UNIVERSIDAD TÉCNICA DEL NORTE



Facultad de Ingeniería en Ciencias Aplicadas

Carrera de Ingeniería en Sistemas Computacionales

DESARROLLO DE UN ALGORITMO PARALELO PARA DETERMINAR LA DENSIDAD DE MALEZA EN CULTIVOS DE MAÍZ, BASADO EN EL ALGORITMO K-MEANS.

Trabajo de grado previa a la obtención del título de Ingeniera en Sistemas Computacionales

Autor:

Ruth Maritza Paspuezan Andrango

Director:

Msc. Marco Remigio Pusdá Chulde

Ibarra-Ecuador

2023

UNIVERSIDAD TÉCNICA DEL NORTE

BIBLIOTECA UNIVERSITARIA

AUTORIZACIÓN DE USO Y PUBLICACIÓN A FAVOR DE LA UNIVERSIDAD TÉCNICA DEL NORTE

1. IDENTIFICACIÓN DE LA OBRA

En cumplimiento del Art. 144 de la Ley de Educación Superior, hago la entrega del presente trabajo a la Universidad Técnica del Norte para que sea publicado en el Repositorio Digital Institucional, para lo cual pongo a disposición la siguiente información:

DATOS DE CONTACTO			
CÉDULA DE IDENTIDAD:	1003104328		
APELLIDOS Y NOMBRES:	PASPUEZAN ANDRANGO RUTH MARITZA		
DIRECCIÓN:	BARRIO IMBABURA- CALLE VENTANAS Y PICHINCHA		
EMAIL:	rmpaspuezana@utn.edu.ec		
TELÉFONO FIJO:	062546022	TELÉFONO MÓVIL:	0939758354

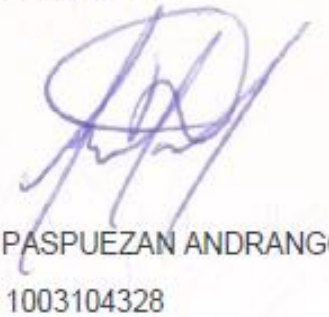
DATOS DE LA OBRA				
TÍTULO:	"DESARROLLO DE UN ALGORITMO PARALELO PARA DETERMINAR LA DENSIDAD DE MALEZA EN CULTIVOS DE MAÍZ, BASADO EN EL ALGORITMO K-MEANS."			
AUTOR (ES):	PASPUEZAN ANDRANGO RUTH MARITZA			
FECHA:	07/09/2023			
PROGRAMA:	PREGRADO	X	POSGRADO	
TITULO POR EL QUE OPTA:	INGENIERO EN SISTEMAS COMPUTACIONALES			
ASESOR /DIRECTOR:	MSc. MARCO REMIGIO PUSDÁ CHULDE			

2. CONSTANCIAS

La autora manifiesta que la obra objeto de la presente autorización es original y se la desarrolló, sin violar derechos de autor de terceros, por lo tanto, la obra es original y que es el titular de los derechos patrimoniales, por lo que asume la responsabilidad sobre el contenido de la misma y saldrá en defensa de la Universidad en caso de reclamación por parte de terceros.

Ibarra, a los 7 días del mes de septiembre de 2023

LA AUTORA:



PASPUEZAN ANDRANGO RUTH MARITZA
1003104328

FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS

CERTIFICACIÓN DEL DIRECTOR

Por medio de la presente yo Msc. Marco Remigio Pusdá Chulde, certifico que la Srta. Ruth Maritza Paspuezan Andrango, portadora de la cedula de ciudadanía Nro. 1003104328, ha trabajado en el desarrollo del proyector de Tesis "**DESARROLLO DE UN ALGORITMO PARALELO PARA DETERMINAR LA DENSIDAD DE MALEZA EN CULTIVOS DE MAÍZ, BASADO EN EL ALGORITMO K-MEANS**", previo a la obtención del título de Ingeniería en Sistemas Computacionales, lo cual ha realizado en su totalidad con responsabilidad y esmero.

Es todo cuanto puedo certificar en honor a la verdad.

Ibarra, a los 7 días del mes de septiembre de 2023.

Atentamente

0401200951
MARCO
REMIGIO PUSDA
CHULDE

Firmado digitalmente
por 0401200951 MARCO
REMIGIO PUSDA
CHULDE
Fecha: 2023.09.07
12:56:08 -05'00'

Msc. Marco Remigio Pusdá Chulde
Tutor de trabajo de grado

DEDICATORIA

Dedico este trabajo de titulación a Dios, a quien he confiado mis esfuerzos y aspiraciones de superación a lo largo de este recorrido educativo. Dios ha sido mi guía en los momentos más desafiantes, iluminando mi camino y fortaleciendo tanto mi mente como mi corazón en este proceso.

A mis padres: **Nelly y José**, por su amor incondicional, apoyo constante y sacrificio incansable. Sin ustedes, este logro no sería posible.

A mi hija: **Nicole**, por ser mi fuente de inspiración y sostén emocional a lo largo de este viaje académico. Tu amor y apoyo han sido mi mayor motivación.

A **mis familiares**, quienes han estado a mi lado desde el principio. Su amor y aliento me han impulsado a dar lo mejor de mí.

A **mis amigos y compañeros** de estudio, quienes han compartido risas, desafíos y momentos inolvidables a lo largo de esta travesía. Gracias por ser mi red de apoyo.

Al **Director de tesis**, quien desempeñó un papel crucial para avanzar hacia la meta establecida, le agradezco sinceramente su apoyo inquebrantable y su dedicación que fueron fundamentales para culminar con éxito.

AGRADECIMIENTO

Agradezco profundamente a **Dios**, quien ha sido mi fuente de inspiración y mi guía constante en esta travesía de la vida. Le agradezco por concederme la fortaleza y la firmeza necesarias para lograr este objetivo académico. Su amor y su compasión han sido mi refugio en momentos de oscuridad y ha sido mi fuente de inspiración en cada paso del camino.

A mis padres **Nelly y José** que de alguna manera me han apoyado incondicionalmente para cumplir mis metas.

De manera especial a mi tutor **Ing. Marco Pusdá, MSc**, quien con sus conocimientos me guío para la elaboración del presente trabajo.

A mis amigos **Vinicio y Erika** por haber hecho que mi experiencia universitaria fuera más agradable y memorable.

Y finalmente a la **Universidad Técnica del Norte** por compartir sus enseñanzas conmigo y permitirme conocer a excelentes profesores y amistades valiosas.

RESUMEN

El objetivo del presente proyecto consiste en desarrollar un algoritmo paralelo para determinar la densidad de maleza en cultivos de maíz, basado en el algoritmo K-means para una gestión agrícola más eficiente y sostenible.

El algoritmo paralelo aprovecha el poder de cómputo distribuido para acelerar el proceso de segmentación de imágenes de campo, clasificando automáticamente la maleza y el maíz en diferentes clústeres utilizando K-means.

Los resultados demuestran una mejora significativa en la precisión y velocidad de detección de maleza en comparación con métodos tradicionales, lo que hace que este enfoque sea valioso en la agricultura de precisión y la toma de decisiones agrícolas. Este desarrollo representa un avance importante en la automatización y optimización de la gestión de cultivos de maíz reduciendo el tiempo de verificación de densidades de maleza y demostrando ser una alternativa viable para la solución rápida de problemas de visión computacional.

ABSTRACT

The objective of this project is to develop a parallel algorithm to determine weed density in corn crops, inspired on the K-means algorithm for a more efficient and sustainable agricultural management.

The parallel algorithm leverages the power of distributed computing to accelerate the field image segmentation process, automatically classifies weeds and corn into different clusters by means of K-means.

The results demonstrate a significant improvement in the accuracy and speed of weed detection compared to traditional methods, making this approach valuable in precision farming and agricultural decision making. This development represents a breakthrough in the automation and optimization of corn, crop management by reducing the time to check weed densities and proving to be a viable alternative for rapid computer vision problem solving.

INDICE DE CONTENIDO

INDICE DE CONTENIDO ix
INDICE DE FIGURAS xii
INDICE DE TABLAS xiv
INTRODUCCIÓN 1
PROBLEMA 1
Antecedentes 1
Situación Actual 1
Prospectiva 3
Planteamiento del problema 4
OBJETIVOS 4
ALCANCE 4
JUSTIFICACIÓN 6
CAPÍTULO 1 9
Marco teórico 9
1.1 Algoritmo K-means 9
1.1.1 Definición 9
1.1.2. Algoritmo 11
1.1.3. Elegir el valor de K 12
1.1.4. Cuantificación de imágenes con k-means 14
1.1.5. Segmentación de imágenes con k-means 17
1.1.6. Casos de uso del algoritmo k-means 17
1.1.7. Beneficio del algoritmo Kmeans 19
1.2. Aprendizaje en las redes neuronales 19
1.2.1 Aprendizaje Supervisado 20
1.2.2. Aprendizaje no Supervisado 21
1.3 Arquitecturas paralelas 22
1.3.1 Single Instruction, Single Data (SISD) 23
1.3.2 Multiple Instruction, Single Data (MISD) 24
1.3.3 Single Instruction, Multiple Data (SIMD): 25
1.3.4. Multiple Instruction, Multiple Data (MIMD): 26
1.4 Lenguajes de programación 27
1.4.1 OpenMP (Open Multi-Processing) 27

1.4.2. PYTHON 30
1.5. Trabajos Relacionados 32
CAPITULO 2 36
DESARROLLO 36
2.1 Análisis del algoritmo Kmeans 36
2.1.1 El algoritmo básico de K-means 37
2.2.2 Elección del Centroide inicial 41
2.2 Diseño del Algoritmo Kmeans 43
2.3 Implementación del Algoritmo Kmeans 49
Eclipse for Parallel Application Developers: 49
GCC 50
MakeFile 51
2.3.1 Herramientas 52
Implementación del algoritmo 54
Equipo de procesamiento 54
2.4 Ejecución del Algoritmo Kmeans 55
CAPITULO 3 57
RESULTADOS 57
3.1 EVALUACIÓN DE TIEMPOS DE EJECUCIÓN 57
3.1.1 Tiempos de ejecución algoritmo Kmeans con 4 clústers 57
3.1.2 tiempos de ejecución algoritmo Kmeans con 6 clústers 60
3.1.3 tiempos de ejecución algoritmo Kmeans con 8 clústers 63
3.1.4 tiempos de ejecución algoritmo Kmeans con 10 clústers 66
3.2. REDUCCIÓN DE TIEMPOS DE EJECUCIÓN 69
3.2.1. Reducción de tiempo algoritmo Kmeans con 4 clústers 69
3.2.2. Reducción de tiempo algoritmo Kmeans con 6 clústers 72
3.2.3. Reducción de tiempo algoritmo Kmeans con 8 clústers 74
3.2.4. Reducción de tiempo algoritmo Kmeans con 10 clústers 76
3.3Tiempo de ejecución promedio 78
3.4 Análisis e interpretación de resultados 79
3.5 Análisis de impactos 82
Impacto científico 82
Impacto tecnológico 83
Impacto Económico 83

CONCLUSIONES 84
RECOMENDACIONES 86
BIBLIOGRAFÍA 88

INDICE DE FIGURAS

Figura 1: Antes de la aplicación del algoritmo Kmeans 11
Figura 2: Después de la aplicación del algoritmo Kmeans 11
Figura 3: Algoritmo K-means 11
Figura 4: Gráfica lineal de la inercia con respecto del número de Clústers-Método del codo 13
Figura 5: Resultado del Método del codo 13
Figura 6: Canales de color RGB de una imagen 15
Figura 7: Matriz de imágenes reformadas 16
Figura 8: Imagen orinal 17
Figura 9: Imagen a color Cuantificada 17
Figura 10: Proceso SISD 24
Figura 11: Proceso MISD 25
Figura 12: Proceso SIMD 26
Figura 13: Proceso MIMD 26
Figura 14: Lanzamiento de múltiples hilos en regiones paralelas 28
Figura 15: Flujograma del algoritmo K-means 38
Figura 16: Uso del algoritmo K-means con tres clústers en datos de muestra 39
Figura 17: Centroide inicial 42
Figura 18: Función segmentación de imagen 44
Figura 19: Función inicialización de centroides 45
Figura 20: Función asignación de pixeles 46
Figura 21: Función actualizar centroides 47
Figura 22: Función actualizar 48
Figura 23: Función cálculo error cuadrático 49
Figura 24: Ubicación de los archivos principales 53
Figura 25: Contenido del archivo Makefile 54
Figura 26: Compilación de los Algoritmos Serial y Paralelo en Imágenes con Dron 56
Figura 27: Tiempos de ejecución imágenes grandes 4 clústers obtenidas con Dron 58
Figura 28: Tiempos de ejecución imágenes medianas con 4 clústers obtenidas con Dron 59
Figura 29: Tiempos de ejecución imágenes pequeñas con 4 clústers obtenidas con Dron 60
Figura 30: Tiempos de ejecución imágenes grandes con 6 clústers obtenidas con Dron 61
Figura 31: Tiempos de ejecución imágenes medianas con 6 clústers obtenidas con Dron 62
Figura 32: Tiempos de ejecución imágenes pequeñas con 6 clústers obtenidas con Dron 63

Figura 33: Tiempos de ejecución imágenes grandes con 8 clústers obtenidas con Dron..... 64
Figura 34: Tiempos de ejecución imágenes medianas con 8 clústers obtenidas con Dron..... 65
Figura 35: Tiempos de ejecución imágenes pequeñas con 8 clústers obtenidas con Dron..... 66
Figura 36: Tiempos de ejecución imágenes grandes con 10 clústers obtenidas con Dron..... 67
Figura 37: Tiempos de ejecución imágenes medianas con 10 clústers obtenidas con Dron.... 68
Figura 38: Tiempos de ejecución imágenes pequeñas con 10 clústers obtenidas con Dron.... 69
Figura 39: Tiempos de ejecución imágenes grandes con 4 clústers obtenidas con Dron..... 70
Figura 40: Tiempos de ejecución imágenes medianas con 4 clústers obtenidas con Dron..... 71
Figura 41: Tiempos de ejecución imágenes pequeñas con 4 clústers obtenidas con Dron..... 72
Figura 42: Tiempos de ejecución imágenes grandes con 6 clústers obtenidas con Dron..... 73
Figura 43: Tiempos de ejecución imágenes medianas con 6 clústers obtenidas con Dron..... 73
Figura 44: Tiempos de ejecución imágenes pequeñas con 6 clústers obtenidas con Dron..... 74
Figura 45: Tiempos de ejecución imágenes grandes con 8 clústers obtenidas con Dron..... 75
Figura 46: Tiempos de ejecución imágenes medianas con 8 clústers obtenidas con Dron..... 75
Figura 47: Tiempos de ejecución imágenes pequeñas con 8 clústers obtenidas con Dron..... 76
Figura 48: Tiempos de ejecución imágenes grandes con 10 clústers obtenidas con Dron..... 77
Figura 49: Tiempos de ejecución imágenes medianas con 10 clústers obtenidas con Dron.... 77
Figura 50: Tiempos de ejecución imágenes pequeñas con 10 clústers obtenidas con Dron.... 78
Figura 51: Tiempos de ejecución promedio en imágenes grandes obtenidas con Dron..... 79
Figura 52: Reducción de tiempo de ejecución promedio para imágenes grandes 80
Figura 53: Reducción de tiempo de ejecución promedio para imágenes medianas 81
Figura 54: Reducción de tiempo de ejecución promedio para imágenes pequeñas 82

INDICE DE TABLAS

Tabla 1: Tabla de notación..... 41
Tabla 2: Especificaciones técnicas de Laptop con Ubuntu Linux..... 55
Tabla 3: Tiempos ejecución imágenes grandes con 4 clústers obtenidas con Dron 57
Tabla 4: Tiempos ejecución imágenes medianas 4 clústers obtenidas con Dron..... 58
Tabla 5: Tiempos ejecución imágenes pequeñas con 4 clústers obtenidas con Dron..... 59
Tabla 6: Tiempos ejecución imágenes grandes con 6 clústers obtenidas con Dron 60
Tabla 7: Tiempos ejecución imágenes medianas con 6 clústers obtenidas con Dron..... 61
Tabla 8: Tiempos ejecución imágenes pequeñas con 6 clústers obtenidas con Dron..... 62
Tabla 9: Tiempos ejecución imágenes grandes con 8 clústers obtenidas con Dron 63
Tabla 10: Tiempos ejecución imágenes medianas con 8 clústers obtenidas con Dron..... 64
Tabla 11: Tiempos ejecución imágenes pequeñas con 8 clústers obtenidas con Dron..... 65
Tabla 12: Tiempos ejecución imágenes grandes con 10 clústers obtenidas con Dron 66
Tabla 13: Tiempos ejecución imágenes medianas con 10 clústers obtenidas con Dron..... 67
Tabla 14: Tiempos ejecución imágenes pequeña con 10 clústers obtenidas con Dron 68
Tabla 15: Tiempos ejecución promedio de imágenes con Dron..... 78
Tabla 16: Tiempos de ejecución promedio de imágenes Grandes con dron 79
Tabla 17: Tiempos de ejecución promedio de imágenes Medianas con dron 80
Tabla 18: Tiempos de ejecución promedio de imágenes pequeñas con dron 81

INTRODUCCIÓN

PROBLEMA

Antecedentes

"El procesamiento digital de imágenes es una herramienta ampliamente utilizada en la automatización de procesos industriales, debido a que presenta confiabilidad, eficacia y rapidez en el procesamiento. La industria agrícola ha comenzado a utilizar este tipo de tecnología para monitorear aspectos relevantes de los cultivos" (Li et al., 2009)

En la actualidad con el desarrollo de algoritmos secuenciales solo es posible obtener un buen trabajo cuando se explotan los detalles de la arquitectura del computador. Al involucrar procesos digitales de imágenes o ejecutar aplicaciones de visión por computador se necesita tiempo y recursos, "una ejecución paralela se permite ejecutar varias operaciones simultáneamente, procesar varios hilos de ejecución concurrentemente, o incluso ejecutar al mismo tiempo diferentes aplicaciones en varios procesadores integrados en un único computador." (Román et al., 2018)

En la actualidad es muy común poseer máquinas con una arquitectura robusta que ejecuten una cantidad cada vez mayor de cargas de trabajo, como el aprendizaje profundo y la inteligencia artificial (IA), obteniendo un aumento significativo en el rendimiento y llevando el entrenamiento de varios problemas del mundo real a un rango factible y viable. (Corporación Intel, 2022)

Situación Actual

Actualmente La Ley Orgánica del Régimen de la Soberanía Alimentaria contiene "políticas públicas agroalimentarias para fomentar la producción, conservación, intercambio, transformación, comercialización y consumo de alimentos sanos y nutritivos, preferentemente provenientes de la micro, pequeña y mediana producción campesina de las organizaciones económicas populares respetando y protegiendo la Agrobiodiversidad, los conocimientos y formas de producción tradicionales y ancestrales, bajo los principios de equidad, solidaridad, inclusión, sustentabilidad social y ambiental."(LORSA, 2010)

El surgimiento de nuevas tecnologías para la computación de alto rendimiento, así como el surgimiento de arquitecturas multi-core y multi-GPU, "aligeran la solución de problemas de alto coste computacional, para optimizar el rendimiento de estos ya sea para el procesamiento de imágenes digitales en tareas agrícolas en ente caso determinar la densidad de maleza en campos de cultivo de maíz utilizando dispositivos autónomos en tiempo real". (Espeso, 2017)

Investigaciones llevadas a cabo con referencia al control de malezas en el cultivo de maíz, han tomado en cuenta los efectos de las diferentes concentraciones de los herbicidas, repercutiendo en la reducción de la salud en las plantas cultivadas, la capacidad de productividad de los cultivos, la eficiencia de la fertilización y de la irrigación. Por otro lado, debido a que las malezas disminuyen los niveles de agua, nutrientes, luz y $C0^2$, provocan la disminución drástica de la calidad del cultivo y el rendimiento agrícola, sumado a que son además hospedadoras de plagas y enfermedades. (Martínez Carriel et al., 2021)

La visión por computador ha sido de mucha utilidad para el procesamiento de imágenes de cultivo en conjunto con la agricultura de precisión (AP), busca aumentar la productividad en las actividades agrícolas, como por ejemplo "determinar la ubicación y eliminación de malezas en cultivos, mejorando de manera considerable la calidad de los cultivos, además de proveer

información de mucha utilidad para cosechas futuras , como la aplicación adecuada de menos herbicidas, bajando así los costos de producción y proporcionando un mayor rendimiento económico".(EOS Data Analytics inc., 2020)

La Agricultura de Precisión (AP) tiene como objetivo optimizar la productividad y rentabilidad de los cultivos mediante una gestión más práctica teniendo en cuenta los recursos agrícolas, lo que implica la recopilación y el procesamiento intensivo, en el tiempo y el espacio, de datos e información sobre los cultivos, por tanto se requiere la obtención de mapas precisos del desarrollo de los cultivos, la presencia de maleza, la aparición de plagas de insectos, la falta de nutrientes y otras condiciones del cultivo y del suelo. (Bordón López, 2018)

Prospectiva

El presente proyecto pretende desarrollar un algoritmo que permita fortalecer el procesamiento de imágenes de cultivos usando herramientas de programación paralela (OPENMP, C++), donde ésta permita determinar las zonas que tengan mayor densidad de maleza en cultivos de maíz, de esta se ayudaría al agricultor a tomar decisiones que permitan erradicar las malas hierbas en las zonas previamente determinadas por el algoritmo.

Después de analizar las herramientas de programación, se realizará una comparación de las mismas, seleccionando la óptima para el desarrollo del algoritmo que procese imágenes agrícolas para la determinar la densidad de malezas en los campos de cultivos; de esta manera se estaría sistematizando tareas para reducir el tiempo de verificación de densidades de maleza, demostrando ser una alternativa viable para la solución rápida de problemas de visión computacional.

Planteamiento del problema

En la Zona 1 del Ecuador la mayoría de las actividades agrícolas es realizada de forma manual y son escasos los procesos sistematizados, en comparación a otros países o regiones del mundo donde se incluye la tecnología como parte de la solución de problemas de cultivos de maíz, las aplicaciones existentes en el campo de la visión artificial para el sector agrícola se enfocan en el procesamiento de imágenes de diferentes cultivos y características.

OBJETIVOS

Objetivo General

Desarrollar un algoritmo paralelo para determinar la densidad de maleza en cultivos de maíz, basado en el algoritmo K-means.

Objetivos Específicos

- Elaborar un marco teórico respecto a lenguajes de programación y algoritmos de aprendizaje no supervisado para determinar densidad de maleza.
- Implementar el algoritmo K-means para el procesamiento de imágenes en lenguaje de programación paralela.
- Evaluar Resultados de la investigación propuesta.

ALCANCE

Para el desarrollo de la presente investigación se utilizará herramientas de desarrollo para entornos paralelos bajo el sistema operativo Linux que permitan sistematizar tareas de

agricultura de precisión utilizando procesamiento de imágenes obtenidas con DRON, permitiendo detectar la densidad de maleza mediante aprendizaje automático no supervisado en cultivos de maíz.

Posteriormente se desarrollará el algoritmo Kmeans que consiste en agrupar datos similares para descubrir patrones que a simple vista se desconocen. (Martínez Heras José, 2020) menciona que, para conseguir dichos patrones, el algoritmo busca un número fijo (k) de clústers en el Dataset siguiendo los pasos:

1. **Inicialización**: se elige la localización de los centroides de los K grupos aleatoriamente
2. **Asignación:** se asigna cada dato al centroide más cercano
3. **Actualización**: se actualiza la posición del centroide a la media aritmética de las posiciones de los datos asignados al grupo.

Finalmente se realizará pruebas de verificación del algoritmo implementado en una arquitectura paralela.

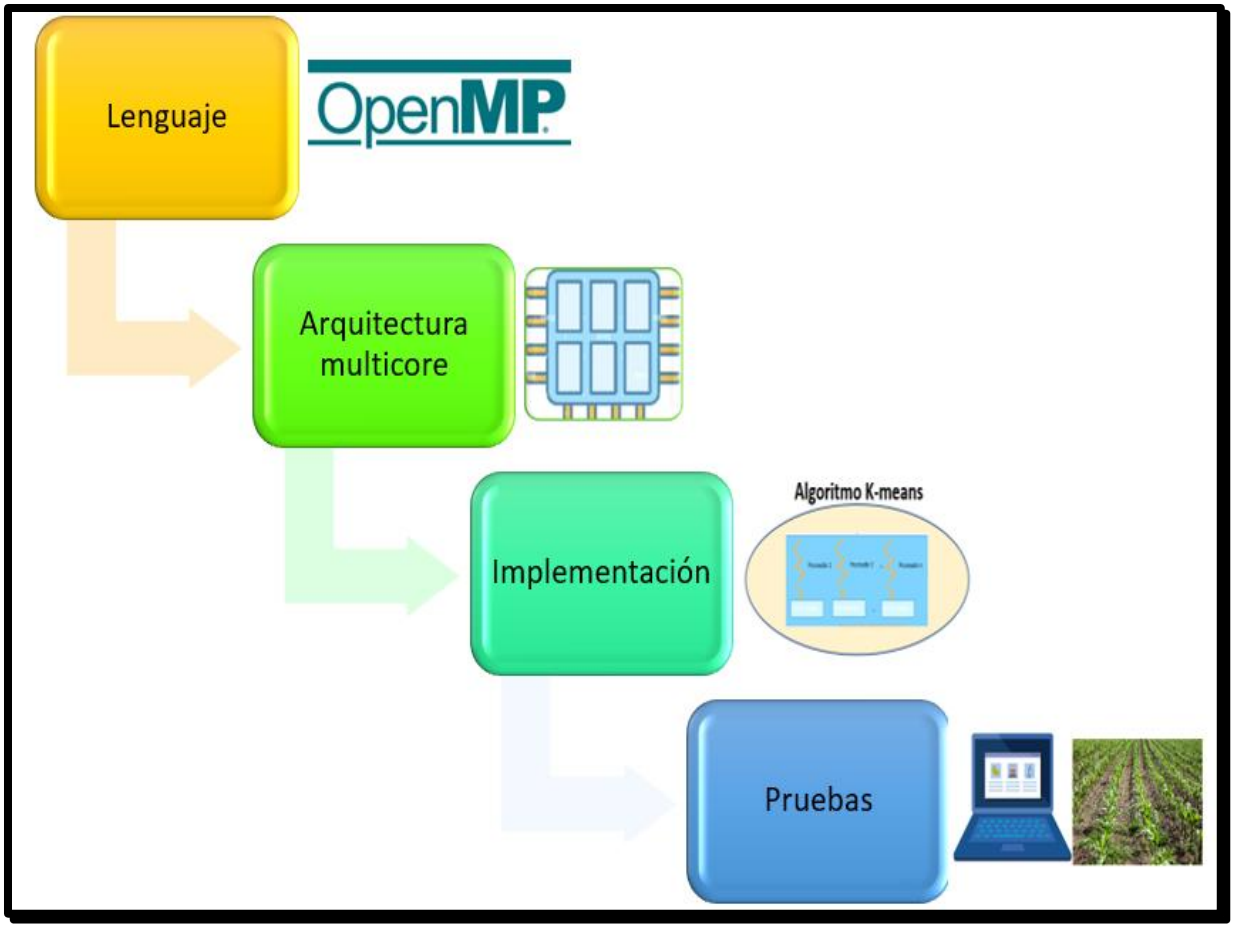


Ilustración 1: Alcance del Proyecto

JUSTIFICACIÓN

Las razones de implementar este proyecto es investigar y recolectar información, ya que no existe un proyecto que determine la densidad de maleza en entornos paralelos como es OpenMp aplicados a la agricultura de precisión. Además, se busca destacar la importancia y las ventajas de estas herramientas en el ámbito de desarrollo en entornos paralelos mediante programación de alto rendimiento.

El presente proyecto tiene un enfoque hacia (Objetivo de Desarrollo Sostenible) ODS 2: Hambre cero y ODS 9: Industria, Innovación e Infraestructura, ya que trata de “mejorar la productividad agrícola y los ingresos de los productores de alimentos en pequeña esca-

la”(NACIONES UNIDAS EN ECUADOR, 2016), además de “apoyar al desarrollo de tecnologías, la investigación y la innovación nacionales en los países en desarrollo, incluso garantizando un entorno normativo propicio a la diversificación industrial y la adición de valor a los productos básicos, entre otras cosas.”(NACIONES UNIDAS EN ECUADOR, 2016)

Justificación Tecnológica. - La implementación de algoritmos de alto nivel se puede ejecutar de forma concurrente en un dispositivo logrando acelerar los tiempos de ejecución con respecto a los ejecutados de forma secuencial ayudando a resolver el problema de manera más rápida.

Justificación Metodológica. - Utilizar la metodología correcta nos permite analizar los resultados que se pueden obtener en el campo de la agricultura de precisión.

Metodología.

