UNIVERSIDAD TÉCNICA DEL NORTE

FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS

CARRERA DE INGENIERÍA EN SISTEMAS COMPUTACIONALES

TRABAJO DE GRADO PREVIO APROBACIÓN PARA LA OBTENCIÓN

DEL TÍTULO DE INGENIERA EN SISTEMAS COMPUTACIONALES

TEMA:

ESTUDIO DE HERRAMIENTAS DE DESARROLLO EN ENTORNOS PARALELOS PARA APLICACIONES DE AGRICULTURA DE PRECISIÓN.

AUTORA:

ERIKA PATRICIA PONCE GALLEGOS

DIRECTOR:

MSC. MARCO REMIGIO PUSDÁ CHULDE

IBARRA – ECUADOR

2022

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:	0401609508
APELLIDOS Y NOMBRES:	PONCE GALLEGOS ERIKA PATRICIA
DIRECCIÓN:	MIRA - BARRIO SAN NICOLÁS - CALLE: 2 DE FEBRERO Y RICARDO RUALES
EMAIL:	epponceg@utn.edu.ec
TELÉFONO MÓVIL:	0986390387

DATOS DE LA OBRA	
TÍTULO:	"ESTUDIO DE HERRAMIENTAS DE DESARROLLO EN ENTORNOS PARALELOS PARA APLICACIONES DE AGRICULTURA DE PRECISIÓN."
AUTOR (ES):	PONCE GALLEGOS ERIKA PATRICIA
FECHA: DD/MM/AAAA	14 de mayo de 2022
PROGRAMA:	[X] **PREGRADO** [ ] **POSGRADO**
TITULO POR EL QUE OPTA:	INGENIERO EN SISTEMAS COMPUTACIONALES
ASESOR /DIRECTOR:	MSc. MARCO REMIGIO PUSDÁ CHULDE

2. CONSTANCIAS

El autor 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 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 14 días del mes de mayo de 2022

EL AUTOR:

Erika Patricia Ponce Gallegos
0401609508

CERTIFICADO DEL DIRECTOR DE TRABAJO DE GRADO

UNIVERSIDAD TÉCNICA DEL NORTE

FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS

CERTIFICACIÓN DEL DIRECTOR

Por medio del presente yo MSc. Marco Remigio Pusdá Chulde, certifico que la Srta. Erika Patricia Ponce Gallegos, portadora de la cédula de ciudadanía Nro. 0401609508. Ha trabajado en el desarrollo del proyecto de tesis "ESTUDIO DE HERRAMIENTAS DE DESARROLLO EN ENTORNOS PARALELOS PARA APLICACIONES DE AGRICULTURA DE PRECISIÓN", 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.

En la ciudad de Ibarra, a los 11 días del mes de mayo del 2022

Atentamente

MSc. Marco Remigio Pusdá Chulde

TUTOR TRABAJO DE GRADO

DEDICATORIA

A Dios, que me ha guiado en cada uno de los pasos que he dado en mi vida.

A mis padres: **Elsa y Edgar**, quiénes supieron inculcar en mí valores, confianza, respeto hacia los demás, siempre brindándome apoyo en cada decisión, y que nunca me dejaron renunciar en mis metas propuestas.

A mis hermanos: **Javier, Jonathan y Vinicio**, quiénes con consejos y apoyo supieron hacer que cada día sea diferente para no decaer.

A mis abuelitos**: María y Carlos**, quiénes ya descansan a lado de nuestro señor, les doy gracias por tenerme muchos años al cuidado de ellos, supieron apoyarme, inculcarme valores, cuidarme y darme mucho amor gracias por todo el tiempo compartidos juntos.

A **toda mi familia**, quiénes de una u otra manera se hicieron presentes con palabras de aliento siempre me orientaron a seguir adelante y terminar mis metas.

Al **Director de tesis**, quién fue una pieza fundamental para seguir adelante en la meta propuesta gracias por su apoyo y dedicación para culminar.

A **mi amigo confidente Fernando**, quién siempre me apoyo, me brindo amor y con una palabra me hacía más fuerte para no decaer y sentirme triunfante.

Erika Patricia Ponce Gallegos

AGRADECIMIENTO

A todas las personas que de una o de otra manera, aconsejaron y guiaron en la elaboración del presente trabajo.

Facultad de Ingeniería en Ciencias Aplicadas FICA, de la Universidad Técnica del Norte y a cada uno de los señores docentes y personal administrativo.

De manera especial al **Ingeniero Marco Pusdá**, tutor de tesis quien con sus conocimientos me guío para la elaboración del presente trabajo.

A **mis compañeros** quienes a lo largo de la carrera me brindaron la ayuda necesaria cada día en cada paso que se dio.

A grandes amigos como: **Mónica, Mario, Jonathan, Maritza** quiénes fueron parte de las aventuras e hicieron que la estadía en la Universidad sea inolvidable.

A la **Universidad Técnica del Norte** por brindarme sus conocimientos y poder conocer a tan buenos docentes y amigos.

ÍNDICE DE CONTENIDO

INDICE DE FIGURAS

INDICE DE TABLAS

INTRODUCCIÓN

PROBLEMA

Antecedentes

La infraestructura de computación híbrida entre procesadores y tarjetas gráficas es un entorno que permite la ejecución de soluciones paralelas que demandan elevados requisitos de cómputo o que realizan el procesamiento de grandes flujos de datos en tiempo real. En este entorno existen diferentes infraestructuras paralelas para la solución de varios problemas informáticos que aprovechan las potencialidades que ofrece las arquitecturas de alto rendimiento.

En algunas aplicaciones informáticas los algoritmos son secuenciales debido a la simplicidad de control de flujo y consistencia de memoria que ofrece la arquitectura de Von-Neumann. En aplicaciones de visión por computador y análisis de imágenes se necesitan mucho tiempo de procesamiento y otros recursos computacionales a diferencia de la ejecución secuencial (Briceño-Coronado, 2012),(Sucar & Gómez, 2011).

Actualmente es común poseer máquinas con arquitectura multicore (CPU – Central Processing Unity) y many-cores (GPU – Graphic Processing Units), incluso los procesadores utilizados por los dispositivos móviles (Frati, 2015), (Trujillo Rasúa, 2009), por lo que las aplicaciones secuenciales deben adaptarse para aprovechar las altas prestaciones de cómputo de forma que permitan resolver problemas más complejos de un problema informático (Sanz, De Giusti, & Naiouf, 2014)

Situación Actual

La computación de alto rendimiento permite diseñar algoritmos de visión por computador utilizando técnicas de programación de alto rendimiento (algoritmos paralelos) para su implementación sobre plataformas multiprocesador que permitan optimizar el rendimiento de los mismos en el procesamiento de imágenes digitales en diferentes tareas agrícolas como detección de líneas de cultivo, malas hierbas, plagas, enfermedades, personas, animales y obstáculos en diferentes campos de cultivo utilizando dispositivos autónomos en tiempo real.

La gran demanda de gráficos de alta calidad motivó el incremento de la potencia de cálculo transformando a las GPU en potentes coprocesadores paralelos para resolver tareas no

vinculadas con actividades gráficas utilizándolas en programación de aplicaciones de propósito general (Piccoli María F., 2011),(Andrés J. Demski, Andrés L. Di Donato, Santiago F. Maudet, 2015), en algunas aplicaciones el uso de los GPU llegan a ser 30 veces más rápido que con los CPU normales dependiendo de la aplicación(Wang et al., 2014),(Echevarría, 2008); en consecuencia, el modelo de programación en memoria compartida se impuso sobre el modelo de programación secuencial como modelo por excelencia para obtener el máximo desempeño de estas arquitecturas optimizando los algoritmos existentes(Printista et al., 2011), ya que más allá de las mejoras en las arquitecturas físicas, el mayor desafío se centra en cómo aprovechar al máximo su potencia (Naiouf et al., 2012), (Álvarez Pastor, 2017).

La revolución tecnológica de la electrónica y la informática ha influido en diferentes ámbitos de la ciencia, entre ellas a la agricultura, lo que ha llevado al auge de nuevos campos de investigación como la agricultura de precisión y el guiado automático de vehículos agrícolas en tiempo real (Rovira-Más et al., 2003).

La Agricultura de Precisión no es más que la consecuencia del desembarco de las Tecnologías de la Información y la Comunicación en la agricultura o agroTICs. Las agroTICs permiten la adquisición de datos del cultivo y de su medio, el procesado de esos datos convierte en información útil del tratamiento y almacenaje de información para la toma de decisiones y actuaciones. Disponen de información detallada sobre características del cultivo y de su medio (el suelo, el clima, etc.) ayudan al agricultor y al técnico experimentado a tomar decisiones mejor fundamentadas. Todo agricultor sabe que sus campos no producen exactamente lo mismo en toda su superficie. Aun así, no es hasta que se cuantifican esas diferencias en la productividad que se ve la magnitud que puede llegar a tener dicha variabilidad. El tener en cuenta esa variabilidad en el manejo de las parcelas es el objetivo de la Agricultura de Precisión (García & Flego, 2005).

La Agricultura de Precisión está llamada a ser la agricultura del siglo XXI. Sin embargo, su implantación comercial es desigual según el tipo de cultivo y está llevando mucho tiempo. No obstante, debe tenerse en cuenta que los grandes cambios necesitan tiempo para generalizarse(García & Flego, 2005). Sin embargo, la aceptación e introducción de las tecnologías de Agricultura de Precisión está siendo lenta en nuestro país. A modo de ejemplo, sólo el 35% de los aplicadores de fertilizantes son vendidos con equipos de control de peso de precisión, que son absolutamente necesarios para ajustar la cantidad y dirección de la dosificación(Díaz, 2018).

Prospectiva

El presente estudio se centra en estudiar tres herramientas de Programación Paralela (OpenMp, MPI, CUDA) que facilite el aprovechamiento de los recursos de Hardware para el procesamiento de imágenes de cultivos agrícolas en la sistematización de tareas agrícolas, con el fin de indagar y comprobar si las nuevas herramientas se adaptan a la sistematización de tareas agrícolas.

Después de analizar las herramientas, se realizará una comparación de las mismas que nos ayudará a definir la herramienta más apropiada para desarrollar algoritmos paralelos para una aplicación de detección automática de malezas en campos agrícolas; de esta manera se estaría sistematizando tareas agrícolas con lenguajes de alto rendimiento.

Planteamiento del problema

El bajo uso de herramientas de programación de alto rendimiento como (OpenMp, MPI, CUDA) en aplicaciones agrícolas utilizando procesamiento de imágenes, tiende a tener inconvenientes en la utilización de nuevas arquitecturas heterogéneas disponibles en el mercado como (Multicore, GPU, FGPA) para la sistematización de tareas de agricultura de precisión.

OBJETIVOS

Objetivo General

Estudiar las herramientas de desarrollo en entornos paralelos (OpenMp, MPI, CUDA), que permita fortalecer el procesamiento de imágenes en agricultura de precisión.

Objetivo Específicos

- Elaborar un marco teórico respecto al procesamiento de imágenes en agricultura de precisión y las diferentes herramientas de desarrollo en paralelo (OpenMp, MPI, CUDA).
- Definir los parámetros de eficiencia en los algoritmos de procesamiento de imágenes en entornos paralelos.
- Desarrollar un algoritmo de procesamiento de imágenes en un lenguaje de programación paralela, basados en los parámetros de calidad.
- Evaluar Resultados de la investigación propuesta.

Alcance

Se analizó las herramientas de desarrollo para entornos paralelos que permitan sistematizar tareas de agricultura de precisión utilizando procesamiento de imágenes de cultivos de papas o maíz.

Se determinó mediante métricas para verificar la calidad de las herramientas de programación, las cuales será valorados cualitativa y cuantitativamente mediante una comparación de sus principales características., lo que permitirá definir el mejor lenguaje de programación que permita sistematizar tareas agrícolas. Se realizó un análisis de las herramientas de programación paralela en función de parámetros establecidos en las diferentes normas y de esta manera determinar eficiencia o calidad de la herramienta a ser elegida.

Utilizando la herramienta seleccionada en los pasos anteriores se desarrollará un prototipo de procesamiento de imágenes en cultivos de maíz o papa aportando con nuevos hallazgos e ideas innovadoras aplicadas a la agricultura de precisión.

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

ARQUITECTURA PROPUESTA

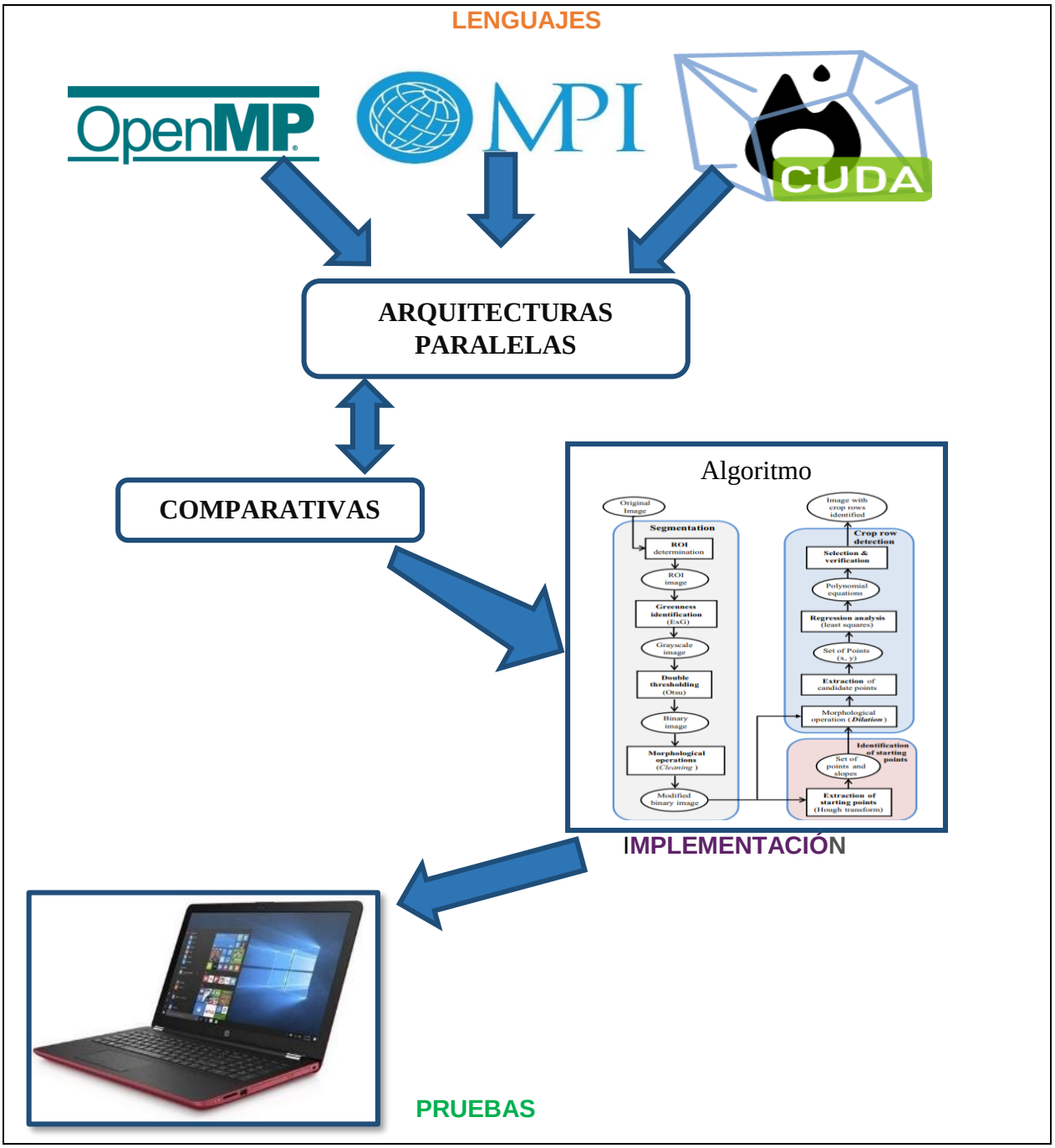


Ilustración 2: Árbol de Problemas
Fuente: Propia

Justificación

Las razones de implementar este proyecto de investigar y recolectar información, ya que no existe un estudio de herramientas de desarrollo en entornos paralelos como son (OpenMp, MPI, CUDA) para sistemas informáticos innovadores aplicados a la agricultura de precisión, como también dar a conocer la importancia de estas herramientas en el ámbito de desarrollo en entornos paralelos aplicada a la agricultura de precisión, en donde se dará a conocer las ventajas que brindan estas herramientas de programación de alto rendimiento.

El presente proyecto tiene un enfoque hacia los objetivos de desarrollo sostenible:

Nº8: Trabajo Decente y Crecimiento Económico:

8.2 Lograr niveles más elevados de productividad económica mediante la diversificación, la modernización tecnológica y la innovación, entre otras cosas centrándose en los sectores con gran valor añadido y un uso intensivo de la mano de obra.

8.a Aumentar el apoyo a la iniciativa de ayuda para el comercio en los países en desarrollo, en particular los países menos adelantados, incluso mediante el Marco Integrado Mejorado para la Asistencia Técnica a los Países Menos Adelantados en Materia de Comercio.

Nº9: Industria, Innovación e Infraestructura:

9.b Apoyar el 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.

9.c Aumentar significativamente el acceso a la tecnología de la información y las comunicaciones y esforzarse por proporcionar acceso universal y asequible a Internet en los países menos adelantados de aquí a 2020(ONU, 2015).

Nº15: Vida de Ecosistemas Terrestres:

15.3 Para 2030, luchar contra la desertificación, rehabilitar las tierras y los suelos degradados, incluidas las tierras afectadas por la desertificación, la sequía y las inundaciones, y procurar lograr un mundo con una degradación neutra del suelo.

15.c Aumentar el apoyo mundial a la lucha contra la caza furtiva y el tráfico de especies protegidas, en particular aumentando la capacidad de las comunidades locales para promover oportunidades de subsistencia sostenibles(Unidas, 2020).

Económico

Esto implica disminuir gastos y tiempos de ejecución en algoritmos secuenciales para el procesamiento de imágenes en tareas agrícolas, teniendo algoritmos más eficientes no se va a necesitar comprar algoritmos de alto consto, sino procesar algoritmos en máquinas normales de visión por computador utilizando técnicas de programación de alto rendimiento sin pérdidas económicas. A la vez se podría beneficiar a la productividad y competitividad de los agricultores, influyendo positivamente al bienestar económico-social de los mismos.

Ambiental

El impacto a nivel ambiental del sistema de automatización de procesos en el uso de la tecnología reduce el daño al ambiente y el mejoramiento de la calidad de vida de los usuarios. El uso de la tecnología en las aplicaciones agrícolas puede reducir las tareas manuales, el costo de producción del cultivo y contribuir a la productividad y competitividad de los agricultores, asegurando los suministros agrícolas.

Social

Al realizar el estudio se pretende contribuir al aseguramiento de la soberanía alimentaria de la provincia de Imbabura utilizando tecnología amigable con el medio ambiente.

En la constitución de la república (2008), el artículo 385 en su literal 3 indica que se debe desarrollar tecnologías e innovaciones que impulsen la producción nacional, eleven la eficiencia y productividad. La política 5.6 del Plan Nacional de Desarrollo 2017-2021 indica que el estado debe promover la investigación, la transformación, la capacitación, el desarrollo y la transferencia tecnológica, la innovación y el emprendimiento para impulsar el cambio productivo en el procesamiento de imágenes de tareas agrícolas (Siteal, 2018).

Las Políticas Agropecuarias al 2025 en la Problemática 7, menciona la Baja investigación, innovación y transferencia de tecnología aplicada a necesidades productivas; por lo que el lineamiento 7.4 del Ministerio de Agricultura y Ganadería (MAGAP) señala Identificar y desarrollar

tecnologías basadas en la investigación, para diversificar las actuales formas de producción, y los productos del multisector, en el uso de nuevas tecnologías agropecuarias.

Justificación Teórica

El estudio de las herramientas (OpenMp, MPI, CUDA) de desarrollo en entornos paralelos que se aplicará para la agricultura de precisión es parte de las herramientas tecnológicas hacen que tengan mayor valor institucional y tiendan a ser pioneras en la región en donde se encuentran desempeñando su función.

Justificación Metodológica

Utilizar la metodología adecuada que nos permita llevar a el análisis de los resultados que se va a obtener en el campo de la agricultura precisión. El método adecuado para contextualizar este proyecto ha sido seleccionado, considerando la investigación que se va a realizar y sea decido utilizar lo que es el método aplicado el cual es el óptimo para mejorar la calidad de vida y contribuir con la construcción del conocimiento nuevo.

CAPÍTULO 1

Marco teórico

1.1 Agricultura de Precisión (AP).

La definición más simple de AP la establece como un grupo de tecnologías que permiten la aplicación de insumos agrícolas, tales como fertilizantes, semillas, pesticidas, fungicidas, etc, en forma variable dentro de una parcela de terreno, de acuerdo con los requerimientos o potencial productivo de varios sectores homogéneos (Moreno et al., 2019).

La AP es considerada como un Sistema Alternativo Sostenible, utilizado en la producción agropecuaria, en el cual se emplean diferentes métodos o herramientas tecnológicas. Con el propósito de recopilar información sobre lo que sucede o puede suceder en los suelos y en los cultivos y, con dicha información, poder proceder a la toma de decisiones, que permita el incremento de los rendimientos, la disminución de los costos de producción y la reducción de los impactos ambientales(Balcázar Guerrero Mario, 2011), (Martínez-Rodríguez., 2016).

La AP busca aumentar la productividad de las actividades agrícolas a nivel mundial, en este marco de desarrollo la navegación autónoma es un pilar fundamental ya que la mayoría de las actividades agrícolas implican desplazamientos extensos de personas o vehículos (Moreno et al., 2019). Actualmente se requiere que la producción agrícola minimice los impactos ecológicos negativos de sus actividades y sea competitiva en mercados globalizados cada vez más exigentes en precios y calidades (Leiva, 2003).

La AP parte de un concepto novedoso que busca optimizar el manejo de la producción agrícola teniendo en cuenta la variabilidad del agroecosistema. De esta manera se establecen estrategias para usar los recursos necesarios en la cantidad requerida, en el sitio adecuado y en el momento oportuno (Leiva, 2003). Tal y como se le conoce en Europa y USA, su desarrollo se basa en tecnologías electrónicas, de telecomunicación y de informática, especialmente adaptado para la aplicación diferenciada de insumos según las necesidades del cultivo o del suelo, en aspectos ambientales y económicos, que definen la contribución de la AP al desarrollo de una agricultura sostenible y competitiva., En países tropicales, dada la heterogeneidad de sus agroecosistemas, el concepto de manejo de la variabilidad adquiere plena vigencia, pero se requieren adaptaciones tecnológicas apropiadas al medio (Jimenez et al., 2013).

La AP es una estrategia de gestión que utiliza las TIC (Tecnologías de la Información y la Comunicación) para obtener información con el fin de apoyar la toma de decisiones, considerando los aspectos ambientales y económicos para optimizar las tareas del agricultor y brindar productos de calidad al cliente. La aplicación de AP en la agricultura puede reducir el tiempo dedicado a las actividades manuales, evitar el uso indiscriminado de productos químicos, aumentar los costos de producción, el deterioro del suelo y la contaminación ambiental. Actualmente la AP aprovecha diversos avances tecnológicos como la visión computacional, arquitecturas heterogéneas (Multicore, GPU, FGPA) y técnicas de inteligencia artificial (Machine learning, Deep learning), que ha permitido sistematizar una variedad de actividades agrícolas, tales como detección de enfermedades, recuento de plantas e identificación de malezas, plagas de insectos aplicados en diferentes cultivos (Pusdá et al., 2019), (Pajares-Martinsanz, Gonzalo; De La Cruz-García, 2008)

1.2 Sistematización de Tareas Agrícolas.

En los últimos tiempos se acoge la AP, como una moderna herramienta agrícola, la cual inicia con la mecanización y sistematización de la producción, enfocándose en la nutrición del suelo, la fertilización del cultivo, la productividad, las ventajas económicas y la minimización del impacto ambiental producto de las labores de campo, como también, la gestión y manejo de semillas, la distribución de plantación, control de siembra y administración de los recursos, en cultivo(Rodríguez González, 2020).

Las posibilidades de usar herramientas modernas vinculadas a los sistemas de información geográfica permiten participar en distintas etapas del proceso productivo de la agricultura de precisión, ya por medio de imágenes satelitales, mapas de prescripción de siembra variables, procesamiento de monitores de rendimiento, etc. Muchos de estos servicios se realizan sin la necesidad de ir al campo, es decir, desde la computadora se puede realizar diferentes procedimientos e intercambio de información(Schiaffino, 2018).