En el presente proyecto se aplicarán buenas prácticas de programación (Martin, 2012) y metodología en cascada que consiste en:

- Análisis: Se considerará los requisitos necesarios para sistematizar la detección de densidad en líneas de cultivo mediante imágenes offline adquiridas con dron.
- Diseño: Se llevará a cabo la adquisición, preprocesamiento, procesamiento y clasificación de las imágenes para el diseño del algoritmo.
- Implementación: Se desarrollará el algoritmo utilizando OpenMp para las diferentes fases del procesamiento de imágenes agrícolas.

- Verificación: Se realizará pruebas de funcionamiento utilizando imágenes RGB en formato jpg, con diferentes métricas como dimensiones de imágenes y características obtenidas en cultivos de maíz.

- Mantenimiento: Se corregirá posibles errores del algoritmo para la detectar la densidad de maleza en cultivos presentes en las imágenes agrícolas. (Digital Guide IONOS, 2019)

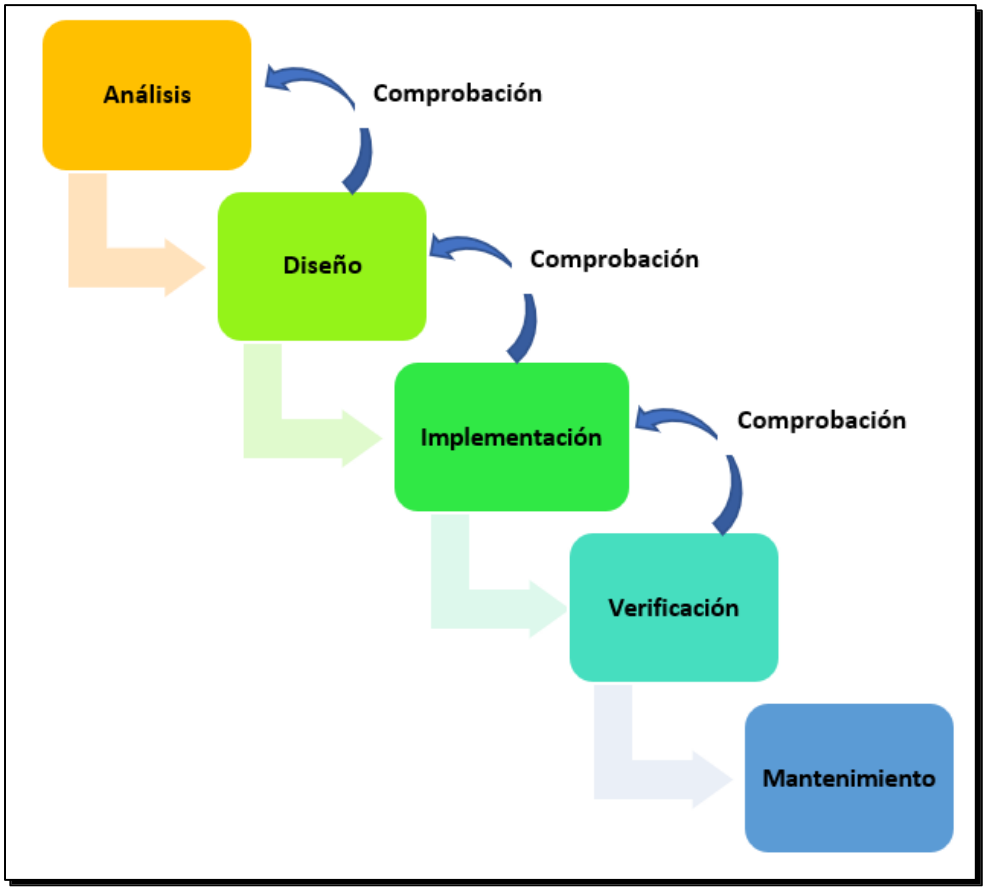


Ilustración 2: Metodología de cascada

Una vez que se desarrolle el algoritmo se realizarán evaluaciones de rendimiento con un conjunto de Imágenes Offline (no a tiempo real) previamente adquiridas con dron para evaluar los tiempos de ejecución.

CAPÍTULO 1

Marco teórico

1.1 Algoritmo K-means

El algoritmo K-means desarrollado por MacQueen en 1967 es el algoritmo de agrupación más reconocido y utilizado debido a su simplicidad y eficacia, el principal objetivo es crear una partición de un conjunto de n observaciones en k grupos, donde cada grupo está representado por el promedio de los puntos que lo componen. El representante de cada grupo se denomina centroide. La cantidad de grupos a precisar, k, es un parámetro que se debe establecer previamente. (Galiano Casas & García Gonzalo, 2014)

1.1.1 Definición

K-means es un algoritmo de clasificación no supervisada (cauterización) que organiza objetos en k grupos en base a las características. La agrupación de los objetos se realiza minorando la suma de distancias entre cada objeto y el centroide de su grupo o clúster, para dicho proceso se suele usar la distancia cuadrática. (Pastrán Ramírez & Gongora Aya, 2021).

Los objetos se representan con vectores reales de d dimensiones) y el algoritmo K-means construye k grupos donde se minimiza la suma de distancias de los objetos, dentro de cada grupo S=($S_1, S_2 \ldots, S_k$), a su centroide.

(Pastrán Ramírez & Gongora Aya, 2021), pudo formular el problema de la siguiente forma (Ecuación 1):

$$\min_{S} E(\mu_i) = \min_{S} \sum_{i=1}^{k} \sum_{x_{j \in S_i}} \|x_j - \mu_i\|^2$$

Ecuación 1. *Ecuación para minimizar la suma de distancias*

donde ***S*** es el conjunto de datos cuyos elementos son los objetos x_j representados por vectores, donde cada uno de sus elementos representa una característica o atributo. Se tendrán k grupos o clústers con su correspondiente centroide μ_i.

En cada actualización de los centroides, desde el punto de vista matemático, se impone la condición necesaria de extremo a la función $E(\mu_i)$que, para la función cuadrática (Ecuación 2), sea:

$$\frac{\partial E}{\partial \mu_i} = 0 \rightarrow \mu_i{}^{t+1} = \frac{1}{\left|S_i{}^{(t)}\right|} \sum_{x_{j \in S_i{}^{(t)}}} x_j$$

Ecuación 2. *Ecuación para actualizar los centroides*

y se toma el promedio de los elementos de cada grupo como nuevo centroide.

Figura 1: *Antes de la aplicación del algoritmo Kmeans*
Fuente: (Angulo et al., 2022)

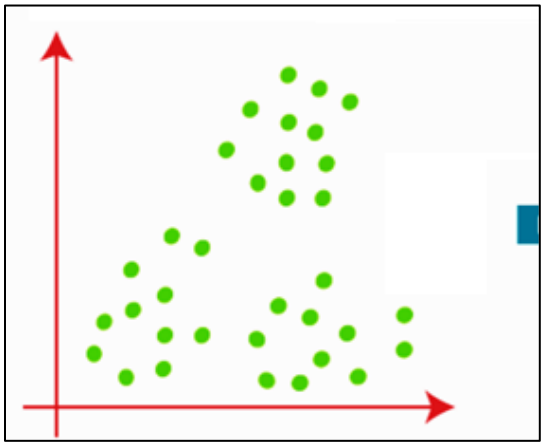
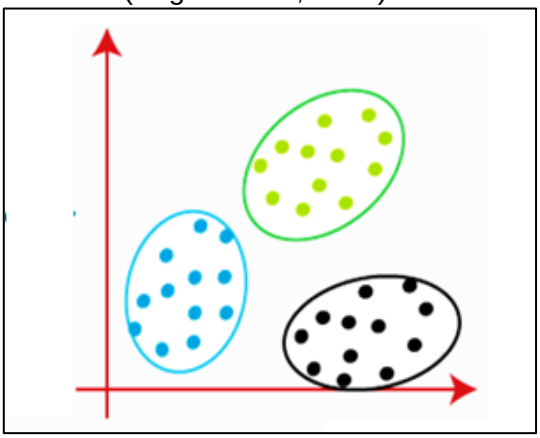


Figura 2: *Después de la aplicación del algoritmo Kmeans*
Fuente: (Angulo et al., 2022)

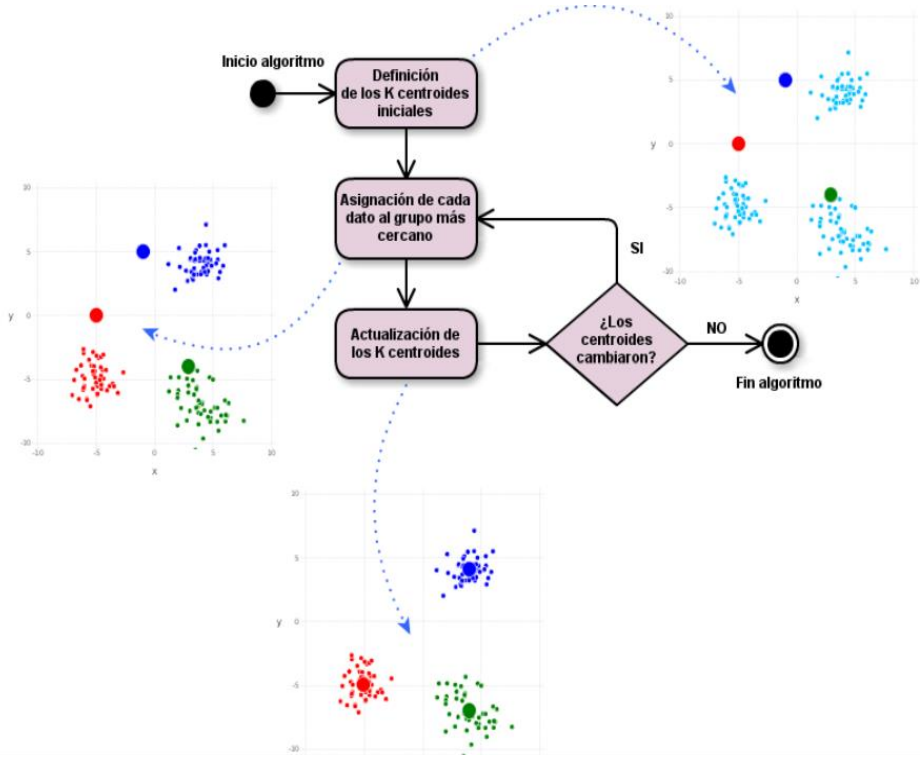


1.1.2. Algoritmo

(Pastrán Ramírez & Gongora Aya, 2021) menciona que: el algoritmo K-means se realiza en 3 pasos:

1. **Inicialización:** una vez escogido el número de grupos, k, se establecen k centroides en el espacio de los datos, por ejemplo, escogiéndolos aleatoriamente.
2. **Asignación objetos a los centroides:** cada objeto de los datos es asignado a su centroide más cercano.
3. **Actualización centroides:** se actualiza la posición del centroide de cada grupo tomando como nuevo centroide la posición del promedio de los objetos pertenecientes a dicho grupo y finalmente se repiten los pasos 2 y 3 hasta que los centroides no se mueven, o se mueven por debajo de una distancia umbral en cada paso.

Figura 3: *Algoritmo K-means*
Fuente: (Universidad Nacional de La Plata, 2016)



Aunque el algoritmo termina siempre, no se garantiza el obtener la solución óptima. De la misma manera, el algoritmo es susceptible a la aleatoriedad en la elección de los k centros iniciales. Esta es la razón por la que, se utiliza el algoritmo del K-means numerosas veces sobre un mismo conjunto de datos para intentar minimizar este efecto, sabiendo que a centros iniciales lo más espaciados posibles dan mejores resultados. (De La Cruz Gutiérrez Keyla et al., 2015)

1.1.3. Elegir el valor de K

El algoritmo k-means trabajaba preseleccionando un valor de K. Para determinar el número de clústers en los datos, en necesario ejecutar el algoritmo con un intervalo de valores K, examinar los resultados y realizar la comparación con las características de los grupos obtenidos. No existe una forma exacta para determinar el valor K, pero se puede realizar un cálculo

aproximado para estos valores siguiendo ciertas normas que nos ayudan a estimar el número de grupos:

El método del codo: utiliza los valores de la inercia obtenidos tras aplicar el K-means a diferente número de Clústers (desde 1 a N Clústers), siendo la inercia la suma de las distancias al cuadrado de cada objeto del Clúster a su centroide. Una vez obtenidos los valores de la inercia tras aplicar el K-means de 1 a N Clústers, representamos en una gráfica lineal la inercia respecto del número de Clústers. En la figura 7 se debería de apreciar un cambio brusco en la evolución de la inercia, teniendo la línea representada una forma similar a la de un brazo y su codo. El punto en el que se observa ese cambio brusco en la inercia nos dirá el número óptimo de Clústers a seleccionar para ese data set; o, dicho de otra manera: el punto que representaría al codo del brazo será el número óptimo de Clústers para ese data set (figura 4). (Moya Ricardo, 2016)

Figura 4: *Gráfica lineal de la inercia con respecto del número de Clústers-Método del codo*
Fuente: (Moya Ricardo, 2016)

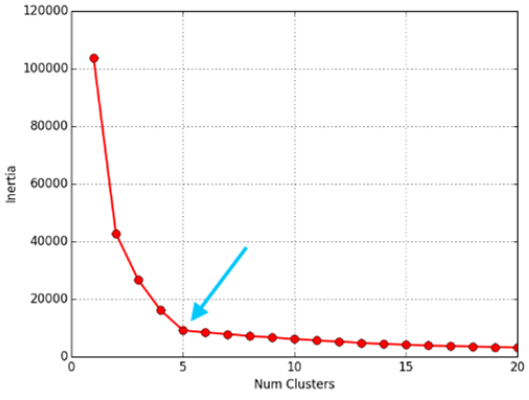
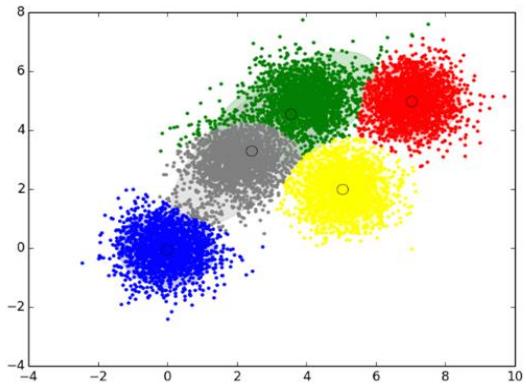


Figura 5: *Resultado del Método del codo*
Fuente: (Moya Ricardo, 2016)



Como se puede observar en la Figura 5, el método del codo devuelve valores coherentes y se ajusta a los resultados. Es posible que al usar este método para otro conjunto de datos no se aprecie “el codo” o incluso se observen dos o más codos.

Análisis de la silueta: mide la calidad del agrupamiento o clustering. Mide la distancia de separación entre los clústers. Nos indica como de cerca está cada punto de un clúster a puntos de los clústers vecinos. Esta medida de distancia se encuentra en el rango [-1, 1]. Un valor alto indica un buen clustering.

Si es 0 -> la muestra está muy cerca de los clústers vecinos.

Si es 1 –> la muestra está lejos de los clústers vecinos.

Si es -1 -> la muestra se asigna a los clústers incorrectos.

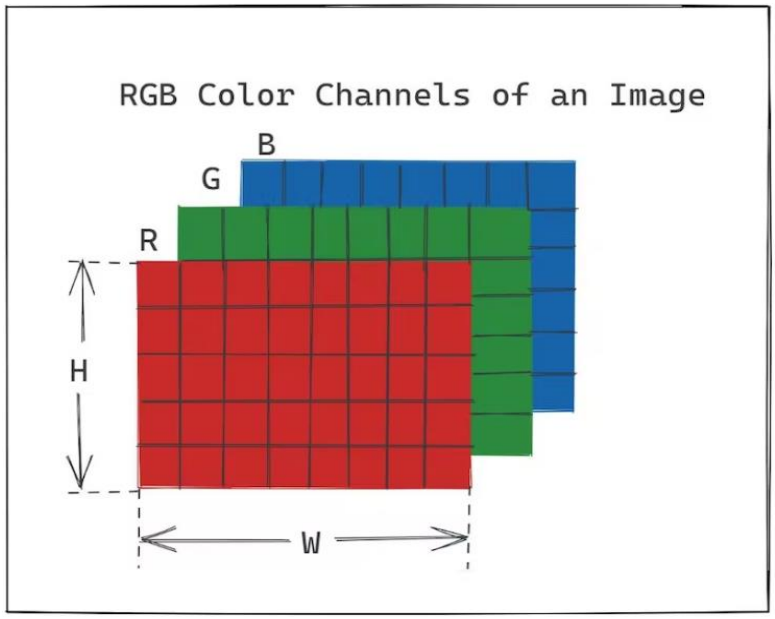
Los coeficientes de silueta cercanos a +1 indican que la observación se encuentra lejos de los clústers vecinos. Un valor del coeficiente de 0 indica que la observación está muy cerca o en la frontera de decisión entre dos clústers. Valores negativos indican que esas muestras quizás estén asignadas al clúster erróneo. El método de la silueta calcula la media de los coeficientes de silueta de todas las observaciones para diferentes valores de k. El número óptimo de clústers k es aquel que maximiza la media de los coeficientes de silueta para un rango de valores de k.

1.1.4. Cuantificación de imágenes con k-means

Cuantificación es una técnica de compresión que permite concentrar todo un rango de valores en uno solo, cuando cuantificamos el color de una imagen, minimizamos el número de colores que se necesita para representarla y como resultado disminuye el tamaño del fichero.

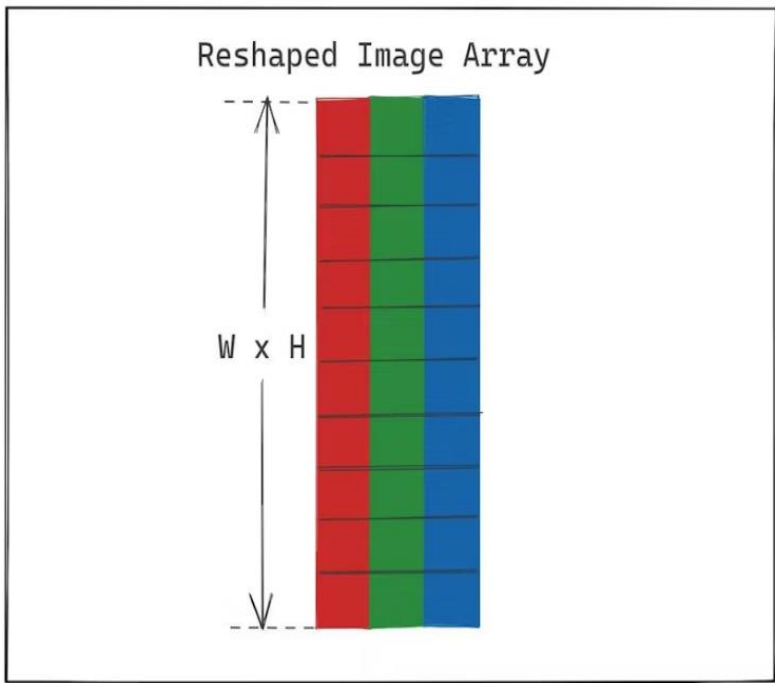
En una imagen en color, cada píxel es una tupla de valores de rojo, verde y azul (R, G, B). Se puede mezclar rojo, verde y azul en proporciones variables para obtener una amplia gama de colores. Por lo tanto, una imagen que tiene W píxeles de ancho y H píxeles de alto y tiene los canales de color R, G y B se puede representar como se muestra en la figura 6. (Bala Priya C., 2022)

Figura 6: *Canales de color RGB de una imagen*
Fuente: (Bala Priya C., 2022)



Para cuantificar la imagen de modo que contenga solo K colores, se considera que los valores R, G, B son características en sí mismos si podemos cambiar la forma de la matriz de imágenes como se muestra a continuación; básicamente, estamos aplanando las matrices de color de modo que cada punto con sus valores RGB sea un punto de datos.

Figura 7: *Matriz de imágenes reformadas*
Fuente: (Bala Priya C., 2022)



De este modo se puede realizar el agrupamiento de K-means, los centros de clúster que se obtienen son puntos en el mismo espacio, lo que significa que también serán tonos de color, con valores RGB válidos, cada punto de datos es un color, de esta forma se ha reducido el espacio de color de la imagen a K colores; los K colores son los colores de los centros de los grupos.

Figura 8: *Imagen orinal*
Fuente: (Bala Priya C., 2022)

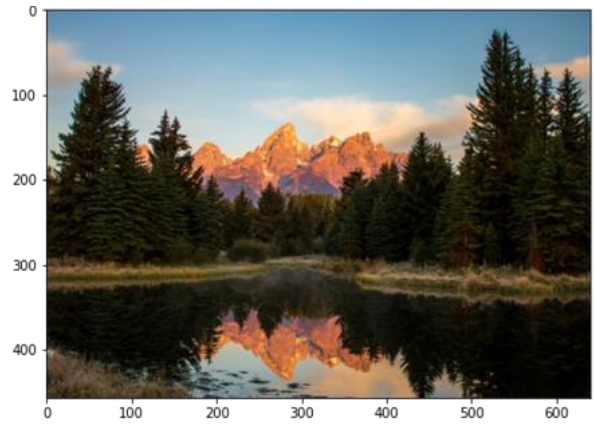
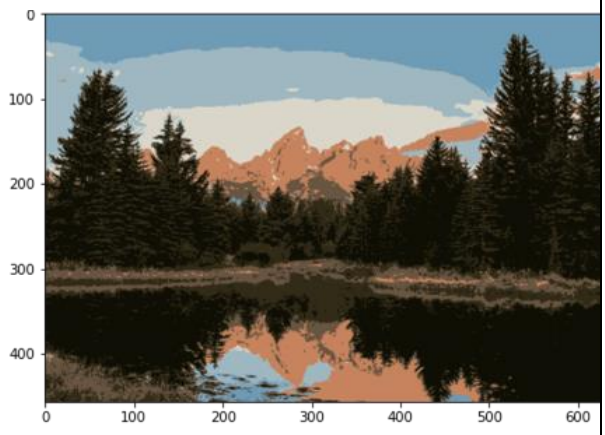


Figura 9*: Imagen a color Cuantificada*
Fuente: (Bala Priya C., 2022)



1.1.5. Segmentación de imágenes con k-means

La segmentación consiste en la división de una imagen en áreas que muestran características internas comunes en distintos elementos que luego podemos estudiar por separado, se puede segmentar una imagen utilizando el color, formas, bordes, gradientes, ubicación absoluta o relativa. El proceso es similar al de cuantización de imágenes, la diferencia es el objetivo con el que se agrupan los píxeles: se agrupan los píxeles para separar los elementos significativos de una imagen y poder extraer cierta información de alguno de ellos. (Galiano Casas & García Gonzalo, 2014)

1.1.6. Casos de uso del algoritmo k-means

Los algoritmos no supervisados de clustering como k-means tienen la capacidad de identificar agrupaciones que no son evidentes o que no están previamente definidas en datos no etiquetado, los resultados obtenidos pueden servir para validar o descartar alguna presunción sobre los datos. Además, se usan para descubrir conexiones y patrones entre los datos, ya que manualmente sería difícil de determinar.

Una vez ejecutado el algoritmo se obtiene los grupos o etiquetas, obteniendo nuevos valores para clasificar, tenemos algunos casos de uso del algoritmo Kmeans(ALVARO, 2019):

Segmentación del comportamiento(usuarios/clientes):

- Segmento por historial de compras.
- Segmento por actividades en la aplicación, sitio web o plataforma.
- Definir a las personas en función de sus intereses.
- Crear perfiles basados en la supervisión de actividades.

Categorización de inventario:

- Agrupar el inventario por actividad de ventas.
- Agrupar el inventario por métricas de fabricación.

Clasificación de las medidas de los sensores:

- Detectar tipos de actividad en los sensores de movimiento.
- Agrupar imágenes.
- Audio separado.
- Identificar grupos en la vigilancia de la salud.

Detección de robots o anomalías:

- Separar grupos de actividades válidos de los bots.
- Agrupar actividad válida para limpiar la detección de valores atípicos.

1.1.7. Beneficio del algoritmo Kmeans

Los beneficios del algoritmo K-means es que es un método sencillo y rápido, se usa como preprocesamiento para otros algoritmos, como buscar una configuración inicial para ajustarse fácilmente con heurísticas que se aplican en áreas como la segmentación de mercados, visión por computadoras, geoestadística, astronomía y minería de datos y agricultura. (Pastrán Ramírez & Gongora Aya, 2021)

Según (International Business Machines Corporation, 2019), al igual que cualquier algoritmo de machine learning, k-means tiene sus puntos fuertes:

- **Fácil de implementar:** La simplicidad y precisión del algoritmo, es uno de los primeros clasificadores que aprenderá un nuevo científico de datos.
- **Se adapta fácilmente**: A medida que se agregan nuevas muestras de entrenamiento, el algoritmo se ajusta para tener en cuenta cualquier dato nuevo, ya que todos los datos de entrenamiento se almacenan en la memoria.
- **Pocos hiperparámetros**: Kmeans solo requiere un valor k y una métrica de distancia, que es baja en comparación con otros algoritmos de machine learning.

1.2. Aprendizaje en las redes neuronales

El aprendizaje es el proceso donde una red neuronal ajusta sus pesos en función de la información de entrada que recibe. Los cambios que se producen durante la etapa de aprendizaje se reducen a la destrucción (el peso de la conexión toma el valor 0), modificación y creación (el peso de la conexión toma un valor distinto de 0) de conexiones entre las neuronas.(Hilera González & Martínez Hernando, 1995)

El aprendizaje en las redes neuronales en la agricultura tiene aplicaciones prometedoras en diversos aspectos de la industria agrícola, desde la optimización de los rendimientos de

los cultivos hasta el diagnóstico de enfermedades de las plantas. Las redes neuronales, como parte del campo más amplio del aprendizaje automático, pueden ayudar a mejorar la productividad, la eficiencia y la sostenibilidad en la industria agrícola. Al analizar grandes conjuntos de datos y reconocer patrones complejos, las redes neuronales pueden ayudar a los agricultores a tomar decisiones más informadas y precisas, optimizando así los resultados agrícolas. Podemos distinguir diferentes tipos de aprendizaje con las que un algoritmo de aprendizaje automático puede aprender, como el aprendizaje supervisado y aprendizaje no supervisado.

1.2.1 Aprendizaje Supervisado

El aprendizaje supervisado consiste en aprender iterativamente de los datos para permitir que las computadoras localicen información escondida sin tener que programar de manera explícita dónde buscar, el objetivo consiste en dejar que la máquina aprenda sin indicaciones de los datos, conforme aprenda debe ajustar los resultados y agrupaciones permitiendo que la máquina comprenda los datos y los procese de manera eficaz.(TIBCO Software Inc, 2022)

Existen varios tipos de algoritmos de aprendizaje supervisado, incluyendo árboles de decisión, regresión lineal, regresión logística, máquinas de vectores de soporte (SVM) y redes neuronales. Cada algoritmo tiene sus propias fortalezas y debilidades, y la elección del algoritmo depende del problema específico que se esté abordando.

El aprendizaje supervisado se utiliza en varios campos, incluyendo la medicina, la banca, la industria manufacturera y la publicidad en línea. Es una herramienta poderosa para automatizar tareas y tomar decisiones precisas basadas en datos. Sin embargo, el aprendizaje supervisado también tiene limitaciones, como la necesidad de datos etiquetados y la posibilidad de sobreajuste del modelo.(IBM, s.f.)

1.2.2. Aprendizaje no Supervisado

El aprendizaje no supervisado en la agricultura implica utilizar técnicas de aprendizaje automático para descubrir patrones, estructuras y relaciones ocultas en los datos agrícolas sin la necesidad de etiquetas o información de salida previa. A diferencia del aprendizaje supervisado, en el que se utilizan datos etiquetados para entrenar un modelo, el aprendizaje no supervisado se basa en la extracción de información útil a partir de datos no etiquetados.(IBM, s.f.)

El aprendizaje no supervisado no requiere influencia externa, es decir se presenta una serie de datos de entrada, pero no los correspondientes datos de salida, no existe conocimiento previo, la red no recibe ninguna información por parte del entorno que le indique si la salida generada en respuesta a una determinada entrada es o no correcta; por este motivo se describe que estas redes tienen la capacidad de "auto-organizarse", el objetivo es identificar las características, patrones, relaciones o categorías que pueden haber entre los datos de la entrada.

(López Takeyas, 2007) menciona que existen varias posibilidades en cuanto a interpretación :

- La salida representa el grado de familiaridad o similitud entre la información de entrada y las informaciones mostradas con anterioridad.
- **Cauterización o establecimiento de categorías**, consiste en que la red determina a qué categoría corresponde la información de entrada generando una salida que de modo autónomo establece las correlaciones adecuadas.
- **Codificación de los datos de entrada**, consiste en generar una versión de salida que contenga menos bits, pero conserve la información esencial de los datos originales.

- **Mapeo de características**, se crea una disposición geométrica que representa un mapa topográfico de las características de los datos de entrada.

Existen dos tipos de aprendizaje no supervisado, suelen ser:

Aprendizaje hebbiano: procura evaluar la similitud o identificar características de los datos de entrada. Este aprendizaje consiste básicamente en el ajuste de los pesos de las conexiones de acuerdo con la correlación de los valores de activación (salidas) de las dos neuronas conectadas.(Hilera González & Martínez Hernando, 1995)

Aprendizaje competitivo y comparativo: busca agrupar o categorizar los datos de entrada. En el aprendizaje competitivo, si un patrón nuevo se determina que pertenece a una clase previamente reconocida, la inclusión de este patrón en esa clase cambiará la representación de dicha clase. Si el patrón de entrada se determinó que no pertenece a ninguna de las clases reconocidas anteriormente, entonces la estructura y los pesos de la red neuronal serán ajustados para reconocer la nueva clase. (Matich Damián Jorge, 2001)

1.3 Arquitecturas paralelas

La computación paralela basa en resolver problemas de gran magnitud dividiendo una tarea en varias tareas de menor complejidad donde pueden ser resueltos en paralelo, es decir, de forma concurrente. En esta técnica de programación se utilizan dos o más procesadores para que se encarguen de la resolución de las tareas, es significativo tener en cuenta que, para diseñar un programa bajo los parámetros de la computación paralela, se deben considerar el tipo de arquitectura sobre la que será ejecutada el programa y sus necesidades de tiempo y espacio, así como el mecanismo de programación paralelo que sea el adecuado, esto se distingue de la computación secuencial en que varias operaciones pueden ocurrir simultáneamente. La sincronización en los programas en paralelo necesita orden en los procesos e hilos, para

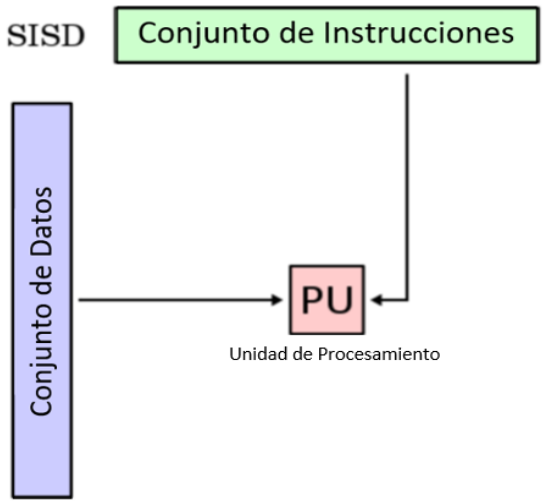
una ejecución correcta, de manera que intercambien información dependiendo de cómo está organizada la memoria en el hardware. El balanceo de carga distribuye cantidades equitativas de trabajo entre todas las tareas de modo que se mantengan ocupadas todo el tiempo., en cambio la concurrencia opera actividades al mismo tiempo obteniendo varios procesos corriendo cada uno en un procesador o varios procesos pueden correr en un solo procesador. (Bernal et al., 2020)

En el mundo de la computación el paralelismo se puede aplicar a otras áreas, la más conocida es la clasificación de Flynn quien tomó dos aspectos a consideración: el flujo de instrucciones y el flujo de datos, de acuerdo con la multiplicidad de cada uno, se clasifican en los siguientes:

1.3.1 Single Instruction, Single Data (SISD)

SISD (Instrucción Única, Flujo de Datos Único), es el computador secuencial convencional, de acuerdo con el modelo de Von Neumann, se toma el conjunto de datos, un conjunto de instrucciones y unidades de procesamiento. En SISD, obtenemos una instrucción del conjunto de instrucciones y una unidad de datos del conjunto de datos y enviamos la instrucción y la unidad de datos a la unidad de procesamiento para su ejecución, el resultado es guardado de vuelta en el almacenamiento de datos.(Bernal et al., 2020)

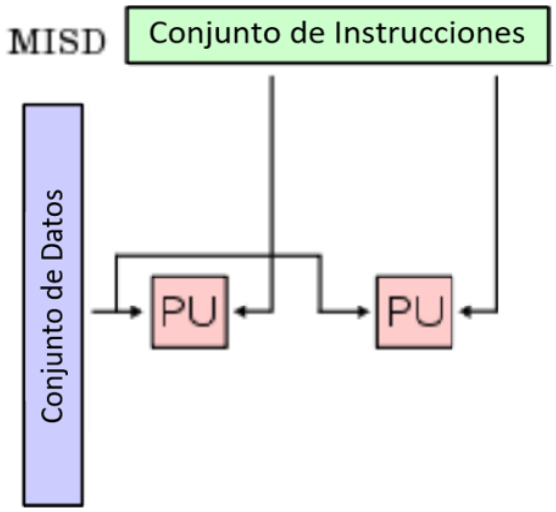
Figura 10: *Proceso SISD*
Fuente: (Bernal et al., 2020)



1.3.2 Multiple Instruction, Single Data (MISD)

Hay múltiples elementos de procesamiento, en el que cada cual tiene memoria privada del programa, pero se tiene acceso común a una memoria global de información. En cada paso, cada elemento de procesamiento accede a la misma información de la memoria y luego carga una instrucción de la memoria privada del programa, las instrucciones posiblemente diferentes de cada unidad son ejecutadas en paralelo, usando la información (idéntica) recibida anteriormente. Este modelo es muy restrictivo y no se ha usado en ningún computador de tipo comercial. (Bernal et al., 2020)

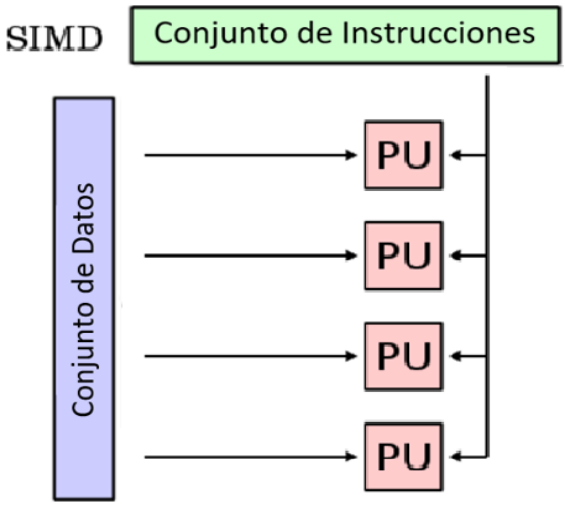
Figura 11: *Proceso MISD*
Fuente:(Bernal et al., 2020)



1.3.3 Single Instruction, Multiple Data (SIMD):

Hay múltiples elementos de procesamiento, en el que cada cual tiene acceso privado a la memoria de información (compartida o distribuida). No obstante, existe una única memoria de programa, desde la cual una unidad de procesamiento especial obtiene y envía las instrucciones. En cada paso, cada unidad de procesamiento recibe la misma instrucción y carga desde su memoria privada un elemento de información y extrae un elemento de información de su memoria privada para ejecutar esta instrucción en dicho elemento. Posteriormente, la instrucción es ejecutada simultánea y sincrónicamente en paralelo por todos los elementos de procesamiento sobre varios elementos de información. Para aplicaciones con un grado significante de paralelismo de información, este acercamiento puede ser muy eficiente. Ejemplos pueden ser aplicaciones multimedia y algoritmos de gráficos de computadora.(Bernal et al., 2020)

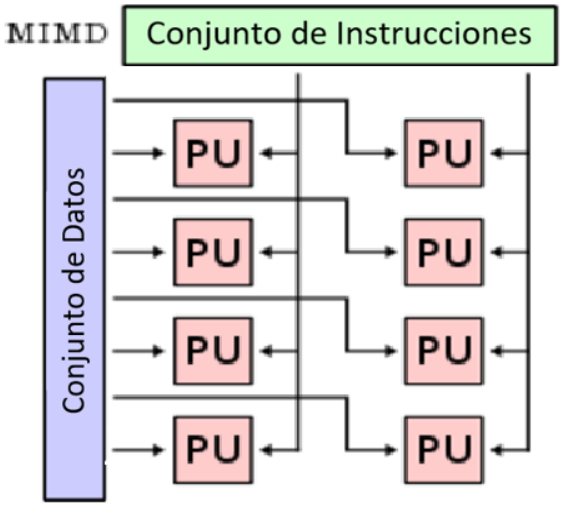
Figura 12: *Proceso SIMD*
Fuente: (Bernal et al., 2020)



1.3.4. Multiple Instruction, Multiple Data (MIMD):

Existen varias unidades de procesamiento, y cada una tiene sus propias instrucciones y datos por separado. Cada elemento ejecuta una instrucción diferente sobre un conjunto distinto de datos. Estos elementos de procesamiento funcionan de forma asíncrona e independiente.. Los clústers son ejemplo son ejemplos del modelo MIMD. (Bernal et al., 2020)

Figura 13: *Proceso MIMD*
Fuente: (Bernal et al., 2020)



1.4 Lenguajes de programación

El lenguaje de programación es el principal medio de comunicación y es el que permite que interactuemos entre humanos y máquinas, la programación paralela es: lenguajes, API, frameworks y otras herramientas que permiten aprovechar el potencial del hardware de forma paralela principalmente subdividiendo un gran problema en mínimas partes con el objetivo de resolver el problema general haciendo uso de lenguajes, API, librerías y capa o niveles donde interactúan, ya con software programados bajo el paradigma de la programación serial o enfocada para ejecución de un proceso a la vez en la línea de ejecución. Sin embargo, la paralelización automática de un programa secuencial sigue siendo un problema que resolver a pesar de los muchos esfuerzos que se han hecho en los últimos 40 años. (Rodríguez Erikson, 2019)

De forma general, los lenguajes de programación paralela pueden clasificarse en niveles en función de la arquitectura subyacente. En este caso, el nivel más alto de abstracción de la programación paralela mantiene toda la complejidad del paralelismo a nivel de hardware.

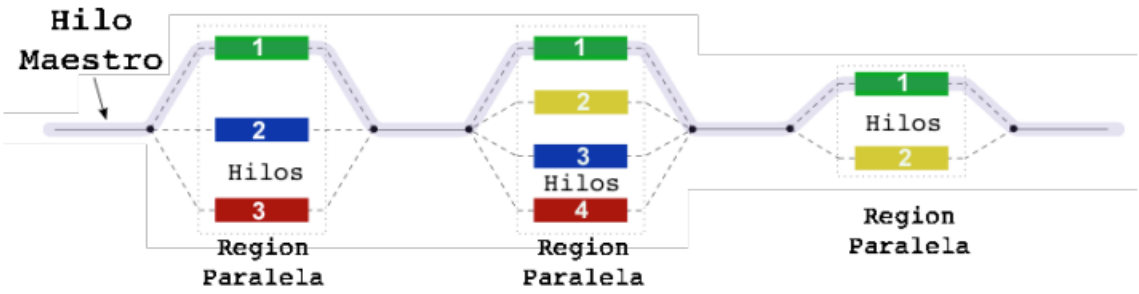
En este trabajo se expone algunos de los modelos de programación paralela, ellos se agrupan de acuerdo con: el método de comunicación entre los diferentes elementos de cómputo, los mecanismos de acceso a la memoria de datos del programa, la forma de gestionar la ejecución de las tareas y más recientemente de acuerdo con la heterogeneidad del hardware sobre el cual pueden ser implementados.(Gürsoy & Cengiz, 2001)

1.4.1 OpenMP (Open Multi-Processing)

OpenMp es una interfaz de programación de aplicaciones (API) para programación multiproceso de memoria compartida en múltiples plataformas que fue publicada por la Architecture Review Board en 1997. Su modelo de ejecución se basa en memoria compartida multihilos

Figura 14. (Barbara Chapman, 2007), demás podría definirse como un modelo de programación portable y escalable que proporciona a los programadores una interfaz simple y flexible para el desarrollo de aplicaciones paralelas para las plataformas que van desde las computadoras de escritorio hasta los supercomputadores, su especificación fue realizada para lenguajes C, C++ y Fortran, funciona en la mayoría de las arquitecturas de procesador y sistemas operativos, incluyendo Solaris, AIX, HP - UX, Linux, Mac OS X, y las plataformas de Windows. (Rodríguez Erikson, 2019)

Figura 14: *Lanzamiento de múltiples hilos en regiones paralelas*
Fuente:(Rodríguez Erikson, 2019)



OpenMP utiliza memoria compartida, lo que implica que todos los procesos pueden acceder al mismo conjunto de direcciones de memoria, una opción sería utilizar memoria distribuida, que se emplea sobre todo en clústeres, donde los datos deben ser transferidos de forma explícita entre distintos espacios de direcciones. Muchos programadores encuentran la memoria compartida más fácil de programar, dado que no tienen que preocuparse de mover los datos, pero es usualmente más difícil de implementar en hardware de una manera escalable, OpenMP proporciona la flexibilidad para que los hilos puedan compartir memoria sin que el programador lo haga. (Tom Christie, s.f.)

Además, OpenMP soporta la interacción con el modelo de paso por mensajes (MPM) permitiendo la integración de la librería MPI en las aplicaciones, lo que amplía aún más las alternativas de programación.

Un programa desarrollado en OpenMP debe manejar la siguiente estructura básica:

- Fichero principal: <omp.h>: hace el llamado de la librería.
- Directivas #pragma de compilación: son sentencias especiales que determinan la forma de trabajar de los procesadores asignados a una aplicación, se usa la siguiente sintaxis.
- #pragma omp nombre_directiva. [cláusulas, ...] \n
- Clausulas: son los parámetros que definen una condición en las funciones.
- Variables de entorno: son los posibles valores que cambian dinámicamente, afectando cualquier proceso definido en un ambiente de programación OpenMP.

Una particularidad de OpenMP es su habilidad para la gestión de recursos en rutinas que incluyen cálculos iterativos. Esto se consiguen distribuyendo las iteraciones de los ciclos entre los diferentes hilos controlados por la aplicación, utilizando determinadas funciones especiales conocidas como constructores. A estos constructores se añade una cláusula especial llamada "Schedule" que incluye parámetros que definen cómo se distribuirán las iteraciones. También se pueden emplear el anidamiento, que permite crear secciones paralelas adicionales dentro de los mismos hilos, generando así una estructura de ramificación que acelera y facilita el procesamiento de grandes cantidades de datos. Al igual que MPI, OpenMP también cuenta con rutinas especiales para atención y detección de errores, que son de gran ayuda para comprender el modo de trabajo de las diferentes directivas. (Acosta et al., 2012)

En el contexto del algoritmo K-means, OpenMP se puede utilizar para paralelizar la computación de los clústers y mejorar el rendimiento del algoritmo. Algunas partes del algoritmo K-means, como el cálculo de distancias y la asignación de puntos a los clústers, se pueden realizar de manera independiente para diferentes puntos o clústers, lo que permite la paralelización.

1.4.2. PYTHON

Python es un lenguaje de programación desarrollado en el año 1980 por Guido Van Rossum, se trata de un código que se basa en C y es un lenguaje de alto nivel que permite crear operaciones simples y complejas. Además, cuenta con multitud de librerías y extensiones que hacen que soporte otros códigos como Java, C, C++ o JSON. (Downey et al., 2002)

Python está presente en multitud de aplicaciones y sistemas operativos y podemos encontrarlo corriendo en servidores, en aplicaciones iOS, Android, Linux, Windows o Mac. Esto es debido a que cuenta con una curva de aprendizaje moderada ya que su filosofía hace hincapié en ofrecer una sintaxis de código legible.(Robledano, 2019)

Python es una multiplataforma versátil y multiparadigma que puede ser programado mediante POO (Programación Orientada a Objetos) y se destaca por su código legible y limpio. Una de las razones de su éxito es que cuenta con una licencia de código abierto que permite su utilización en cualquier escenario, esto hace que sea uno de los lenguajes de iniciación de muchos programadores, es ideal para trabajar con grandes volúmenes de datos ya que, el ser multiplataforma, favorece su extracción y procesamiento de datos, junto con su sencillez, ha producido que pudiera situarse en la escena del Big Data y en especial del desarrollo de algoritmos de aprendizaje automatizado.(Robledano, 2019)

El procesamiento paralelo es un modo de operación en el que la tarea se ejecuta simultáneamente en múltiples procesadores en la misma computadora. Está destinado a reducir el

tiempo total de procesamiento, in embargo, suele haber un poco de sobrecarga cuando se comunica entre procesos, lo que en realidad puede aumentar el tiempo total necesario para tareas pequeñas en lugar de disminuirlo. (Prabhakaran, 2018)

En Python, el módulo de multiproceso se utiliza para ejecutar procesos paralelos independientes mediante el uso de subprocesos permitiendo aprovechar múltiples procesadores en una máquina tanto Windows como Unix, lo que significa que los procesos se pueden ejecutar en ubicaciones de memoria completamente separada, a diferencia de otros códigos hace que el proceso de depuración de código sea mucho más sencillo y eficiente facilitando la ejecución y ahorro de tiempo en la misma a largo plazo.(Prabhakaran, 2018)

Python viene con una gran biblioteca de módulos que se pueden usar para hacer toda clase de tareas que abarcan desde programación para web a gráficos, seguida de la explosión de la Inteligencia Artificial, Machine Learning, Deep Learning y la consolidación de la ciencia de datos como una nueva área de trabajo con especialistas propios, ha revolucionado el panorama (Robledano, 2019).

Características del lenguaje de programación Python (RootStack, 2021):

- Python, puede resolver problemas completos en menor tiempo y con menos líneas de códigos, este lenguaje facilita muchos procesos con su fácil aplicación.
- Como lenguaje de alto nivel, permite prescindir de las complicaciones asociadas a tareas complejas, como la gestión de la memoria, que son imprescindibles al programar en C++.
- Python es multiplataforma, donde se puede crear y correr aplicaciones en Windows, Mac y Linux.

- Python cuenta con una comunidad inmensa de desarrolladores, dispuestos a ayudar en alguna tarea.
- Python tiene un gran ecosistema disponible con librerías, frameworks y herramientas.

1.5. Trabajos Relacionados

A continuación, veremos algunos proyectos relacionados que influyeron para llevar a cabo el desarrollo de este trabajo Se hablará brevemente de cada uno, dando a conocer los puntos más sobresalientes que han servido de apoyo y referencia para el presente trabajo de tesis.

En el estudio de (Wu & Wen, 2009) se investigó la máquina de vectores de soporte, como una herramienta de clasificación para identificar las malezas en los campos de maíz en la etapa temprana de crecimiento, haciendo uso de una SVM (Support Vector Machine) un clasificador de SVM para reconocer las malezas y plántulas de maíz, la segmentación de la imagen se realizó mediante transformación, las imágenes se utilizaron para obtener la textura de la maleza y el maíz; estás características de textura fueron utilizadas en el procedimiento de clasificación. Los resultados indicaron que los clasificadores SVM con diferentes selecciones de características podían identificar con éxito maleza-maíz con una mayor precisión que oscilaba entre 92,31 y 100 %. Se realizó un estudio de comparación de las capacidades de reconocimiento de SVM y el clasificador de redes neuronales de retropropagación (BP) utilizando el mismo conjunto de datos.

En (Kucukyilmaz, 2014) se llevó a cabo la implementación del algoritmo paralelo de k-means. Este algoritmo está optimizado de manera que minimiza el número total de bloqueos de

memoria realizados por los procesadores, con el algoritmo implementado, el paso de recálculo del centroide se divide en dos partes. En la primera parte, cada procesador encuentra errores cuadráticos medios parciales para todos los centroides en paralelo, eliminando la necesidad de bloqueos globales en la memoria, y en el segundo paso se acumulan los resultados parciales donde los procesadores se sincronizan proporcionando un análisis que, en términos de complejidad computacional, tanto el número de instancias, el número de grupos y el número de dimensiones en el conjunto de datos tienen casi la misma importancia. Los resultados muestran que se mantienen si el conjunto de datos es suficientemente grande.

En el trabajo de (Tang et al., 2017) construye un modelo de identificación de malezas basado en el aprendizaje de características de K-means combinado con una red neuronal convolucional este documento establece el aprendizaje de características no supervisado de k-means como proceso de preentrenamiento y reemplaza los pesos de inicialización aleatorios de los parámetros tradicionales, este método hace que los parámetros puedan obtener valores más razonables antes de la optimización para obtener una mayor precisión en la identificación de malas hierbas. El aprendizaje de características no supervisado basado en K-means se utiliza como proceso de preentrenamiento basado en las características de los parámetros que se pueden optimizar, y sobre esta base se construye un modelo de identificación de malezas para lograr este objetivo de identificación automática de malezas en el campo.

En la investigación de (García Mera & Zambrano Moreira, 2022) se desarrolló una implementación programación paralela para cultivos agrícolas para optimizar diferentes variables como la producción y beneficio, optimizando las técnicas de computación paralela, y usando herramientas como Google Colab, Python 3.8, la librería Torch, en la fase de codificación, logrando como resultados excelentes tiempos de ejecución, donde se destaca una mejoría del 200%, frente al algoritmo no paralelizado, obteniendo así mejores resultados en menor tiempo

y ayudando a resolver problemas de alta complejidad computacional que no se suelen abordar, ya que tarda semanas o incluso meses en obtener buenos resultados con las estrategias existentes en la actualidad.

En (Zhang et al., 2022), se investigó técnicas de identificación de malezas basadas en el aprendizaje profundo, este documento revisó sistemáticamente cinco tecnologías emergentes basadas en el aprendizaje profundo como la transferencia de aprendizaje y búsqueda de arquitectura neuronal para la identificación temprana de malezas, A partir del desarrollo de tecnologías emergentes, pudieron reconocer que las tecnologías emergentes han logrado algunos logros, la investigación y la práctica en el campo de la identificación de malezas aún se encuentra en una etapa inicial.

En el trabajo de (Hnin Hlaing & Soe Khaing, 2014) desarrollaron un método de segmentación y la clasificación basada en el método de umbralización de áreas entre cultivos para distinguir las malas hierbas con una tasa de error del 33.3 %. Parra llevar a cabo este proceso se calcula valores absolutos de rojo y azul en cada píxel de la imagen. Si los valores exceden un límite determinado, el píxel se etiqueta como planta; en caso contrario, se etiqueta como mala hierba. Los píxeles de las plantas se evalúan con una función de decisión proporcionada por un clasificador para discriminar entre cultivos y malezas.

En la investigación realizada por (Gómez et al., 2020) existen varios métodos de extracción de características y técnicas de clasificación de imágenes usando distintos tipos de algoritmos. Hoy, la visión artificial es un punto clave en la tecnología dado que ayuda a identificar con precisión las plantas de maíz y diferenciar de las malezas para poder aplicar una fertilización apropiada.

En (Pérez Ortega et al., 2018) se propuso equilibrar el tiempo de procesamiento y la calidad de la solución de los algoritmos de clústers de k-means cuando se aplican a instancias

donde el número n de objetos es grande aprovechando el problema de la complejidad computacional de K-Means sin que esto implique una pérdida considerable de su calidad. Como resultado se propusieron nuevas metaheurísticas, las cuales, por una asignación temprana de objetos a grupos, redujeron significativamente el número de cálculos de distancia de objetos a centroides.

En (Olszewski, 2016) propone la mejora del algoritmo K-means donde consiste el uso de un número fijo de objetos en las agrupaciones, a la hora de computar las similitudes asimétricas en cada ciclo del proceso de agrupaciones, así, el algoritmo puede manejarse de forma eficaz , aunque los datos estén contenidos en una agrupación de diferente número de objetos, para esto, se introducen los coeficientes de grupo los cuales transmiten la información sobre el número actual de objetos en las agrupaciones.

CAPITULO 2

DESARROLLO

2.1 Análisis del algoritmo Kmeans

La definición simple de agrupación en clústeres de k-means, es clasificar los datos en grupos de objetos en función de los atributos/características en un número K de grupos. K es un número entero positivo. K-means es una técnica de agrupamiento basada en prototipos (basada en el centro) que es uno de los algoritmos que resuelven problemas de agrupamiento. (Karoussi & Bouhmala, 2012)

Como ventajas usando K-means, hay algunas ideas que se encuentran en el trabajo de (MacQueen, 1966):

"*El proceso, llamado "k-means", parece dar particiones que son razonablemente eficientes en el sentido de variación dentro de la clase, corroborado hasta cierto punto por el análisis matemático y la experiencia práctica. Además, el procedimiento k-means se programa fácilmente y es computacionalmente económico, por lo que es factible procesar muestras muy grandes en un computador digital".*

Y la otra es la misma idea que resume en la parte introductoria de su trabajo las ventajas de utilizar K-means(Barbakh & Fyfe, 2008):

"El algoritmo K-means es uno de los primeros que un analista de datos utilizará para investigar un nuevo conjunto de datos porque es algorítmicamente simple, relativamente robusto y brinda respuestas "suficientemente buenas" en una amplia variedad de conjuntos de datos".

2.1.1 El algoritmo básico de K-means

La técnica de agrupación en clústeres de k-means es uno de los algoritmos más simples (Karoussi & Bouhmala, 2012):

Se determina un punto de datos, D= (X1… Xn), primero se elige entre los puntos de datos, K centroide inicial, donde k es el parámetro de usuario, el número de grupos deseado. Luego, cada punto se asigna al centroide más cercano.

La idea consiste en elegir centros de clústers aleatorios, uno para cada clúster. Luego, se actualiza el centroide de cada clúster en función de la media de cada grupo que se asigna como nuevo centroide.

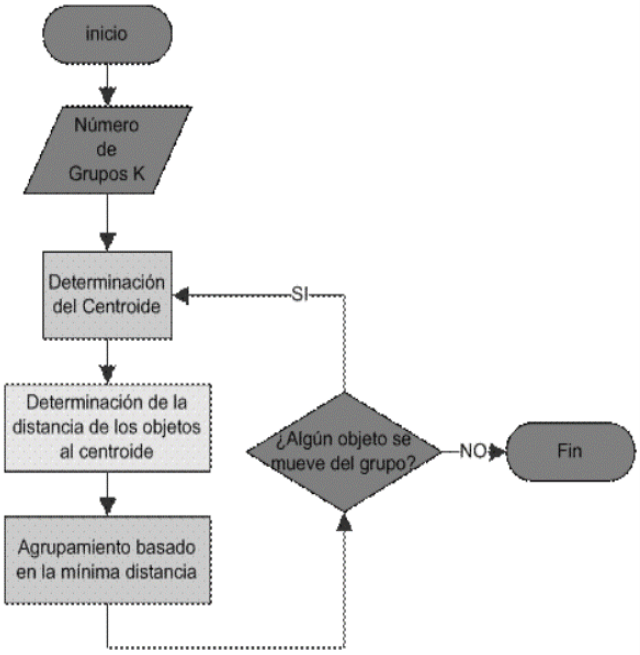
Repetimos la asignación y el centroide actualizado hasta que ningún punto cambie, lo que significa que ningún punto no navega de un grupo a otro o, de manera equivalente, cada centroide sigua siendo el mismo.

La figura 15 muestra los pasos para realizar la agrupación mediante el algoritmo k-means son los siguientes:

1. Elegir un valor de k, número de clúster a formar.
2. Seleccione aleatoriamente k puntos de datos del conjunto de datos como los centroides/centros del grupo inicial
3. Para cada punto de datos, calcule la distancia entre el punto de datos y el centroide del grupo y asigne el punto de datos al centroide más cercano.
4. Para cada clúster, calcule la nueva media en función de los puntos de datos en el clúster.

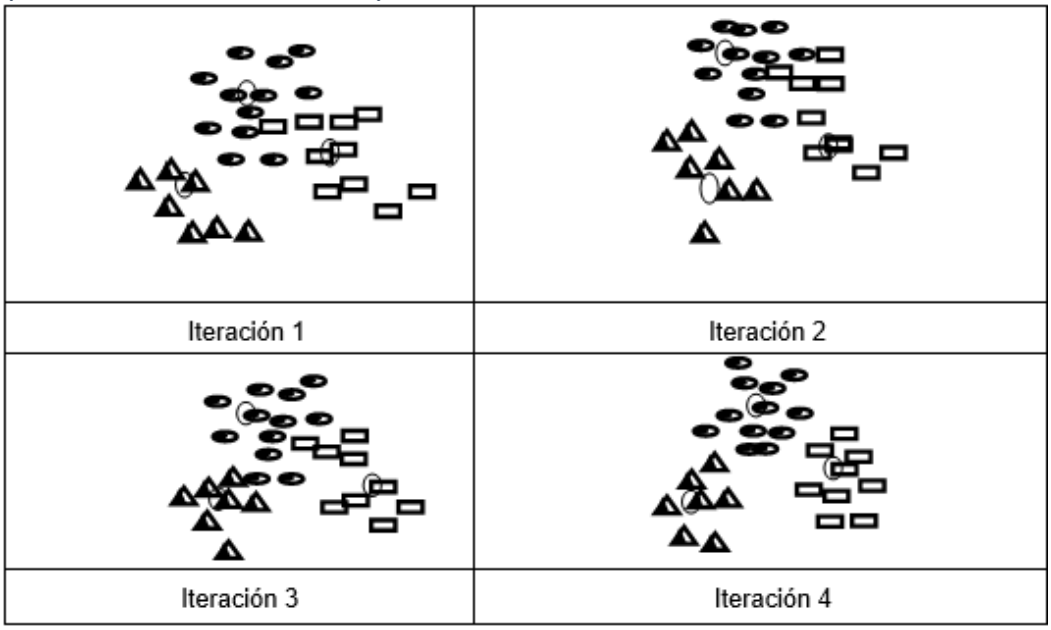
5. Repetir los pasos 3 y 4 hasta que la media de los grupos deje de cambiar o se alcance el número máximo de iteraciones.