La identificación y clasificación de enfermedades a partir de las imágenes de la planta, en la etapa temprana, es una de las áreas de investigación más relevante en el campo de la informática y la agricultura, en un 75 % las personas de poblaciones rurales se dedican a la agricultura ya que es un medio de sustento para sus familias (Shah et al., 2016). Una enfermedad en las plantas da como resultado una gran pérdida económica para los agricultores cada año.

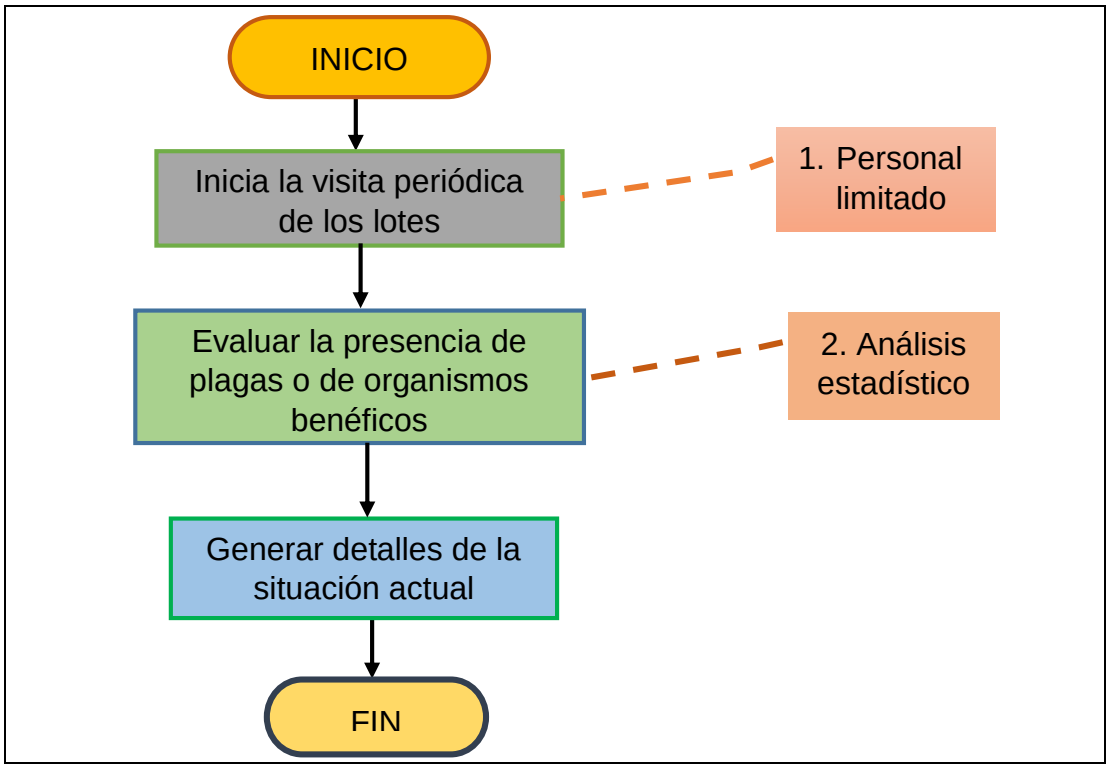
Por lo tanto, un diagnóstico rápido, oportuno y preciso de la enfermedad evita la pérdida de producto y también mejora la calidad de la producción.(Shrivastava et al., 2019)

El uso de las imágenes satelitales, juntamente con los sistemas de información geográfica (SIG) y los datos generados por la agricultura de precisión, están en la base de los servicios, en los cuales la generación, sistematización e interpretación se vuelven centrales. El uso de modernas tecnologías asociadas a los sistemas de información geográfica (SIG), los sistemas de posicionamiento global (GPS), la teledetección y la programación, ofrecen servicios de consultoría, desarrollo de sistemas, tecnología geoespacial aplicada, sistemas de análisis de datos, visitas y soluciones a campo, capacitación y soporte. La combinación y sistematización de la información, a partir de algoritmos y modelos predictivos, permite realizar mapas de rendimiento proyectados a una escala de 10 metros x 10 metros(Schiaffino, 2018).

En el trabajo de (Jimenez et al., 2013), establece modelos del comportamiento de variables fenológicas de las regiones cultivables, de un lote completo o de una planta individual, mediante el uso del procesamiento digital de imágenes adquiridas por sensores remotos satelitales. El algoritmo desarrollado relaciona la información espectral de un sistema de adquisición de imágenes en cultivos agrícolas, pues emplea los análisis básicos de procesamiento de imágenes para reconocer patrones espacio- temporales de lo que ocurre con las plantas en algunas etapas fenológicas, valorar su estado nutricional y detectar plagas y enfermedades.

Para brindar una definición de lo que es un Sistema de Monitoreo en el contexto de la Agricultura de Precisión hay que referirse al significado de las palabras que lo componen. Por tanto, sistema no es más que un conjunto de dispositivos que se relacionan entre sí de manera ordenada. Luego monitoreo es supervisar o controlar las entradas, los procesos intermedios y las salidas para identificar fortalezas y debilidades. Además de formular propuestas de acciones prácticas a implementar y tomar los pasos necesarios para alcanzar los resultados esperados (Otero Barrera, 2019). El monitoreo de cultivos consiste en visitar periódicamente los lotes de cultivos, evaluando la presencia de plagas, de organismos benéficos y la interacción que ambos tienen con las áreas cultivadas en base a una cierta cantidad de muestras, obsérvese en la Figura 1, con las cuales el productor pueda tomar una mejor decisión(Balcázar Guerrero Mario, 2011).

Figura 1: Diagrama de Flujo del Proceso del Monitoreo de cultivos
Fuente: (Balcázar Guerrero Mario, 2011).



El proceso de monitoreo de cultivos presenta dos debilidades, las que son descritas en la Tabla 1.

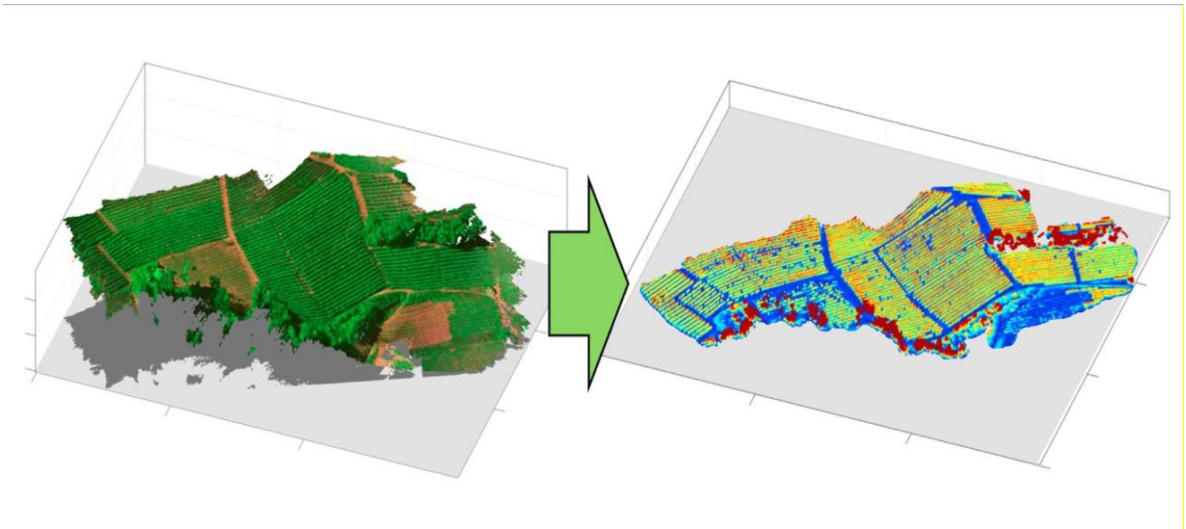
Tabla 1: Debilidades del Proceso de Monitoreo de Cultivos

DEBILIDADES	EXPLICACIÓN
1. Personal limitado	El personal encargado de realizar las visitas periódicas toma muestras de los cultivos y hace el análisis a partir de las muestras obtenidas en el campo. Los períodos entre visitas pueden ser prolongadas en el caso de tener campos de cultivos extensos.
2. Análisis estadísticos	De las muestras obtenidas se realiza estimaciones sobre el estado actual del cultivo. Se genera detalles y se toman las decisiones basadas en estas estimaciones.

Nota. En la Tabla 1 *se muestra las dos debilidades, encargadas de observar y analizar el estado del cultivo para tener un monitoreo adecuado.*

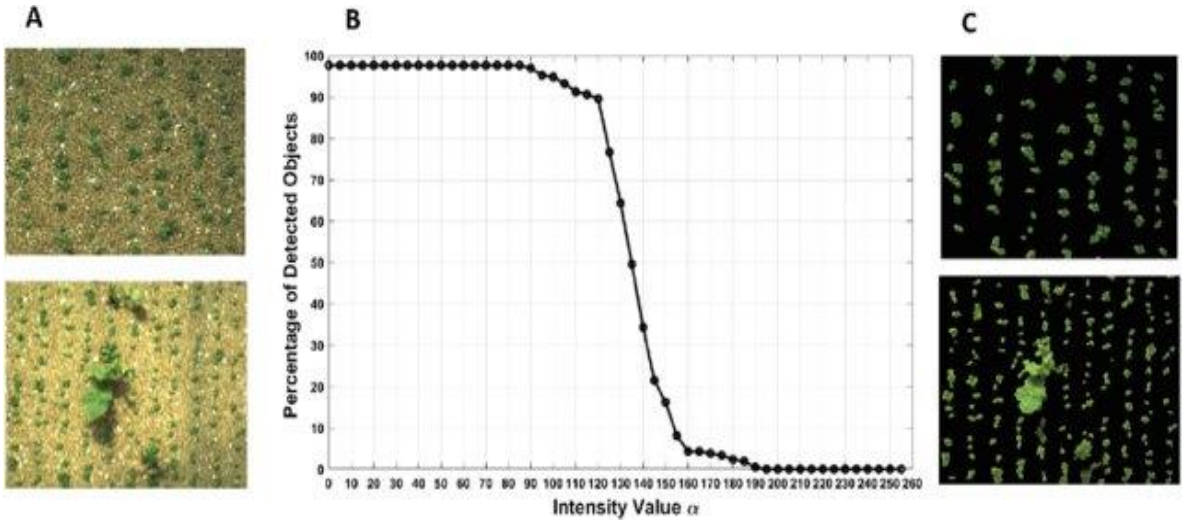
Una gestión eficaz de los procesos de viticultura de precisión se basa en sólidos procedimientos de seguimiento de cultivos y, en un futuro próximo, en una máquina autónoma para la gestión automática de cultivos en un lugar específico. En este contexto, la detección exacta de viñedos a partir de mapas de nubes de puntos en 3D, generados a partir de imágenes multiespectrales de vehículos aéreos no tripulados (UAV) (Comba et al., 2018), (Pérez-Ortiz et al., 2016), (Ahmad et al., 2018). Como se presenta en la Figura 2 las imágenes tridimensionales juegan un papel crucial, tanto para lograr datos mejorados de detección remota como para administrar la ruta y el funcionamiento de los vehículos no tripulados.

Figura 2: Mapa de nube de puntos (3D)
Fuente: (Comba, Biglia, Aimonino, & Gay, 2018)



La visión artificial y el procedimiento de imágenes se utilizan ampliamente en aplicaciones vegetales, donde la detección y la ubicación precisa de las malas hiervas se puede obtener mediante visión por computadora aplicando varios métodos de procesamiento de imágenes, basado en seleccionar características capaces de discriminar plantas de cultivo y malezas de imágenes RGB. Lo primero a describir es el protocolo de adquisición de imágenes, realizado en condiciones reales al aire libre, con diversas condiciones de iluminación. Las imágenes RGB son capturadas por una cámara de 20 megapíxeles con una resolución espacial de 5120 x 3840 pixeles, montado con un objeto de 35mm. Una distancia típica entre la cámara y la hilera de cultivo es de 120 cm., después, se entrena y se prueba el clasificador de SVMs basado en Kernel polinomial dentro de características de forma e intensidad extraídas de objetos segmentados, y evaluados en el desempeño de la clasificación con respecto a la configuración diferente para el clasificador de SVMs, Figura 3 (Ahmad et al., 2018).

Figura 3: Imágenes adquiridas en condiciones de luz diferentes.
Fuente:(Ahmad et al., 2018)



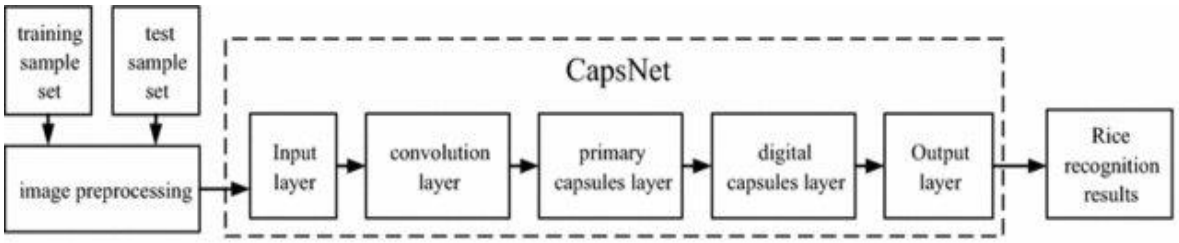
1.3 Redes Neuronales en Agricultura de Precisión

Las redes neuronales (RNA) son sistemas computacionales utilizados como herramientas eficaces para realizar tareas de clasificación, reconocimiento de imágenes, detección/localización de objetos, generación de imágenes y otras tareas relacionadas con la visión por computador obteniendo resultados adecuados para la toma de decisiones necesarias en los problemas agrícolas utilizando visión artificial (Moreno Díaz, 2020).

Un tipo de red neuronal utilizado en el estudio de (Li et al., 2019), para monitorear el crecimiento del producto y prevenir las enfermedades y plagas mediante imágenes capturadas por un dron fue una red de capsulas (CapsNet) de cinco capas, la misma que está construida para reconocer imágenes y preprocesan mediante el método de ecualización de histograma en imágenes en escala de grises y mediante el algoritmo de superpíxeles. La función de CapsNet es realizar un análisis inverso de imágenes de arroz, Figura 4.

- Una capa de entrada,
- Una capa de convolución (la cual transforma dos funciones en una nueva),
- Una capa de cápsulas primaria,
- Una capa de cápsulas digitales,
- Una capa de salida.

Figura 4: Proceso de Reconocimiento de Imágenes.
Fuente: (Li et al., 2019)



Las Redes Neuronales Convolucionales (CNN), son algoritmos que pertenecen al campo de Machine Learning. Estos toman imágenes como entrada, detectan una serie de características dentro de cada imagen y en base a estas características son capaces de realizar diversos procesamientos(Moreno Díaz, 2020), (Kerkech et al., 2018). En los últimos tiempos, las técnicas de Deep Learning han obtenido un alto rendimiento en el reconocimiento de imágenes, segmentación de imágenes, reconocimiento de voz, procesamiento de lenguaje natural, reconocimiento de emociones, sistema de recomendaciones, entre otros, obteniendo excelentes resultados (Shrivastava et al., 2019).

La aplicación de las CNN son diversas en cultivos y tareas diferentes, para una rápida identificación de insectos se utiliza CNN, para la extracción de características, una red de propuesta de región (ROI) y un mapa de puntuación sensible a la posición para la detección de objetos. Las diferentes formas biológicas de los insectos de varias especies se diferencian en el tamaño, forma y brillo de color, entre las cuales las diferencias de forma se encuentran en las antenas, cabeza y placa posterior (Zhichao et al., 2020).

El enfoque de Deep Learning ha demostrado resultados impresionantes, sin embargo, este enfoque necesita una cierta cantidad de datos, por lo que esta tarea requiere mucho tiempo. Sin embargo, se propone un método de aprendizaje totalmente automático que utiliza CNN con recopilación de conjuntos de datos de entrenamiento no supervisados para la detección de malezas a partir de imágenes de UAV (Bah et al., 2018).

1.4 Arquitecturas Paralelas.

Se debe tomar muy en cuenta las características del computador en el cual se va a trabajar para la programación paralela, la combinación de varias configuraciones nos puede llevar a distintas arquitecturas en las cuales de acuerdo con el número de procesadores o el tamaño de memoria se puede elegir la más adecuada para resolver una aplicación. En la informática la

Arquitectura Paralela da un enfoque constructivo del conocimiento del lenguaje basado en la computación paralela que es el uso de múltiples recursos computacionales, los cuales requieren de una gran capacidad de procesamiento y de espacio de memoria, debido a complejas operaciones que se deben realizar, esto se distingue de la computación secuencial en que varias operaciones pueden ocurrir simultáneamente. La sincronización en los programas en paralelo necesita la coordinación de procesos e hilos, para una ejecución correcta, asociados de manera que los procesos o hilos intercambian información dependiendo de cómo está organizada la memoria en el hardware, el mapping se hace por el sistema en tiempo de ejecución, también puede ser manipulado por el programador. El balanceo de carga distribuye cantidades equitativas de trabajo entre todas las tareas de modo que se mantengan ocupadas todo el tiempo. El paralelismo en sí son varias actividades que tiene lugar al mismo tiempo, implicando que se puede tener varios procesos corriendo cada uno en un procesador, 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. La arquitectura de Von Neumann guarda instrucciones de los procesos y datos en una memoria electrónica, compuesta por, memoria, unidad de control, unidad aritmética lógica y entrada/ salida (Daniel, 2004).

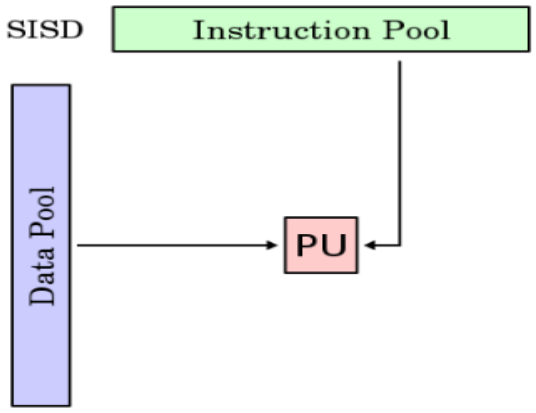
El paralelismo a nivel de bit se habla cuando se aumenta el tamaño de la palabra del procesador (tamaño de la cadena de bits a procesar). Este aumento reduce el número de instrucciones que tiene que ejecutar el procesador en variables cuyos tamaños sean mayores a la longitud de la cadena, el paralelismo a nivel de instrucción consiste en cambiar el orden de las instrucciones de un programa y juntarlas en grupos para posteriormente ser ejecutados en paralelo sin alterar el resultado final del programa, el paralelismo a nivel de datos, aquí cada procesador realiza la misma tarea sobre un subconjunto independiente de datos, en el paralelismo a nivel de tareas, cada hilo realiza una tarea distinta e independiente de las demás. Los hilos se distinguen de los procesos por ser independientes, los cuales llevan bastante información de estados e interactúan solo a través de mecanismos de comunicación dados por el sistema. Por otra parte, los hilos comparten otros recursos directamente. En diferentes sistemas operativos muchos proveen facilidades para los hilos, es más rápido cambiar de un hilo a otro dentro del mismo proceso, que cambiar de un proceso a otro. Los datos en paralelo, conocido como PGAS (Partitioned Global Address Scape) una serie de tareas trabajan de manera colectiva en la misma estructura de datos, realizando la misma operación, pero cada tarea trabaja en una partición diferente(Peña & Mosquera, 2017)

En la clasificación de la arquitectura de computadores, se encuentra la taxonomía de Flynn (Flynn & Rudd, 2004) quien tomó dos aspectos a consideración: el flujo de instrucciones y el flujo de datos, de acuerdo a la multiplicidad de cada uno, se clasifican en los siguientes:

1.4.1 SISD Single Instruction, Single Data

SISD (Instrucción Única, Flujo de Datos Único) Un elemento de procesamiento tiene acceso a un único programa y a un almacenamiento de datos, en cada paso, el elemento de procesamiento carga una instrucción y un único flujo de datos para ser ejecutada, siendo el resultado guardado de vuelta en el almacenamiento de datos, donde SISD es el computador secuencial convencional, de acuerdo al modelo de Von Neumann(Daniel, 2004). También son computadoras de una CPU; especialmente usadas para la ejecución de programas secuenciales en donde las instrucciones están bajo el mando de una única unidad de control (Flynn & Rudd, 2004).

Figura 5: Procesamiento de SISD.
Fuente: (Daniel, 2004)

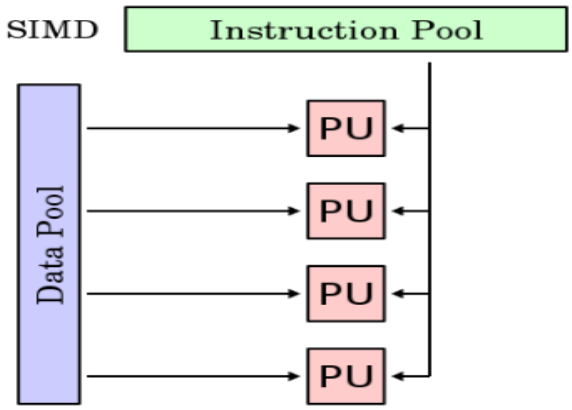


1.4.2 SIMD Single Instruction, Multiple Data

SIMD (Instrucción Única, Flujo de Datos Múltiples)., Incluye procesadores vectoriales, así como procesadores paralelos, en los que se encuentra múltiples elementos de procesamiento, en el que cada cual tiene acceso privado a la memoria de información (compartida o distribuida). Sin embargo, hay una sola memoria de programa, desde la cual una unidad de procesamiento especial obtiene y despacha instrucciones. En cada paso, cada unidad de procesamiento obtiene la misma instrucción y carga desde su memoria privada un elemento de información y ejecuta la instrucción en dicho elemento, aplicada en paralelo por todos los elementos de proceso a diferentes elementos de información siendo un proceso muy eficiente que puede ser aplicado en

aplicaciones multimedia y algoritmos de gráficos de computadora(Peña & Mosquera, 2017). Se caracteriza por tener un único flujo de instrucciones y múltiple flujo de datos, su arquitectura se caracteriza por contar con varias CPU, estás pueden ejecutar el mismo flujo de instrucciones en distintos datos; las SIMD incluyen los computadores vectoriales, la unidad de control se encarga de enviar la orden para ejecutar la instrucción sobre diferentes grupos de datos que provienen de varios flujos (Flynn & Rudd, 2004).

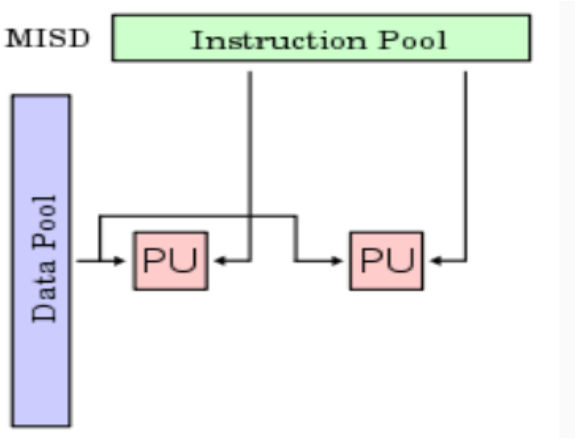
Figura 6: Procedimiento de SIMD.
Fuente: (Peña & Mosquera, 2017)



1.4.3 MISD Multiple Instruction, Single Data

MISD(Instrucción Múltiple, Flujo de Datos Únicos), En MISD hay múltiples elementos de procesamiento, que cada cual tiene memoria privada del programa, pero tiene acceso común a una memoria global de información. Cada paso, cada elemento de procesamiento obtiene la misma información de la memoria, ejecutadas en paralelo, siendo un modelo muy restrictivo y no se ha usado en ningún computador de tipo comercial (Daniel, 2004). Es decir, tienen múltiple flujo de instrucciones y un único flujo de datos, para ello las diferentes unidades de procesamiento trabajan sobre el único flujo de datos con diferentes instrucciones, ninguna de las arquitecturas implementan este tipo de computadora(Flynn & Rudd, 2004).