Figura 15: *Flujograma del algoritmo K-means*
Fuente:*(Karoussi & Bouhmala, 2012)*



La figura 16, muestra el funcionamiento de K-means, que ilustra cómo a partir de 3 centroides los clústers finales se encuentran en cuatro iteraciones.

Figura 16: *Uso del algoritmo K-means con tres clústers en datos de muestra*
Fuente:*(Karoussi & Bouhmala, 2012)*



En la iteración 1 se muestra el centroide y la asignación de puntos a esos centroides y en la segunda iteración, se asignan los puntos a los centroides actualizados. En las iteraciones 2, 3, 4 que se muestran en la figura 16, los centroides se mueven a los pequeños grupos de puntos en la parte inferior de la figura. En la iteración 4, el algoritmo de K-means termina porque no se producen más cambios.

Cada paso del algoritmo K-means se considera con más detalle y, a continuación, proporcionamos un análisis de la complejidad espacial y temporal del algoritmo(Amoroso Ordóñez & Ávila Flores, 2015) :

- **Asignación de puntos al centroide más cercano:** Para asignar un punto al centroide más cercano, se requiere una medida de proximidad que cuantifique el punto más cercano para los datos específicos que se están considerando. La distancia euclidiana (L2) se usa a menudo para puntos de datos. Sin embargo, puede haber varios tipos de medidas de proximidad que sean apropiadas para

un dato dado. Como ejemplo, la distancia Manhattan (L1) se puede usar para datos euclidianos, mientras que la medida Jaccard se usa a menudo para documentos.

A veces, el cálculo de la medida de similitud de cada punto lleva mucho tiempo; en el espacio euclidiano es posible evitar acelerar el algoritmo K-means.

- **Centroide y función objetivo:** El paso 4 del algoritmo es "Recalcular el centroide de cada grupo", ya que el centroide puede ser variable según el objetivo del agrupamiento. Por ejemplo, para medir la distancia, minimizada la distancia al cuadrado de cada punto al centroide más cercano, es el objetivo de la agrupación que depende de la proximidad del punto a otro, que se expresa mediante una función objetivo.

- **Datos en el espacio euclidiano:** Considere que la medida de proximidad es la distancia euclidiana. Para nuestra función objetivo usamos la suma del error cuadrático (SSE), que se conoce como dispersión. Para más, calculamos el error de cada punto de datos.

  La fórmula del error cuadrático SSE se define formalmente de la siguiente manera:

$$SSE = \sum_{i=1}^{K} \sum_{x \in c_i} dist(C_i, x)^2$$

Ecuación 3. *Ecuación para el error cuadrático SSE*

Donde **dist.** es la distancia estándar euclidiana (L2) entre dos objetos en el espacio euclidiano.

La Tabla 1 describe cada símbolo de la fórmula:

Tabla 1: *Tabla de notación*

Símbolos	Descripción
X	Un objeto
C_i	el grupo i
c_i	El centroide del grupo i
C	El centroide de todos los puntos.
m_i	El número de objetos en i clúster
metro	El número de objetos en el conjunto de datos.
k	El número de clúster

Según la notación de la tabla 1, el centroide (media) del clúster i está definido por la Ecuación 4.

$$C_i = \frac{1}{m_i} \sum_{x \in C_i} x$$

Ecuación 4. *Ecuación para el centroide*

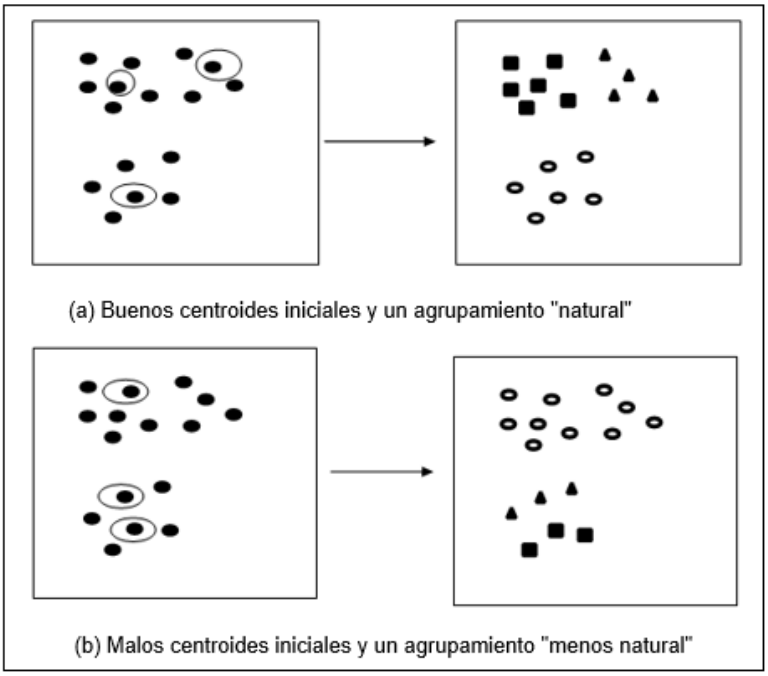
Los pasos 3 y 4 del algoritmo intentan directamente minimizar el SSE. El paso 3 se forma un grupo asignando los puntos al centroide más cercano que minimiza el SSE para el centroide dado y el paso 4 vuelve a calcular el centroide para minimizar aún más el SSE.

2.2.2 Elección del Centroide inicial

Cuando se usa la inicialización aleatoria del centroide, diferentes ejecuciones de K-means tienen diferentes resultados en la suma total del error cuadrático. Un enfoque común es elegir el centroide inicial al azar, pero la elección puede ser deficiente y causa resultados deficientes (con un error cuadrático más alto).

La figura 18 ilustra un ejemplo muy sencillo de tres clústers y 16 puntos, donde la parte (a) indica que la agrupación "natural" debida a los centroides iniciales está bien distribuida, por el contrario, la parte (b) es el resultado de seleccionar mal el centroide inicial, que es "menos natural".

Figura 17: *Centroide inicial*
Fuente: *(Karoussi & Bouhmala, 2012)*



Otra técnica que se usa comúnmente para abordar el problema de elegir el centroide inicial es realizar múltiples ejecuciones, cada una con centroides iniciales elegidos al azar, y luego seleccionar el conjunto de grupos con SSE mínimo.

Como se menciona anteriormente, un clúster de un conjunto de datos es óptimo siempre que cada punto esté más cerca del centroide de su propio clúster que del centroide de otro clúster disponible. La proximidad se mide en términos de distancia euclidiana dentro del espacio de características del conjunto de datos.

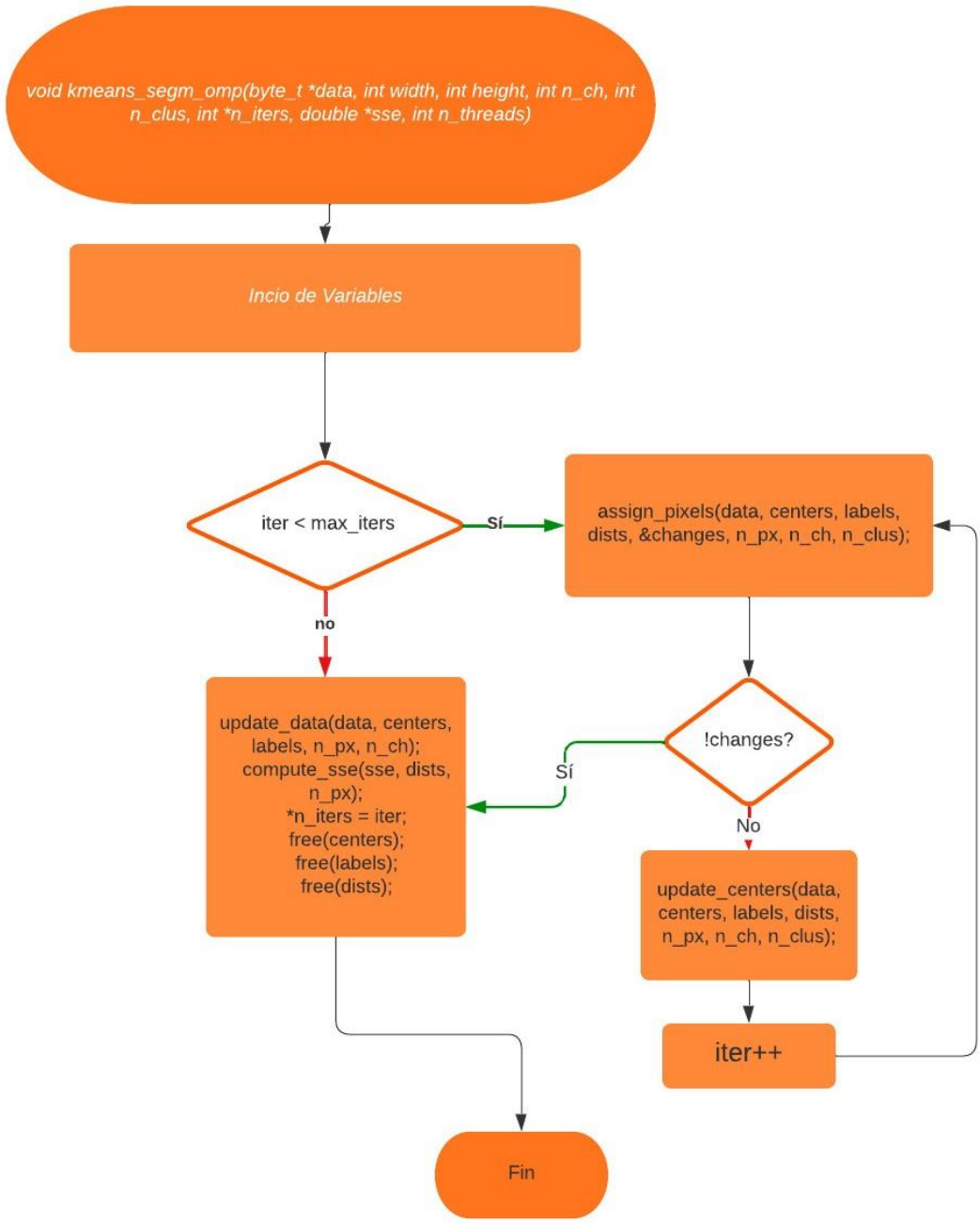
2.2 Diseño del Algoritmo Kmeans

En este apartado se presenta el diagrama de flujo del algoritmo paralelo basado en Kmeans para el procesamiento de imágenes agrícolas como parte de la detección de densidad de maleza utilizando OpenMP para la paralelización e implementado el lenguaje de programación C (GCC) en el sistema operativo Ubuntu.

En la búsqueda de una mayor eficiencia de procesamiento, la utilización de la robusta tecnología de paralelización OpenMP ocupa un lugar destacado. Esta herramienta permite distribuir las tareas entre múltiples hilos de ejecución, aprovechando todo el potencial de los recursos informáticos disponibles. Como resultado, consigue una notable aceleración en los complejos cálculos que sustentan la funcionalidad del algoritmo.

Función principal (void kmeans_segm_omp): Toma como entrada una imagen (ancho y alto), número de canales de color(n_ch), número de clústeres(n_clus), número de iteraciones (n_iters), número de hilos (n_threads) y devuelve el número de iteraciones que se realizaron y la suma de errores cuadráticos (SSE) entre los píxeles y sus centros de clúster asignados. Se muestra el algoritmo en la figura 18.

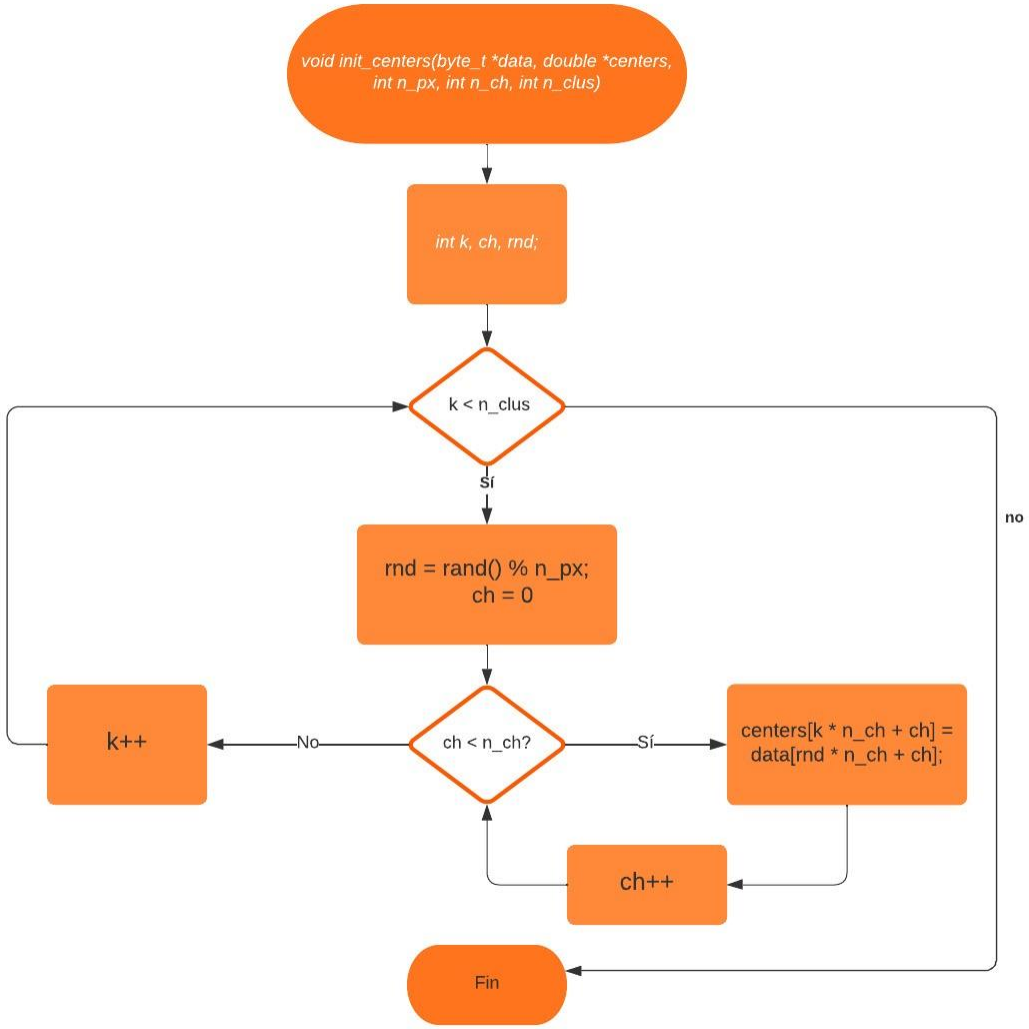
Figura 18: *Función segmentación de imagen*



Función void init_centers: Inicializa los centroides de los clústeres para el conjunto de los datos dados. Para cada clúster, selecciona de forma aleatoria un píxel de la imagen para cada centro y copia los valores de los canales de color que corresponde al centro del clúster.

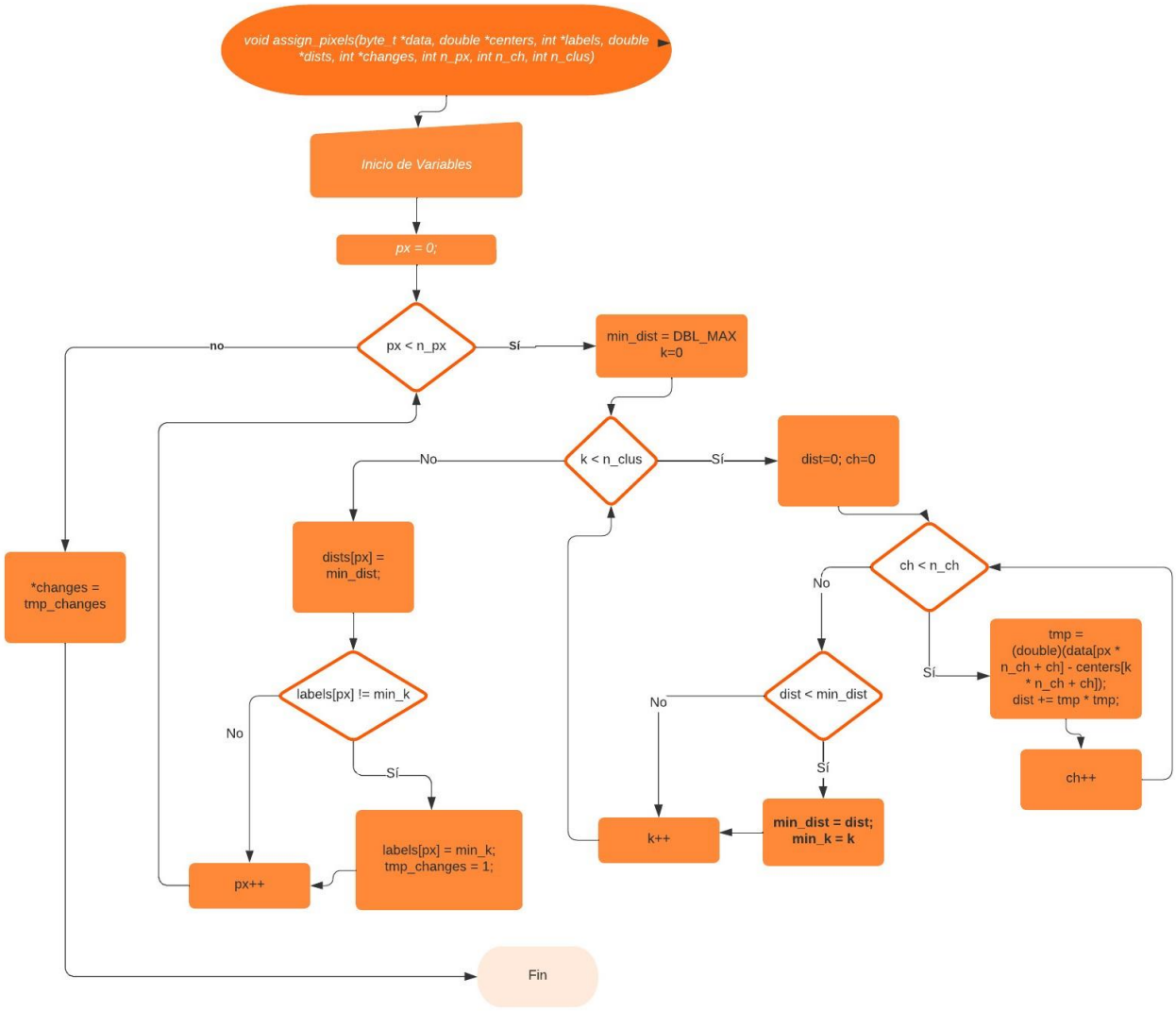
Esto se repite para todos los clústeres hasta que se hayan inicializado todos los centros. Se muestra el algoritmo en la figura 19.

Figura 19: *Función inicialización de centroides*



Función void assign_pixels: Asigna cada píxel a un grupo cuyo centro está más cerca de él calculando la distancia Euclidiana entre el píxel y los centros de los clústeres, si la etiqueta del píxel cambia durante el proceso la función establece el indicador en 1 para mostrar que se han realizado cambios. Se muestra el algoritmo en la figura 20.

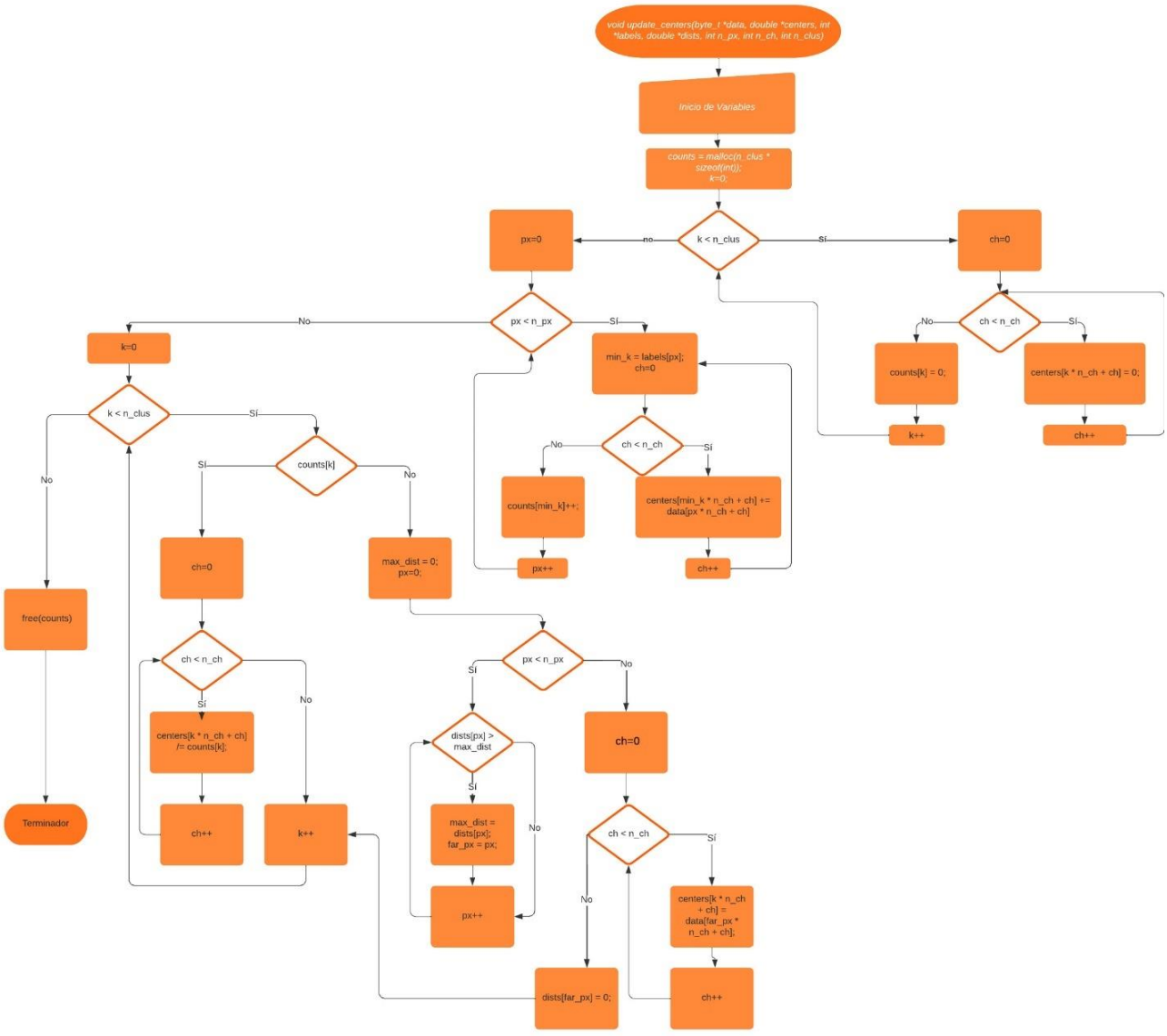
Figura 20: *Función asignación de pixeles*



Función void update_centers: inicializa los contadores de pixeles asignados y restablece los centros, luego calcula las sumas parciales de los centros y actualiza los contadores de pixeles seguidamente divide las sumas por el número de píxeles asignados a cada cluster para obtener los valores de los nuevos centros.

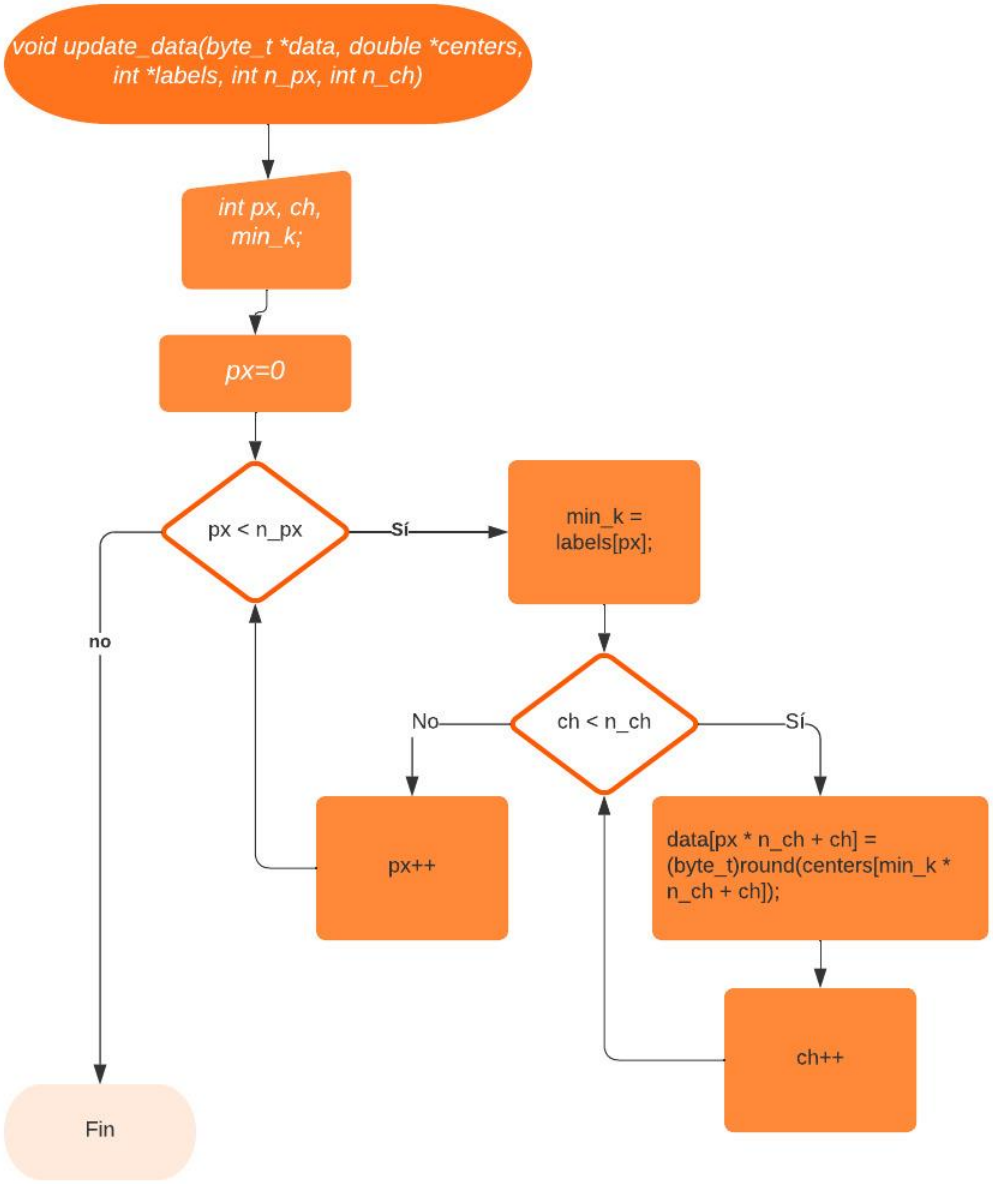
Si un cluster está vacío, se localiza el píxel más alejado al centro del cluster y se establece los valores de ese píxel como el nuevo centro. Finalmente, la función libera la memoria asignada para los contadores de clúster. Se muestra el algoritmo en la figura 21.