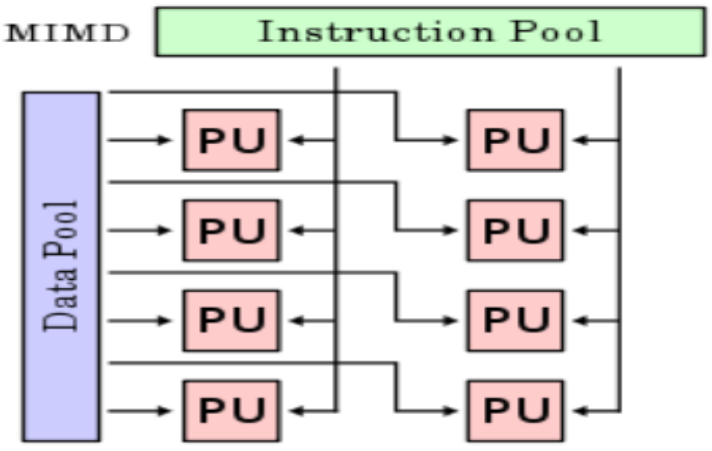
Figura 7: Procedimiento de MISD.
Fuente:(Daniel, 2004)



1.4.4 MIMD Multiple Instruction, Multiple Data

MIMD (Instrucción Múltiple, Flujo de Datos Múltiple) En las que se encuentra múltiples unidades de procesamiento, en cada una tiene tanto instrucciones como información separada. Cada elemento ejecuta una instrucción distinta en un elemento de información. Los elementos de proceso trabajan asíncronamente. Los clusters son ejemplos del modelo MIMD(Daniel, 2004). Este tipo de computadoras tienen un múltiple flujo de instrucciones y un múltiple flujo de datos, las unidades de instrucciones son independientes y cada una maneja su propio flujo de datos, en MIMD existen dos tipos de computadoras: las de multiprocesadores, son arquitecturas con memoria compartida; y las Multicomputadoras que manejan la arquitectura con memoria distribuida (Flynn & Rudd, 2004).

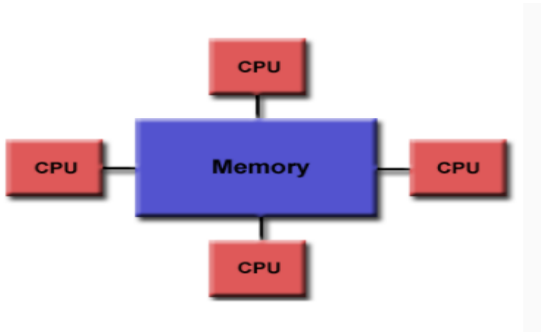
Figura 8: Procedimiento de MIMD.
Fuente:(Daniel, 2004)



1.4.5 UMA Uniform Memory Access

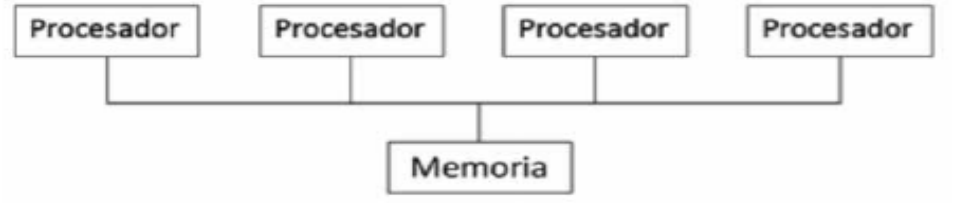
UMA (Acceso Uniforme a Memoria) Los procesos comparten un espacio de memoria en común en los cuales escriben y leen de manera asíncrona, de las que no es necesario especificar cómo se comunican los datos de las tareas, se usan semáforos o locks para controlar el acceso a la memoria compartida. Representado hoy por las máquinas Symmetric Multiprocessor (Daniel, 2004)(SMP)(Symmetric Multi- Processing), con procesadores idénticos de igual acceso y tiempos de acceso a la memoria, si un procesador actualiza una ubicación en memoria compartida, todos los demás procesadores saben sobre la actualización, esto es llamado coherencia del cache (Acosta et al., 2012).

Figura 9: Procedimiento de UMA.
Fuente: (Daniel, 2004)



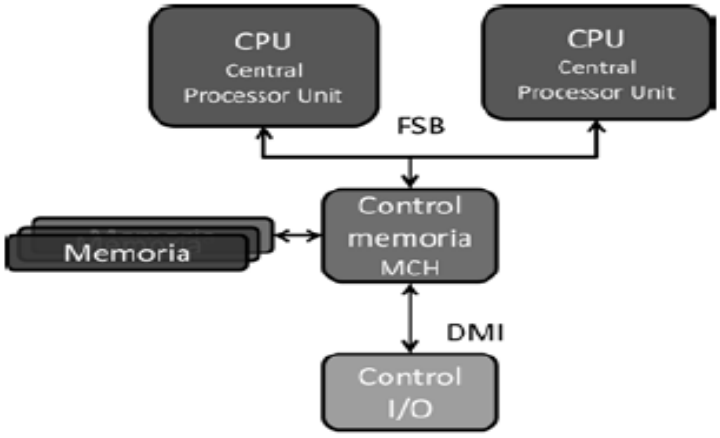
SMM-Shared Memory Model, es una abstracción del modelo de multiprocesamiento centralizado, que se compone por una colección de procesadores en donde cada uno tiene acceso a una memoria local compartida en la que se almacenan variables que pueden ser usadas por todos los procesos. El método de acceso a memoria es llamado UMA (Uniform Memory Access), también conocido como SMP (Symmetric Multi- Processing). Básicamente, describe que cada procesador de un arreglo de procesadores idénticos tiene los mismos tiempos de acceso a una memoria compartida equidistante a cada núcleo, mediante un bus de interconexión (Acosta, Segura, & Ospina, 2012)(Acosta et al., 2012).

Figura 10: Esquema de Hardware para SMM
Fuente: (Acosta et al., 2012)



El hardware en SMM está basado en FSB (front-side bus) que a su vez es el modelo acceso a memoria usado en UMA (Uniform Memory Access). Consta de un controlador de memoria (MCH) al que se conecta a la memoria general, las CPU interactúan con el MCH cada vez que necesitan acceder a memoria (Acosta et al., 2012).

Figura 11: Hardware para el modelo UMA
Fuente:(Acosta et al., 2012)

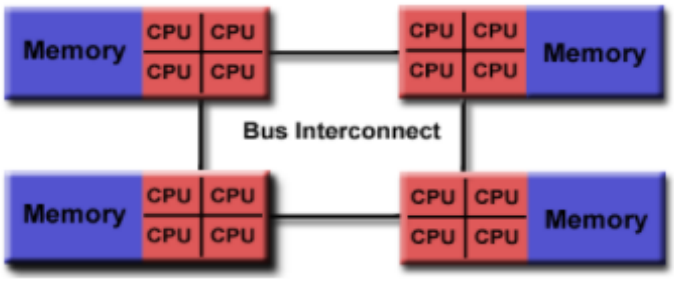


A diferencia de un modelo de memoria compartida (SMM), es un programa basado en MPM los procesos típicos permanecen activos en todo momento de la ejecución del programa; por el contrario, en el SMM el número de conexiones activas es dinámico (se pueden crear y destruir tantos procesos como sean necesarios) (Acosta et al., 2012) (Acosta, Segura, & Ospina, 2012).

1.4.6 NUMA Non-Uniform Memory Access

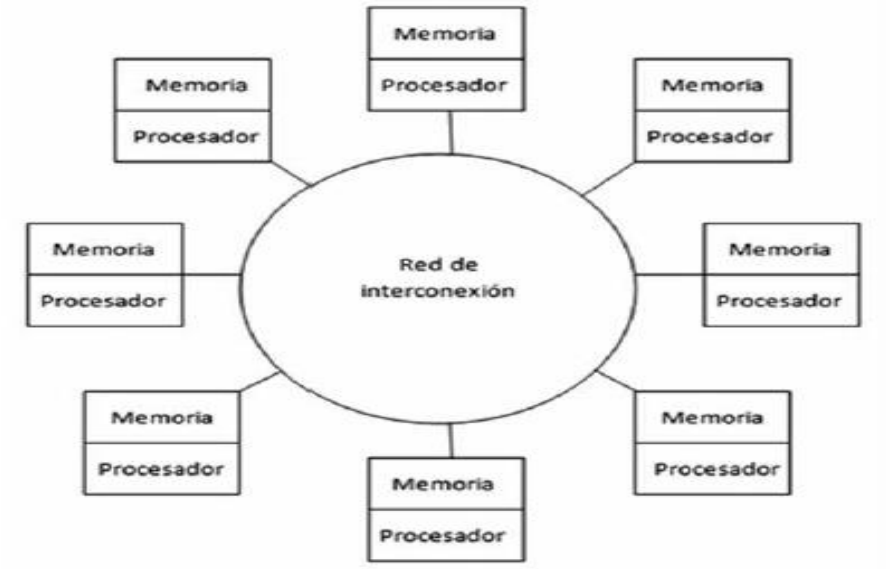
NUMA (Acceso a Memoria no Uniforme),Creado por la vinculación física de dos o más SMP, puede acceder directamente a la memoria de otro SMP, no todos los procesadores tienen igual tiempo de acceso a toda la memoria, el acceso a la memoria es más lento, si se mantiene la coherencia del caché(Daniel, 2004).

Figura 12: Procedimiento de NUMA.
Fuente:(Daniel, 2004)



MPM - Modelo de Paso por Mensajes, es aprovechado por los múltiples CORES, que se encuentran en equipos de nueva tecnología. Con MPM el hardware, como diferentes procesadores (dentro o fuera de una máquina en el caso de clusters) a través de una red de interconexión donde cada uno tiene acceso directo y exclusivo a la información almacenada en su propia memoria local (Acosta et al., 2012).

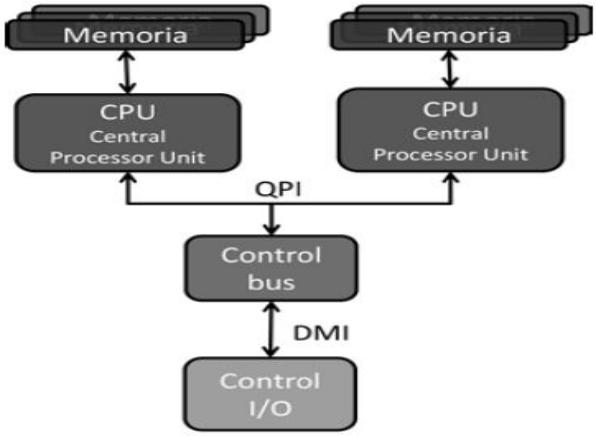
Figura 13: Esquema del modelo de paso por mensajes MPM
Fuente:(Acosta et al., 2012)



MPM maneja un tipo de acceso a memoria denominado NUMA (Nom-Uniform Memory Access). En el que se describe los procesos que se comunican por la red escalable de interconexión, manejan un espacio específico en memoria para hacer la interacción y comunicación con los demás procesos por medio de mensajes. En el hardware, cada memoria está conectada a una CPU, en lugar de estar conectada a un controlador de memoria (modelo

NUMA), en cuanto, los controladores de memoria se encuentran integrados a las distintas CPU que están en la red. (Daniel, 2004)Además, las CPU están conectadas con un I/O hub, permitiendo un ordenamiento en los datos y reduciendo los problemas de tráfico (Acosta, Segura, & Ospina, 2012)(Acosta et al., 2012).

Figura 14: Hardware para el modelo NUMA.
Fuente: (Acosta et al., 2012)



La comunicación y la transferencia de datos entre procesos se hace usando mensajes estructurados que poseen información esencial, referente a los datos, las funciones, las variables de estado y mensajes de sincronización de las memorias locales de los procesadores. Para este proceso se usan una serie de funciones definidas en la librería(Peña & Mosquera, 2017).

Los mensajes de MPM contienen:

- La variable en la que reposan los datos que se envían.
- La cantidad de datos que se envían.
- El proceso receptor, el que recibe el mensaje.
- Los datos que se esperan recibir por parte del receptor.
- El tipo de dato que se envía.
- El proceso emisor del mensaje.
- La ubicación o espacio en memoria (puntero en C) donde se almacenarán los datos.

La memoria distribuida se basa en múltiples procesadores con su propia memoria física privada, las tareas operan solo con información local y se necesita de la comunicación para obtener información remota, hay cuatro implementaciones tales como: múltiples sistemas operativos, Middlewares, clusters y grids (Daniel, 2004).

1.5 Lenguaje de Programación.

Las herramientas de programación paralela son: lenguajes, API, frameworks que permiten aprovechar el potencial del hardware de forma paralela. Sin embargo, la paralelización automática de un programa secuencial sigue siendo un problema por resolver a pesar de los muchos esfuerzos que se han hecho en los últimos 40 años(Peña & Mosquera, 2017). Los lenguajes de programación paralelos se pueden agrupar en niveles de acuerdo con la arquitectura subyacente, en donde el nivel de abstracción de programación paralela es el más alto y mantiene toda la complejidad del paralelismo a nivel de hardware explotando su uso de forma eficiente mediante librerías y herramientas de alto nivel(Briceño-Coronado, 2012). El modelo de memoria distribuida proporciona un modelo de datos, en el que cada procesador tiene su propio espacio de direcciones de ubicaciones de memoria inaccesibles para otros procesadores. La elección del modelo de datos determina cuando los procesadores se comunican entre sí, en cambio un modelo de memoria compartida, se comunican leyendo y escribiendo ubicaciones compartidas, pero en un modelo de memoria distribuida, se comunican enviando y recibiendo mensajes. A continuación, se da a conocer tres modelos diferentes de la programación en paralelo, basado en MPI, OpenMP y CUDA, generando información básica y oportuna para definir la mejor. La programación en paralelo ha surgido como una nueva alternativa para aprovechar más eficientemente los recursos de los computadores(Acosta et al., 2012).

Las arquitecturas de computadora multinúcleo y muchos núcleos son más atractivas para lograr la ejecución de alto rendimiento de aplicaciones científicas a costos relativamente bajos. Para las arquitecturas de computadora, la vectorización eficiente es crucial para lograr un desempeño satisfactorio(Stpiczynski, 2018).

1.5.1 MPI (Message Pasaing Interface)

MPI es una librería estándar que se usa en programas que aprovechen arquitecturas de múltiples procesadores. El desarrollo de MPI comienza en los años ochenta por las diferentes casas productoras de equipos de cómputo, en 1993 se crea la primera versión como un estándar abierto por un gran número de organizaciones, principalmente desarrollada por el Centro de investigación en Computación paralela de Williamsbrug, libre de un desarrollo que contenía una librería de paso por mensajes diseñada para sistemas de computadores intercomunicados con conexiones en paralelo denominada PVM V3 (Parallel Virtual machine). Enseguida, en mayo 1994 se ajustaron algunos complementos de la librería de PVM y nace la primera versión de MPI

1.0 (Message Passing Interface). Una característica importante de MPI es que permite trabajar con grupos de procesadores definidos según el programador lo disponga mediante objetos virtuales denominados comunicadores, que permiten distribuir los recursos según la tarea a realizar. Con la librería MPI, sólo se puede declarar una única vez el ambiente en paralelo (sección comprendida entre las funciones MPI_Init y MPI_Finalize) y que todo el código que este dentro de la zona se ejecutará en simultáneo por todos los procesos(Acosta et al., 2012). Sus lenguajes de especificación son C, C++ y Fortran. Existen implementaciones para Python, Ocalm, Java, .NET y PHP.

Cualquier librería, MPI cuenta con una serie de funciones que permiten llevar a cabo tareas específicas dentro del ambiente en paralelo, la sintaxis general de una función de MPI es del tipo:

MPI_función. [argumento1, argumento2, …] \n

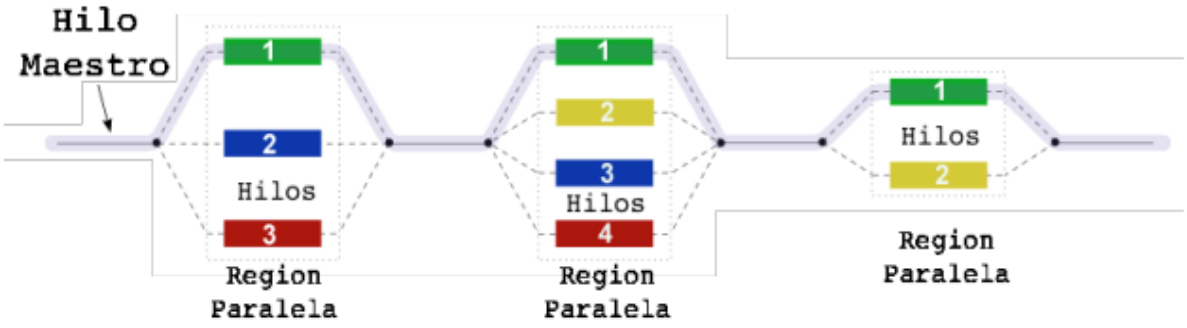
Dentro de los grupos de funciones para MPI se destacan algunas como:

- *Funciones de control de flujo:* permiten crear y establecer parámetros de la sección en paralelo como número de procesos a usar, el comunicador, los ID de los procesos de la aplicación, etc.
- *Funciones para el manejo de grupos y comunicadores:* facilitan la conformación de los grupos de procesadores.
- *Funciones de administración de recurso*.
- *Funciones de comunicación:* permiten la interacción (enviar y recibir información) entre diferentes procesos. Según el número de procesos presentes en la comunicación ésta se clasifica en punto a punto y multipunto.
- *Funciones para comunicación punto a punto:* implican la interacción de dos procesos exclusivamente (maestro y esclavo), que según el tipo de petición para establecer la conexión se dividen en método bloqueante y no bloqueante.
- *Funciones para comunicación multipunto:* interactúan con múltiples procesos simultáneamente, el uso de ellas requiere que el desarrollador tenga claro el recurso con el que cuenta.

1.5.2 OpenMP (Open Multi-Processing)

OpenMP es una API para programación multiproceso de memoria compartida en múltiples plataformas, denominada como una interfaz portable para computadores o redes que soportan SMM, adoptada como un estándar informal en 1997 por científicos que buscaban generalizar los programas basados en SMM (Abdul Khalib et al., 2020),(Daniel, 2004). Su modelo de ejecución se basa en memoria compartida multihilos. Se considera el sucesor más sofisticado de Posix Threads (Portable operating system interface for Linux, es una librería para sistemas operativos). Dispone de directivas que apoyan al programador para convertir algoritmos secuenciales a paralelos de forma eficiente, introduciendo el código necesario para lanzar al mismo tiempo múltiples hilos. La gran portabilidad es una característica de MPI debido a que está soportado para C, C++ y Fortran, disponible para sistemas operativos como Solaris, AIX, HP-UX, GNU/Linux, MAC OS, y Windows(Sterling et al., 2018).

Figura 15: Lanzamiento de múltiples hijos en regiones paralelas.
Fuente:(Ignacio Molina Cuquerella, 2013)



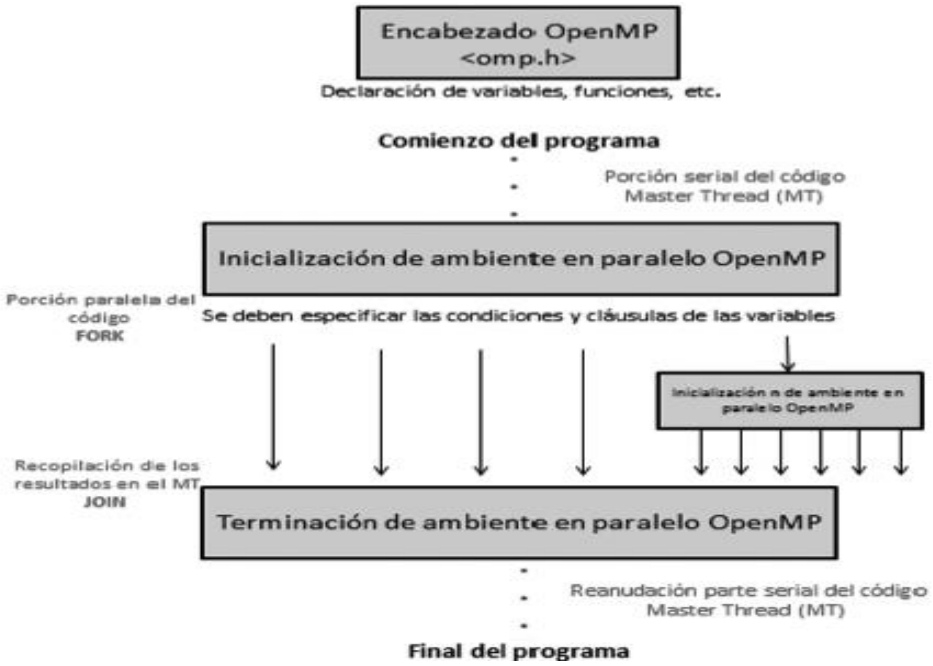
Los compiladores y herramientas de desarrollo Intel C/C++ ofrecen muchas extensiones basadas en lenguajes que se pueden utilizar para simplificar el proceso de desarrollo de programas paralelos de alto rendimiento. OpenMP es un estándar bien conocido para la programación paralela de memoria compartida de alto nivel. OpenMP en una extensión de compilador C/C ++ y Fortran que le da al programador la capacidad de agregar paralelismo en un código fuente sin la necesidad de reescribir el código de manera significativa (Abdul Khalib et al., 2020). Con el fin de optimizar la utilización de la CPU, las directivas del compilador se utilizan para especificar la ejecución paralela de partes seleccionadas de programas informáticos (es decir, bucles y secciones de código). Las directivas también se utilizan para proporcionar construcciones de sincronización explícitas. Dichas directivas pueden contener cláusulas que definan el intercambio de datos entre subprocesos. Además, OpenMP consta de un pequeño

conjunto de rutinas de biblioteca y variables de entorno que influyen en el comportamiento del tiempo de ejecución. Las versiones más recientes del estándar definen construcciones de programación más avanzadas, como tareas y compatibilidad con SIMD (Stpiczynski, 2018).

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 más las alternativas de programación (Acosta et al., 2012). 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, determinan la forma de trabajar de los procesadores asignados a una aplicación.
- #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.

Figura 16: Estructura general de un programa en OpenMP
Fuente:(Acosta et al., 2012)



Una importancia de OpenMP es que permite administrar el recurso en rutinas que contienen cálculos iterativos, mediante la distribución de iteraciones de los ciclos en los diferentes

threads (hilos) que maneja la aplicación mediante funciones especiales denominadas constructores. Los constructores se combinan con una cláusula especial llamada Schedule que contiene parámetros que definen la forma de distribución de las iteraciones. OpenMP, consiste en crear nuevas secciones paralelas dentro de los mismos threads formando una especie de ramificación que facilitan el procesamiento de grandes volúmenes de información (Acosta et al., 2012).

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. Con respecto a la tecnología multinúcleo, se ha realizado trabajos para evaluar el rendimiento de diferentes lenguajes de programación paralelos, construcciones, herramientas y bibliotecas. Sin embargo, apenas se está conociendo el rendimiento de un código paralelo con diferentes tipos de programación OpenMP y tamaños de fragmentos variados en un procesador multinúcleo de memoria compartida. La falta de pautas sobre la asignación del tipo de programación y el tamaño del fragmento adecuados podría hacer que un usuario de OpenMP deje la configuración como predeterminada. Dado que el tipo de programación decide cómo se divide la carga de trabajo entre los subprocesos, mientras que el tamaño del fragmento define la granularidad del trabajo, es necesario determinar el tipo de programación junto con el tamaño de fragmento apropiado de un código paralelo. No obstante, aún no se ha identificado cuánta contribución dan estas directivas en términos de mejora del rendimiento de diferentes procesadores multinúcleo (Abdul Khalib et al., 2020).

1.5.3 CUDA (Computing Unified Device Architecture)

CUDA, es una API de computación en paralelo, que incluye un compilador y una serie de herramientas de desarrollo creadas por NVIDIA para implementar algoritmos en sus GPU. Permite programador mediante una variación del lenguaje de programación C/C++, Java y Fortran y existen múltiples aplicaciones en Python. CUDA intenta aprovechar el gran paralelismo, y el alto ancho de banda de la memoria en las GPU en aplicaciones con un gran coste aritmético frente a numerosos accesos a memoria principal(Daniel, 2004).

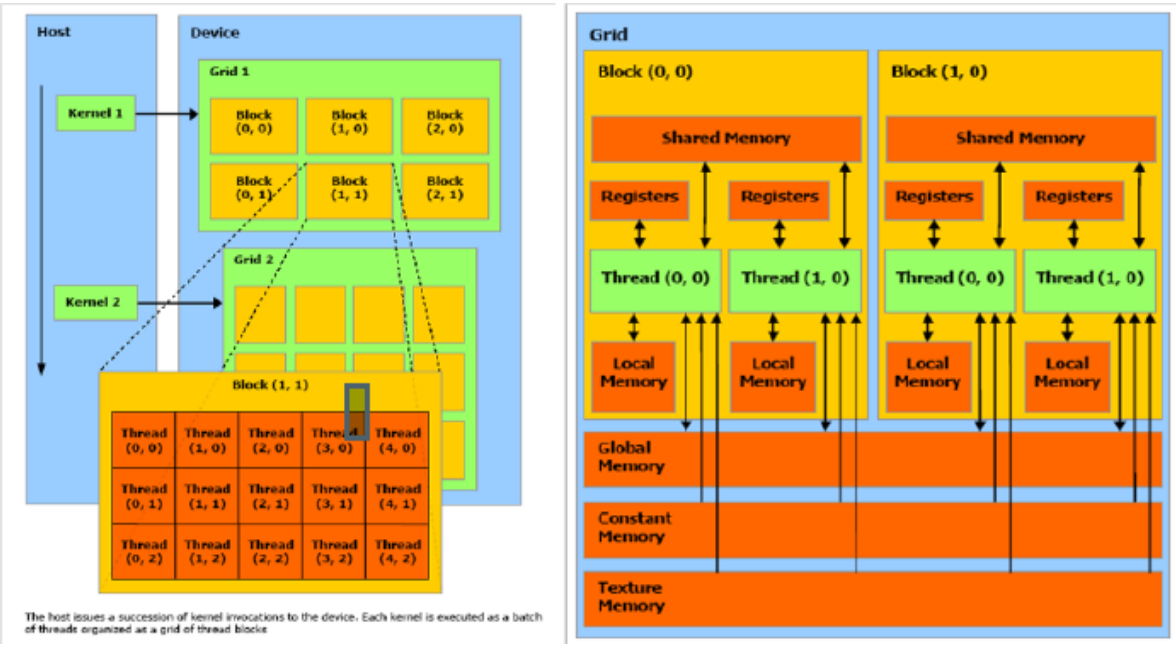
CUDA, nace a partir de la necesidad de desarrollar nuevas alternativas para el procesamiento de grandes volúmenes de información, aplicables en el desarrollo de videojuegos, reproductores de video y simulaciones complejas (meteorología y distribución térmica). Poco después surge Nvidia como una compañía especializada en el desarrollo de tecnologías para el

procesamiento de información, que a mediados del 2000 lanzó la primera GPU (Graphic Processing Unit) que se caracterizaba por la gran capacidad para hacer operaciones de punto flotante, esta hizo que la comunidad se volcará a la tecnología. En 2003 se desarrolla el primer modelo de programación extendido a C usando una GPU que fue sometido a algunos ajustes de hardware y software, generando así en 2006 lo que hoy se conoce como CUDA (Acosta et al., 2012)

Arquitectura de CUDA

Un compilador genera código ejecutable para el dispositivo CUDA. La CPU considera un dispositivo CUDA como un coprocesador de múltiples núcleos. Los subprocesos CUDA acceden a datos de varios espacios de memoria durante su ejecución (Pan et al., 2019). Hay tres niveles de estructura de memoria los cuales se presentan en la siguiente **¡Error! No se encuentra el origen de la referencia.¡Error! No se encuentra el origen de la referencia.**:

Figura 17: Estructura de Memoria CUDA.
Fuente:(Pan et al., 2019)



El primer nivel consiste en la memoria local privada y el registro que son propiedad de cada hilo. El segundo nivel es la memoria compartida a la que pueden acceder todos los

subprocesos del bloque. El tercer nivel es la memoria global, la memoria constante y la memoria de textura a la que todos los hilos pueden acceder(Pan et al., 2019).

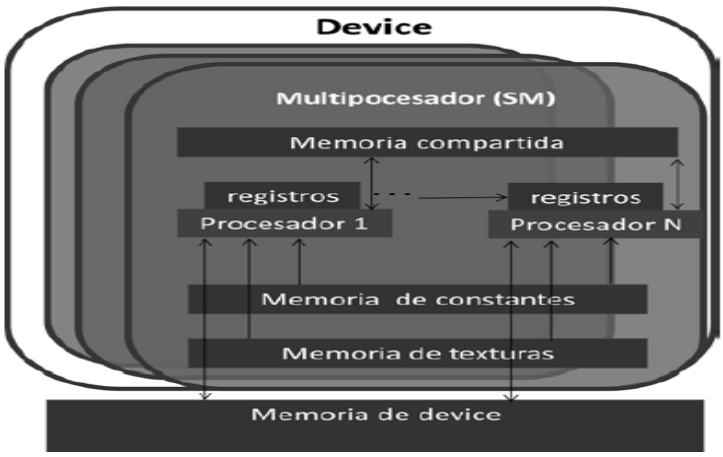
CUDA es un estándar que migra de un modelo centralizado a un modelo de cooprocesamiento en el que se dividen los datos ordenadamente entre la CPU y la GPU. El hardware de CUDA se compone por la CPU de un equipo principal o *host* y una GPU de un dispositivo externo o *device*(Acosta et al., 2012). El nivel de código de programación con las GPU se obtienen ventajas como:

- Código compacto: una instrucción define N operaciones.
- Reduce la frecuencia de los saltos en el código.
- No se requiere de hardware adicional para detectar el paralelismo.
- Permite la ejecución en paralelo asumiendo N flujos de datos paralelos.
- No maneja dependencias.
- Usa patrones de acceso a memoria continua.

Un SM cuenta con varias memorias, que pueden ser introducidas en cualquier parte del código:

- Memoria compartida de lector /escritura.
- Memoria de constantes.
- Memoria de texturas

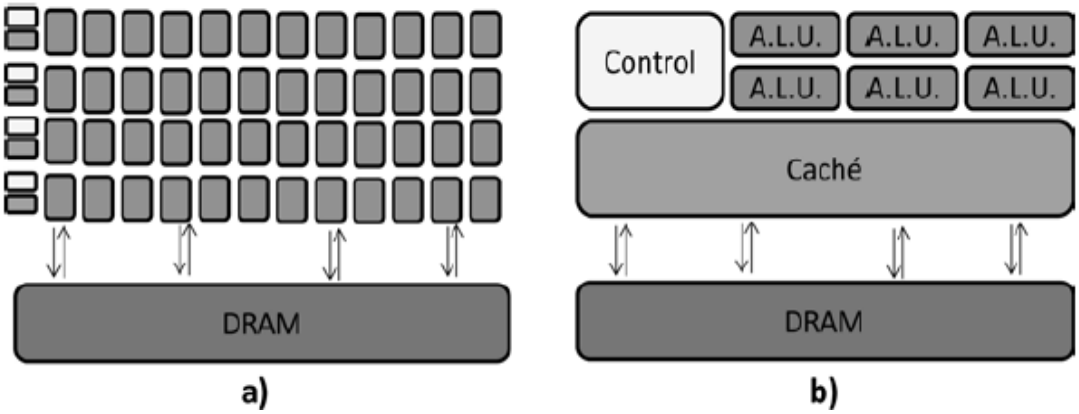
Figura 18: Esquema de Memoria de una GPU
Fuente:(Acosta et al., 2012)



Por el contario una CPU cuenta con bloques únicos para el manejo de caché, para el control y para el manejo de la RAM; y en muchos casos múltiples ALU (dependiendo de las

características). Por su parte, la GPU cuenta con una unidad de control o Kernel y una caché por cada ALU dispuesta (según el modelo de la tarjeta de video), lo que permite aumentar el poder de cómputo de manera considerable.

Figura 19: Composición de a) la GPU, vs. b) La CPU.
Fuente:(Acosta et al., 2012)



Técnicamente existen grandes diferencias entre una CPU y la GPU como, Tabla 2:

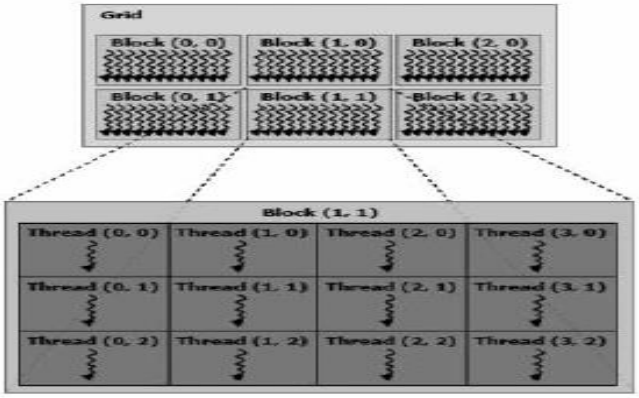
Tabla 2: Diferencias CPU vs. GPU.
Fuente:(Acosta et al., 2012)

CPU	GPU
▪ Procesados Escalar	▪ Procesador Vertical
▪ Acceso aleatorio a memoria	▪ Acceso secuencial a memoria
▪ Modelo de programación muy conocido	▪ Modelo de la programación muy poco conocido
▪ 20GB/s de Ancho de banda	▪ 100 – 150 GB/S de Ancho de banda
▪ 0,1 Tflops	▪ Consumo de 10 Gflops/watt
▪ Consumo de 1 Gflop/watt	

Modelo de Programación

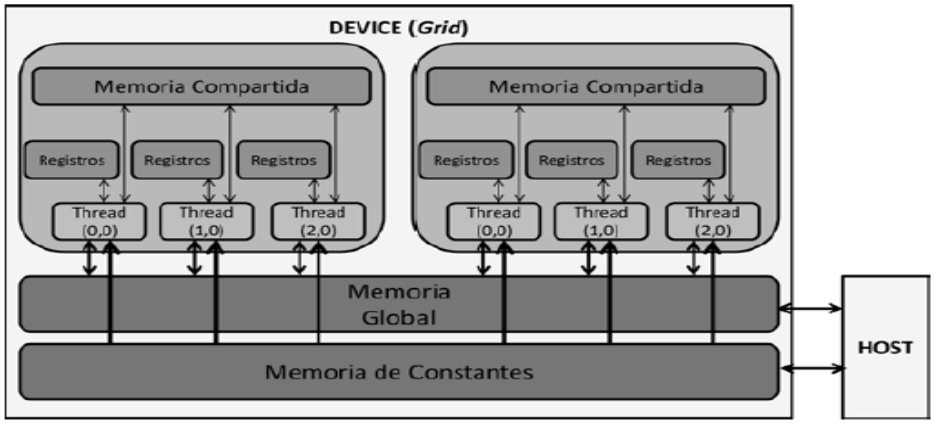
CUDA maneja un concepto de programación en el que se puede ejecutar código de forma secuencial mediante la CPU o paralela usando la GPU. El kernel hace la paralelización del código, asignando tareas específicas a los diferentes threads dispuestos para la aplicación. Para definir la cantidad de *threads* se maneja una jerarquía donde los *threads* se agrupan en estructuras de 1, 2 o 3 dimensiones denominados bloques, los bloques a su vez, se agrupan en un *grid* de bloques que también puede ser de 1, 2 o 3 dimensiones (Ignacio Molina Cuquerella, 2013), (Acosta et al., 2012).

Figura 20: Esquema de división de threads
Fuente: (Acosta, Segura, & Ospina, 2012)



Para la comunicación entre threads en CUDA se usan una serie de memorias que tienen jerarquía en donde los resultados de las operaciones son comunes entre *los threads*. Cada bloque maneja una memoria compartida visible por todos los *threads* del bloque, que dura el tiempo que dure activo el bloque, por otra parte, los bloques también manejan otra memoria común entre ellos mismos y el grid denominada memoria global (DRAM) (Acosta, Segura, & Ospina, 2012).

Figura 21: Mapa de memoria del device
Fuente: (Acosta, Segura, & Ospina, 2012)



La estructura de un código basado en CUDA contiene:

- Declaración de funciones y kernel (función que realiza el host).
- Copiado de variables de la memoria del Host al Device.
- Paralelización y ejecución de las tareas por realizar en la GPU.
- Ordenamiento de los resultados de las operaciones paralelas.
- Copiado de los resultados de la memoria del Device al Host.

- Liberación de memorias.

Los tres diferentes modelos de programación en paralelo que se ha mencionado, es posible resaltar que el costo de la implementación de los modelos depende mucho de los recursos con que se cuente. En el caso de que se tenga con una red de máquinas como los clúster o salas de computadores es más fácil utilizar MPM, ya que no es necesario realizar cambios significativos a nivel de hardware y software para su implementación (Ignacio Molina Cuquerella, 2013).

La aplicabilidad de los modelos está ligada al nivel de granularidad de las tareas, es decir, en el caso de que una tarea se pueda dividir en muchas partes se obtienen mejores resultados usando MPI. En el caso de que se tenga el recurso (GPU) CUDA sería la mejor opción ya que, ofrece gran cantidad de Core para el procesamiento de las tareas y libera la CPU. En el caso de que las tareas sean más densas de nada sirve tener muchos recursos, porque no todos están trabajando sobre la tarea, en este caso sería mejor usar OpenMP porque el direccionamiento de información es más eficiente al interactuar directamente con la memoria compartida. La paralelización de tareas usando la librería MPI está basada en el modelo de paso por mensajes que conceptualmente es más asimilable porque las tareas se dividen según el número de procesadores físicos con los que cuenta el sistema. MPI permite un control más riguroso sobre los procesos que llevan a cabo las tareas dentro del flujo, debido a que es posible administrar quién ejecuta una tarea en específico, dando como resultado un nivel de paralelización más modificable que facilita el desarrollo de la aplicación. Además, la estructura del código es la más semejante de las tres librerías a la usada a un código de C (Acosta, Segura, & Ospina, 2012), (Ignacio Molina Cuquerella, 2013).

Cuando el diseño de una aplicación requiere varias secciones secuenciales y paralelas, OpenMP facilita el trabajo, gracias a su modelo de programación que combina la interacción entre rutinas secuenciales y paralelas que lo diferencia de MPI en donde sólo se puede declarar un único ambiente en paralelo. El mayor rendimiento de CUDA se obtiene cuando se realiza una operación puntual que no requiere de una lógica muy extensa dentro del kernel, puesto que CUDA está diseñado para manejar una gran cantidad de datos durante un corto tiempo para optimizar el procesamiento. Para las tres librerías de programación en paralelo mencionadas, es posible afirmar que cualquiera de las tres opciones optimiza los tiempos de procesamiento de un código respecto a la ejecución secuencial del mismo, de acuerdo con esto, CUDA es la librería que presenta los tiempos más reducidos de procesamiento cuando se ejecutan tareas que implican el manejo de un gran volumen de datos (Ignacio Molina Cuquerella, 2013).

1.5.4 MATLAB

Millones de ingenieros y científicos en todo el mundo utilizan MATLAB® para analizar y diseñar los sistemas y productos que transforman nuestro mundo. El lenguaje de MATLAB, basado en matrices, es la forma más natural del mundo para expresar las matemáticas computacionales. Puede ejecutar sus análisis en conjuntos de datos de mayor tamaño y expandirse a clusters y nubes. El código de MATLAB se puede integrar con otros lenguajes, lo que le permite implementar algoritmos y aplicaciones en sistemas web, empresariales o de producción(Matlab, 2022).

MATLAB, es el lenguaje de cálculo técnico desarrollado por MathWorks, es un entorno de programación para el desarrollo de algoritmos, análisis de datos, visualización y cálculo numérico. Simulink es un entorno gráfico para simulación y diseño basado en modelos de sistemas dinámicos multi-dominio e integrados. MathWorks es una multinacional especializada en el desarrollo de software de ingeniería. Es un lenguaje de cuarta generación. Fue fundada en 1984 por Jack Little y Cleve Moler, que identificaron la necesidad entre ingenieros y científicos de un entorno de computación más potente y productivo más allá de los proporcionados por lenguajes como Fortran y C. Combinaron su experiencia en matemáticas, ingeniería e informática para desarrollar MATLAB. Así mismo, Mathworks pone a su disposición más de ochenta productos para tareas especializadas como el análisis de datos y el procesamiento de imágenes (A. et al., 2008).

Algunos beneficios:

- Facilita el desarrollo de la simulación científica gracias a la biblioteca incorporada.
- Alta eficiencia e codificación y productividad, ya que no requiere un compilador para su ejecución.
- Ideal para desarrollar aplicaciones de investigación científica.
- Es una plataforma independiente.

Los usos de MATLAB incluyen cálculos matriciales, desarrollo y ejecución de algoritmos, creación de interfaces de usuarios (UI) y visualización de datos. El entorno informático numérico de paradigmas múltiples permite a los desarrolladores interactuar con programas desarrollados en diferentes lenguajes, lo que permite aprovechar las fortalezas únicas de cada idioma para diversos fines (Reclu IT, 2020).

MATLAB se utiliza generalmente para tareas, como: Procesamiento de la señal, optimización de funciones, diseño del sistema de control, procesamiento de imagen y audio, aprendizaje automático y aprendizaje profundo (PUCP, 2019).

Algunas características principales de Matlab son las siguientes:

MATLAB proporciona métodos de cálculo numérico para análisis de datos, desarrollo de algoritmos y creación de modelos. Para ello se incluyen funciones matemáticas que utilizan librerías optimizadas por procesador para conseguir una ejecución rápida de los cálculos de vectores y matrices.

- Se puede comportarse como una calculadora o como un lenguaje de programación.
- Combina muy bien el cálculo y el trazado gráfico.
- Es relativamente fácil de aprender, puede interactuar con diferentes lenguajes.
- Se interpreta (no se compila), los errores son fáciles de corregir.
- Está optimizado para ser relativamente rápido cuando se realizan operaciones matriciales.
- Tiene algunos elementos orientados a objetos.
- Acceso y procesamiento de datos, accede archivos de imágenes (.jpg, PNG) archivos de audio (.mp) y datos en tiempo real de JDBC / ODBC.
- Entorno interactivo: proporciona una GUI (interfaz gráfica de usuario), también cuenta con herramientas para la depuración y el desarrollo de cualquier software. Importar y exportar archivos en MATLAB a través de la GUI.
- Simulink: es una función en la que se puede modelar los sistemas de control y ver su comportamiento en tiempo real.
- Interfaz de programación de aplicaciones (API) de MATLAB: consta de una API extensa. A través de esta API, podemos vincular nuestros programas C / C++ directamente a MATLAB. MATLAB se puede utilizar como herramienta de cálculo y análisis.

Machine Learning, aprendizaje profundo, y la visión de computadora se puede realizar en MATLAB. También se puede crear e interconectar capas de una red neuronal profunda. Podemos construir bucles de entrenamiento personalizados y capas de entrenamiento con diferenciación automática. Para el aprendizaje automático, puede usar clústeres y ruido en los datos. Para la

visión por computadora, se puede hacer seguimiento de objetos, reconocimiento de objetos, reconocimiento de gestos y procesamiento de nubes de puntos 3D (Acervo, 2022).

El lenguaje de MATLAB es un lenguaje de matriz/array o arreglo de alto nivel para la visualización de datos bidimensionales y tridimensionales, procesamiento de imágenes, animación y gráficos de presentación, el cual contiene funciones, estados de flujo de control, estructuras de datos, in/out, y otras características comunes de la programación orientada a objetos(Reclu IT, 2020).

MATLAB ® es una plataforma de programación diseñado específicamente para los ingenieros y científicos, contiene tareas de computación, visualización, las cuales se puede manejar de forma fácil, en comparación con otro software de programación y las soluciones o resultados se expresan en notación matemática conocida, para analizar y diseñar sistemas y productos que transforman nuestro mundo. MATLAB es un lenguaje basado en matrices que permite la expresión más natural de las matemáticas computacionales(Acervo, 2022).

Entre las tareas más comunes que están:

- Matemáticas y computación.
- Desarrollo de algoritmos.
- Modelado, simulación y prototipos.
- Análisis, exploración y visualización de datos.
- Gráficos científicos y de Ingeniería.
- Desarrollo de aplicaciones, incluyendo la construcción de la interfaz Gráfica de Usuario.

Matlab es un programa para realizar cálculos numéricos con vectores y matrices, también se puede trabajar con números escalares (tanto reales como complejos), con cadenas de caracteres y con otras estructuras de información más complejas. Matlab es un lenguaje de alto rendimiento para cálculos técnicos, es al mismo tiempo un entorno y un lenguaje de programación. Uno de sus puntos fuertes es que permite construir nuestras propias herramientas reutilizables. Se puede crear fácilmente nuestras propias funciones y programas especiales (conocidos como M-archivos) en código Matlab, los podemos agrupar en Toolbox (también llamadas librerías): colección especializada de M-archivos para trabajar en clases particulares de problemas. Matlab, a parte del cálculo matricial y álgebra lineal, también puede manejar polinomios, funciones, ecuaciones diferenciales ordinarias, gráficos, etc (PUCP, 2019).

Ventajas de MATLAB

- Interfaz fácil de usar: con las funciones que desea utilizar está a un clic de distancia.
- Una gran base de datos de algoritmos incorporada: MATLAB cuenta con numerosos algoritmos incorporados, y solo tiene que llamarlos en su código.
- Amplia visualización y procesamiento de datos: podemos procesar una gran cantidad de datos en MATLAB y visualizarlos usando gráficos y figuras.
- Fácil depuración de códigos: muchas herramientas integradas como analizador y depurador para el análisis y la depuración de códigos escritos en MATLAB.
- Fácil manipulación simbólica: operaciones matemáticas y herramientas de manipulación simbólicas en MATLAB.

Desventajas de MATLAB

- MATLAB es lento ya que es un lenguaje interpretado, los programas de MATLAB no se convierten a lenguaje máquina, sino que se ejecutan mediante software externo, por lo que a veces puede ser lento.
- No crea archivos de SALIDA en MATLAB.
- No permite realizar funciones en un solo archivo .m como en otros lenguajes de programación.
- A veces, los mensajes de error no son muy informativos, se debe averiguar el error por sí mismo.

1.5.5 PYTHON

Es el caso en particular del lenguaje de programación Python, el cual proporciona una inmensa cantidad de posibilidades desde el programa (software) para el manejo de muchos sensores, protocolos de comunicación y datos en grandes cantidades. Así mismo, existen en el mercado programas de uso comercial que proveen las funcionalidades de un sistema SIG, con el cual se evidencia que las herramientas en términos de programas (software) tienen un amplio rango de opciones que permiten obtener gran cantidad de mediciones en tiempo real y que logran ser almacenadas en lo que se conoce como la nube (Tovar Soto et al., 2019).

Python es un lenguaje multiplataforma que puede ser programado mediante POO (Programación Orientada a Objetos), es de código abierto. Su sintaxis es simple, y fácil de

entender, por lo que ahorra tiempo y recursos, al contrario de otros lenguajes como Java, C, PHP y JavaScript en la ausencia de paréntesis y la no incorporación de punto y coma al final de cada instrucción, entre otros aspectos. Es un lenguaje muy completo, con soporte para bases de datos relacionales, no relacionales u orientados a objetos, capaz de incorporar ORM´s, con frameworks de desarrollo muy complejos (Yeeply, 2017).

Python es un lenguaje versátil que puede tener múltiples aplicaciones, como, la Inteligencia Artificial, gracias a bibliotecas como Keras o TensorFlow. También es de utilidad para aplicaciones de Big Data, por cuanto a bibliotecas de procesamiento de datos (Yeeply, 2017).

Además, soporta el uso de módulos y paquetes, y los programas pueden ser diseñados en un estilo modular y el código puede ser reutilizado en varios proyectos. Una vez desarrollado un módulo o paquete, se puede escalar para su uso en otros proyectos y es fácil de importar y exportar. Uno delos beneficios más importantes de Python es que tanto la librería estándar como el intérprete están disponibles gratuitamente, tanto en forma binaria como en forma de fuente (Centro de formación técnica para la industria, 2018).

También, se puede utilizar para procesar texto, mostrar números o imágenes, resolver ecuaciones científicas y guardar datos. El código lo convierte a (código de byte). Ya que este código no puede ser entendido por la CPU. Así que se necesita el intérprete llamado Máquina Virtual Python (PVM) que ejecuta los códigos de bytes (Centro de formación técnica para la industria, 2018).

Este lenguaje de programación también se emplea en el desarrollo web, sobre todo por sus frameworks Django o Flask, creación de software y procesamiento de datos, entre otros propósitos.

Python fue creado por Guido van Rossum a principios de los años 90 en Holanda. Su nombre proviene de la afición de su creador por el grupo de humoristas británicos de Monty Python. Siendo desarrollado hoy en día por la Python Software Foundation. Es un lenguaje similar a Perl, pero con una sintaxis muy limpia y que favorece un código legible(Yeeply, 2017).

Características del lenguaje de programación Python

- Python es un lenguaje intérpretado. Se considera sucesor del lenguaje ABC y usa conceptos de otros lenguajes como Modula -3, lisp, entre otros.
- Python no obliga a los programadores a adoptar un estilo particular de programación.