Figura 21: *Función actualizar centroides*



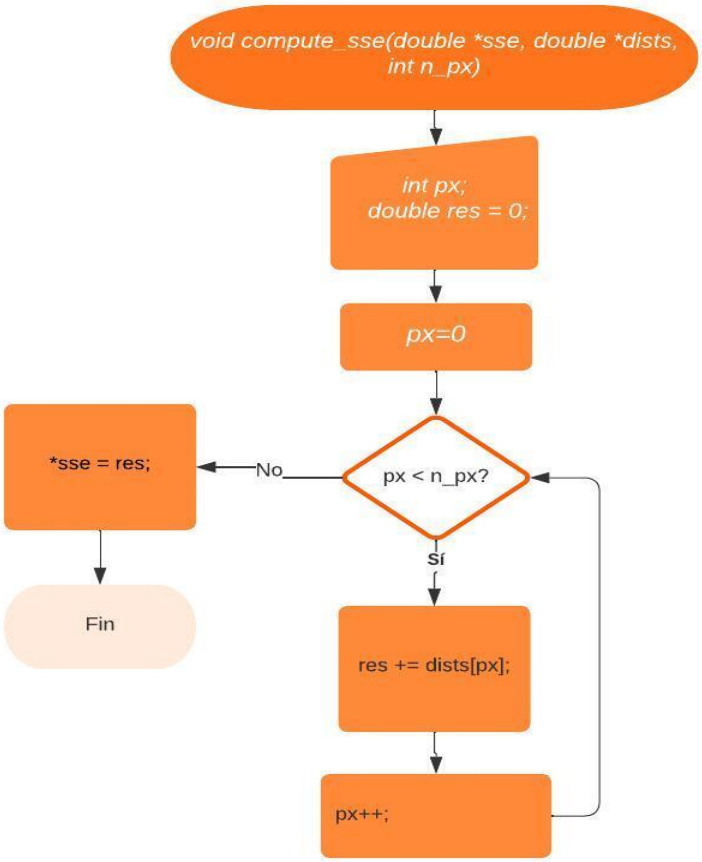
Función void update_data: Actualiza y asigna los valores a los pixeles que se toma del centro del cluster correspondiente, finalmente el valor se redondea al entero mas cercano con respecto a la distancia euclidiana. Se muestra el algoritmo en la figura 22.

Figura 22: *Función actualizar*



Función void compute_sse: Calcula la suma de error cuadrático (SSE) como la suma de las distancias al cuadrado entre cada píxel y el centro de clúster. Al finalizar el resultado se guarda en el puntero asignado como: ”see”. Se muestra el algoritmo en la figura 23.

Figura 23: *Función cálculo error cuadrático*



2.3 Implementación del Algoritmo Kmeans

Para el desarrollo del algoritmo se implementaron las siguientes herramientas de programación:

Eclipse for Parallel Application Developers:

Es una distribución de Eclipse que se centra en el desarrollo de aplicaciones paralelas. Proporciona características y herramientas específicas para facilitar el desarrollo, depuración y optimización de aplicaciones paralelas en diferentes paradigmas de programación.

Incluye herramientas para C, C ++, Fortran y UPC, incluidos MPI, OpenMP, OpenACC, un depurador paralelo y aplicaciones de creación, ejecución y supervisión de forma remota, (Eclipse Foundation, 2013), como:

- Herramientas de desarrollo C/C++
- Eclipse EGit
- Lista de tareas Mylyn
- Plataforma de herramientas paralelas (PTP) Tiempo de ejecución del usuario final
- Explorador de sistemas remotos
- Herramientas y editores XML Eclipse

GCC

GCC (GNU Compiler Collection) es un conjunto de compiladores de código abierto desarrollado por el proyecto GNU. Es uno de los compiladores más populares y ampliamente utilizados en el desarrollo de software. GCC es compatible con varios lenguajes de programación, incluyendo C, C++, Objective-C, Fortran, Ada y otros.

El compilador GCC convierte el código fuente escrito en un lenguaje de programación en código ejecutable para una plataforma o arquitectura específica. Puede generar código máquina para una amplia gama de sistemas operativos y procesadores. Además, GCC proporciona herramientas y utilidades relacionadas, como el enlazador y el depurador, que facilitan el proceso de desarrollo de software.

GCC está disponible en múltiples plataformas, incluyendo Linux, macOS y Windows. Se usa en proyectos de software libre y de código abierto disponibles fácil y gratuitamente a través de Git. Además, GCC ha influido en el desarrollo de otros compiladores y ha establecido estándares para la implementación de lenguajes de programación.(Moraguez Erick Roch, n.d.)

En general, GCC es conocido por su estabilidad, rendimiento y soporte de lenguajes, lo que lo convierte en una opción popular para desarrollar software en plataformas y entornos.

MakeFile

Un archivo MakeFile utiliza el comando Make para la creación de *n* archivos ejecutables o programas, su instalación y la limpieza de archivos temporales basados en C, C++. Al existir muchos ficheros fuente (muchos .c y muchos .h) repartidos por varios directorios, este identifica qué archivos hay que recompilar y si hay cambios en algún archivo fuente, sólo se recompilarán aquellos archivos que dependan del modificado, además guarda los comandos de compilación con todos sus parámetros para encontrar librerías, archivos de cabecera (.h), y archivos dependientes, no es necesario escribir largas líneas de compilación con varias opciones, sólo se realiza una vez o las veces que haya cambios en el código fuente, facilitando el trabajo en proyectos grandes(Chuidiang, 2007).

El formato del archivo MakeFile sigue una estructura básica. Cada regla se define mediante un objetivo (target) y una serie de dependencias y comandos asociados. Algunas de las secciones y directivas más comunes en un archivo Makefile son (Lambert Chase, 2021):

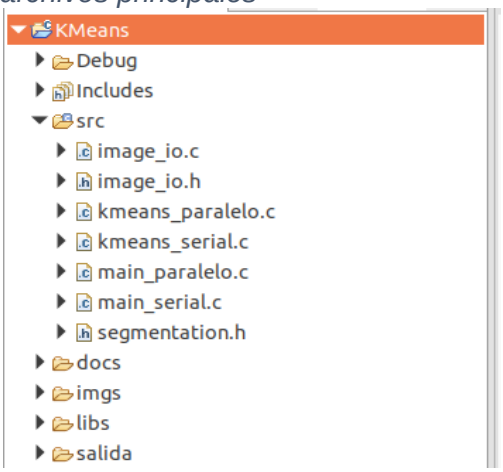
- **Directiva "target":** Especifica el objetivo principal del Makefile. Por lo general, es el nombre del archivo ejecutable o biblioteca que se desea ejecutar.

- **Directiva "dependencies":** Lista las dependencias del objetivo principal. Estas dependencias pueden ser archivos fuente, archivos de encabezado u otros objetivos. El objetivo se volverá a compilar si alguna de sus dependencias ha cambiado.
- **Reglas de compilación:** Definen cómo se compila cada archivo fuente. Se especifica la regla para generar el archivo objeto (. o) a partir del archivo fuente (.c, .cpp, etc.). Esto incluye las opciones de compilación y los comandos de compilación que se deben ejecutar.
- **Regla de enlace:** Define cómo se enlazan los archivos objeto para generar el archivo ejecutable o biblioteca final. Esto incluye las opciones de enlazado y los comandos de enlace necesarios.
- **Reglas adicionales:** Pueden incluirse otras reglas para tareas como limpiar (clean) el proyecto donde elimina los archivos objeto y el archivo ejecutable., generar documentación, ejecutar pruebas, entre otras.

2.3.1 Herramientas

El proyecto está compuesto por 7 archivos principales, como se presenta en la Figura 19, los archivos se encuentran en la carpeta src, dos archivos de cabecera (image.h y segementacion.h) y cinco archivos fuente (image_io.c, kmeas_paralelo.c, kmeas_serial.c, main_paralelo.c, main_serial.c)

Figura 24: *Ubicación de los archivos principales*



El archivo image_io.c contiene los métodos para leer una imagen de entrada para el procesamiento y un método para guardar la imagen procesada mediante los algoritmos.

El archivo main_paralelo.c contiene los métodos para obtener le tiempo de ejecución en segundos, un método para impresión de resultados en pantalla y el método principal que contiene sentencias y método para el algoritmo Kmeans

El archivo main_serial.c contiene los métodos para obtener el tiempo de ejecución en segundos, un método para impresión de resultados en pantalla y el método principal que contiene sentencias y método para el algoritmo serial.

El archivo segmentation.c contiene los métodos necesarios para el proceso de imágenes mediante la segmentación de manera paralela.

La carpeta imgs contiene las 10 imágenes de entrada a ser evaluadas por los algoritmos implementados, las imágenes tienen una sola medida (grande) para el respectivo procesamiento.

La carpeta Includes contiene librerías del lenguaje de programación c, c++ y OpenMP que se utilizan el desarrollo de los algoritmos.

La carpeta libs contiene librerías para el procesamiento de imágenes digitales, incluye estructuras de datos compatibles para leer y escribir imágenes en formatos más utilizados jpg, bmp, png.

La carpeta salida contiene las imágenes resultado del procesamiento de las imágenes mediante los algoritmos implementados.

En la Figura 25, se presenta el contenido del archivo Makefile que contiene las tareas necesarias para compilar los algoritmos serial y paralelo:

Figura 25: *Contenido del archivo Makefile*

Makefile

```
CC = gcc
CC_FLAGS = -Wall
CC_OMP = -fopenmp

all: serial.out paralelo.out

clean:
    rm serial.out omp.out salida.jpg

serial.out: src/main_serial.c src/image_io.h src/image_io.c src/segmentation.h src/kmeans_serial.c
    $(CC) $(CC_FLAGS) -o serial.out src/main_serial.c src/image_io.c src/kmeans_serial.c -lm

paralelo.out: src/main_paralelo.c src/image_io.h src/image_io.c src/segmentation.h src/kmeans_paralelo.c
    $(CC) $(CC_FLAGS) $(CC_OMP) -o paralelo.out src/main_paralelo.c src/image_io.c src/kmeans_paralelo.c -lm
```

Implementación del algoritmo

La paralelización es la tarea de modificar el código fuente de un programa de tal manera que muchos cálculos puedan realizarse simultáneamente por diferentes núcleos de un mismo procesador. El primer paso hacia una paralelización efectiva, que garantice una aceleración significativa, fue evaluar cuáles son las secciones del algoritmo donde el programa hace la mayor parte del trabajo y comprender si en esas secciones los cálculos se pueden realizar simultáneamente por múltiples hilos.

Equipo de procesamiento

Las pruebas del algoritmo serial y paralelo, se realizaron en una Laptop Lenovo Ideapad ES15MITG con Linux Ubuntu 18. de 64 bits, procesador Intel Core I7, 8GB de memoria RAM.

Las características técnicas del computador y del equipo utilizado para el procesamiento serial y paralelo se presentan en la Tabla 2:

Tabla 2: *Especificaciones técnicas de Laptop con Ubuntu Linux*

Detalle	Información
Sistema operativo	Ubuntu 18.04 bionic
Núcleo de Linux	.4.0-87-generic
Memoria RAM	8 Gb
Arquitectura	X86_64B
Procesador	Intel(R) Core (TM) i7-10750H CPU @ 2.60GHz 2.59 GHz
Núcleos	6
Procesadores lógicos	12

2.4 Ejecución del Algoritmo Kmeans

Para la ejecución del programa se utiliza la terminal de Linux para ejecutar los programas compilados a partir de los ejecutables generados por el comando Make. El comando Make compila el contenido del archivo MakeFile para depurar el código fuente y construir los archivos *.out*.

El algoritmo realizado en OpenMP basado en lenguaje C++, extrae características relevantes de las imágenes, como el color, la textura o la forma de las malezas y el fondo, luego decide el número de clústeres (K) que desea utilizar para la segmentación. Cada clúster representará una clase: maleza o fondo. Una vez segmentada la imagen cada píxel se asigna al color o etiqueta correspondiente al clúster finalmente detecta las regiones de malezas en fun-

ción de las etiquetas o colores asignados a los clústers. En la Figura 26, se presenta las compilaciones de los algoritmos serial y paralelo para procesar imágenes obtenidas con Dron.

Figura 26: *Compilación de los Algoritmos Serial y Paralelo en Imágenes con Dron*



a) Imagen de entrada

```
ruth@RuthPc:~/eclipse-workspace/KMeans$ ./serial.out -k 10 -m 4 -o
salida/SerPic20G-10K.jpg imgs/im20g.jpg

    DETALLES EJECUCIÓN

Tamaño imágen          : 1920 x 1280
Canales imagen         : 3
Número de clústers     : 10
Número of iteraciones  : 4
Errores de suma cuadrados  : 491437666.813729
Tiempo ejecución       : 2.335813
```

b) Algoritmo serial

```
ruth@RuthPc:~/eclipse-workspace/KMeans$ ./paralelo.out -k 10 -m 4 -t 2 -o
salida/Par20G-10K-4-2.jpg imgs/im20g.jpg

    DETALLES EJECUCIÓN PARALELO

Tamaño imagen          : 1920 x 1280
Canales imagen         : 3
Número de clústers     : 10
Número of núcleos      : 2
Número of iteraciones  : 4
Errores de suma cuadrados  : 449170196.629559
Tiempo ejecución       : 1.057294
```

c) Algoritmo paralelo con 2 núcleos

```
ruth@RuthPc:~/eclipse-workspace/KMeans$ ./paralelo.out -k 10 -m 4 -t 4 -o
salida/Par20G-10K-4-4.jpg imgs/im20g.jpg

    DETALLES EJECUCIÓN PARALELO

Tamaño imagen          : 1920 x 1280
Canales imagen         : 3
Número de clústers     : 10
Número of núcleos      : 4
Número of iteraciones  : 4
Errores de suma cuadrados  : 477233756.886300
Tiempo ejecución       : 0.944529
```

d) Algoritmo paralelo con 4 núcleos

```
ruth@RuthPc:~/eclipse-workspace/KMeans$ ./paralelo.out -k 10 -m 4 -t 6 -o
salida/Par20G-10K-4-6.jpg imgs/im20g.jpg

    DETALLES EJECUCIÓN PARALELO

Tamaño imagen          : 1920 x 1280
Canales imagen         : 3
Número de clústers     : 10
Número of núcleos      : 6
Número of iteraciones  : 4
Errores de suma cuadrados  : 352262335.162361
Tiempo ejecución       : 0.603763
```

e) Algoritmo paralelo con 6 núcleos

```
ruth@RuthPc:~/eclipse-workspace/KMeans$ ./paralelo.out -k 10 -m 4 -t 8 -o
salida/Par20G-10K-4-8.jpg imgs/im20g.jpg

    DETALLES EJECUCIÓN PARALELO

Tamaño imagen          : 1920 x 1280
Canales imagen         : 3
Número de clústers     : 10
Número of núcleos      : 8
Número of iteraciones  : 4
Errores de suma cuadrados  : 344820946.811040
Tiempo ejecución       : 0.614287
```

f) Algoritmo paralelo con 8 núcleos

```
ruth@RuthPc:~/eclipse-workspace/KMeans$ ./paralelo.out -k 10 -m 4 -t 10 -
o salida/Par20G-10K-4-10.jpg imgs/im20g.jpg

    DETALLES EJECUCIÓN PARALELO

Tamaño imagen          : 1920 x 1280
Canales imagen         : 3
Número de clústers     : 10
Número of núcleos      : 10
Número of iteraciones  : 4
Errores de suma cuadrados  : 337643890.536574
Tiempo ejecución       : 0.560749
```

g) Algoritmo paralelo con 10 núcleos

```
ruth@RuthPc:~/eclipse-workspace/KMeans$ ./paralelo.out -k 10 -m 4 -t 12 -
o salida/Par20G-10K-4-12.jpg imgs/im20g.jpg

    DETALLES EJECUCIÓN PARALELO

Tamaño imagen          : 1920 x 1280
Canales imagen         : 3
Número de clústers     : 10
Número of núcleos      : 12
Número of iteraciones  : 4
Errores de suma cuadrados  : 472143948.891679
Tiempo ejecución       : 0.566101

ruth@RuthPc:~/eclipse-workspace/KMeans$
```

h) Algoritmo paralelo con 12 núcleos

CAPITULO 3

RESULTADOS

3.1 EVALUACIÓN DE TIEMPOS DE EJECUCIÓN

En esta sección se documenta los resultados obtenidos en la evaluación del algoritmo serial y paralelo (2,4,6,8,10 y 12 núcleos)

3.1.1 Tiempos de ejecución algoritmo Kmeans con 4 clústers

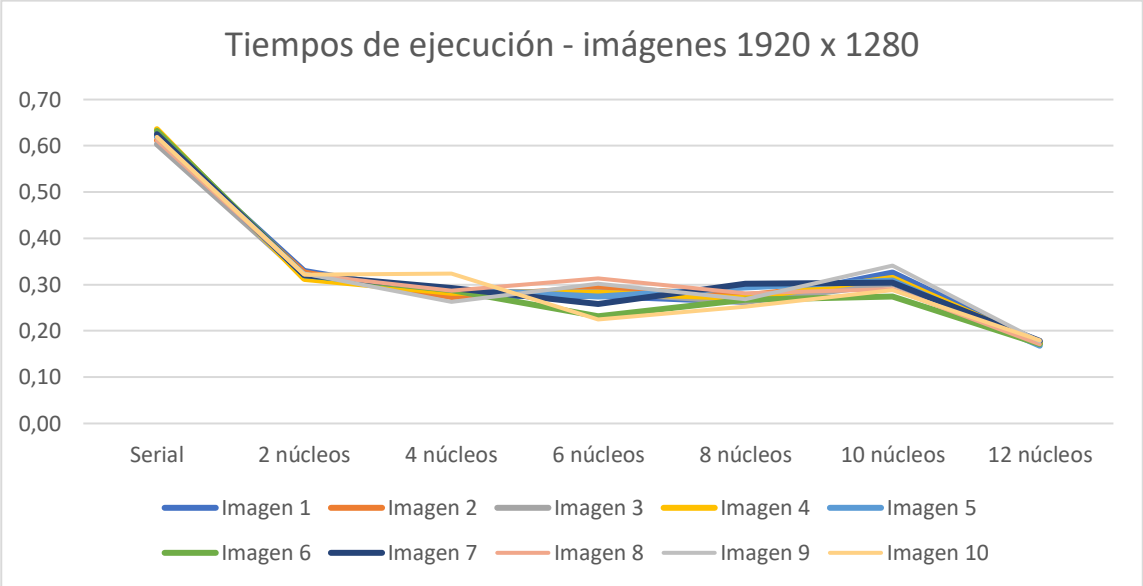
En la tabla 3 se muestra los tiempos de ejecución obtenidos por el algoritmo serial y paralelo para el procesamiento de las imágenes por dron con dimensiones (1920x1280 pixeles).

Tabla 3: *Tiempos ejecución imágenes grandes con 4 clústers obtenidas con Dron*

Tiempos de ejecución imágenes Grandes – 1920 x 1280							
Imagen	Serial	2 núcleos	4 núcleos	6 núcleos	8 núcleos	10 núcleos	12 núcleos
1	0,62	0,33	0,28	0,28	0,26	0,33	0,17
2	0,61	0,33	0,27	0,29	0,28	0,31	0,17
3	0,60	0,32	0,28	0,28	0,26	0,30	0,18
4	0,64	0,31	0,28	0,28	0,27	0,31	0,17
5	0,62	0,32	0,29	0,27	0,29	0,31	0,17
6	0,63	0,32	0,29	0,23	0,27	0,27	0,17
7	0,63	0,32	0,29	0,26	0,30	0,30	0,18
8	0,61	0,32	0,29	0,31	0,28	0,29	0,17
9	0,62	0,32	0,26	0,30	0,27	0,34	0,18
10	0,62	0,32	0,32	0,23	0,25	0,29	0,18
Promedio	**0,62**	**0,32**	**0,29**	**0,27**	**0,27**	**0,31**	**0,17**

En la Figura 27 se representa de manera gráfica los tiempos de ejecución de los algoritmos serial y paralelo.

Figura 27: *Tiempos de ejecución imágenes grandes 4 clústers obtenidas con Dron*



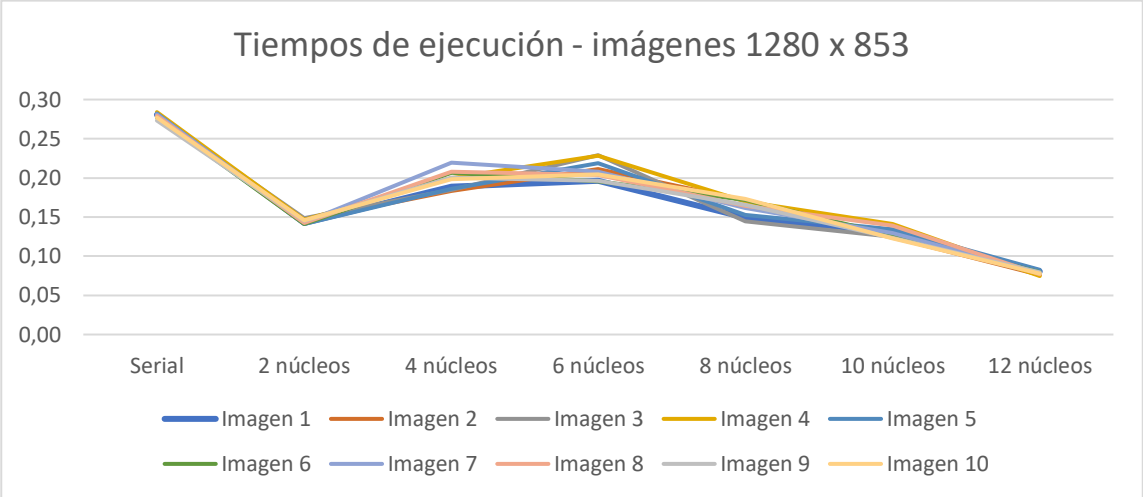
En la tabla 4 se muestra los tiempos de ejecución obtenidos por el algoritmo serial y paralelo para el procesamiento de las imágenes por dron con dimensiones (1280x853 pixeles).

Tabla 4: *Tiempos ejecución imágenes medianas 4 clústers obtenidas con Dron*

Tiempos de ejecución imágenes Medianas –1280x853							
Imagen	Serial	2 núcleos	4 núcleos	6 núcleos	8 núcleos	10 núcleos	12 núcleos
1	0,28	0,14	0,19	0,20	0,15	0,13	0,08
2	0,28	0,15	0,18	0,21	0,17	0,12	0,08
3	0,28	0,15	0,18	0,23	0,14	0,13	0,08
4	0,28	0,15	0,20	0,23	0,17	0,14	0,07
5	0,28	0,14	0,19	0,22	0,15	0,13	0,08
6	0,28	0,14	0,21	0,20	0,17	0,13	0,08
7	0,28	0,14	0,22	0,21	0,16	0,13	0,08
8	0,28	0,14	0,21	0,20	0,16	0,14	0,08
9	0,27	0,15	0,20	0,20	0,17	0,12	0,08
10	0,28	0,15	0,20	0,20	0,17	0,12	0,08
Promedio	**0,28**	**0,14**	**0,20**	**0,21**	**0,16**	**0,13**	**0,08**

En la Figura 28 se representa de manera gráfica los tiempos de ejecución de los algoritmos serial y paralelo

Figura 28: *Tiempos de ejecución imágenes medianas con 4 clústers obtenidas con Dron*



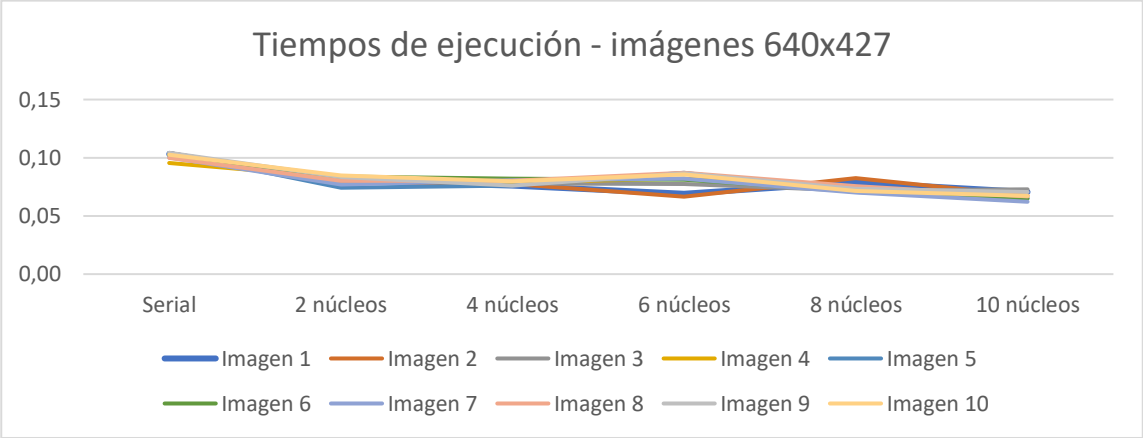
En la tabla 5 se muestra los tiempos de ejecución obtenidos por el algoritmo serial y paralelo para el procesamiento de las imágenes por dron con dimensiones (640x427 pixeles).

Tabla 5: *Tiempos ejecución imágenes pequeñas con 4 clústers obtenidas con Dron*

Tiempos de ejecución imágenes Pequeñas – 640x427							
Imagen	Serial	2 núcleos	4 núcleos	6 núcleos	8 núcleos	10 núcleos	12 núcleos
1	0,10	0,08	0,08	0,07	0,08	0,07	0,04
2	0,10	0,08	0,08	0,07	0,08	0,07	0,04
3	0,10	0,08	0,08	0,08	0,07	0,07	0,04
4	0,10	0,08	0,08	0,08	0,07	0,07	0,04
5	0,10	0,07	0,08	0,09	0,07	0,06	0,04
6	0,10	0,08	0,08	0,08	0,07	0,06	0,04
7	0,10	0,08	0,08	0,08	0,07	0,06	0,05
8	0,10	0,08	0,08	0,09	0,08	0,07	0,04
9	0,10	0,08	0,08	0,09	0,07	0,07	0,04
10	0,10	0,08	0,08	0,09	0,07	0,07	0,04
Promedio	**0,10**	**0,08**	**0,08**	**0,08**	**0,07**	**0,07**	**0,04**

En la Figura 29 se representa los resulta los tiempos de ejecución de manera gráfica de los algoritmos serial y paralelo

Figura 29: *Tiempos de ejecución imágenes pequeñas con 4 clústers obtenidas con Dron*



3.1.2 tiempos de ejecución algoritmo Kmeans con 6 clústers

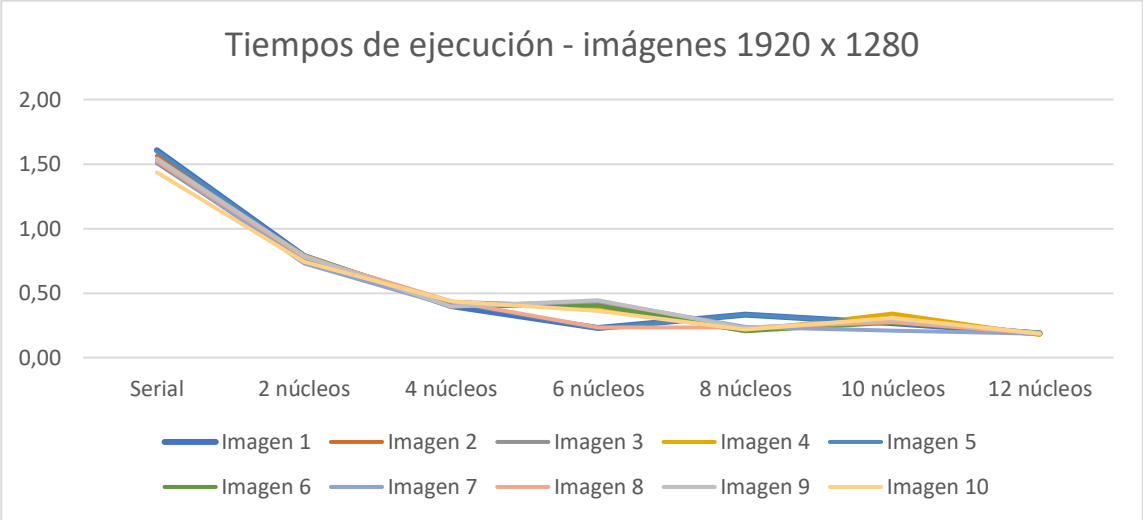
En la tabla 3 se muestra los tiempos de ejecución obtenidos por el algoritmo serial y paralelo para el procesamiento de las imágenes por dron con dimensiones (1920x1280 pixeles).

Tabla 6: *Tiempos ejecución imágenes grandes con 6 clústers obtenidas con Dron*

Tiempos de ejecución imágenes Grandes –1920x1280							
Imagen	Serial	2 núcleos	4 núcleos	6 núcleos	8 núcleos	10 núcleos	12 núcleos
1	1,61	0,79	0,40	0,23	0,33	0,27	0,19
2	1,56	0,76	0,40	0,41	0,21	0,29	0,19
3	1,51	0,76	0,43	0,40	0,20	0,31	0,19
4	1,54	0,79	0,41	0,39	0,21	0,34	0,18
5	1,60	0,78	0,43	0,23	0,33	0,26	0,19
6	1,53	0,78	0,40	0,40	0,21	0,27	0,19
7	1,52	0,73	0,40	0,44	0,24	0,21	0,19
8	1,52	0,78	0,44	0,23	0,24	0,27	0,19
9	1,54	0,79	0,39	0,45	0,23	0,30	0,19
10	1,44	0,74	0,43	0,36	0,22	0,31	0,18
Promedio	**1,54**	**0,77**	**0,41**	**0,35**	**0,24**	**0,28**	**0,19**

En la Figura 30 se presenta los tiempos de ejecución de manera gráfica de los algoritmos serial y paralelo

Figura 30: *Tiempos de ejecución imágenes grandes con 6 clústers obtenidas con Dron*



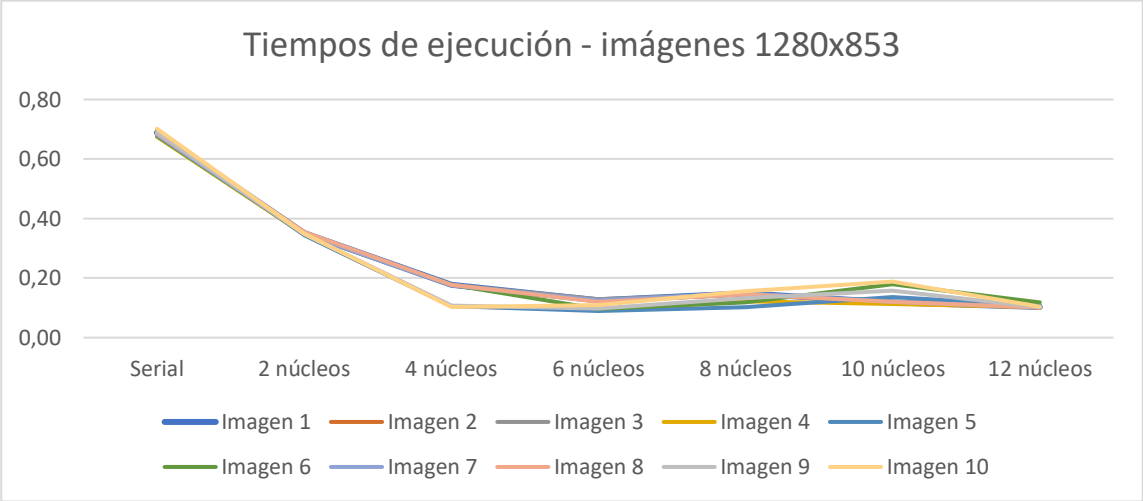
En la tabla 7 se muestra los tiempos de ejecución obtenidos por el algoritmo serial y paralelo para el procesamiento de las imágenes por dron con dimensiones (1280x853 pixeles).

Tabla 7: *Tiempos ejecución imágenes medianas con 6 clústers obtenidas con Dron*

Tiempos de ejecución imágenes Medianas –1280x853							
Imagen	Serial	2 núcleos	4 núcleos	6 núcleos	8 núcleos	10 núcleos	12 núcleos
1	0,69	0,35	0,18	0,13	0,15	0,12	0,10
2	0,69	0,35	0,18	0,12	0,15	0,12	0,10
3	0,70	0,35	0,18	0,13	0,15	0,12	0,11
4	0,68	0,35	0,11	0,09	0,12	0,11	0,10
5	0,69	0,34	0,11	0,09	0,10	0,14	0,11
6	0,68	0,35	0,18	0,10	0,12	0,18	0,12
7	0,68	0,35	0,18	0,13	0,15	0,12	0,10
8	0,69	0,35	0,18	0,12	0,14	0,12	0,10
9	0,69	0,35	0,11	0,10	0,13	0,16	0,10
10	0,70	0,35	0,10	0,11	0,16	0,19	0,10
Promedio	**0,69**	**0,35**	**0,15**	**0,11**	**0,14**	**0,14**	**0,10**

En la Figura 22 se presenta los tiempos de ejecución de manera gráfica de los algoritmos serial y paralelo.

Figura 31: *Tiempos de ejecución imágenes medianas con 6 clústers obtenidas con Dron*



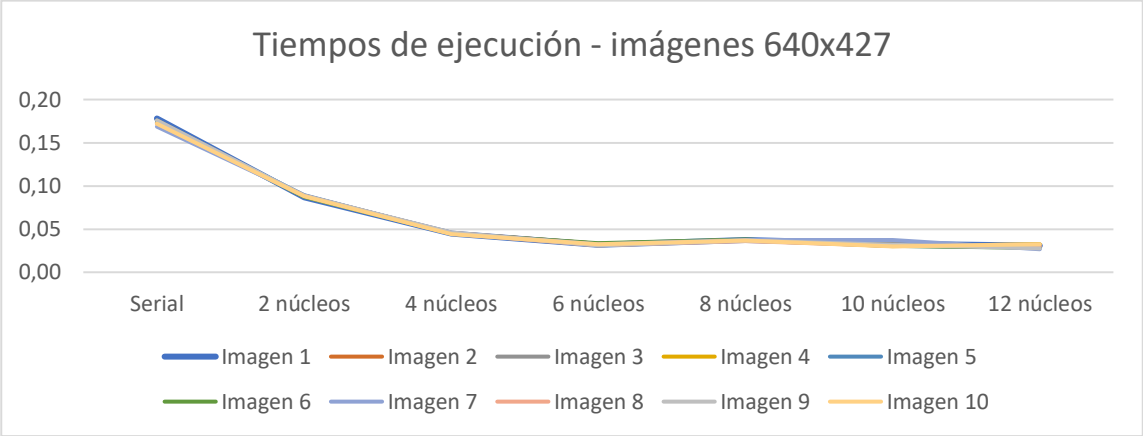
En la tabla 8 se muestra los tiempos de ejecución obtenidos por el algoritmo serial y paralelo para el procesamiento de las imágenes por dron con dimensiones (640x427 pixeles).

Tabla 8: *Tiempos ejecución imágenes pequeñas con 6 clústers obtenidas con Dron*

Tiempos de ejecución imágenes Pequeñas – 640x427							
Imagen	Serial	2 núcleos	4 núcleos	6 núcleos	8 núcleos	10 núcleos	12 núcleos
1	0,18	0,09	0,05	0,03	0,04	0,03	0,03
2	0,17	0,09	0,05	0,03	0,04	0,03	0,03
3	0,17	0,09	0,04	0,03	0,04	0,03	0,03
4	0,18	0,09	0,04	0,03	0,04	0,03	0,03
5	0,17	0,09	0,04	0,03	0,04	0,03	0,03
6	0,17	0,09	0,04	0,03	0,04	0,03	0,03
7	0,17	0,09	0,05	0,03	0,04	0,04	0,03
8	0,17	0,09	0,04	0,03	0,04	0,03	0,03
9	0,18	0,09	0,05	0,03	0,04	0,03	0,03
10	0,17	0,09	0,04	0,03	0,04	0,03	0,03
Promedio	**0,17**	**0,09**	**0,05**	**0,03**	**0,04**	**0,03**	**0,03**

En la Figura 32 se presenta los tiempos de ejecución de manera gráfica de los algoritmos serial y paralelo.

Figura 32: *Tiempos de ejecución imágenes pequeñas con 6 clústers obtenidas con Dron*



3.1.3 tiempos de ejecución algoritmo Kmeans con 8 clústers

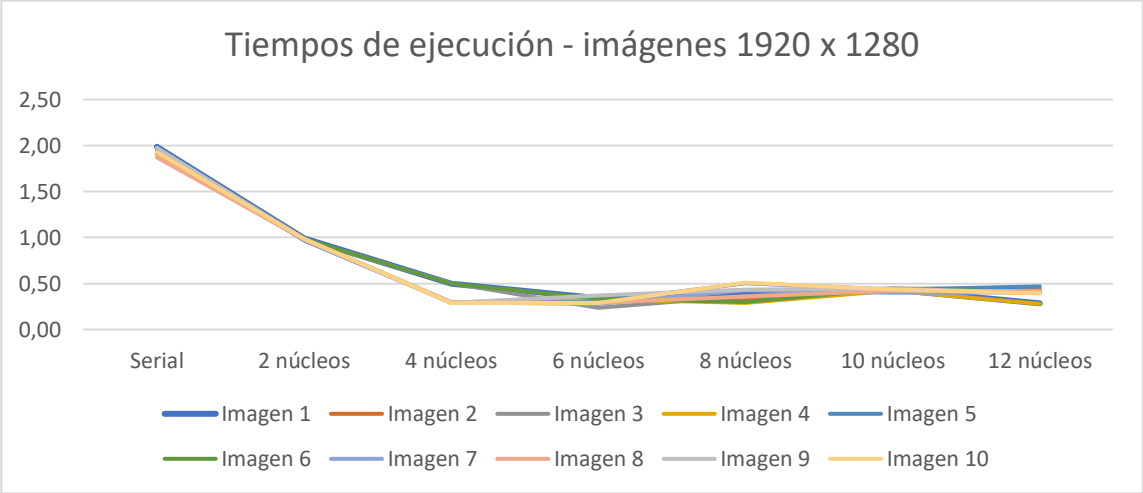
En la tabla 9 se muestra los tiempos de ejecución obtenidos por el algoritmo serial y paralelo para el procesamiento de las imágenes por dron con dimensiones (1920x1280 pixeles).

Tabla 9: *Tiempos ejecución imágenes grandes con 8 clústers obtenidas con Dron*

Tiempos de ejecución imágenes Grandes – 1920 x 1280							
Imagen	Serial	2 núcleos	4 núcleos	6 núcleos	8 núcleos	10 núcleos	12 núcleos
1	1,99	0,99	0,50	0,34	0,39	0,43	0,28
2	1,96	0,98	0,29	0,33	0,32	0,44	0,44
3	1,96	0,97	0,50	0,23	0,36	0,42	0,40
4	1,91	0,99	0,29	0,33	0,29	0,42	0,28
5	1,96	0,96	0,29	0,30	0,50	0,44	0,47
6	1,93	0,99	0,50	0,33	0,30	0,44	0,41
7	1,94	0,97	0,30	0,28	0,43	0,40	0,41
8	1,87	0,98	0,29	0,29	0,36	0,42	0,42
9	1,98	0,98	0,29	0,37	0,43	0,44	0,40
10	1,93	0,98	0,29	0,29	0,51	0,43	0,39
Promedio	**1,94**	**0,98**	**0,35**	**0,31**	**0,39**	**0,43**	**0,39**

En la Figura 22 se presenta los tiempos de ejecución de manera gráfica de los algoritmos serial y paralelo

Figura 33: *Tiempos de ejecución imágenes grandes con 8 clústers obtenidas con Dron*



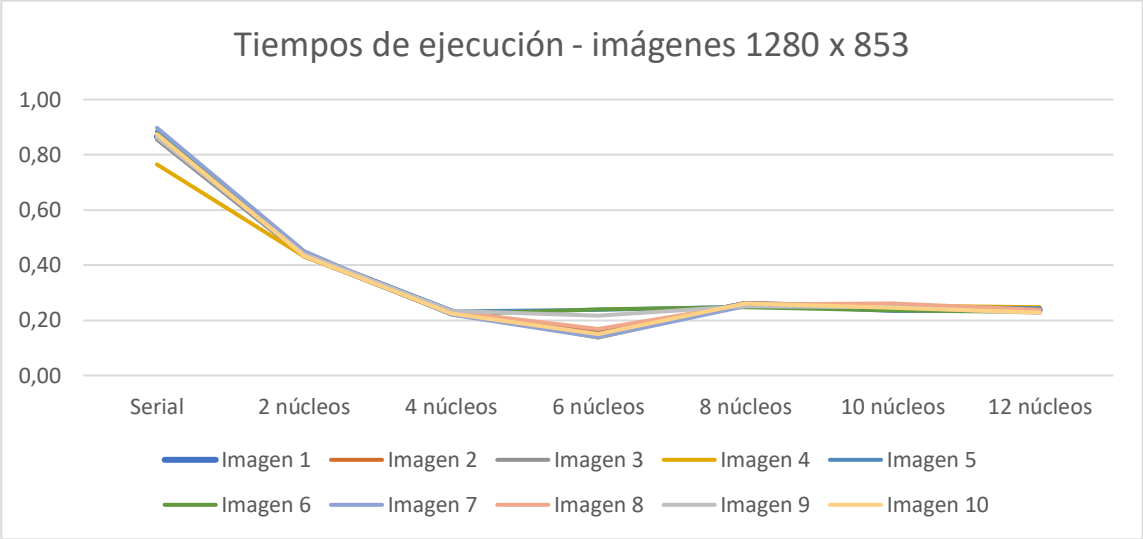
En la tabla 10 se muestra los tiempos de ejecución obtenidos por el algoritmo serial y paralelo para el procesamiento de las imágenes por dron con dimensiones (1280x853 pixeles).

Tabla 10: *Tiempos ejecución imágenes medianas con 8 clústers obtenidas con Dron*

Tiempos de ejecución imágenes Medianas –1280x853							
Imagen	Serial	2 núcleos	4 núcleos	6 núcleos	8 núcleos	10 núcleos	12 núcleos
1	0,87	0,44	0,23	0,15	0,26	0,24	0,24
2	0,87	0,43	0,22	0,15	0,26	0,25	0,23
3	0,85	0,43	0,22	0,15	0,26	0,25	0,24
4	0,77	0,43	0,23	0,14	0,26	0,25	0,25
5	0,88	0,44	0,23	0,24	0,25	0,25	0,24
6	0,88	0,44	0,22	0,24	0,25	0,24	0,23
7	0,90	0,45	0,22	0,14	0,25	0,25	0,24
8	0,86	0,44	0,23	0,17	0,26	0,26	0,24
9	0,86	0,44	0,23	0,22	0,25	0,25	0,23
10	0,87	0,43	0,22	0,15	0,26	0,25	0,23
Promedio	**0,86**	**0,44**	**0,23**	**0,17**	**0,26**	**0,25**	**0,24**

En la Figura 34 se presenta los tiempos de ejecución de manera gráfica de los algoritmos serial y paralelo

Figura 34: *Tiempos de ejecución imágenes medianas con 8 clústers obtenidas con Dron*



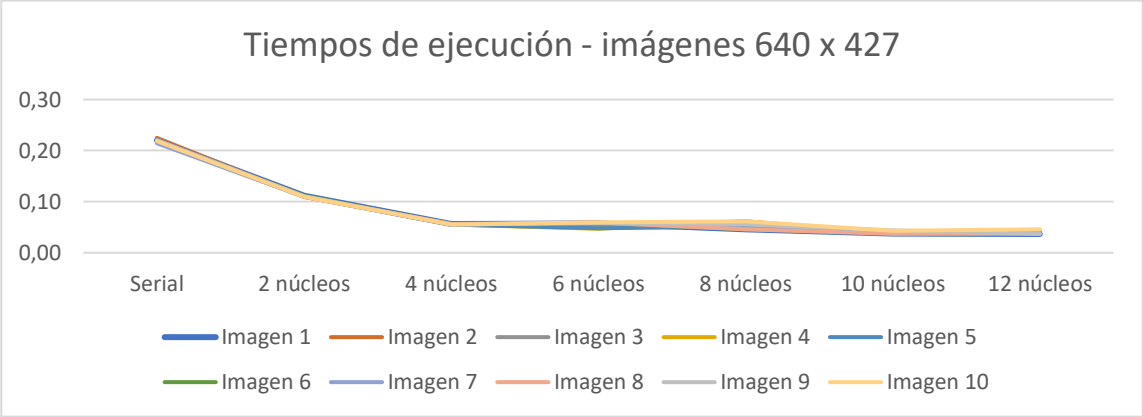
En la tabla 3 se muestra los tiempos de ejecución obtenidos por el algoritmo serial y paralelo para el procesamiento de las imágenes por dron con dimensiones (640x427 pixeles).

Tabla 11: *Tiempos ejecución imágenes pequeñas con 8 clústers obtenidas con Dron*

Tiempos de ejecución imágenes Pequeñas – 640x427							
Imagen	Serial	2 núcleos	4 núcleos	6 núcleos	8 núcleos	10 núcleos	12 núcleos
1	0,22	0,11	0,06	0,06	0,05	0,04	0,04
2	0,23	0,11	0,06	0,06	0,05	0,04	0,04
3	0,22	0,11	0,06	0,05	0,06	0,04	0,04
4	0,22	0,11	0,06	0,05	0,06	0,04	0,04
5	0,22	0,11	0,06	0,05	0,05	0,04	0,04
6	0,22	0,11	0,06	0,06	0,05	0,04	0,04
7	0,21	0,11	0,06	0,06	0,05	0,04	0,04
8	0,22	0,11	0,06	0,06	0,05	0,04	0,04
9	0,22	0,11	0,06	0,06	0,06	0,04	0,04
10	0,22	0,11	0,06	0,06	0,06	0,04	0,05
Promedio	0,20	0,11	0,06	0,06	0,05	0,04	0,04

En la Figura 22 se presenta los tiempos de ejecución de manera gráfica de los algoritmos serial y paralelo

Figura 35: *Tiempos de ejecución imágenes pequeñas con 8 clústers obtenidas con Dron*



3.1.4 tiempos de ejecución algoritmo Kmeans con 10 clústers

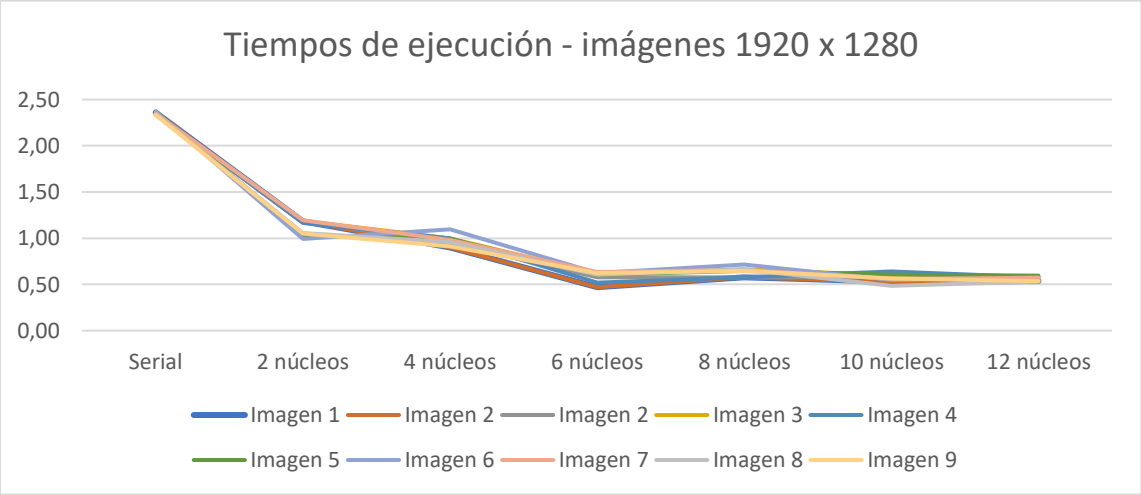
En la tabla 12 se muestra los tiempos de ejecución obtenidos por el algoritmo serial y paralelo para el procesamiento de las imágenes por dron con dimensiones (1920x1280 pixeles).

Tabla 12: *Tiempos ejecución imágenes grandes con 10 clústers obtenidas con Dron*

Tiempos de ejecución imágenes Grandes – 1920 x 1280							
Imagen	Serial	2 núcleos	4 núcleos	6 núcleos	8 núcleos	10 núcleos	12 núcleos
1	2,36	1,19	0,90	0,47	0,57	0,53	0,54
2	2,36	1,19	0,95	0,58	0,58	0,58	0,55
3	2,33	1,19	1,00	0,61	0,67	0,60	0,55
4	2,36	1,16	1,00	0,52	0,58	0,64	0,57
5	2,37	1,03	0,99	0,60	0,65	0,60	0,60
6	2,37	0,99	1,10	0,62	0,71	0,55	0,58
7	2,35	1,19	0,98	0,63	0,66	0,56	0,57
8	2,34	1,06	0,96	0,61	0,66	0,48	0,53
9	2,33	1,05	0,91	0,62	0,65	0,56	0,53
10	2,34	1,06	0,94	0,60	0,61	0,56	0,57
Promedio	**2,35**	**1,11**	**0,97**	**0,59**	**0,63**	**0,57**	**0,56**

En la Figura 36 se presenta los tiempos de ejecución de manera gráfica de los algoritmos serial y paralelo

Figura 36: *Tiempos de ejecución imágenes grandes con 10 clústers obtenidas con Dron*



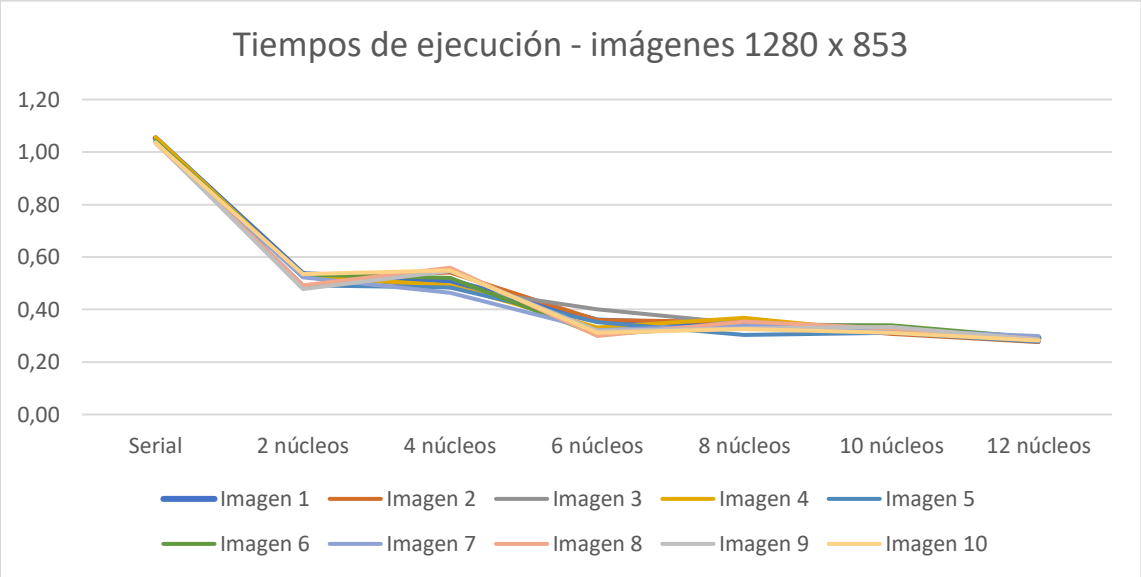
En la tabla 13 se muestra los tiempos de ejecución obtenidos por el algoritmo serial y paralelo para el procesamiento de las imágenes por dron con dimensiones (1920x1280 pixeles).

Tabla 13: *Tiempos ejecución imágenes medianas con 10 clústers obtenidas con Dron*

Tiempos de ejecución imágenes Medianas –1280x853							
Imagen	Serial	2 núcleos	4 núcleos	6 núcleos	8 núcleos	10 núcleos	12 núcleos
1	1,05	0,53	0,51	0,36	0,34	0,31	0,29
2	1,05	0,53	0,54	0,36	0,35	0,31	0,28
3	1,05	0,53	0,48	0,40	0,35	0,34	0,28
4	1,06	0,53	0,49	0,33	0,37	0,31	0,29
5	1,04	0,49	0,48	0,35	0,30	0,31	0,28
6	1,04	0,53	0,52	0,31	0,34	0,34	0,29
7	1,04	0,52	0,46	0,32	0,34	0,32	0,30
8	1,03	0,49	0,56	0,30	0,35	0,33	0,29
9	1,04	0,48	0,55	0,32	0,33	0,33	0,29
10	1,04	0,54	0,55	0,31	0,33	0,31	0,28
Promedio	**1,04**	**0,52**	**0,51**	**0,34**	**0,34**	**0,32**	**0,29**

En la Figura 37 se presenta los tiempos de ejecución de manera gráfica de los algoritmos serial y paralelo

Figura 37: *Tiempos de ejecución imágenes medianas con 10 clústers obtenidas con Dron*



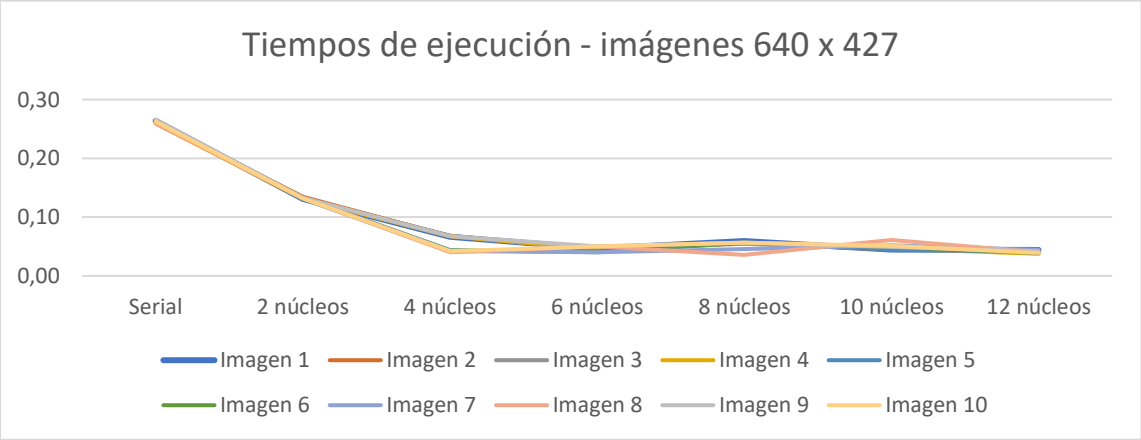
En la tabla 3 se muestra los tiempos de ejecución obtenidos por el algoritmo serial y paralelo para el procesamiento de las imágenes por dron con dimensiones (643x427 pixeles).

Tabla 14: *Tiempos ejecución imágenes pequeña con 10 clústers obtenidas con Dron*

Tiempos de ejecución imágenes Pequeñas – 640x427							
Imagen	Serial	2 núcleos	4 núcleos	6 núcleos	8 núcleos	10 núcleos	12 núcleos
1	0,26	0,13	0,07	0,05	0,06	0,04	0,04
2	0,26	0,13	0,07	0,05	0,06	0,04	0,04
3	0,26	0,13	0,07	0,04	0,06	0,05	0,04
4	0,26	0,13	0,07	0,05	0,06	0,05	0,04
5	0,26	0,13	0,04	0,04	0,06	0,04	0,04
6	0,26	0,13	0,04	0,04	0,06	0,05	0,04
7	0,26	0,13	0,04	0,04	0,05	0,05	0,04
8	0,26	0,13	0,04	0,05	0,04	0,06	0,04
9	0,27	0,13	0,07	0,05	0,06	0,05	0,04
10	0,26	0,13	0,04	0,05	0,06	0,05	0,04
Promedio	**0,26**	**0,13**	**0,05**	**0,05**	**0,05**	**0,05**	**0,04**

En la Figura 38 se presenta los tiempos de ejecución de manera gráfica de los algoritmos serial y paralelo

Figura 38: *Tiempos de ejecución imágenes pequeñas con 10 clústers obtenidas con Dron*

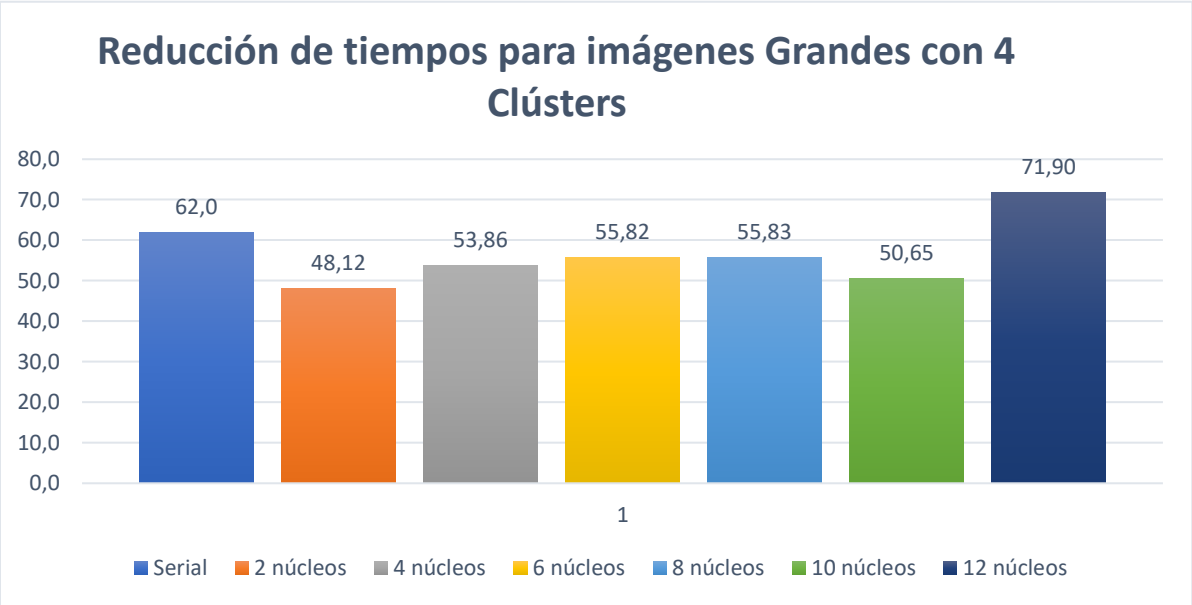


3.2. REDUCCIÓN DE TIEMPOS DE EJECUCIÓN

3.2.1. Reducción de tiempo algoritmo Kmeans con 4 clústers

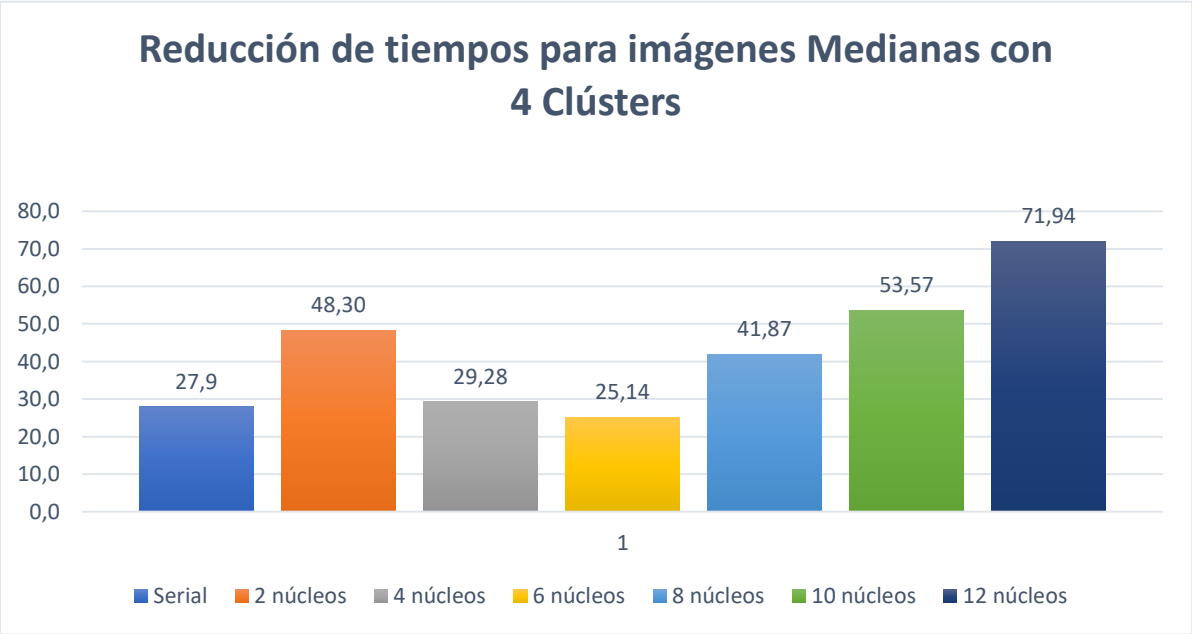
Basado en la tabla 3 se representa de manera gráfica la reducción de tiempos de ejecución con 4 clústers para imágenes grandes (1920x1280) figura 39, donde la máxima reducción que se alcanza de forma paralela es con 12 núcleos reduciendo un 71.90% de tiempo en la ejecución de imágenes, con el resto de los núcleos el porcentaje de reducción es inferior.