- Se puede instalar en varias plataformas: Windows, Linux, etc. Con menores cambios puede trasladarse entre ellas.
- Es software libre y de código abierto con licencia GPL (General Public License). Se puede instalar modificar y distribuir proporcionando al código fuente.
- El código escrito en Python es legible, sin marcas para definir bloques como en otros lenguajes. No requiere símbolos de fin de líneas.
- Python es un lenguaje de tipado dinámico: conecta un método y un nombre de variable durante la ejecución del programa.
- Es un lenguaje seguro. Realizar chequeo dinámico de tipos de datos y de índices.
- Python es un lenguaje de propósito general. Algunas aplicaciones son: Aprendizaje de la programación, desarrollo de prototipos, computación científica y matemática, estadística y optimización, desarrollo de interfaz visual, desarrollo de aplicaciones web, interfaces para manejo de bases de datos. Educación y juegos, desarrollo de software.

Python se basa en los lenguajes C y C++ y tiene sus raíces en el sistema operativo UNIX. Python existe desde años atrás, pero fue a mediados del 2000 cuando salió al mundo del desarrollo web, con el auge de sitios como Wordpress.

Las ventajas de Python

- Su curva de aprendizaje es muy corta, se aprenderá rápidamente.
- Se obvia los puntos y coma.
- Si está aprendiendo inglés, Python es el lenguaje ideal para practicarlo.
- Es muy poderoso y por su versatilidad puede ser usado en casi cualquier área de la tecnología.

1.6 Comparativa de las Lenguajes de Programación.

Tabla 3: Comparativa de Herramientas de Programación
MPI, OpenMP, CUDA, MATLAB, PYTHON

COMPARATIVA DE LAS HERRAMIENTAS DE PROGRAMACIÓN					
BIBLIOTECA	**MPI**	**OpenMP**	**CUDA**	**MATLAB**	**PYTHON**
Significado	Message Passing Interface	Open Multi-Processing	Computer Unified Device Arquitecture	Matrix Laboratory	

Se utiliza	Para arquitecturas de memoria distribuida	Para arquitecturas de memoria compartida	Para las unidades de proceso gráfico de NVIDIA.	Para analizar y diseñar los sistemas y productos que transformen el mundo.	Para desarrollar aplicaciones de todo tipo, es de código abierto.
Plataforma de cómputo	Clúster Beowulf	Clúster SMP	GPU	Unix, Windows, MacOS y GNU/Linux.	Linux, Mac OS, MS-DOS, OS/2 y Windows.
A que hace referencia cada una de las herramientas	Es un estándar de programación de aplicaciones paralelas que define la sintaxis y semántica de un conjunto de operaciones de paso de mensajes que permiten explotar el paralelismo en arquitecturas de múltiples procesadores.	Es un interfaz de programación de aplicaciones compuesto por un conjunto de funciones, directivas de compilación y variables de entorno que proporcionan un modelo de ejecución paralelo bajo arquitecturas de memoria compartida (Sterling et al., 2018).	Es un framework de desarrollo de aplicaciones paralelas de propósito general mediante el uso de dispositivos de aceleración gráfica (GPU's) (Acosta et al., 2012).	Es el lenguaje de cálculo técnico desarrollado por MathWorks, es un entorno para el desarrollo de algoritmos, análisis de datos, visualización y cálculo numérico. Simulink es un entorno grafico para simulación y diseño basado en modelos de sistemas dinámicos multi-dimonio e integrados.(A. et al., 2008)	Es un lenguaje muy completo, con soporte para bases de datos relacionales, no relacionales u orientados a objetos, capaz de incorporar ORM´s, con frameworks de desarrollo muy complejos (Yeeply, 2017)..
Su forma	Están formadas por un conjunto	formadas por varios procesadores,	Desarrollada por NVIDIA en el campo de la GPU	Se identifico un entorno de computación	ahorra tiempo y recursos, al contrario de

	de procesadores, cada uno tiene su propia memoria local y cuyo espacio de direcciones de memoria no está mapeado al del resto.	cada uno de los cuales tiene acceso a una zona de almacenamiento global a través de algún bus o red de interconexión y bajo un único espacio de direccionamiento.	y compuesta por un nuevo modelo de programación que permite desarrollar el acceso a los dispositivos con capacidades CUDA.	más potente y productivo más allá de los proporcionados por lenguajes Fortran y C.	otros lenguajes como Java, C, PHP y JavaScript en la ausencia de paréntesis y la no incorporación de punto y coma al final de cada instrucción.
Principios	MPI fue creado en 1993 como un estándar abierto y fue desarrollada por el Centro de investigación en Computación paralela en Williamsbrug.	OpenMP fue desarrollada por varias empresas y fue publicada por la Architecture Review Board en 1997. OpenMP es un estándar para la industria de la computación, entre ellas Intel Corporation, IBM, Silicon Graphics Inc.	CUDA, nace a partir de la necesidad de desarrollar nuevas alternativas para el procesamiento de grandes volúmenes de información.	Es un lenguaje de cuarta generación, fundada por 1984 por Jack Little y Cleve Moler, quienes identificaron la necesidad de un entorno de computación más potente y productivo de los lenguajes como Fortran y C.	Python fue creado por Guido van Rossum, a principios de los años 90 en Holanda.
	MPI es una librería estándar para ser usada en programas que	OpenMP es una API para programación multiproceso de memoria compartida en múltiples	CUDA es una API de computación en paralelo, que incluye un compilador y una serie de herramientas de	MATLAB consta de una API extensa, en la cual podemos vincular nuestros programas C /	Python se basa en los lenguajes C y C++ y tiene sus raíces en el sistema

	aprovechen arquitecturas de múltiples procesadores. Sus lenguajes de especificación son C, C++ y Fortran. Existen implementaciones para Python, Ocalm, Java, .NET y PHP.(Acosta et al., 2012)	plataformas. Su modelo de ejecución se basa en memoria compartida multihilos. 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.	desarrollo creadas por NVIDIA para implementar algoritmos en sus GPU. Permite programador en C. Mediante wrapers se puede usar en C/C++, Java y Fortran y existen múltiples aplicaciones en Python.	C++ directamente a Matlab, también se utiliza generalmente para tareas, como: Procesamiento de la señal, optimización de funciones, diseño del sistema de control, procesamiento de imagen y audio, aprendizaje automático y aprendizaje profundo(Acervo, 2022)	operativo UNIX.

CAPITULO 2

DESARROLLO

2.1 Evaluación de algoritmos paralelos

Existen varios criterios que se pueden utilizar con el fin de evaluar la calidad de los algoritmos paralelos. Los criterios usados en el presente trabajo son los recomendados por (Briceño Coronado, 2012).

2.1.1 Tiempos de ejecución

Cuando se aborda la solución de un problema computacional se tiene la disyuntiva de seleccionar entre varios algoritmos y la cual se realiza basándose fundamentalmente en los siguientes aspectos: el algoritmo fácil de entender, codificar, depurar versus el algoritmo que se ejecute con la mayor rapidez posible, siempre con el objetivo de hacer un uso eficiente de los recursos de la máquina. La razón principal del diseño de algoritmos paralelos es usarlo de tal manera que una tarea computacional pueda ser completada rápidamente. El tiempo de ejecución es un buen indicador de la funcionalidad de un algoritmo paralelo. La medición del tiempo de ejecución utilizada está representada por el conteo de las unidades de segundos comprendidas entre el momento de inicio y termino de ejecución de algoritmo para la segmentación de imágenes agrícolas.

2.1.2 Rapidez

En computación paralela la rapidez se refiere a que tan rápido es un algoritmo en comparación con otro que resuelve el mismo problema a través de la medición de tiempos. Para la determinar la rapidez se utilizó la rapidez absoluta (Sp); la misma que es la razón de los tiempos entre el algoritmo serial (Ts) y el tiempo del algoritmo paralelizado (Tp). La medición de rapidez está determinada por la fórmula 1.

$$Sp = \frac{Ts}{Tp} \qquad (1)$$

Donde Ts es el tiempo de ejecución del algoritmo secuencial y Tp es el tiempo de ejecución del algoritmo paralelo con p procesadores.

2.1.3 Eficiencia

En computación se considera eficiencia (Ep) a la proporción del tiempo que se pasa desarrollando trabajo útil. La medición de la eficiencia está determinada por la fórmula 2.

$$Ep = \frac{Sp}{p} \qquad (2)$$

Donde Sp es la rapidez previamente calculada del algoritmo y p es el número de procesadores utilizados en la ejecución del algoritmo.

2.1.4 Costo

El costo (Cp) de un algoritmo paralelo es el límite sobre un total de operaciones ejecutadas colectivamente por los procesadores. La medición del costo está determinada por la fórmula 3.

$$Cp = \frac{Ts}{Ep} \qquad (3)$$

Donde Ts es el tiempo de ejecución del algoritmo serial y Ep es la eficiencia previamente calculada del algoritmo.

2.2 Imágenes Digitales

Una imagen se define como una función bidimensional f (x, y) donde x, y son coordenadas en el plano y la amplitud f es llamada intensidad o nivel de gris en ese punto. Cuando (x, y) y f son todos finitos [cantidades discretas] llamamos a la función como imagen digital. Es decir, una imagen digital está compuesta por un número finito de elementos llamadas pixeles, con un valor y una posición particular, en la cual contiene la representación digital de una imagen (Luján-Mora Sergio, 2008).

El término píxel [Picture elemento de imagen], trata de la unidad mínima de información de una imagen, la cual aparece como un punto en la pantalla o en una impresora. Cada pixel se compone de tres registros de color, mediante la combinación de cierta cantidad de rojo, verde y azul, el pixel adopta un color particular(García-Santillán, 2008), representando las imágenes de dos formas: como mapas de bits(imagen raster) o de forma vectorial.

Las imágenes bidimensionales son el resultado de una proyección en perspectiva de escenas tridimensionales. Cuando se obtiene una imagen bidimensional del mundo tridimensional desaparece gran cantidad de información.

Las imágenes digitales, ya sean generadas o creadas por medio de un aparato de captura(cámara, dron), suponen la traducción de valores de luminosidad y color a un lenguaje que pueda entender el ordenador y los periféricos relacionados con él, con una ventaja que supone la estabilidad; los ceros y unos que forman una imagen digital permanecerán estables, y así la imagen no variara durante el transcurso del tiempo (Luján-Mora Sergio, 2008).

2.2.1 Imágenes vectoriales

La información de cada uno de los puntos está recogida en forma de ecuación matemática que se relaciona con el resto de los puntos de la imagen.(Luján-Mora Sergio, 2008). Los objetos que componen una imagen vectorial son Líneas curvas o rectas, definidas por vectores y cada uno de estos vectores se componen de un punto inicial y un punto final que se denomina punto de control. Los gráficos vectoriales conservan la nitidez de los bordes y no pierden detalles al cambiar de tamaño puesto que son independientes de la resolución y puesto a que la información de cada punto no es absoluta sino relativa al resto de la imagen (Intefp et al., 2017), (Luján-Mora Sergio, 2008). No es soportado de forma directa por los navegadores de Internet como: Internet Explorer, Netscape Navigator, Firefox, Mozilla, etc. Algunos formatos de este tipo de imágenes son DWG (AutoCAD), SWF y FLA (Flash).

2.2.2 Imágenes Raster o Mapa de Bits

Las imágenes ráster, constan de un número fijo de píxeles y, por tanto, depende de la resolución, también pueden perder detalle y verse pixeladas cuando se agrandan. La imagen ráster es un formato de imagen que usa una cuadrícula rectangular de colores para representar las imágenes en la que cada punto de la cuadricula es un píxel, el formato ráster trabaja con imágenes en mapa de bits, el ráster de una imagen se obtiene multiplicando su anchura por la altura de la pantalla (Iturrioz, 2015). Utilizan una cuadrícula rectangular de elementos de imagen (pixeles) para representar las imágenes. A cada píxel se le asigna una ubicación y un valor de color especifico. La ventaja que presenta este formato es la posibilidad de recoger una amplia gama para representar imágenes capturadas en la realidad. En cambio, la variación de tamaño supondrá modificaciones en la calidad, ya que el número de celdas que forman las imágenes permanecen invariables, por lo que un aumento del tamaño hace que el único recurso posible sea ampliar el tamaño de cada una de las celdas(García-Santillán, 2008).

Dentro de este tipo se encuentran muchos formatos, algunos de los cuales son soportados directamente por los navegadores, siendo el tiempo de imágenes con las que vamos a trabajar. Estas imágenes son creadas por los drones y las cámaras digitales(García-Santillán, 2008).

Formatos de imágenes raster

Los formatos de imágenes están formados por una cabecera de información (formato, versión, datos de tamaño, colores, como leer la imagen, etc.), y de bits que contienen la información gráfica de la imagen. Así también las imágenes digitales se pueden guardar en distintos formatos y tienen prestaciones como: resolución, profundidad de bits, capacidades de color, compresión, soporte de metadatos (Luján-Mora Sergio, 2008).

Los formatos de imágenes más populares en nuestro medio tenemos:

TIFF (Tagged Imagen File Format= formato de Archivo de Imagen Etiquetada): Se trata de un formato de imágenes muy difundido a causa de su facilidad de lectura tanto en PC como en Macintosh, por la compresión de imágenes sin pérdida de calidad. Su principal desventaja es, una vez descomprimidas, las imágenes pueden ser grandes, motivos por el cual no se usa en la Web(García-Santillán, 2008), (Luján-Mora Sergio, 2008).

BMP (Bitmap = Mapa de bits): es el formato tradicional para los usuarios de Windows. Se le puede emplear con propósitos generales, como en la edición de imágenes y tapiz del escritorio de Windows. No siempre puede ser leído por computadoras Macintosh y sur archivos tienden a ser grandes. En general, este formato sólo tiene sentido en la actualidad para almacenar imágenes de uso exclusivo de Windows(García-Santillán, 2008)

GIF (Graphic Interchange Format= Formato de Intercambio Gráfico): Fue creado por la compañía CompuServe y significa formato de intercambio de gráficos. Se trata de un formato de archivo compacto de uso muy común en páginas Web. Su desventaja es que limita las imágenes a solo 256 colores, lo cual puede afectar la calidad de la imagen en la pantalla (Luján-Mora Sergio, 2008), (García-Santillán, 2008).

JPEG (Joint Photographic Experts Group = Grupo de Expertos Fotográficos Unidos): Fue creado por el Grupo Unido de Expertos en Fotografía. En este formato de archivo comprime con posibilidades de escalamiento para producir archivos reducidos. Sin embargo, de acuerdo con el grado de compresión de una imagen JPEG, la calidad de imagen puede variar poco o mucho.

Permite la exhibición de la paleta integra de 16 millones de colores, a diferencia del GIF(García-Santillán, 2008).

PNG (Portable Network Graphics= Gráfico portable para la red): Este formato de archivo es el más reciente en la Web, es similar al JPEG en el sentido de que también permite la exhibición de imágenes de amplio colorido, pero su compresión no reduce la calidad de imagen. Como consecuencia de haber sido diseñado para internet, este formato posee muchas otras características, pero su uso es aún restringido (Luján-Mora Sergio, 2008).

Tabla 4: Comparativa de formatos
Fuente: (Luján-Mora Sergio, 2008)

Nombre	Profundidad de Píxel	Compresión	Color	Web	Comentarios
GIF	1 a 8 bits	LZW	P.S.	SI	Estándar de factor en la web, soporta animaciones.
JPG	24 bits	JPEG	YCC	SI	Estándar de factor en la web.
PNG	1 a 48 bits	Deflate	P.S., RGB	SI	Sustituto de GIF (aprobado por el W3C)
TIF	1 a 64 bits	ITU T6, JPEG	P.S., RGB, YCC, CMYK	NO	Posibilidad de múltiples imágenes en un solo fichero.
BMP	1 a 24 bits	No tiene	P.S.	NO	Otras extensiones: dib, vga, bga, rle, rl4, rl8.
PDF	1 a 64 bits	ITU T6, JPEG	RGB, YCC, CMYK.	NO	Óptimo para ver documentos de páginas múltiples.

Nota: P.S.: Paleta de colores del sistema operativo.
YCC: Sistema de color con un canal para luminosidad (y) y dos crománcia (CC).

2.2.3 Tipos de imágenes digitales

En el procesamiento digital de imágenes (PDI) se maneja cuatro tipos de imágenes básicamente imágenes RGB, imágenes indexadas, imágenes en escala de grises e imágenes binarias, las cuales se aplican a continuación:

Imágenes RGB (Red-Green-Blue)

- Este tipo de imágenes utilizan tres canales para reproducir los colores en la pantalla.
- Utilizan 8 bits por canal (8 bits x3), es decir, 24 bits de color para cada píxel
- Reproduce hasta 16, 7 millones de colores.
- Soporta algunos formatos como: JPG, BMP, PNG, etc.

Imágenes Indexadas

- Este tipo de imágenes reduce los colores de la imagen a un máximo de 256.
- Admite los formatos GIF y PNG-8 y muchas aplicaciones multimedia.
- Reduce el tamaño de archivo porque elimina la información del color.

Imágenes de escala de grises

- Utilizan distintos tonos de gris.
- En imágenes de 8 bits, puede tener hasta 256 tonos de gris.
- Cada píxel tiene un valor de brillo comprimido entre 0 (negro) y 255 (blanco).

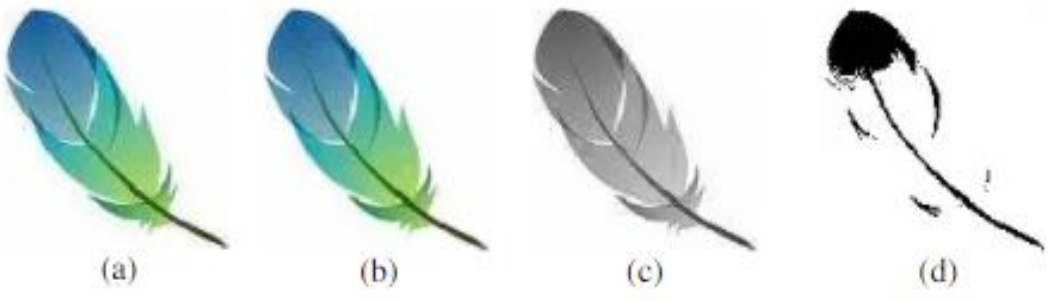
Imágenes binarias

- Tienen una profundidad de color de 1 bit.
- Utiliza uno de los valores de color (blanco o negro) para representar los pixeles de una imagen.

Existen imágenes con una profundidad de píxel de 32 bits. Los 8 bits (1 byte) adicionales de profundidad sobre las imágenes de 24 bits, permitiendo almacenar la transparencia de la imagen. El byte adicional es generalmente llamado macara o canal alfa, y almacena valores de transparencia.

En la (Abdul Khalib, Ng., Elshaikh, & Othman, 2020), Figura 22, se muestra algunos elementos correspondientes a los tipos de imágenes.

Figura 22: Tipo de Imágenes Digitales;
a) RGB, b) Indexada, c) Escala de Grises, d) Binaria
Fuente:(García-Santillán, 2008)



2.2.4 Calidad de una imagen

La imagen digital, bien sea generada por computador o creada a través de algún dispositivo de captura, tal como una cámara digital o un dron, aporta una principal ventaja que es

la estabilidad, mientras que la emulsión de una imagen fotográfica clásica sufre una degradación química con el paso de tiempo, que repercute en la calidad de dicha reproducción, los ceros y nos que componen una imagen digital permanecen estables, con lo que la imagen no variará a lo largo de tiempo(García-Santillán, 2008).

La calidad de la imagen ráster es determinada en el proceso de captura de tres factores: el tamaño del píxel (resolución espacial), la profundidad de píxel (resolución de brillo) y el ruido. El tamaño de píxel es determinado por el rango al cual dron muestrea la imagen. Un intervalo me muestreo largo produce una imagen baja en resolución espacial (Bollatti et al., 2017). El brillo o valor de color de cada píxel es definido por un bit o un grupo de bits. Mientras se usan más bits, más alta es la resolución de brillo. Todas las imágenes tienen cierta cantidad de ruido, ya sea por la cámara, dron o el medio de transmisión de la señal. Por lo general el ruido se manifiesta como pixeles aislados que toman un nivel de gris como pequeñas y aleatorias variaciones en el brillo y el color, los algoritmos de filtrado, permiten eliminar o disminuir el ruido(García-Santillán, 2008).

La resolución de una imagen es el número de píxeles que contiene una imagen, expresada como 640 x 480, 800 x 600, por ejemplo. Es un término que debe ser considerado al utilizar imágenes en determinados trabajos.

En general una baja resolución de imagen se utiliza para:

- Imágenes para páginas Web y correo electrónico.
- La memoria de la cámara es limitada.
- Se dispone de escaso espacio de disco duro para el almacenamiento de imágenes.
- En general se debe reducir a una resolución más alta si:
- Las imágenes están destinadas a impresiones de alta resolución.
- Se dispone de suficiente espacio de almacenamiento, tanto en la cámara como en el disco duro.

El tamaño de una imagen se puede calcular multiplicando la calidad de píxeles horizontales (ancho) por l cantidad de píxeles verticales (alto) y por la profundidad de brillo (en bits). En la Tabla 5, se muestra algunos ejemplos de tamaños de imágenes.

Tabla 5: Tamaño de una imagen

Resolución	Profundidad de píxel	Tamaño del archivo			
		Bits	bytes	Kbytes	Mbytes
640 x 480	X 1 bit	=307,200	=38.400	=37.5	=0.036
640 x 480	X 8 bits	=2’457.600	=307.200	=300	=0.0292
640 x 480	X 24 bits	=7’372.800	=921.600	=900	=0.878
640 x 480	X 32 bits	=9’830.400	=1’228.800	=1200	=1.171

2.2.5 Imágenes de drones

Las imágenes satelitales son una de las herramientas digitales que el agro utiliza en la actualidad. Anteriormente las capturas de imágenes se hacía cada 16 días dado que el satélite demoraba ese período de tiempo para regresar al mismo punto y muchas veces el factor climático no permitía buenas tomas para definir manejos en los cultivos (Bollatti et al., 2017).

De esta dificultad para conseguir las imágenes en el momento oportuno es que muchos investigadores y profesionales comenzaron a sacar fotografías aéreas desde aviones tripulados y entregando la información ya procesada para poder realizar el análisis agronómico correspondiente. Esta actividad se desarrolló y dio buenos resultados agronómicos, pero en algunos casos el factor costo y logística para sacar las fotografías en vuelos programados era una limitante que aún se incrementaba cuando se deseaba hacer un seguimiento de los cultivos (Jiménez Jiménez Sergio Iván, 2020).

El uso de drones para monitoreo agropecuario es una realidad en estos en los últimos años. Al realizar el monitoreo de grandes extensiones de campo es esencial en la actualidad se realiza, con la finalidad de detectar presencia de malezas, fallas en la siembra, estado de los cultivos, o bien reflejar alguna problemática en el campo o áreas urbanas, es necesario contar con información rápida y detallada de la situación. Es por ello que se utilizan aeronaves no tripuladas (UAV), comúnmente llamados drones (Bollatti et al., 2017), (Jiménez Jiménez Sergio Iván, 2020).

El objetivo de este trabajo fue implementar técnicas básicas de procesamiento de imágenes y compartir algunas experiencias realizadas con un dron provisto de una cámara RGB y sus posibles usos relacionados al sector.

El dispositivo utilizado es un equipo de gama media a baja, con buena relación beneficio/costo debido a las prestaciones que ofrece con un valor de mercado relativamente bajo. Dentro de la fotointerpretación, una de las herramientas más utilizadas es el análisis visual para determinar patrones distintos en una imagen dentro del rango que el ojo humano lo permite.

Las imágenes se presentan con un breve detalle del uso posible de esta herramienta para el monitoreo de actividades agropecuarias.

2.3 Procesamiento de imágenes

Las técnicas de procesamiento digital de imágenes son una herramienta que permite la identificación temprana de las plagas o enfermedades en cultivos tempranos para de esta forma, mitigar pérdidas económicas en el sector agrícola. A nivel mundial, alrededor del 40% de los cultivos son desechados por diversas enfermedades y plagas. En la mayoría de los casos, las enfermedades de los cultivos producen síntomas y características visibles durante el crecimiento de las plantas. Debido a la escasez de tecnologías utilizadas en los cultivos, el diagnóstico de las enfermedades y plagas se soporta en gran parte en la inspección humana, generando errores ocasionados por la subjetividad de los individuos. Demostrando que el sistema de diagnóstico está compuesto de la adquisición de imágenes, preprocesamiento de imágenes, la segmentación, la extracción de características, la selección de características y la posterior clasificación de las plagas o enfermedades (Gómez-Camperos et al., 2021).