Figura 39: *Tiempos de ejecución imágenes grandes con 4 clústers obtenidas con Dron*



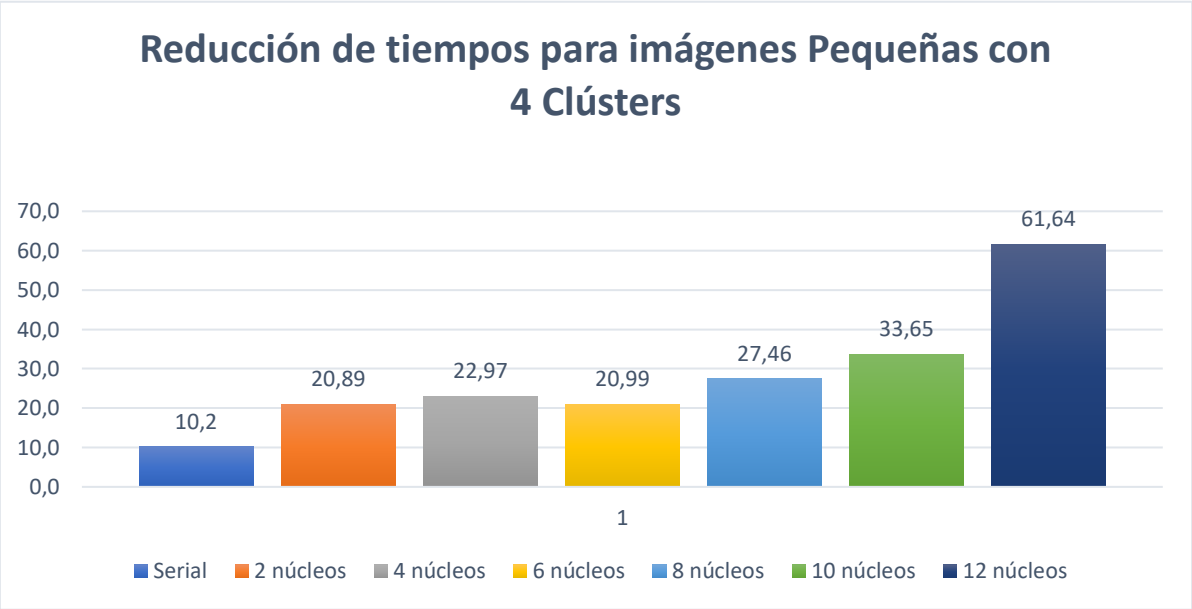
Basado en la tabla 4 se representa de manera gráfica la reducción de tiempos de ejecución con 4 clústers para imágenes medianas (1280x853) figura 40, donde la máxima reducción que se alcanza de forma paralela es con 12 núcleos reduciendo un 71.94% de tiempo en la ejecución de imágenes, con el resto de los núcleos el porcentaje de reducción es inferior.

Figura 40: *Tiempos de ejecución imágenes medianas con 4 clústers obtenidas con Dron*



Basado en la tabla 5 se representa de manera gráfica la reducción de tiempos de ejecución con 4 clústers para imágenes grandes (640x427) figura 41, donde la máxima reducción que se alcanza de forma paralela es con 12 núcleos reduciendo un 61.64% de tiempo en la ejecución de imágenes, con el resto de los núcleos el porcentaje de reducción es inferior.

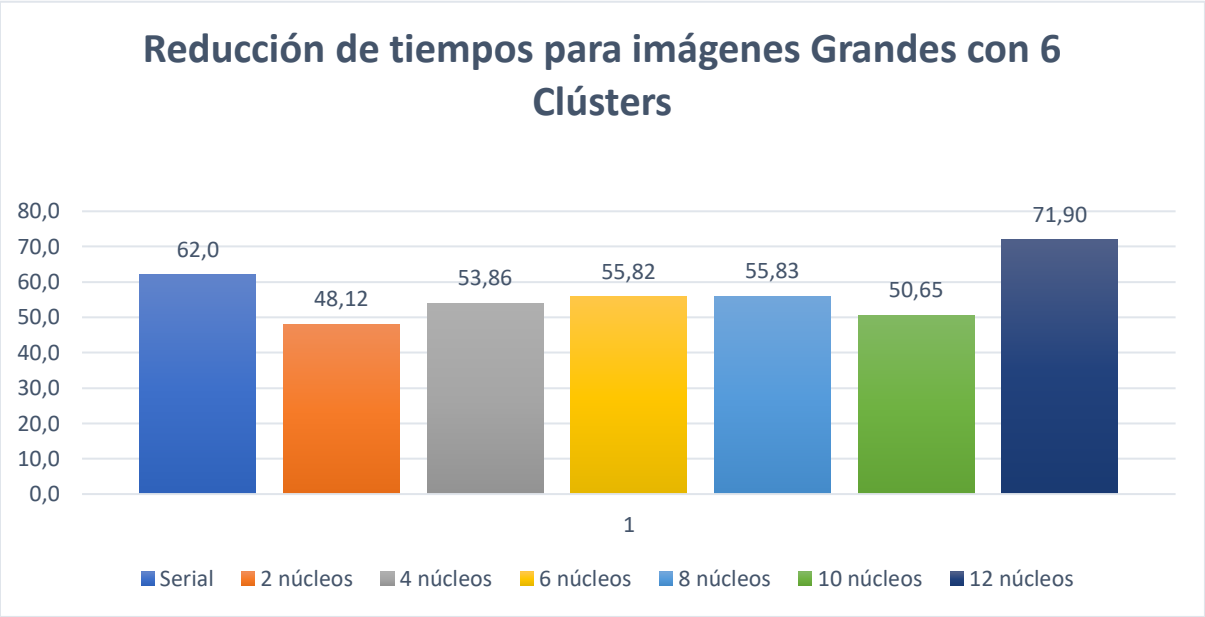
Figura 41: *Tiempos de ejecución imágenes pequeñas con 4 clústers obtenidas con Dron*



3.2.2. Reducción de tiempo algoritmo Kmeans con 6 clústers

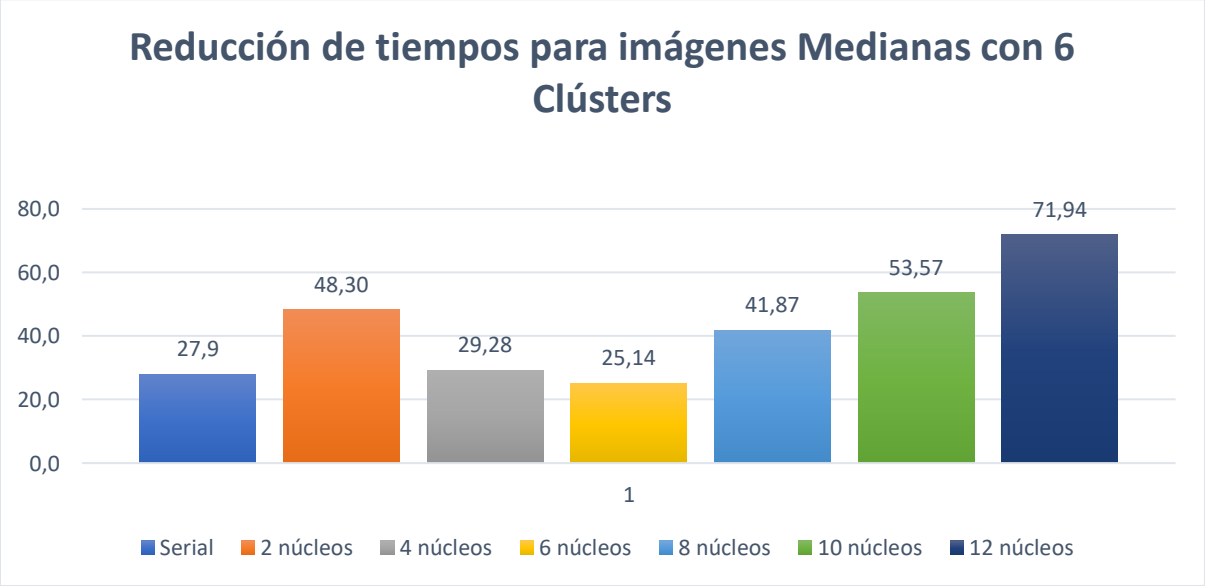
Basado en la tabla 6 se representa de manera gráfica la reducción de tiempos de ejecución con 6 clústers para imágenes grandes (1920x1280) figura 42, donde la máxima reducción que se alcanza de forma paralela es con 12 núcleos reduciendo un 71.90% de tiempo en la ejecución de imágenes, con el resto de los núcleos el porcentaje de reducción es inferior.

Figura 42: *Tiempos de ejecución imágenes grandes con 6 clústers obtenidas con Dron*



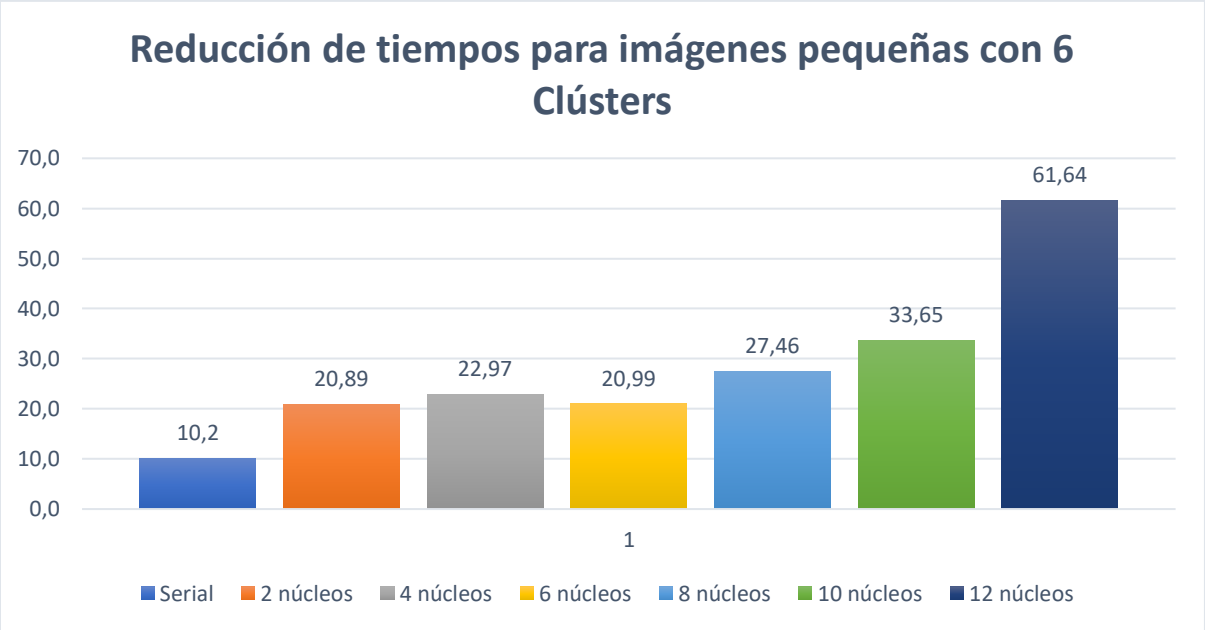
Basado en la tabla 7 se representa de manera gráfica la reducción de tiempos de ejecución con 6 clústers para imágenes medianas (1280x853) figura 43, donde la máxima reducción que se alcanza de forma paralela es con 12 núcleos reduciendo un 71.94% de tiempo en la ejecución de imágenes, con el resto de los núcleos el porcentaje de reducción es inferior.

Figura 43: *Tiempos de ejecución imágenes medianas con 6 clústers obtenidas con Dron*



Basado en la tabla 8 se representa de manera gráfica la reducción de tiempos de ejecución con 6 clústers para imágenes pequeñas (640x427) figura 44, donde la máxima reducción que se alcanza de forma paralela es con 12 núcleos reduciendo un 61.64% de tiempo en la ejecución de imágenes, con el resto de los núcleos el porcentaje de reducción es inferior.

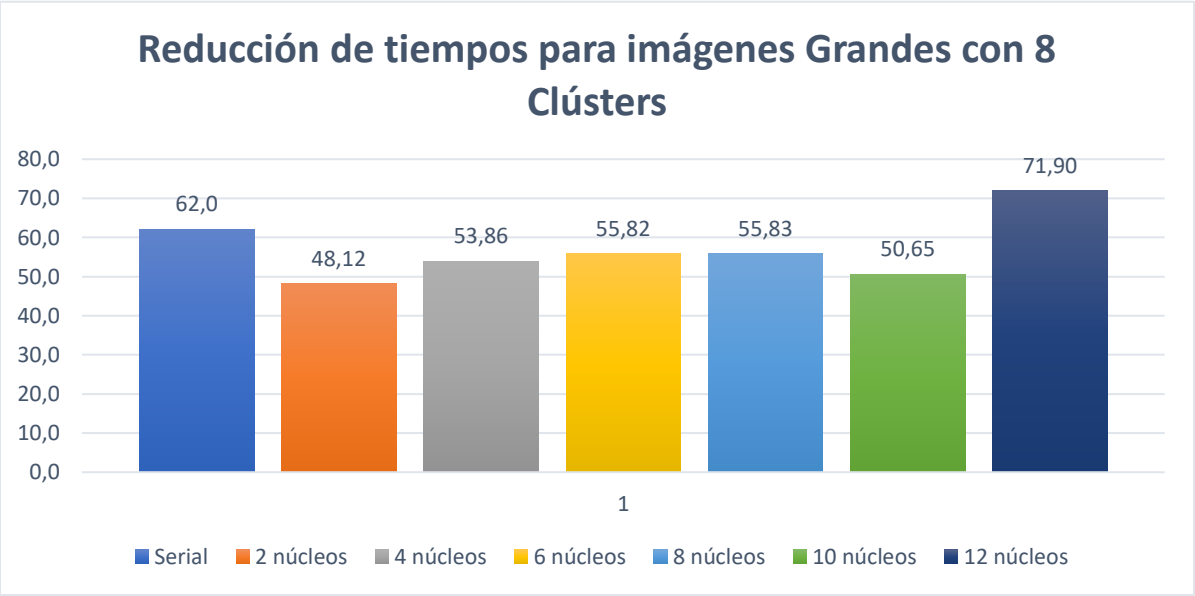
Figura 44: *Tiempos de ejecución imágenes pequeñas con 6 clústers obtenidas con Dron*



3.2.3. Reducción de tiempo algoritmo Kmeans con 8 clústers

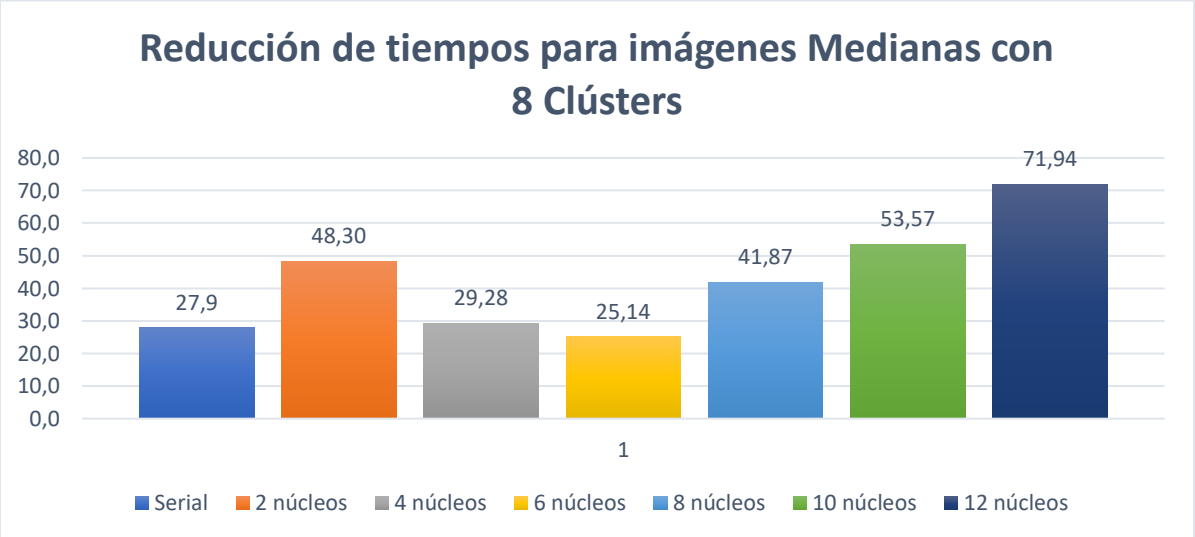
Basado en la tabla 9 se representa de manera gráfica la reducción de tiempos de ejecución con 8 clústers para imágenes grandes (1920x1280) figura 45, donde la máxima reducción que se alcanza de forma paralela es con 12 núcleos reduciendo un 71.90% de tiempo en la ejecución de imágenes, con el resto de los núcleos el porcentaje de reducción es inferior.

Figura 45: *Tiempos de ejecución imágenes grandes con 8 clústers obtenidas con Dron*



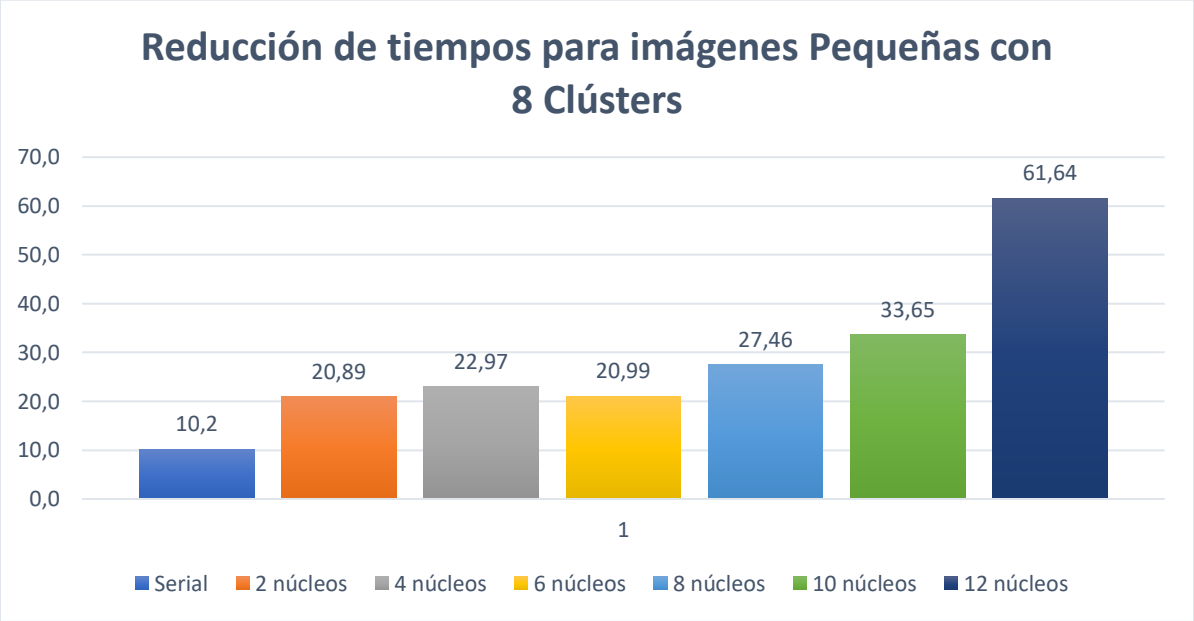
Basado en la tabla 10 se representa de manera gráfica la reducción de tiempos de ejecución con 8 clústers para imágenes medianas (1280x853) figura 46, donde la máxima reducción que se alcanza de forma paralela es con 12 núcleos reduciendo un 71.94% de tiempo en la ejecución de imágenes, con el resto de los núcleos el porcentaje de reducción es inferior.

Figura 46: *Tiempos de ejecución imágenes medianas con 8 clústers obtenidas con Dron*



Basado en la tabla 11 se representa de manera gráfica la reducción de tiempos de ejecución con 4 clústers para imágenes pequeñas (640x427) figura 47, donde la máxima reducción que se alcanza de forma paralela es con 12 núcleos reduciendo un 61.64% de tiempo en la ejecución de imágenes, con el resto de los núcleos el porcentaje de reducción es inferior.

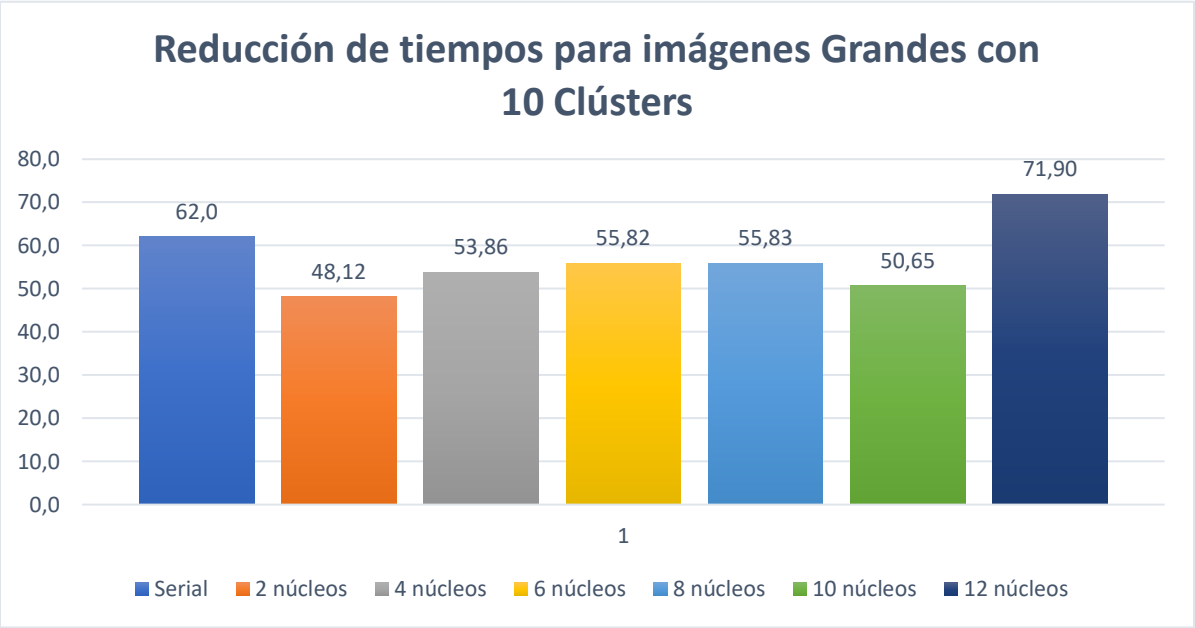
Figura 47: *Tiempos de ejecución imágenes pequeñas con 8 clústers obtenidas con Dron*



3.2.4. Reducción de tiempo algoritmo Kmeans con 10 clústers

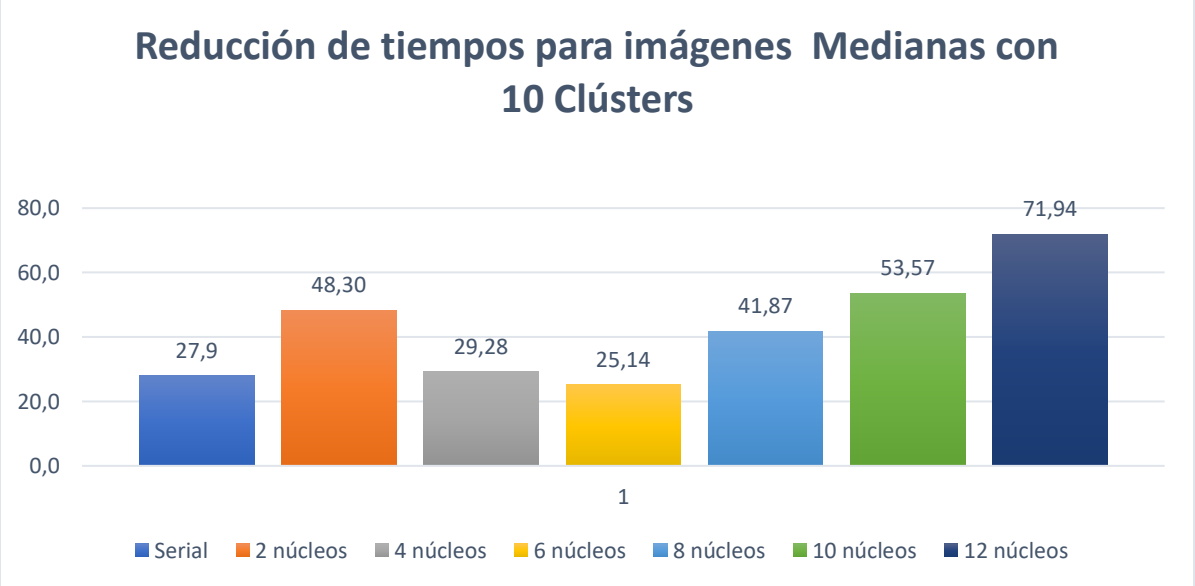
Basado en la tabla 12 se representa de manera gráfica la reducción de tiempos de ejecución con 4 clústers para imágenes grandes (1920 x 1280) figura 48, donde la máxima reducción que se alcanza de forma paralela es con 12 núcleos reduciendo un 71.90% de tiempo en la ejecución de imágenes, con el resto de los núcleos el porcentaje de reducción es inferior.