2.3.1 Extracción de bandas o colores (otsu)

El método de Otsu fue uno de los mejores métodos de selección de umbral para imágenes del mundo real, sin embargo, necesita mucho más tiempo para seleccionar el umbral óptimo. La importancia del método Otsu radica que es automático, es decir, no necesita supervisión humana ni información previa de la imagen antes de su procesamiento. Si bien, otros métodos en este tipo de imágenes no dan buenos resultados debido a la presencia de ruido, histogramas planos o iluminación altamente cambiante. En si el objetivo del método de Otsu es calcular el valor del umbral de forma que la dispersión dentro de cada clase sea lo más pequeña posible, pero al mismo tiempo que la dispersión sea lo más alta posible entre clases diferentes (Riomoros Callejo, 2015).

Profundidad del color: Cantidad de colores diferentes que se pueden representar en una imagen. Para almacenar esta información se realiza una asignación de un número de bits para indicar el color de cada píxel. Si la imagen es bitonal, solo necesitaremos dos valores para indicar los colores:

Blanco = encendido = 1

Negro = apagado = 0

Si queremos representar una imagen usando 16 colores necesitamos (2^4)= 16 combinaciones diferentes para poder representar todos los colores, es decir, 4 bits.

Las imágenes en blanco y negro utilizan 256 tonalidades de gris (la más oscura es el negro y la más luminosa el blanco). Así que necesitamos $(2^8) = 256$ combinaciones, es decir, necesitamos 8 bits (1 btye)(Luján-Mora Sergio, 2008).

La Tabla 6, siguiente muestra las profundidades más frecuentes:

Tabla 6: Profundidades de bits.

Bits	Combinaciones	Colores	Bits por pixel
1	2^1	2	1
8	2^8	256	8 (1 byte)
16	2^{16}	65538	16 (2 bytes)
24	2^{24}	16,7millones	24 (3 bytes)

Así, se aumenta el número de los colores posibles en cada imagen, incrementando el espacio necesario para almacenar, porque se necesita más espacio (más bits) para representar el color de cada píxel.

Una imagen en blanco y negro usa un bit por píxel.

Una imagen a 256 colores usa un byte (8 bits) por píxel.

Una imagen a “truecolor” (16, 7 millones de colores), usa 3 bytes por píxel, es decir, 24 bits por píxel.

Tratamiento del color

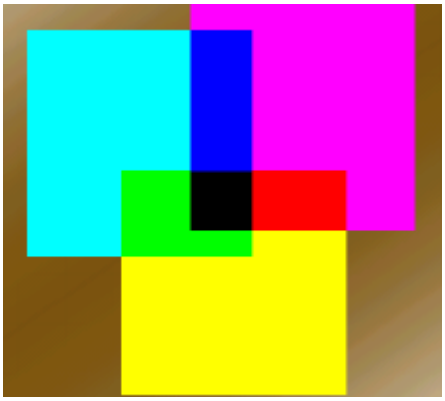
Los distintos sistemas de representación del color están basados en estudios y teorías sobre la percepción humana del color y en modelos matemáticos de representación de los colores. Siendo el ser humano capaz de percibir cuatro matrices principales diferentes: rojo, amarillo, verde y azul. En los modelos de color: RGB, CMYK, HSV (Luján-Mora Sergio, 2008).

RGB: Espacio RGB (Red, Green, Blue Rojo, Verde, Azul): conjunto de tres imágenes con niveles de gris independientes, cada una representa un color: rojo, verde y azul. Llamándose cada imagen en canal o componente de color, obteniendo de un color la suma de los tres canales, colores aditivos(Luján-Mora Sergio, 2008).

Figura 23: Modelo de color RGB

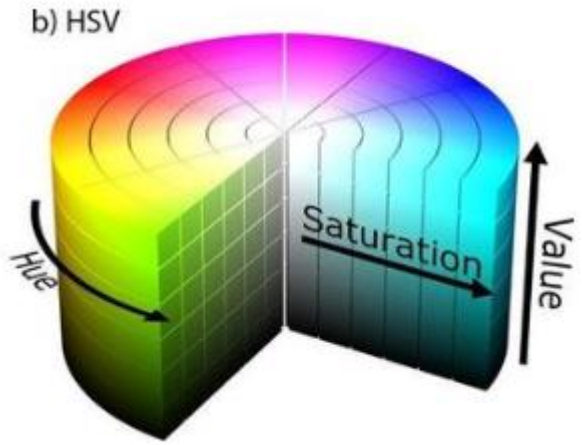
CMYK (Cyan, magenta, yellow), canals o componentes; azul claro, rosa y Amarillo. Se añade la componente negra para extender la gama. En un sistema substractivo. La obtención de colores se realiza restando los tonos de los tres canales.

Figura 24: Modelo en CMYK



HSV (Hue, Saturation, Value; tono, saturación y valor): añade la saturación, el valor y el tono para conseguir colores, es un sistema coordenado cilíndrico, y el subconjunto donde se define el color es una pirámide de base hezagonal. En este modelo los colores más brillantes están contenidos en el área hexagonal. Para medir el tono, se usa el ángulo alrededor del eje S. el rojo se sitúa a 0°, el verde a los 120° y el azul a los 240°. Los colores complementarios se encuentran a 180° de su color primario(Riomoros Callejo, 2015).

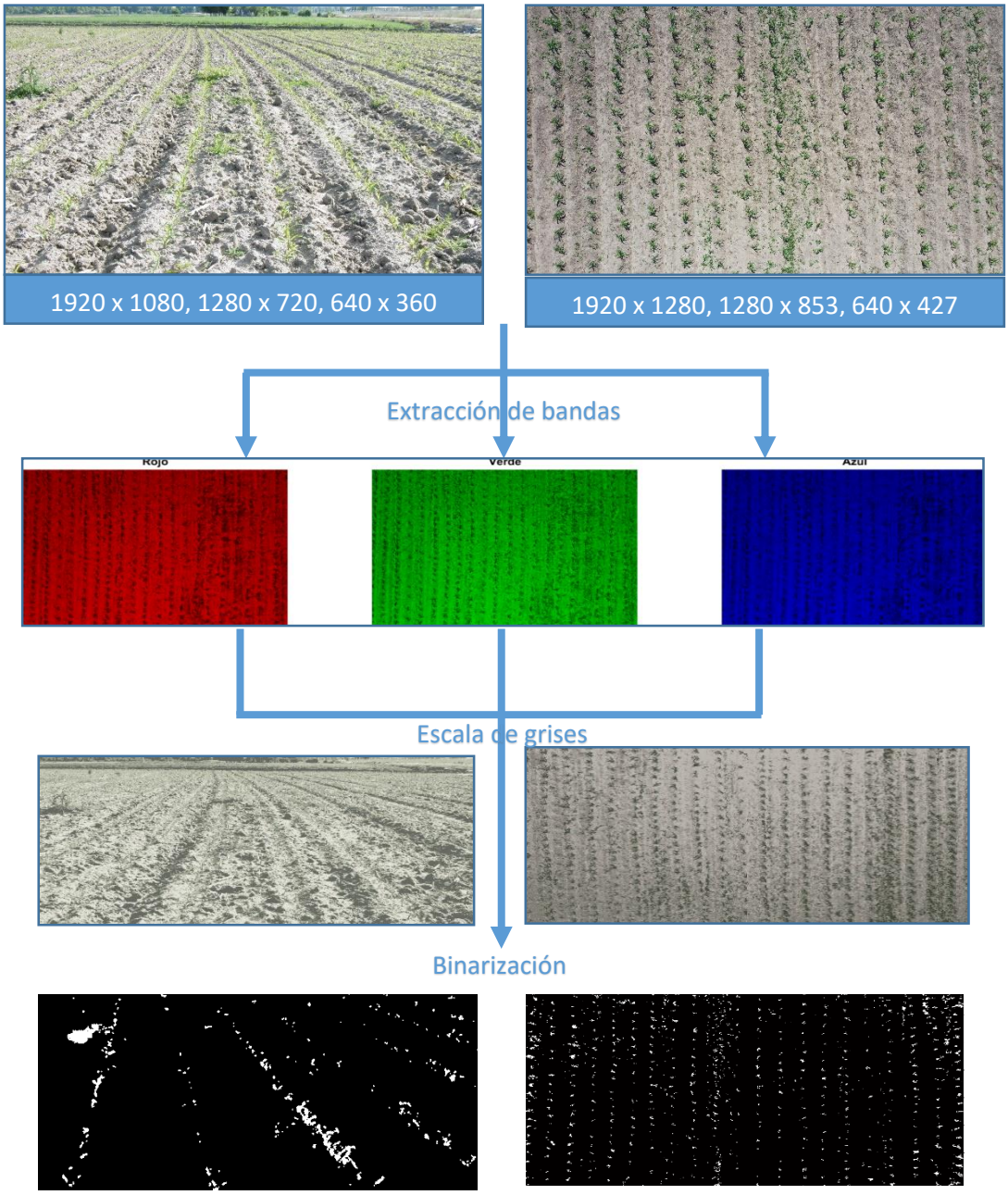
Figura 25: Modelo de HSV
Fuente:(Vazquez Saraullo et al., 2019)



2.4 Algoritmo de Procesamiento

La segmentación de imágenes es el proceso de particionar una imagen digital en sus objetos constituyentes para identificar los principales elementos que componen la escena, es el primer paso hacia una detección efectiva de los objetos presentes en la imagen. Cuanto más precisa sea la fase de segmentación, más probable es que el proceso de reconocimiento de imágenes tenga éxito. Para el propósito del proyecto, se consideró la segmentación simple basada en el color, dividiendo los elementos de interés y discriminantes en regiones blancas y negras respectivamente para convertir los datos en un entorno lógico de ceros y unos y permitir al sistema reconocer y extraer los elementos de interés. En La Figura 26, se presenta el flujograma del algoritmo serial como paralelo para el procesamiento de imágenes digitales.

Figura 26. Arquitectura Lógica de Procesamiento
Fuente: Propia



En el programa, los valores de color obtenidos al escanear la imagen de entrada se almacenan en una matriz asignada en memoria y constituyen el conjunto de datos inicial sobre el que realizar la segmentación. Cada fila de la matriz contiene los valores RGB de 0 a 255 de cada píxel de la imagen. Dada una imagen digital de ancho W y alto H, el conjunto de datos inicial es una matriz formada por filas N = W x H y tres columnas:

$$imagen = \begin{bmatrix} p_0^r & p_0^g & p_0^b \\ p_1^r & p_1^g & p_1^b \\ \cdots & \cdots & \cdots \\ p_{N-1}^r & p_{N-1}^g & p_{N-1}^b \end{bmatrix}$$

,p-i-r., ,p-i-g. y ,p-i-b. son respectivamente los componentes rojo, verde y azul del i-ésimo píxel de la imagen.

2.4.1 Herramientas de programación

Eclipse IDE for Scientific Computing

Herramientas para C, C ++, Fortran y UPC, incluidos MPI, OpenMP, OpenACC, un depurador paralelo y aplicaciones de creación, ejecución y supervisión de forma remota (Fundation, 2021), incluye las siguientes herramientas:

- Herramientas de desarrollo C / C ++
- Integración de Git para Eclipse
- Lista de tareas de Mylyn
- Plataforma de herramientas paralelas
- Editores y herramienta XML de Eclipse

GCC

GCC es una colección de compiladores GNU incluye interfaces para C, C ++, Objective-C, Fortran, Ada, Go y D, así como bibliotecas para estos lenguajes (libstdc ++, entre otras). GCC se escribió originalmente como el compilador del sistema operativo GNU. El sistema GNU fue desarrollado para ser un software 100% libre, gratuito en el sentido de que respeta la libertad del usuario (GNU, 2021). Las actualizaciones del código fuente están disponibles de forma fácil y gratuita a través de Git e instantáneas semanales.

MakeFile

Los archivos Makefile utilizan el comando de linux make para compilar programas basados en C, C++. Presenta muchas ventajas para programas grandes, en los que hay muchos ficheros fuente (muchos .c y muchos .h) repartidos por varios directorios (Chuidiang, 2020). Principalmente aporta dos ventajas:

- Es capaz de saber qué archivos hay que recompilar. Si haya cambios en algún archivo fuente, al compilar con make sólo se recompilarán aquellos archivos que dependan del modificado. Facilita el trabajo en proyectos es grandes.

- 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 de opciones, sólo se realiza una vez o las veces que haya cambios en el código fuente.

2.4.2 DataSet de imágenes

En la evaluación del algoritmo se consideraron imágenes de 24-bits por píxel, y por tanto con 8 bits por cada canal espectral R, G y B, en formato JPG. Las 10 primeras se consideraron del trabajo realizado por (García-Santillán & Pajares, 2018); las mismas fueron capturadas bajo la proyección de perspectiva de 45 grados y almacenadas con una resolución de 3000×2250 pixeles (7 Mpx). Las segundas imágenes se realizaron en un terreno de maíz recién cultivado utilizando un dron DJI MAvic 2Pro (DJI, 2021) capturadas de forma cenital (90 grados) y almacenadas con una resolución original de 5472 x 3626 pixeles (20 Mpx).

Las fotografías con proyección de 45 grados evaluadas con el algoritmo implementado fueron de las siguientes dimensiones 1920 x 1080 (imágenes grandes), 1280 x 720 (imágenes medianas) y 640 x 360 (imágenes pequeñas). Las fotografías utilizadas para evaluación con proyección de 90 grados fueron con dimensiones de 1920 x 1080 (imágenes grandes), 1280 x 720 (imágenes medianas) y 640 x 427 (imágenes pequeñas).

2.4.3 Extracción de bandas

Esta fase consiste en la distinción e identificación de la vegetación de las imágenes y se extraen los valores de los canales en rojo, verde y azul (RGB) para realizar la operación de luminosidad con los bits de la imagen.

2.4.4 Escala de grises

Este proceso consiste en modificar la matriz de información en valores de escala blanco y negro ya que resulta mucho más conveniente a la hora de manipular información contenida en la imagen. Los formatos RGB hacen uso de tres canales que describir la cantidad de rojo, verde y azul en una imagen. Mediante el cambio de formato de escala de grises los valores RGB de

cada píxel en la matriz se modifica a través de la intensidad del color y transformarlo en tonalidad blanco y negro.

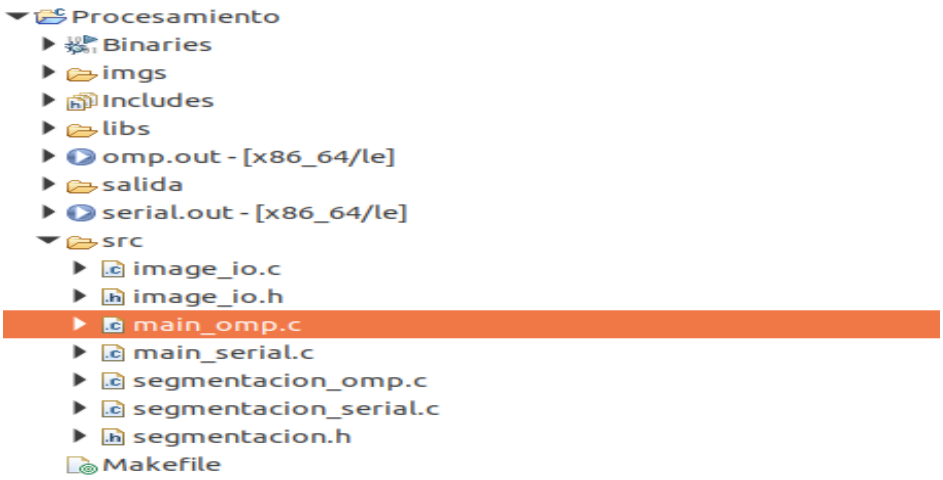
2.5 Estructura del proyecto

Este apartado presenta el trabajo realizado en la paralelización de algoritmos para procesamiento de imágenes agrícolas como parte de la segmentación de imágenes. A continuación, se presentan todos los pasos necesarios para la correcta prueba de todos los algoritmos, que fueron implementados en el lenguaje de programación C (GCC) en el sistema operativo Ubuntu 18.04. El IDE de Eclipse para Computación Científica se utilizó para la programación del algoritmo serial y paralelo, todas las imágenes utilizadas como entrad y salid durante este proyecto están en formato jpg.

2.5.1 Contenido proyecto

El proyecto está compuesto por 7 archivos principales, como se presenta en la Figura 27, los archivos se encuentran en la carpeta src, dos archivos de cabecera (image.h y segementacion.h) y cinco archivos fuente (image_io.c, main_omp.c, main_serial.c, segmentación_omp.c y segmentacion_serial.c).

Figura 27: 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_omp.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 paralelizado.

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

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

La carpeta imgs contiene las 30 imágenes de entrada a ser evaluadas por los algoritmos implementados, las imágenes tienen diferentes medidas (grande, mediana y pequeña) 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 28, se presenta el contenido del archivo Makefile; el mismo que contiene el conjunto de tareas a ser ejecutadas para la compilación de los algoritmos serial y paralelo.

Figura 28: Contenido del Archivo Makefile

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

all: serial.out omp.out

clean:
    rm serial.out omp.out

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

omp.out: src/main_omp.c src/image_io.h src/image_io.c src/segmentacion.h src/segmentacion_omp.c
    $(CC) $(CC_FLAGS) $(CC_OMP) -o omp.out src/main_omp.c src/image_io.c src/segmentacion_omp.c -lm

```

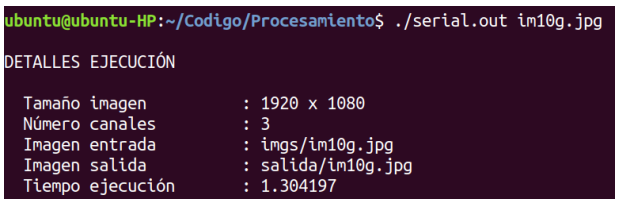
Adicionalmente se incluyen dos archivos .out que son los ejecutables que son utilizados desde la terminal del sistema operativo.

2.5.2 Ejecución de proyecto

Para la ejecución del programa se utiliza la terminal de Linux utilizando los ejecutables previamente generados mediante el comando make. El comando make ejecuta el contenido del archivo Make file para depurar el código fuente y generar los archivos .out.

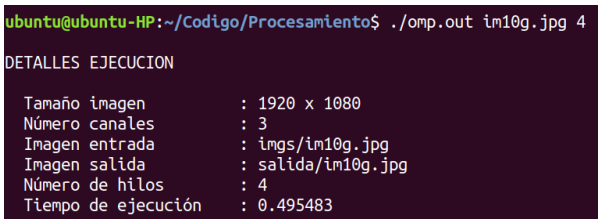
El programa realizado en OpenMP basado en lenguaje C++, realiza la separación de los tres colores (rojo, verde, azul), luego se calcula el umbral de limonosidad (exg) para obtener una imagen con escala de grises, posteriormente se binariza la imagen previa en escala de grises. En la Figura 29 y Figura 27, se presenta las compilaciones de los algoritmos serial y paralelo para procesar imágenes obtenidas con cámara.

Figura 29: Compilaciones de los Algoritmos Serial y Paralelo en Imágenes con Cámara

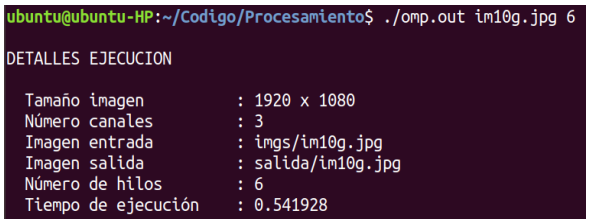


(a) Algoritmo serial

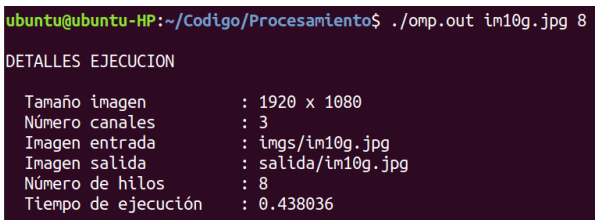
(b) Algoritmo paralelo con 2 núcleos



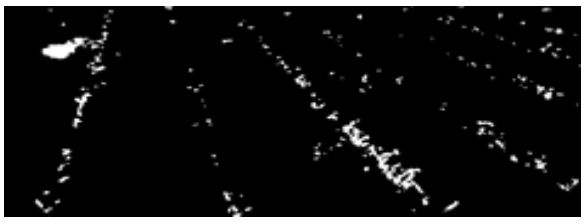
(c) Algoritmo paralelo con 4 núcleos



(d) Algoritmo paralelo con 6 núcleos



(e) Algoritmo paralelo con 8 núcleos



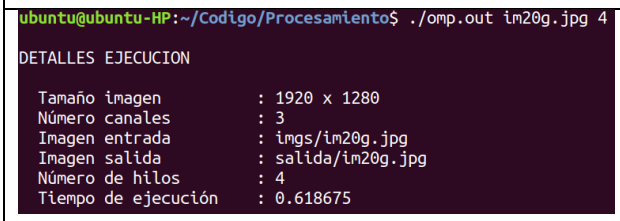
(f) Imagen de salida

En la Figura 30, se presenta las compilaciones de los algoritmos serial y paralelo para procesar imágenes obtenidas con dron.

Figura 30: Compilaciones de los Algoritmos Serial y Paralelo en Imágenes con Dron

(a) Algoritmo serial

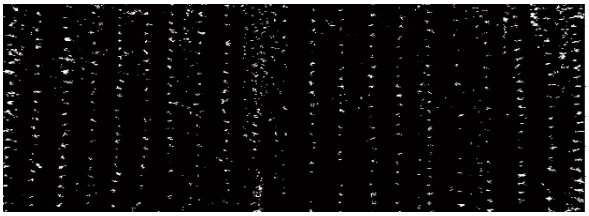
(b) Algoritmo paralelo con 2 núcleos



(c) Algoritmo paralelo con 4 núcleos

(d) Algoritmo paralelo con 6 núcleos

(e) Algoritmo paralelo con 8 núcleos

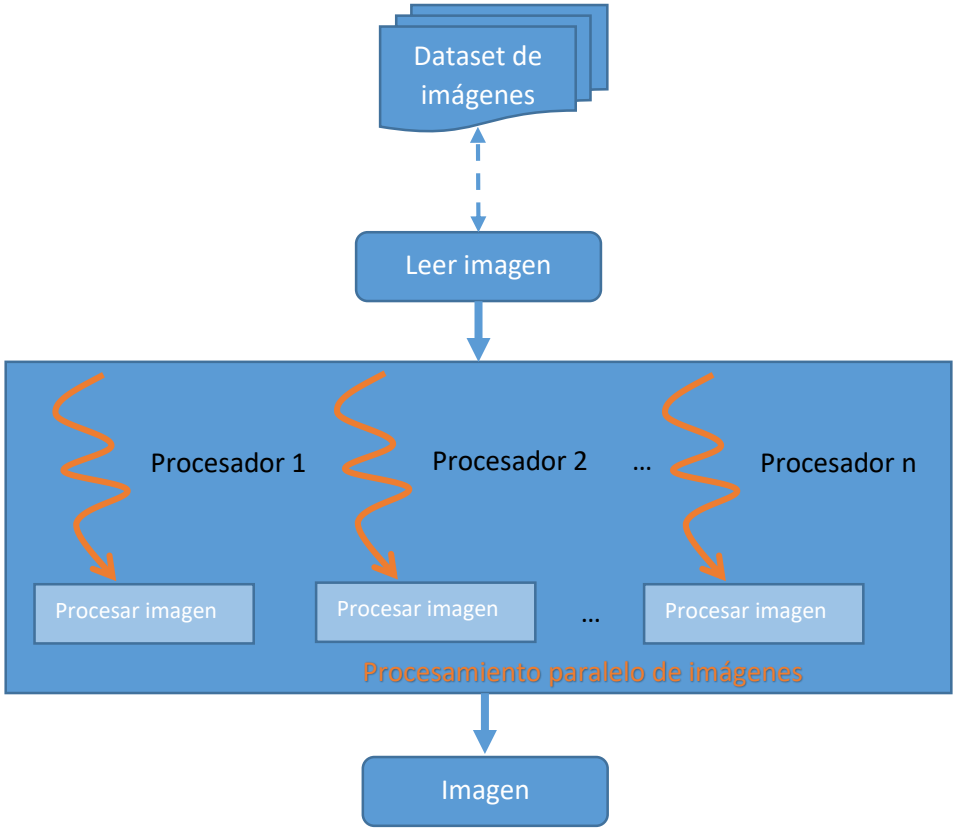


(f) Imagen de salida

2.6 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. Utilizando la Ley de Amdahl (EduRed, 2021), también es posible producir una estimación de la aceleración alcanzable de la implementación paralela.

Figura 31: Procesamiento de imágenes en paralelo



2.6.1 Equipo de procesamiento

Las pruebas del algoritmo serial y paralelo, se realizaron en una Laptop HP EliteBook 14 Noteboox con Linux Ubuntu 18.04 de 64 bits, procesador Intel Core I7, 16 MB de memoria RAM. Las características técnicas del computador y del CPU utilizado para el procesamiento serial y paralelo se presentan en la Tabla 7:

Tabla 7: Especificaciones técnicas de Laptop con Ubuntu Linux y Procesador multicore-CPU

Detalle	**Información**
Sistema operativo	Ubuntu 18.04 bionic
Núcleo de Linux	5.4.0-87-generic
Memoria RAM	8 Gb
Arquitectura	X86_64B
Procesador	Intel® Core™ i7-10510U CPU @ 1.80 Ghz
Núcleos	4
Procesadores lógicos	8
Fabricante	Hewlett-Packard

2.6.2 Algoritmo serial

La versión serial del programa ha sido codificado utilizando el generador de perfiles GNU gprof para obtener el porcentaje del tiempo de ejecución que tarda cada función del programa. En particular, el código del programa está organizado de tal manera que el proceso de segmentación tiene lugar en la función serial segmentación, y cada uno de los pasos del algoritmo se implementa en una sub-rutina independiente.

2.6.3 Algoritmo paralelo

La versión paralela del programa ha sido realizada usando OpenMP. OpenMP es una interfaz de programación de aplicaciones desarrollada específicamente para la programación multiprocesamiento en entornos de memoria compartida utilizando el lenguaje C. Ofrece un conjunto de directivas de compilador, rutinas de biblioteca y variables de entorno que se pueden usar para hacer que un código fuente C se ejecute simultáneamente en múltiples núcleos de un procesador.

OpenMP se basa en un modelo de unión de bifurcación. El flujo de control del algoritmo es estrictamente secuencial y es manejado por el hilo principal. Cuando se alcanza una sección computacionalmente intensiva, los ciclos principales y los cálculos se distribuyen entre todos los subprocesos disponibles. Una vez que todos los hilos finalizan su trabajo, se unen y el control vuelve al hilo principal. Al agregar pragmas al código fuente en serie, fue posible implementar una versión paralela del algoritmo un esfuerzo reducido. Para paralelizar la asignación de cada píxel, se ha utilizado una construcción paralela para compartir el trabajo.

```
#pragma omp parallel for schedule(static) private(…)

for (px = 0; px < n_px; px++) {

        …

}
```

El trabajo que se realiza dentro del bucle for se comparte entre todos los subprocesos disponibles para el programa, de modo que a cada subproceso se le asigna de antemano un número igual de iteraciones del bucle que se va a ejecutar, de tal forma que la distribución del trabajo entre subprocesos está bien equilibrada.

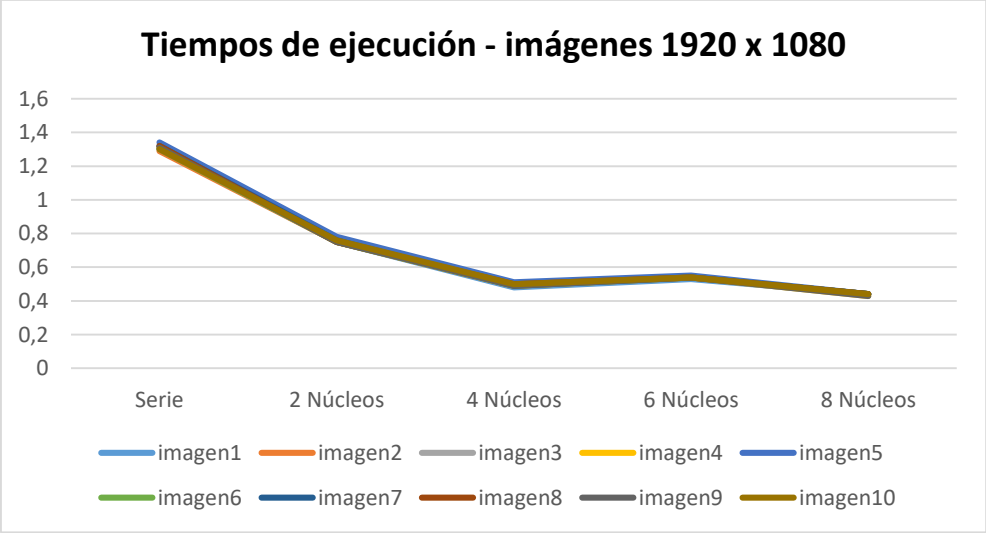
2.6.4 Tiempos de ejecución imágenes grandes

En la Tabla 8 y Figura 32, se presenta los tiempos de ejecución en el procesamiento de imágenes obtenidas con cámara con dimensiones (1920x1080 pixeles) del algoritmo serial y paralelo (2,4,6 y 8 núcleos).

Tabla 8: Tiempos de ejecución imágenes grandes obtenidas con cámara

Imágenes grandes con cámara – 1920 x 1080					
Imagen	**Serial**	**2 núcleos**	**4 núcleos**	**6 núcleos**	**8 núcleos**
1	1.30	0.75	0.48	0.53	0.44
2	1.29	0.75	0.49	0.54	0.44
3	1.32	0.76	0.50	0.54	0.44
4	1.31	0.75	0.49	0.54	0.43
5	1.34	0.78	0.51	0.55	0.44
6	1.30	0.75	0.49	0.54	0.44
7	1.32	0.76	0.49	0.54	0.44
8	1.32	0.75	0.50	0.54	0.44
9	1.31	0.75	0.49	0.54	0.43
10	1.30	0.76	0.5	0.54	0.44
Promedio	**1.31**	**0.76**	**0.49**	**0.54**	**0.44**

Figura 32: Tiempos de ejecución imágenes grandes obtenidas con cámara

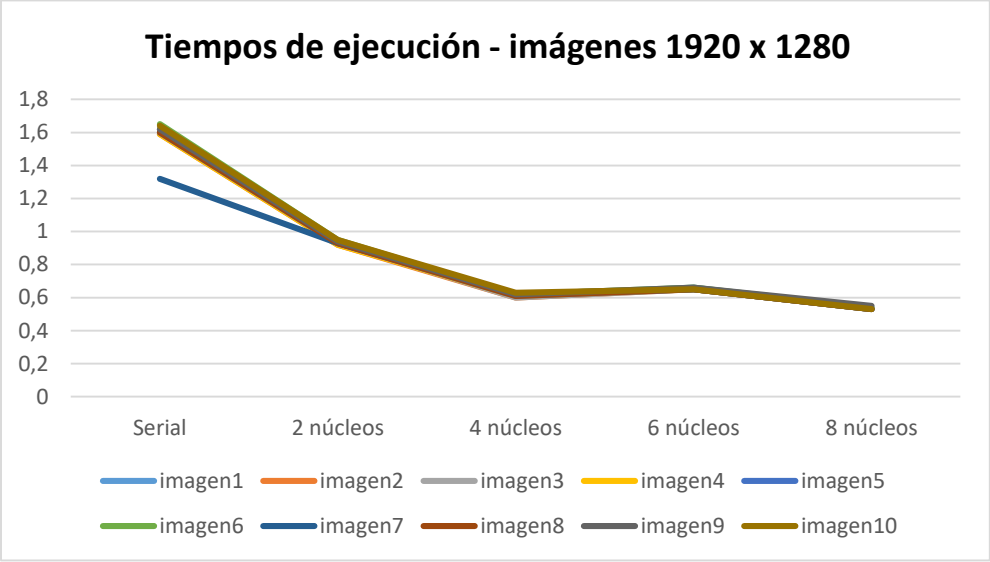


En la Tabla 9 y Figura 33, se presenta los tiempos de ejecución en el procesamiento de imágenes obtenidas con drone con dimensiones (1920x1280 pixeles) del algoritmo serial y paralelo (2,4,6 y 8 núcleos).

Tabla 9 : Tiempos ejecución imágenes grandes obtenidas con dron

Imágenes grandes con dron - 1920x1280					
Imagen	**Serial**	**2 núcleos**	**4 núcleos**	**6 núcleos**	**8 núcleos**
1	1.62	0.94	0.61	0.66	0.53
2	1.59	0.92	0.60	0.65	0.53
3	1.62	0.93	0.60	0.65	0.53
4	1.59	0.92	0.61	0.65	0.53
5	1.60	0.93	0.61	0.65	0.53
6	1.65	0.95	0.62	0.66	0.54
7	1.32	0.93	0.62	0.65	0.53
8	1.60	0.93	0.61	0.65	0.53
9	1.62	0.94	0.62	0.66	0.55
10	1.64	0.95	0.63	0.65	0.53
Promedio	**1.59**	**0.93**	**0.61**	**0.65**	**0.53**

Figura 33:Tiempos ejecución imágenes grandes obtenidas con dron



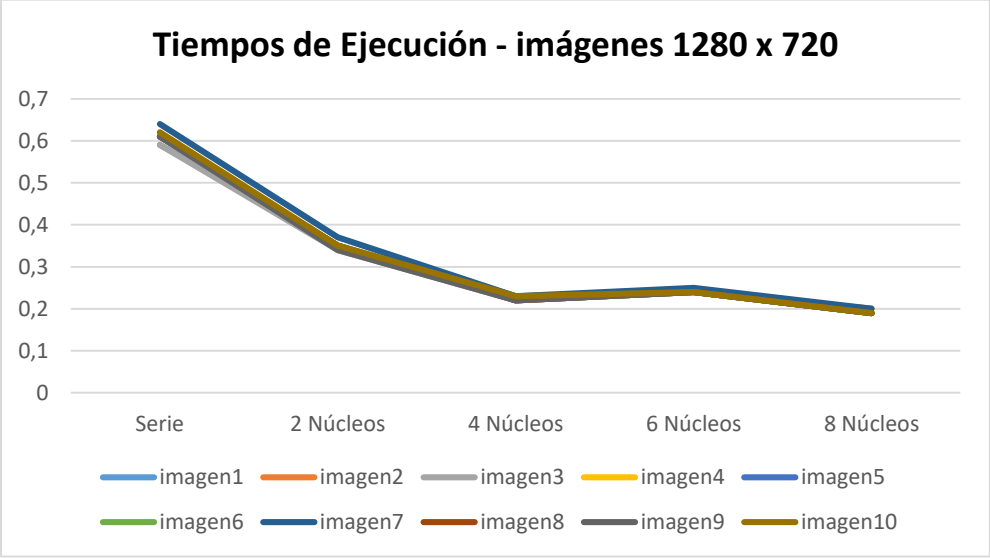
2.6.5 Tiempos de ejecución imágenes medianas

En la Tabla 10 y Figura 34, se presenta los tiempos de ejecución en el procesamiento de imágenes obtenidas con cámara con dimensiones (1280x720 pixeles) del algoritmo serial y paralelo (2,4,6 y 8 núcleos).

Tabla 10: Tiempos de ejecución imágenes medianas obtenidas con cámara

Imágenes medianas con cámara - 1280x720					
Imagen	**Serial**	**2 núcleos**	**4 núcleos**	**6 núcleos**	**8 núcleos**
1	0.59	0.35	0.22	0.24	0.19
2	0.61	0.35	0.22	0.24	0.20
3	0.59	0.34	0.22	0.24	0.19
4	0.62	0.35	0.22	0.24	0.20
5	0.61	0.35	0.22	0.24	0.19
6	0.61	0.35	0.23	0.24	0.19
7	0.64	0.37	0.23	0.25	0.20
8	0.61	0.35	0.22	0.24	0.19
9	0.61	0.34	0.22	0.24	0.19
10	0.62	0.35	0.23	0.24	0.19
Promedio	**0.61**	**0.35**	**0.22**	**0.24**	**0.19**

Figura 34: Tiempos de ejecución imágenes medianas obtenidas con cámara

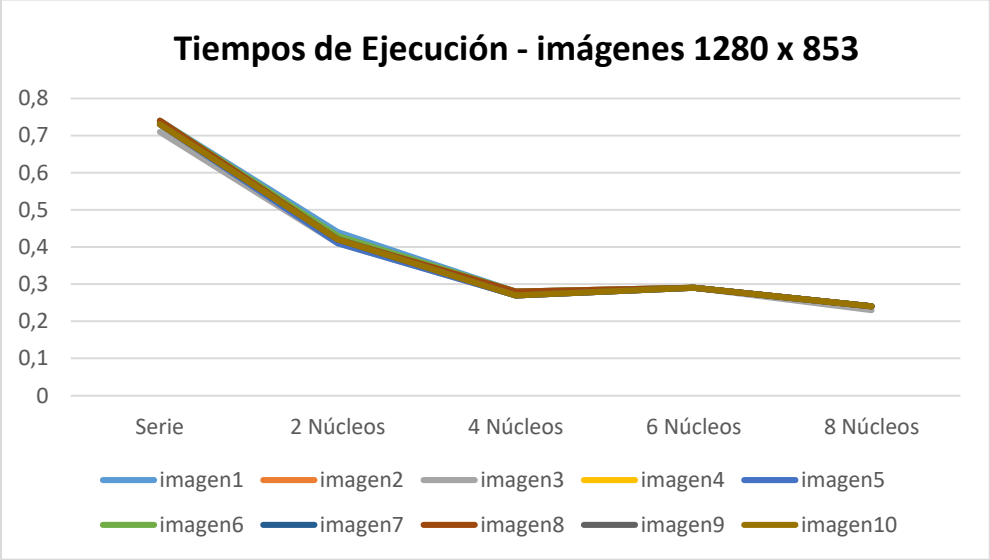


En laTabla 11 y Figura 35: Tiempos de ejecución imágenes medianas obtenidas con DronFigura 35,se presenta los tiempos de ejecución en el procesamiento de imágenes obtenidas con dron con dimensiones (1280x853 pixeles) del algoritmo serial y paralelo (2,4,6 y 8 núcleos).

Tabla 11: Tiempos ejecución imágenes medianas obtenidas con Dron

Imágenes medianas con dron - 1280x853					
Imagen	**Serie**	**2 núcleos**	**4 núcleos**	**6 núcleos**	**8 núcleos**
1	0.74	0.44	0.28	0.29	0.24
2	0.74	0.42	0.28	0.29	0.24
3	0.71	0.41	0.27	0.29	0.23
4	0.73	0.42	0.27	0.29	0.24
5	0.73	0.41	0.27	0.29	0.24
6	0.74	0.43	0.28	0.29	0.24
7	0.73	0.42	0.27	0.29	0.24
8	0.74	0.42	0.28	0.29	0.24
9	0.73	0.42	0.27	0.29	0.24
10	0.73	0.42	0.27	0.29	0.24
Promedio	**0.73**	**0.42**	**0.27**	**0.29**	**0.24**

Figura 35: Tiempos de ejecución imágenes medianas obtenidas con Dron



2.6.6 Tiempos de ejecución imágenes pequeñas

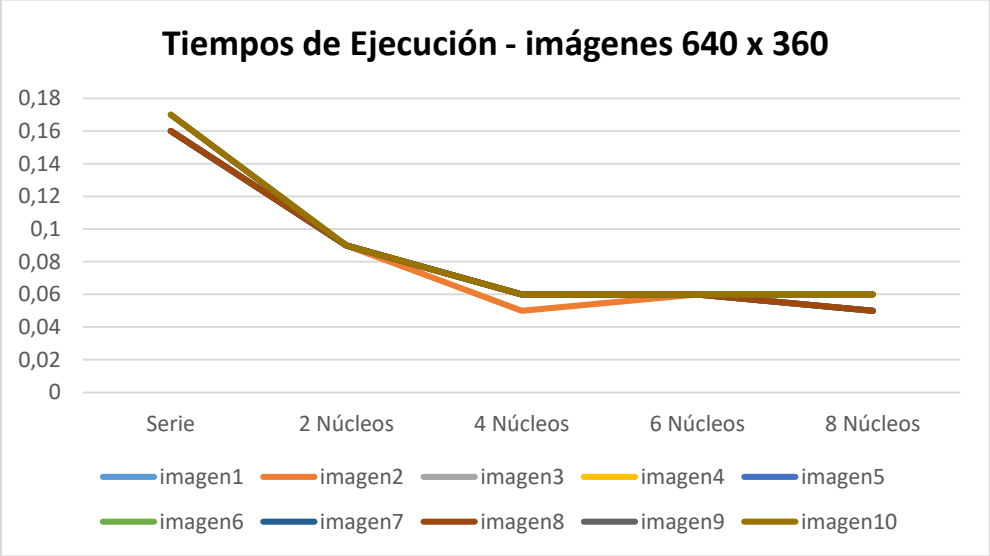
En la Tabla 12 y Figura 36

Figura 36, se presenta los tiempos de ejecución en el procesamiento de imágenes obtenidas con cámara con dimensiones (640x360 pixeles) del algoritmo serial y paralelo (2,4,6 y 8 núcleos).

Tabla 12: Tiempos de ejecución imágenes medianas obtenidas con cámara

Imágenes pequeñas con cámara - 640x360					
Imagen	**Serie**	**2 núcleos**	**4 núcleos**	**6 núcleos**	**8 núcleos**
1	0.16	0.09	0.06	0.06	0.06
2	0.16	0.09	0.05	0.06	0.05
3	0.16	0.09	0.06	0.06	0.05
4	0.16	0.09	0.06	0.06	0.06
5	0.16	0.09	0.06	0.06	0.06
6	0.16	0.09	0.06	0.06	0.05
7	0.16	0.09	0.06	0.06	0.05
8	0.16	0.09	0.06	0.06	0.05
9	0.17	0.09	0.06	0.06	0.06
10	0.17	0.09	0.06	0.06	0.06
Promedio	**0.16**	**0.09**	**0.06**	**0.06**	**0.06**

Figura 36:Tiempos de ejecución imágenes medianas obtenidas con cámara



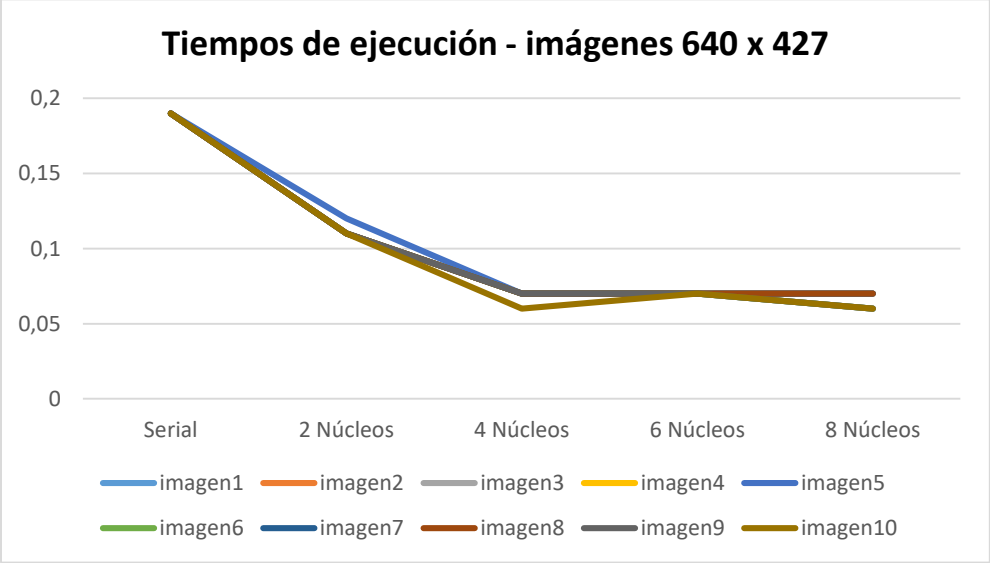
En la Tabla 13 y

Figura 37, se presenta los tiempos de ejecución en el procesamiento de imágenes obtenidas con dron con dimensiones (640x427 pixeles) del algoritmo serial y paralelo (2,4,6 y 8 núcleos).

Tabla 13: Tiempos ejecución imágenes medianas obtenidas con Dron

Imágenes pequeñas con dron - 640x427					
IMAGEN	**Serial**	**2 núcleos**	**4 núcleos**	**6 núcleos**	**8 núcleos**
1	0.19	0.11	0.07	0.07	0.06
2	0.19	0.11	0.07	0.07	0.07
3	0.19	0.11	0.07	0.07	0.06
4	0.19	0.11	0.07	0.07	0.07
5	0.19	0.12	0.07	0.07	0.07
6	0.19	0.11	0.07	0.07	0.06
7	0.19	0.11	0.07	0.07	0.07
8	0.19	0.11	0.07	0.07	0.07
9	0.19	0.11	0.07	0.07	0.06
10	0.19	0.11	0.06	0.07	0.06
Promedio	**0.19**	**0.11**	**0.07**	**0.07**	**0.07**

Figura 37: Tiempos ejecución imágenes medianas obtenidas con Dron



CAPITULO 3

RESULTADOS

3.1 Evaluación de resultados

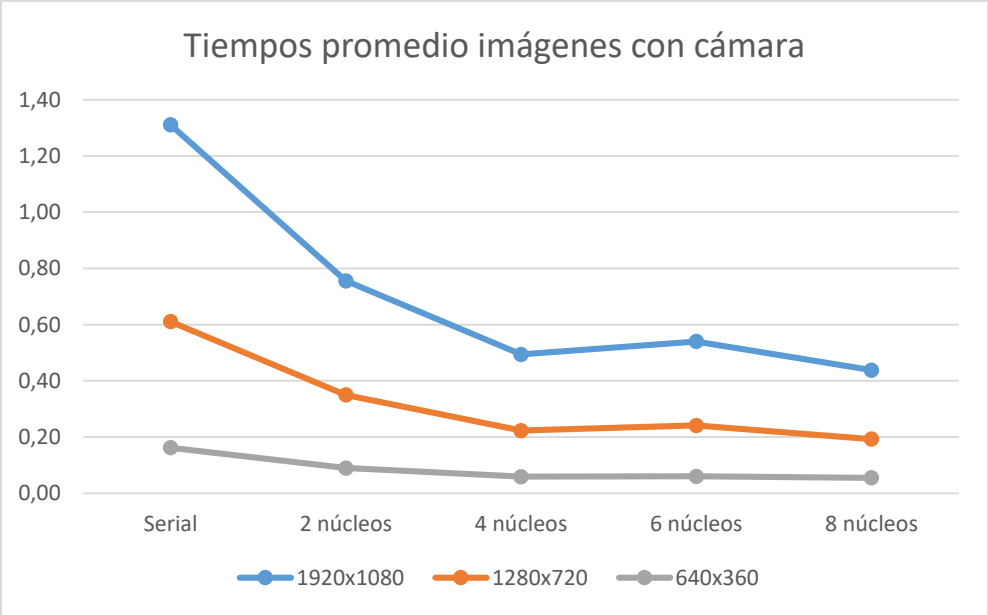
3.1.1 Tiempos

En la Tabla 14 y Figura 38, se presenta los tiempos de ejecución promedio para el procesamiento de imágenes obtenidas con cámara del algoritmo serial y paralelo (2,4,6 y 8 núcleos).

Tabla 14: Tiempos ejecución promedio de imágenes con cámara

Tiempos promedio imágenes con cámara					
Dimensión	**Serial**	**2 núcleos**	**4 núcleos**	**6 núcleos**	**8 núcleos**
1920x1080	1.31	0.76	0.49	0.54	0.44
1280x720	0.61	0.35	0.22	0.24	0.19
640x360	0.16	0.09	0.06	0.06	0.06

Figura 38: Tiempos de ejecución promedio para imágenes con cámara

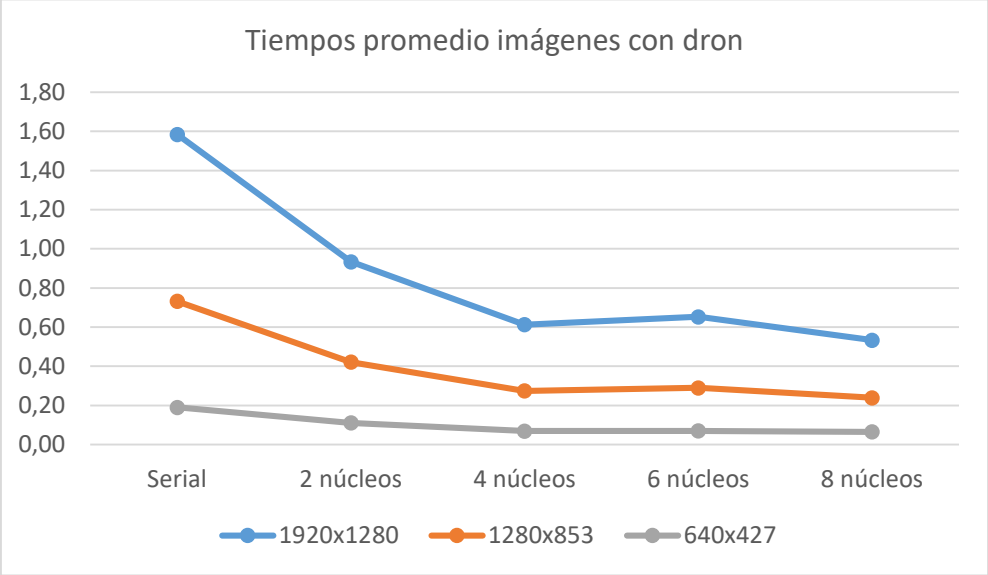


En la Tabla 15 **¡Error! No se encuentra el origen de la referencia.**, se presenta los tiempos de ejecución promedio para el procesamiento de imágenes obtenidas con dron del algoritmo serial y paralelo (2,4,6 y 8 núcleos).

Tabla 15: Tiempos ejecución promedio de imágenes con dron.

Tiempos promedio imágenes con dron					
Imagen	**Serial**	**2 núcleos**	**4 núcleos**	**6 núcleos**	**8 núcleos**
1920x1280	1.59	0.93	0.61	0.65	0.53
1280x853	0.73	0.42	0.27	0.29	0.24
640x427	0.19	0.11	0.07	0.07	0.07

Figura 39: Tiempos ejecución promedio para imágenes con dron



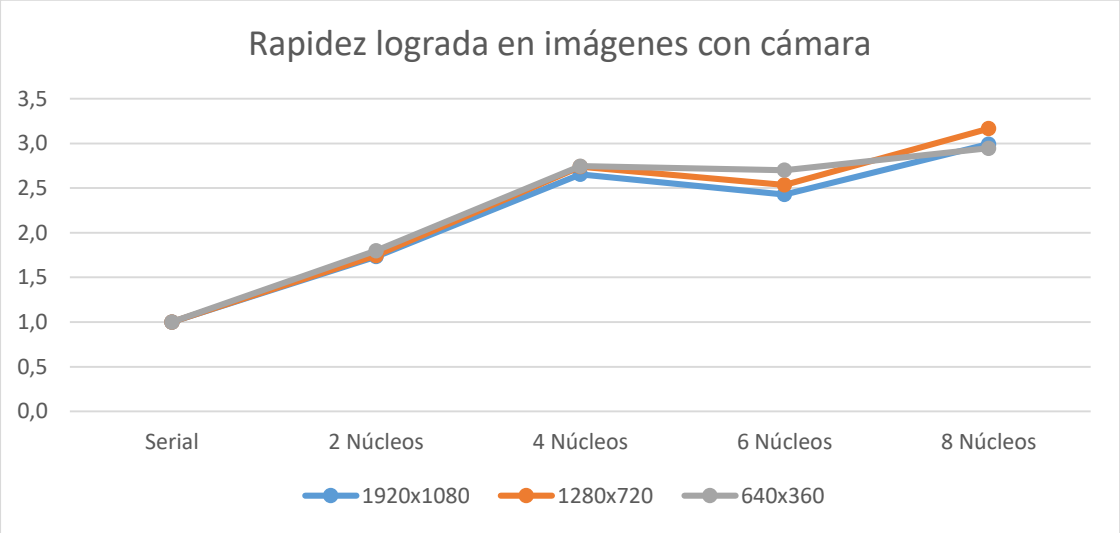
Rapidez

En la Tabla 16 y Figura 40, se presenta la rapidez lograda para el procesamiento de imágenes obtenidas con cámara del algoritmo serial y paralelo (2,4,6 y 8 núcleos).

Tabla 16: Rapidez lograda en imágenes con cámara

Rapidez imágenes con cámara					
Imagen	**Serial**	**2 núcleos**	**4 núcleos**	**6 núcleos**	**8 núcleos**
1920x1080	1.0	1.7	2.7	2.4	3.0
1280x720	1.0	1.7	2.7	2.5	3.2
640x360	1.0	1.8	2.7	2.7	2.9

Figura 40: Rapidez para imágenes con cámara

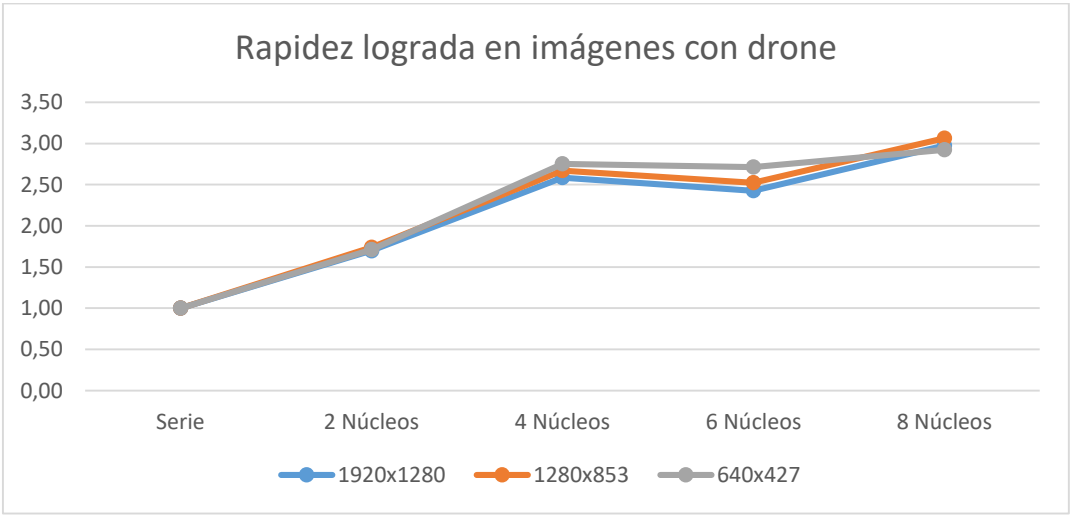


En la Tabla 17 y Figura 41**¡Error! No se encuentra el origen de la referencia.**, se presenta la rapidez lograda para el procesamiento de imágenes obtenidas con dron del algoritmo serial y paralelo (2,4,6 y 8 núcleos).

Tabla 17: Rapidez lograda en imágenes con dron

Rapidez en imágenes con dron					
Imagen	**Serie**	**2 núcleos**	**4 núcleos**	**6 núcleos**	**8 núcleos**
1920x1280	1.00	1.70	2.59	2.43	2.97
1280x853	1.00	1.74	2.67	2.52	3.06
640x427	1.00	1.71	2.75	2.71	2.92

Figura 41: Rapidez para imágenes con dron



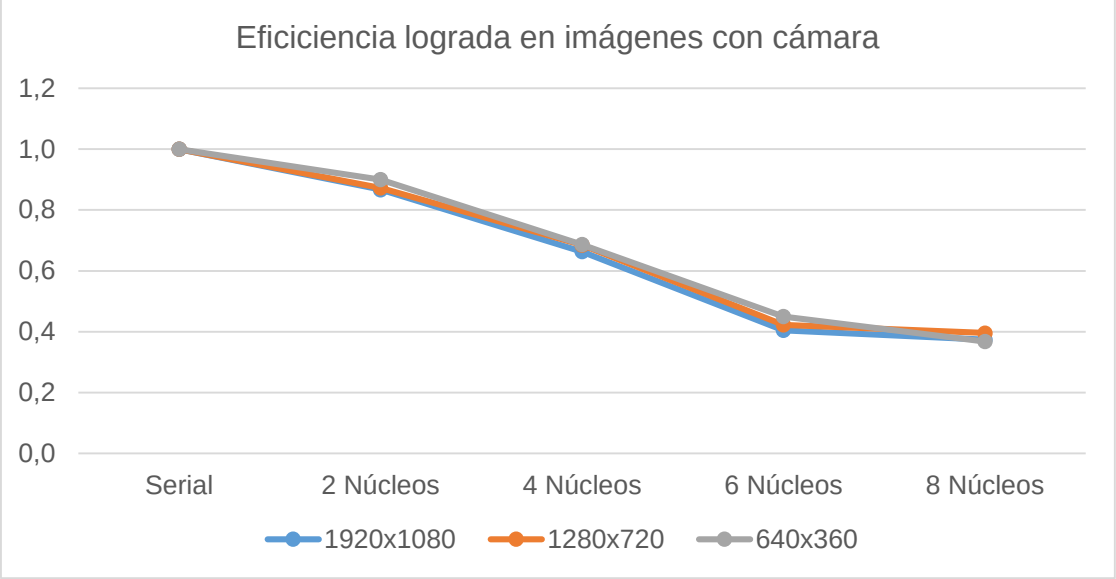
3.1.2 Eficiencia

En la Tabla 18 y Figura 42, se presenta la eficiencia lograda para el procesamiento de imágenes obtenidas con cámara del algoritmo serial y paralelo (2,4,6 y 8 núcleos).

Tabla 18: Eficiencia lograda en imágenes con cámara

Eficiencia imágenes con cámara					
Imagen	**Serial**	**2 núcleos**	**4 núcleos**	**6 núcleos**	**8 núcleos**
1920x1080	1.0	0.9	0.7	0.4	0.4
1280x720	1.0	0.9	0.7	0.4	0.4
640x360	1.0	0.9	0.7	0.5	0.4

Figura 42: Eficiencia para imágenes con cámara

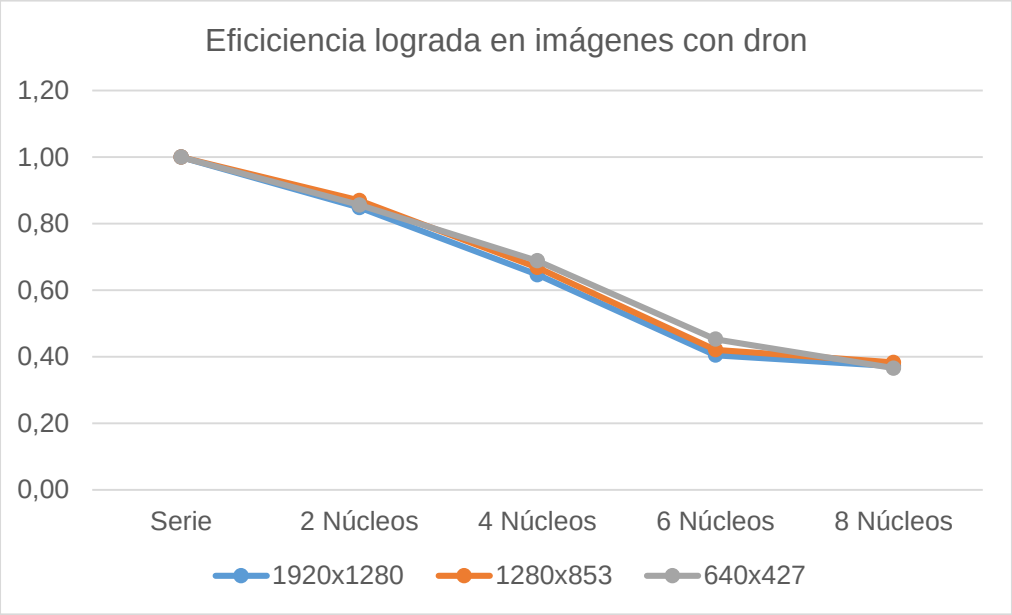


En la Tabla 19 y Figura 43, se presenta la eficiencia lograda para el procesamiento de imágenes obtenidas con dron del algoritmo serial y paralelo (2,4,6 y 8 núcleos).

Tabla 19: Eficiencia lograda en imágenes con dron

Eficiencia en imágenes con dron					