Figura 48: *Tiempos de ejecución imágenes grandes con 10 clústers obtenidas con Dron*



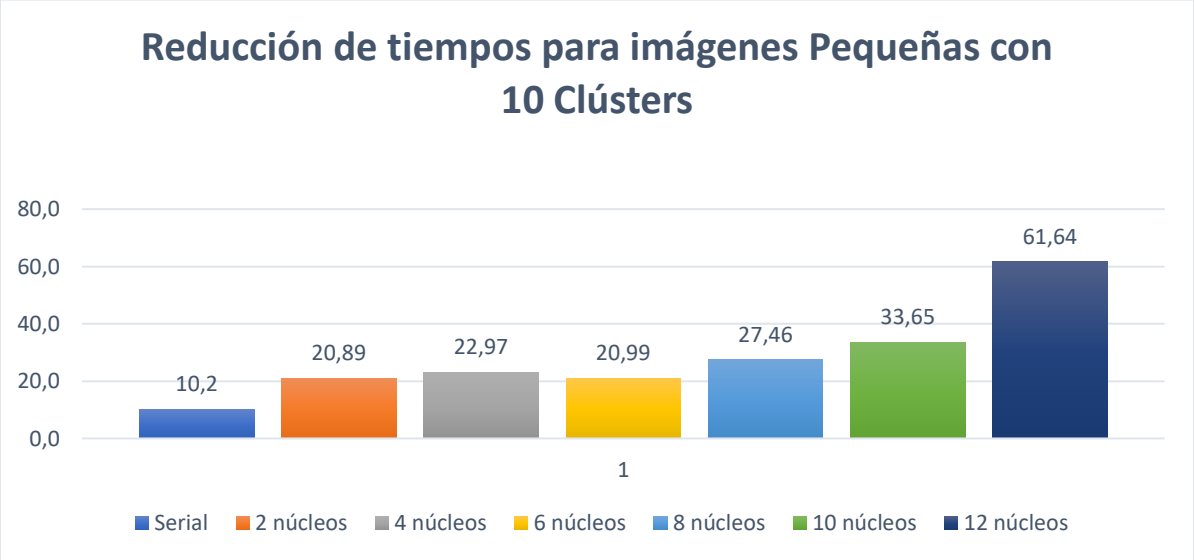
Basado en la tabla 13 se representa de manera gráfica la reducción de tiempos de ejecución con 4 clústers para imágenes medianas (1280x853) figura 49, donde la máxima reducción que se alcanza de forma paralela es con 12 núcleos reduciendo un 71.94% de tiempo en la ejecución de imágenes, con el resto de los núcleos el porcentaje de reducción es inferior.

Figura 49: *Tiempos de ejecución imágenes medianas con 10 clústers obtenidas con Dron*



Basado en la tabla 14 se representa de manera gráfica la reducción de tiempos de ejecución con 4 clústers para imágenes pequeñas (640x427), donde la máxima reducción que se alcanza de forma paralela es con 12 núcleos reduciendo un 61.64% de tiempo en la ejecución de imágenes, con el resto de los núcleos el porcentaje de reducción es inferior.

Figura 50: *Tiempos de ejecución imágenes pequeñas con 10 clústers obtenidas con Dron*



3.3Tiempo de ejecución promedio

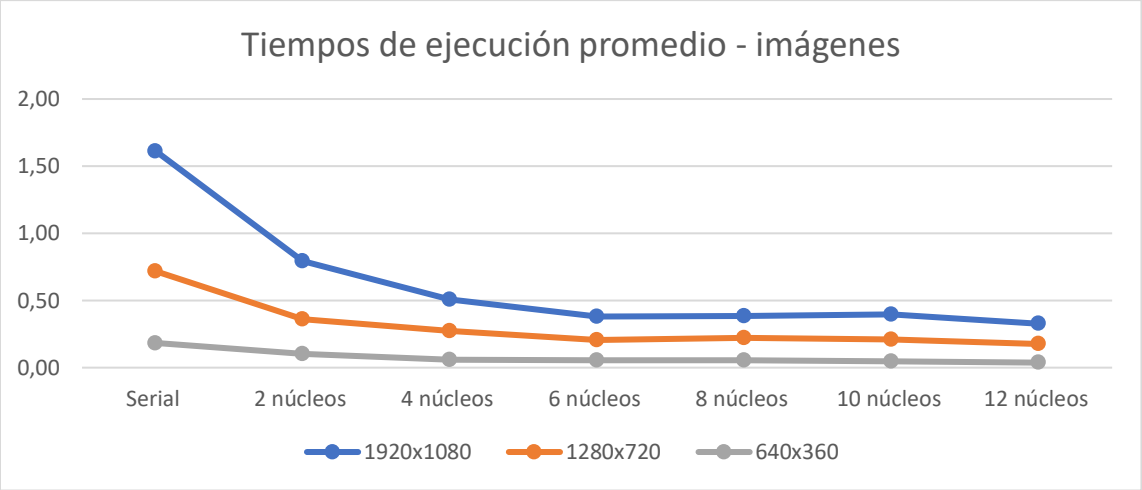
De acuerdo con los resultados obtenidos en la evaluación de los algoritmos se presenta en la Tabla 15 los tiempos de ejecución promedio obtenidos por el algoritmo serial y paralelo (2,4,6,8,10 y 12 núcleos) para el procesamiento de las imágenes por dron (grandes, medianas y pequeñas).

Tabla 15: *Tiempos ejecución promedio de imágenes con Dron*

Tiempos de ejecución promedio de imágenes con dron							
Dimensión	Serial	2 núcleos	4 núcleos	6 núcleos	8 núcleos	10 núcleos	12 núcleos
1920x1280	1,61	0,80	0,51	0,38	0,38	0,40	0,33
1280x853	0,72	0,36	0,27	0,21	0,22	0,21	0,18
640x427	0,18	0,10	0,06	0,05	0,05	0,05	0,04

Figura 51 los tiempos de ejecución promedio obtenidos por el algoritmo serial y paralelo (2,4,6,8,10 y 12 núcleos) para el procesamiento de las imágenes por dron (grandes, medianas y pequeñas).

Figura 51: *Tiempos de ejecución promedio en imágenes grandes obtenidas con Dron*



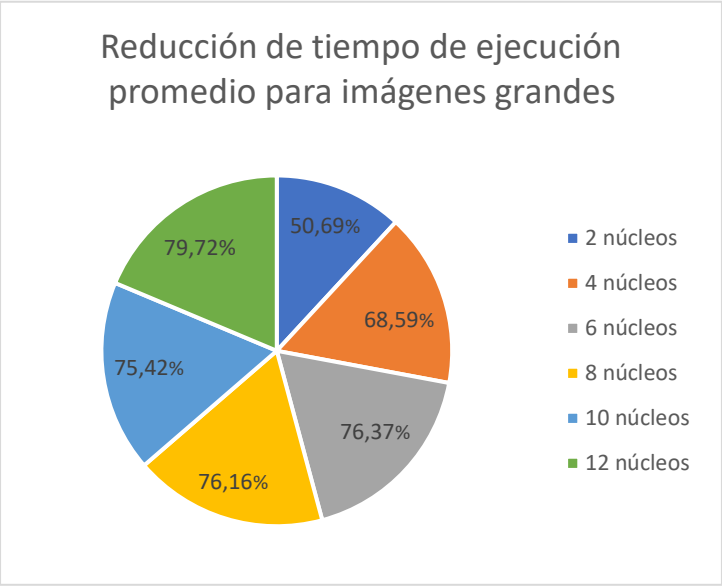
3.4 Análisis e interpretación de resultados

En la tabla 16 y figura 52 se representa la máxima reducción que se alcanza de forma paralela es con 12 núcleos para imágenes grandes (1920x1280), reduciendo un 75.27% de tiempo en la ejecución de imágenes, con el resto de los núcleos el porcentaje de reducción es inferior.

Tabla 16: *Tiempos de ejecución promedio de imágenes Grandes con dron*

Tiempos de ejecución promedio de imágenes Grandes con dron – 1920 x 1280							
N° clústers	Serial	2 núcleos	4 núcleos	6 núcleos	8 núcleos	10 núcleos	12 núcleos
4	0,62	0,32	0,29	0,27	0,27	0,31	0,17
6	1,54	0,77	0,41	0,35	0,24	0,28	0,19
8	1,94	0,98	0,35	0,31	0,39	0,43	0,39
10	2,35	1,11	0,97	0,59	0,63	0,57	0,56
Promedio	**1,61**	**0,80**	**0,51**	**0,38**	**0,38**	**0,40**	**0,33**

Figura 52: *Reducción de tiempo de ejecución promedio para imágenes grandes*

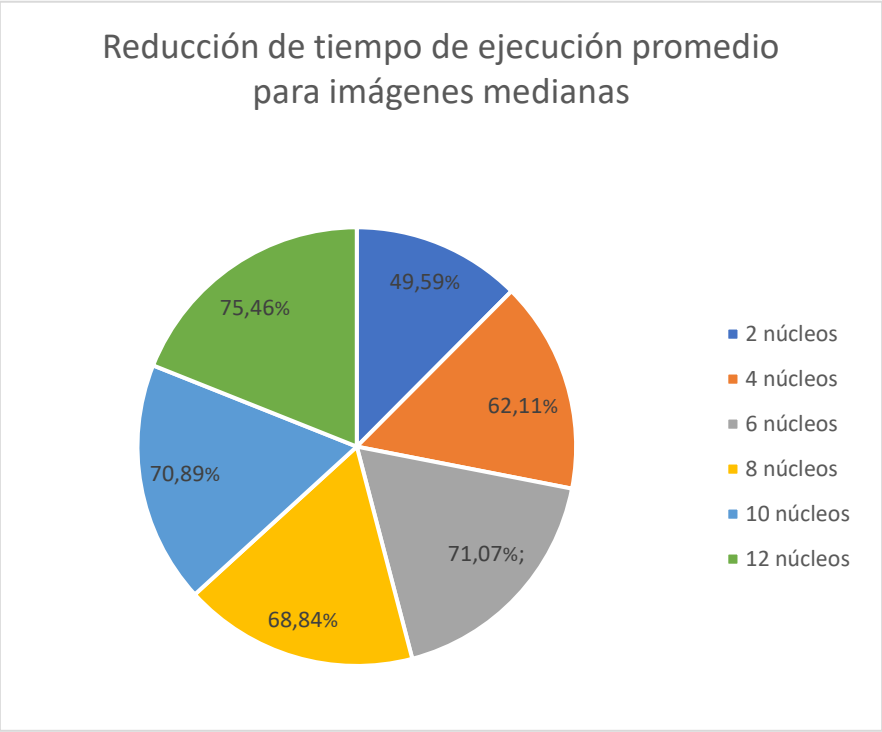


En la Tabla 17 y Figura 53, se presenta la máxima reducción que se alcanza de forma paralela es con 12 núcleos para imágenes grandes (1280x853), reduciendo un 75.27% de tiempo en la ejecución de imágenes, con el resto de los núcleos el porcentaje de reducción es inferior

Tabla 17: *Tiempos de ejecución promedio de imágenes Medianas con dron*

Tiempos de ejecución promedio de imágenes Medianas con dron – 1280 x 853							
N° clústers	Serial	2 núcleos	4 núcleos	6 núcleos	8 núcleos	10 núcleos	12 núcleos
4	0,28	0,14	0,20	0,21	0,16	0,13	0,08
6	0,69	0,35	0,15	0,11	0,14	0,14	0,10
8	0,86	0,44	0,23	0,17	0,26	0,25	0,24
10	1,04	0,52	0,51	0,34	0,34	0,32	0,29
Promedio	**0,72**	**0,36**	**0,27**	**0,21**	**0,22**	**0,21**	**0,18**

Figura 53: *Reducción de tiempo de ejecución promedio para imágenes medianas*

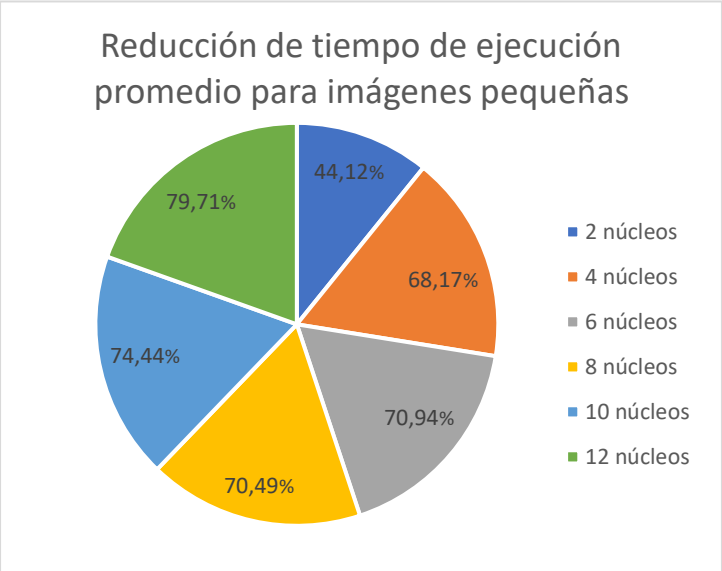


En la Tabla 18 y Figura 54, se presenta la máxima reducción que se alcanza de forma paralela es con 12 núcleos para imágenes pequeñas (640x247), reduciendo un 75.27% de tiempo en la ejecución de imágenes, con el resto de los núcleos el porcentaje de reducción es inferior

Tabla 18: *Tiempos de ejecución promedio de imágenes pequeñas con dron*

Tiempos de ejecución promedio de imágenes Pequeñas con dron – 640 x 427							
N° clústers	Serial	2 núcleos	4 núcleos	6 núcleos	8 núcleos	10 núcleos	12 núcleos
4	0,10	0,08	0,08	0,08	0,07	0,07	0,04
6	0,17	0,09	0,05	0,03	0,04	0,03	0,03
8	0,20	0,11	0,06	0,06	0,05	0,04	0,04
10	0,26	0,13	0,05	0,05	0,05	0,05	0,04
Promedio	**0,18**	**0,10**	**0,06**	**0,05**	**0,05**	**0,05**	**0,04**

Figura 54: *Reducción de tiempo de ejecución promedio para imágenes pequeñas obtenidas con Dron*



3.5 Análisis de impactos

Impacto científico

El algoritmo K-means ha contribuido significativamente en varias áreas científicas como es la ciencia y minería de datos, la bioinformática (agrupación de genes y proteínas), la medicina, el medio ambiente, la economía, el análisis de redes sociales, el análisis de imágenes y visión por computador e Inteligencia artificial, siendo una herramienta eficaz para la identificación de patrones y datos no etiquetados.

En el desarrollo del algoritmo K-means para la detección de maleza se refleja en la capacidad de los investigadores para automatizar y mejorar la precisión de la detección de maleza en imágenes agrícolas, como la segmentación de imágenes, la extracción de características relevantes de las imágenes ya sea el color, la forma o la textura, toda la información obtenida contribuye a la eficiencia y rentabilidad de la agricultura al utilizar lenguajes de programación

paralela en el procesamiento de imágenes agrícolas, al utilizar diferentes herramientas los investigadores se basarían en los resultado

Impacto tecnológico

El desarrollo del algoritmo paralelo para determinar la densidad de maleza en cultivos de maíz puede tener un impacto tecnológico significativo al mejorar la eficiencia, ahorrar tiempo y recursos, optimizar el control de malezas, permitir el monitoreo continuo el cual ayuda a prevenir la propagación de malezas y permite un manejo más eficiente de los cultivos, al mismo tiempo fomentar la investigación agrícola avanzada recopilando datos a través del algoritmo y para poder realizar análisis más profundos, identificar patrones y tendencias, y desarrollar modelos predictivos para mejorar aún más las prácticas agrícolas.

Impacto Económico

El algoritmo Kmeans para la detección de maleza en cultivos de maíz se desarrolló con herramientas y librerías open source (software libre), reduciendo costos para el equipo donde se requiera implementarlo. Al automatizar el proceso de detección de malezas se está reduciendo costos en mano de obra para inspeccionar manualmente los campos, ahora se lo podrá hacer por medio de un Dron que proporciona imágenes para ser analizadas en el programa , la información de las imágenes ayuda a los agricultores a tomar decisiones sobre la rotación de cultivos, la programación de labores agrícolas y la implementación de prácticas de manejo de maleza, estas decisiones estratégicas pueden mejorar la eficiencia operativa y la rentabilidad general de la explotación agrícola , así solo se invertiría en personal para el manejo del aplicativo , además de que el agricultor se encarga de aplicar los herbicidas de manera más selectiva y dirigida aumentando el rendimiento de los cultivos y optimizando la planificación agrícola.

CONCLUSIONES

- El objetivo principal del proyecto fue desarrollar un algoritmo paralelo para determinar la densidad de maleza en cultivos de maíz, basado en el algoritmo K-means con imágenes procedentes de dron el cual se demostró obtener resultados óptimos en tiempos de ejecución.

- Entre los objetivos del proyecto planteado fue investigar y comprender los conceptos más fundamentales del procesamiento de imágenes en la agricultura de precisión haciendo uso de las herramientas de programación paralela como son: OpenMP y Python , donde podemos concluir que las herramientas mencionadas son de fácil integración y los investigadores pueden aprovechar las ventajas de la programación paralela abordando problemas más complejos y obtener resultados más rápidos y precisos para las aplicaciones que se desarrollen en las diferentes áreas científicas.

- la programación paralela en el algoritmo K-means representa una poderosa herramienta para mejorar la eficiencia y escalabilidad en el rendimiento de análisis de datos y aplicaciones de aprendizaje automático. Al entender y aprovechar adecuadamente esta técnica, los investigadores, científicos de datos y desarrolladores pueden abordar problemas más complejos y obtener resultados más rápidos y precisos en una amplia variedad de campos.

- OpenMP es una de API de programación paralela muy popular que ha demostrado ser eficaz en diferentes entornos y aplicaciones además de optimizar el potencial del paralelismo para el desarrollo de algoritmos en arquitectura Multicore.

- Python es un lenguaje de programación que se adapta para el desarrollo de programas paralelos donde ayuda a disminuir el tiempo de ejecución juntamente con la arquitectura paralela.

- Las métricas definidas permitieron evaluar el rendimiento de los algoritmos para imágenes con dron de diferentes dimensiones. Estos resultados confirman que el algoritmo paralelo supera en rendimiento al algoritmo serial.
- Los resultados alcanzados con la implementación del algoritmo paralelo demuestran que el rendimiento de cualquier algoritmo puede mejorarse en las diferentes áreas científicas. La implementación del algoritmo paralelo reduce los tiempos de ejecución y mejora la rapidez a medida que se aumenta el número de núcleos.

RECOMENDACIONES

Se recomienda continuar con el uso del algoritmo en otras etapas para el procesamiento de imágenes u otras aplicaciones para el procesamiento de datos, investigar y adecuar los algoritmos a las necesidades de la industria agrícola, con el objetivo de optimizar los resultados en áreas como el monitoreo de los cultivos, control de plagas, la detección de enfermedades y la reducción del uso de fertilizantes.

Se propone fomentar el estudio con lenguajes de programación paralela que sean compatibles con sistemas operativos Linux, puesto que son de código abierto y permite el uso de comandos y herramientas disponibles en los repositorios de Ubuntu sin la necesidad de adquirir licencias temporales o pagadas, podemos aprovechar las ventajas para optimizar el trabajo como desarrolladores. Además, al tratarse de un sistema operativo muy utilizado en entornos profesionales y empresariales podemos desarrollar habilidades en lenguajes compatibles con Linux abriendo muchas puertas en el ámbito laboral y mejorar las oportunidades en el ámbito tecnológico.

Se recomienda hacer estudios de campo aplicando la arquitectura paralela, ya que hoy es muy importante para la computación, pero su programación requiere mucho trabajo, esfuerzo e investigación para dominarla.

Se debe considerar el lenguaje de programación más apropiado al momento de ejecutar un algoritmo realizando estudios exhaustivos de las arquitecturas paralelas para garantizar un rendimiento óptimo de los resultados y la comprensión de las características, capacidades y limitaciones de los distintos modelos arquitectónicos.

Se recomienda estudiar y comprender la funciones y ventajas que brindan las herramientas de desarrollo en paralelo como OpenMP, Python y C++ con el objetivo de programar aplicaciones en paralelo.

Se recomienda aplicar algoritmos en arquitecturas paralelas para aprovechar los recursos computacionales a medida que aumentan el número de núcleos o procesadores y conseguir una velocidad adecuada además de reducción en tiempos de ejecución de las imágenes.

BIBLIOGRAFÍA

Acosta, F. A., Segura, O. M., & Ospina, A. E. (2012). Guía y fundamentos de la programación en paralelo. In *Revista en Telecomunicaciones e Informática* (Vol. 2, Issue 4).

ALVARO. (2019, March 8). *Segmentación utilizando K-means en Python*. https://machinelearningparatodos.com/segmentacion-utilizando-k-means-en-python/

Amoroso Ordóñez, M. S., & Ávila Flores, H. S. (2015). *APLICACIÓN DE LAS TÉCNICAS DE AGRUPAMIENTO PARA LA DISTRIBUCIÓN CUASI–ÓPTIMA DE UNA RED HÍBRIDA WDM–TDM/PON EN CASCADA MULTINIVEL QUE DA SOPORTE A UNA SMART GRID O SMART CITY*.

Angulo, C., Bocardo, F., Flores, J., & Roman, O. (2022, April 5). *RPubs - K means*. RPubs. https://rpubs.com/JosueEmmanuel/Kmeans

Bala Priya C. (2022, February 15). *Aprenda la agrupación en clústeres de K-Means mediante la cuantificación de imágenes en color en Python*. Https://Hackernoon.Com/Es/Learn-k-Means-Clustering-by-Quantizing-Color-Images-in-Python.

Barbakh, W., & Fyfe, C. (2008). Local vs global interactions in clustering algorithms: Advances over K-means. *International Journal of Knowledge-Based and Intelligent Engineering Systems*, *12*(2), 83–99. https://doi.org/10.3233/KES-2008-12201

Bernal, F., Albarracín, C., Gaona, J., Giraldo, L., & Mosquera, C. (2020). *Programación Paralela*. http://ferestrepoca.github.io/paradigmas-de-programacion/paralela/paralela_teoria/index.html

Chuidiang. (2007). *Conceptos básicos de make y los Makefile*. Chuidiang.Org. https://www.chuidiang.org/clinux/herramientas/makefile.php

De La Cruz Gutiérrez Keyla, Cieza Rivasplata Joel, & Flores Sahuanga Caleb. (2015). *APLICACIÓN DEL MODELO DE CLUSTERIZACIÓN BASADO EN EL ALGORITMO DE K-MEANS PARA LA SEGMENTACIÓN DE LA MORBILIDAD MATERNA EN EL HOSPITAL SAN BARTOLOMÉ DE LA CIUDAD DE LIMA-2012*. 16.

Downey, Allen., Meyer, C., & Elkner, Jeffrey. (2002). *How to think like a computer scientist : learning with Python*. Green Tea Press.

Eclipse Foundation. (2013, February 21). *Eclipse for Parallel Application Developers | Eclipse Packages*. https://www.eclipse.org/downloads/packages/release/juno/sr2/eclipse-parallel-application-developers

Galiano Casas, G., & García Gonzalo, E. (2014, May 13). *El algoritmo k-means aplicado a clasificación y procesamiento de imágenes*. Universidad de Oviedo. https://www.unioviedo.es/compnum/laboratorios_py/new/kmeans.html#segmentacion

García Mera, J. A., & Zambrano Moreira, A. J. (2022). *ALGORITMOS GENETICOS PARALELIZADOS*. Escuela Superior de Agropecuaría de Manabí.

Gómez, M. Y., Ruíz Sergio J., & Farid García. (2020). Clasificación de plantas de maíz y maleza: Hacia la mejora de la fertilización en México. In *Research in Computing Science* (Vol. 149, Issue 8).

Gürsoy, A., & Cengiz, ˙ Ilker. (2001). *Parallel Pruning for K-Means Clustering on Shared Memory Architectures*.

Hilera González, R. J., & Martínez Hernando, V. J. (1995). *REDES NEURONALES ARTIFICIALES: Fundamentos, modelos y aplicaciones*. https://www.researchgate.net/publication/44343683

Hnin Hlaing, S., & Soe Khaing, A. (2014). WEED AND CROP SEGMENTATION AND CLASSIFICATION USING AREA THRESHOLDING. In *IJRET: International Journal of Research in Engineering and Technology*. http://www.ijret.org

IBM. (n.d.). *¿Qué es el aprendizaje supervisado?* Retrieved August 19, 2023, from https://www.ibm.com/mx-es/topics/supervised-learning

International Business Machines Corporation. (2019, December). *¿Qué es el algoritmo de kmedias más cercanos?* IBM. https://www.ibm.com/co-es/topics/knn

Karoussi, E., & Bouhmala, N. (2012). *Data Mining K-Clustering Problem*.

Kucukyilmaz, T. (2014). Parallel K-Means Algorithm for Shared Memory Multiprocessors. *Journal of Computer and Communications*, *02*(11), 15–23. https://doi.org/10.4236/jcc.2014.211002

Lambert Chase. (2021). *Makefile Tutorial By Example*. Makefiletutorial. https://makefiletutorial.com/#include-makefiles

MacQueen, J. B. (1966). *SOME METHODS FOR CLASSIFICATION AND ANALYSIS OF MULTIVARIATE OBSERVATIONS*.

Martinez Heras Jose. (2020). *Clustering (Agrupamiento), K-Means con ejemplos en python - IArtificial.net*. IArtificial.Net. https://www.iartificial.net/clustering-agrupamiento-kmeans-ejemplos-en-python/

Matich Damián Jorge. (2001). *Redes Neuronales: Conceptos Básicos y Aplicaciones*. https://www.frro.utn.edu.ar/repositorio/catedras/quimica/5_anio/orientadora1/monograias/matich-redesneuronales.pdf

Moraguez Erick Roch. (n.d.). *¿Qué es GCC: cómo funciona y para qué sirve?* LovTechnology. Retrieved August 19, 2023, from https://lovtechnology.com/que-es-gcc-gnu-compiler-collection-como-funciona-y-para-que-sirve/

Moya Ricardo. (2016, September 12). *Selección del número óptimo de Clusters*. Jarroba. https://jarroba.com/seleccion-del-numero-optimo-clusters/

Olszewski, D. (2016). Asymmetric k-Means Clustering of the Asymmetric Self-Organizing Map. *Neural Processing Letters*, *43*(1), 231–253. https://doi.org/10.1007/s11063-015-9415-8

Pastrán Ramírez, F. L., & Gongora Aya, S. (2021). *ALGORITMO DE SELECCIÓN Y VALIDACIÓN DEL MÉTODO DE CLUSTERIZACIÓN ÓPTIMO PARA DATOS NO SUPERVISADOS*.

Pérez Ortega, J., Almanza Ortega, N. N., & Romero, D. (2018). Balancing effort and benefit of K-means clustering algorithms in Big Data realms. *PLoS ONE*, *13*(9). https://doi.org/10.1371/journal.pone.0201874

Prabhakaran, S. (2018, October 31). *Procesamiento paralelo en Python: una guía práctica con ejemplos.* Machine Learning Plus. https://www.machinelearningplus.com/python/parallel-processing-python/

Robledano, A. (2019, September 3). *Qué es Python: Características, evolución y futuro | OpenWebinars*. OpenWebinars. https://openwebinars.net/blog/que-es-python/

Rodriguez Erikson. (2019, July 29). *Lenguajes y Framework Disponibles Actualmente para la Programación Paralela - Centro de Computación Paralela y Distribuida*. Universidad Central de Venezuela. https://ccpd.ciens.ucv.ve/lenguajes-y-framework-disponibles-actualmente-para-la-programacion-paralela/

RootStack. (2021, September 7). *Python: características principales de este lenguaje*. Rootstack. https://rootstack.com/es/blog/python-caracteristicas

Tang, J. L., Wang, D., Zhang, Z. G., He, L. J., Xin, J., & Xu, Y. (2017). Weed identification based on K-means feature learning combined with convolutional neural network. *Computers and Electronics in Agriculture*, *135*, 63–70. https://doi.org/10.1016/J.COMPAG.2017.01.001

TIBCO Software Inc. (2022). *¿Qué es el aprendizaje supervisado?* https://www.tibco.com/es/reference-center/what-is-unsupervised-learning#:~:text=El%20aprendizaje%20no%20supervisado%20es,intentar%20entender%20por%20s%C3%AD%20mismo.

Tom Christie. (n.d.). *9: OpenMP - Estructura de Máquinas*. Retrieved December 1, 2022, from https://cc-3.github.io/labs/lab11/

Universidad Nacional de La Plata. (2016). *Estrategia | K-medias*. Universidad Nacional de La Plata. http://163.10.22.82/OAS/Agrupamiento_Kmedias/estrategia.html

Wu, L., & Wen, Y. (2009). Weed/corn seedling recognition by support vector machine using texture features. In *African Journal of Agricultural Research* (Vol. 4, Issue 9). http://www.academicjournals.org/ajar

Zhang, Y., Wang, M., Zhao, D., Liu, C., & Liu, Z. (2022). Early weed identification based on deep learning: A review. *Smart Agricultural Technology*, *3*. https://doi.org/10.1016/j.atech.2022.100123