Imagen	**Serie**	**2 núcleos**	**4 núcleos**	**6 núcleos**	**8 núcleos**
1920x1280	1.00	0.85	0.65	0.40	0.37
1280x853	1.00	0.87	0.67	0.42	0.38
640x427	1.00	0.86	0.69	0.45	0.37

Figura 43: Eficiencia para imágenes con dron



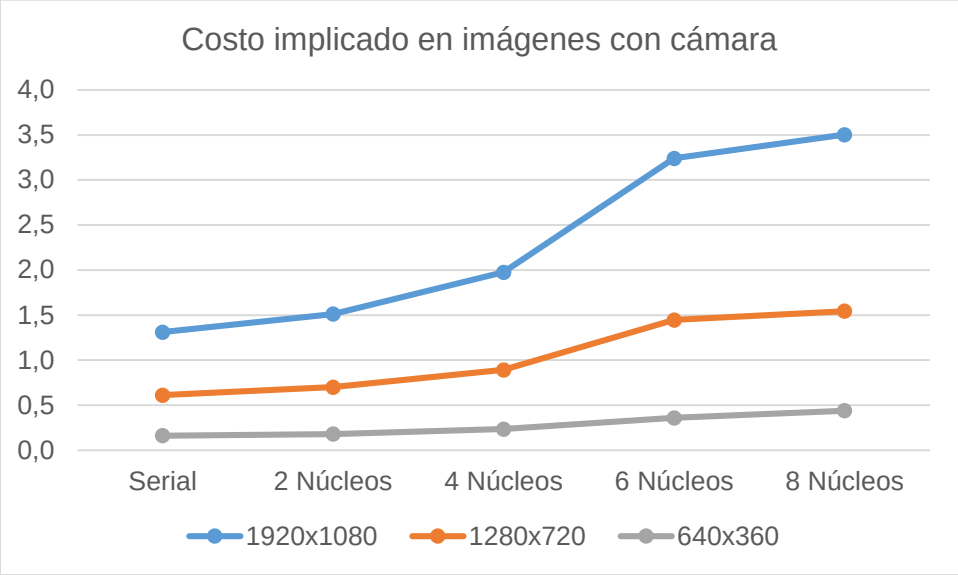
3.1.3 Costo

En la Tabla 20 y Figura 44, se presenta costo computacional implicado para el procesamiento de imágenes obtenidas con cámara del algoritmo serial y paralelo (2,4,6 y 8 núcleos).

Tabla 20: Costo implicado en imágenes con cámara

Costo en imágenes con cámara					
Imagen	**Serial**	**2 núcleos**	**4 núcleos**	**6 núcleos**	**8 núcleos**
1920x1080	1.3	1.5	2.0	3.2	3.5
1280x720	0.6	0.7	0.9	1.4	1.5
640x360	0.2	0.2	0.2	0.4	0.4

Figura 44: Costo para imágenes con cámara

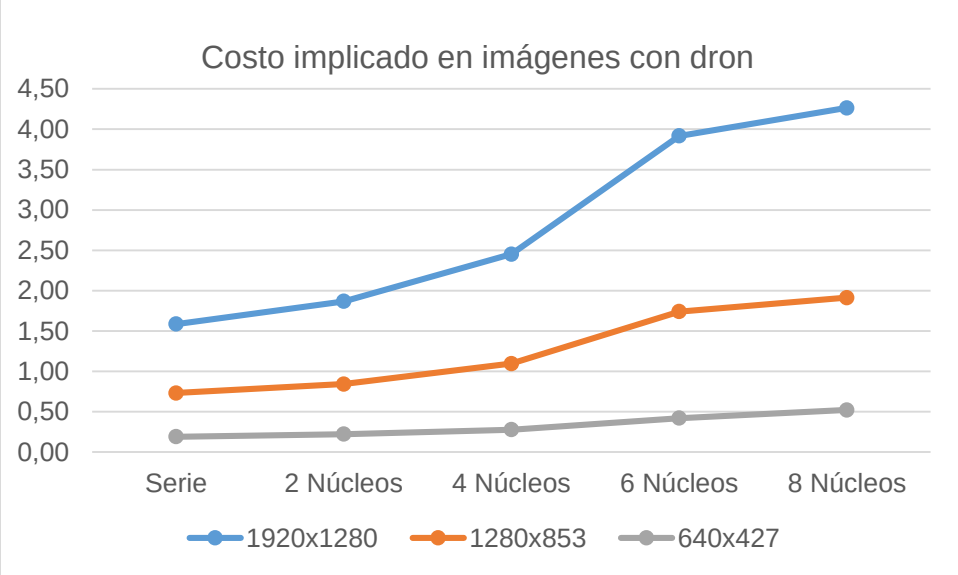


En la Tabla 21 y Figura 45, se presenta el costo computacional implicado para el procesamiento de imágenes obtenidas con dron del algoritmo serial y paralelo (2,4,6 y 8 núcleos).

Tabla 21: Costo implicado en imágenes con dron

Costo en imágenes con dron					
Imagen	**Serie**	**2 núcleos**	**4 núcleos**	**6 núcleos**	**8 núcleos**
1920x1280	1.59	1.87	2.45	3.92	4.26
1280x853	0.73	0.84	1.10	1.74	1.91
640x427	0.19	0.22	0.28	0.42	0.52

Figura 45: Costo para imágenes con dron



3.2 Análisis de resultados

La implementación del algoritmo secuencial y paralelo para la segmentación de imágenes agrícolas fue exitosa, logrando diferenciar la vegetación existente en los cultivos de maíz con 10 imágenes con medidas diferentes; tanto para imágenes adquiridas con cámara e imágenes adquiridas con dron. Los resultados evaluados permitieron verificar con mejor claridad el comportamiento del algoritmo paralelo con diferentes núcleos, ver Tabla 14 y Figura 38. Se puede deducir el comportamiento del algoritmo paralelo disminuye el tiempo de ejecución si se aumenta el número de núcleos para el procesamiento. Aunque se reduce los tiempos de ejecución a medida que aumentan el número de núcleos, la rapidez y eficiencia no presenta la misma tendencia, ver sección 3.1.2 y 3.1.3. el costo computacional para procesar las imágenes con diferentes medidas es relativamente más alto cuando se incrementa el número de núcleos aspecto que se debe tener en cuenta para poder administrar los recursos computacionales de manera razonable.

CONCLUSIONES

Entre los objetivos del proyecto planteado fue investigar y obtener los conceptos más claros del procesamiento de imágenes en la agricultura de precisión como también estudiar un poco más sobre las tres diferentes herramientas de programación en paralelo como son: OpenMP, MPI, CUDA, en las cuales podemos concluir que son herramientas actas y de gran beneficio para investigadores como para programadores que quieran entrarse más a fondo al mundo de las arquitecturas paralelas.

Podemos decir que la computación paralela ha ido poco a poco tomando fuerza en el hardware y software, hoy en día los dispositivos electrónicos incluyen nuevas tecnologías arquitectónicas que son compatibles con lenguajes de programación paralela.

OpenMP se define como una API adecuada para desarrollo de algoritmos en arquitectura Multicore, utilizando el procesamiento de hilos y de memoria mapeada naturalmente ligera, robusto y disponible para ser usada para múltiples códigos sobre diferentes tecnologías disponibles.

Las métricas definidas permitieron evaluar el rendimiento de los algoritmos para imágenes con diferentes dimensiones, dando resultados que verifica la mejora del algoritmo paralelo con relación al algoritmo serial.

OpenMP es una librería compatible con lenguaje de programación de bajo nivel, facilitando el desarrollo e implementación de algoritmos en lenguaje C o C++ en entornos Linux, aprovechando los diferentes núcleos disponibles en los computadores actuales.

Los resultados obtenidos con la implementación del algoritmo paralelo demuestran que es posible mejorar el rendimiento de cualquier algoritmo aplicado en cualquier área de conocimiento. El algoritmo paralelo reduce los tiempos de ejecución a medida que se incrementa el número de núcleos, la rapidez mejora en relación con el número de núcleos. A diferencia que la eficiencia y el costo computacional se reduce con el incremento de núcleos en la ejecución de los algoritmos.

RECOMENDACIONES

Dar continuidad a los algoritmos en otras fases de procesamiento de imágenes u otras aplicaciones informáticas para sistematizar actividades agrícolas.

Revisar y familiarizarse con lenguajes compatibles con sistemas operativos Linux ya que son lenguajes opensource y no requieren de licencias temporales o pagadas. Adicionalmente aprovechar las ventajas de los comandos y herramientas disponibles en los repositorios de Ubuntu para lenguajes de programación.

Las arquitecturas paralelas a medida que va pasando el tiempo requieren de una programación paralela de mucho tiempo de aprendizaje y concentración, por lo que es necesario realizar algunas investigaciones en campo.

Se debe considerar un estudio a fondo sobre arquitecturas paralelas para elegir el lenguaje de programación adecuado para realizar cualquier algoritmo.

Estudiar las funciones y beneficios que ofrecen las herramientas de desarrollo en paralelo como son OpenMP, MPI, CUDA, para realizar una programación paralela.

Implementar los algoritmos en otras arquitecturas paralelas para obtener una rapidez adecuada, una eficiencia aceptable y un costo computacional razonable a medida que incrementa el número de núcleos o procesadores en procesamiento de imágenes.

BIBLIOGRAFÍA

Bernal, F., Albarracín, C., Gaona, J., Giraldo, L., Mosquera, C., Peña, S., . . . Baquero, C. (2017). *Programación Paralela*. Obtenido de http://ferestrepoca.github.io/paradigmas-de-programacion/paralela/paralela_teoria/index.html

Abdul Khalib, Z. I., Ng., H. Q., Elshaikh, M. E., & Othman, M. N. (March de 2020). Optimizing Speedup on Multicore Platform with OpenMP Schedule Clause and Chunk Size. *ResearchGate, IOP Conference Series Materials Science and Engineering 767:012037*. doi:10.1088/1757-899X/767/1/012037

Acosta, F. A., Segura, O. M., & Ospina, Á. E. (Julio - Diciembre de 2012). Guía y fundamentos de la programación en paralelo. (I. 2215-8200, Ed.) *Repositorio Universidad Pontificia Bolivariana, Vol. 2 No. 4 (2012) [6]*. doi:https://repository.upb.edu.co/bitstream/handle/20.500.11912/6577/Gu%c3%ada%20y%20fundamentos%20de%20la%20programaci%c3%b3n%20en%20paralelo.pdf?sequence=1&isAllowed=y

Ahmad, A., Guyonneau, R., Mercier, F., & Belin, É. (June de 2018). An Image Processing Method Based on Features Selection for Crop Plants and Weeds Discrimination Using RGB Images. *Researchers*. doi:10.1007/978-3-319-94211-7_1

Álvarez Pastor, M. L. (2017). *Configuración y ejecución de algoritmos de visión artificial en la tarjeta Nvidia Jetson TK1 DevKit.* Obtenido de Universidad de Alcalá Escuela Politécnica Superior.: https://dspace.uah.es/dspace/bitstream/handle/10017/30204/TFG-Álvarez-Pastor-2017.pdf?sequence=1&isAllowed=y

Baggio, G. (May de 2021). Compositionality in a Parallel Architecture for Language Processing. *ResearchGate, License: CC BY-NC-ND 4.0*. doi:10.1111/cogs.12949

Balcázar-Guerrero, M. (2011). *DESARROLLO DE UN MÓDULO SIG PARA EL MANEJO DE IMÁGENES MULTIESPECTRALES ORIENTADO A LA AGRICULTURA DE PRECISIÓN.* Tesis, Pontificia Universidad Católica del Perú, Lima.

Basso, M., & Pignaton-de-Freitas, E. (March de 2020). A UAV Guidance System Using Crop Row Detection and Line Follower Algorithms. *Springer, Journal of Intelligent & Robotic Systems 97(1)*. doi:10.1007/s10846-019-01006-0

Basso, M., Stocchero, D., Bayan- Henriquez, R. V., Vian, A. L., Bredemeier, C., Konzen, A. A., & Pignaton de Freitas, E. (December de 2019). Proposal for an Embedded System Architecture Using a GNDVI Algorithm to Support UAV-Based Agrochemical Spraying. *Researchers, Sensors 19(19):5397*. doi:10.3390/s19245397

Briceño Coronado, A. A. (29 de 09 de 2012). *Algoritmos paralelos de visión computacional para reconstrucción tridimensional.* editorial academica española eae. Obtenido de https://www.eae-publishing.com/catalogue/details/es/978-3-659-05307-8/algoritmos-paralelos-de-visi%C3%B3n-computacional

Chuidiang. (6 de Diciembre de 2020). *Ejemplos java y C/linux*. Obtenido de http://www.chuidiang.org/clinux/herramientas/makefile.php

Comba, L., Biglia, A., Aimonino, D. R., & Gay, P. (October de 2018). Unsupervised detection of vineyards by 3D point-cloud UAV photogrammetry for precision agriculture. *ScienceDirect, 155*, 84-95. Obtenido de https://www.researchgate.net/publication/328232372_Unsupervised_detection_of_vineyards_by_3D_point-cloud_UAV_photogrammetry_for_precision_agriculture

Demski, A. J., Di Donato, A. L., Maudet, S. F., & Furfaro, A. (2015). *Algoritmo de visión estereo en tiempo real implementado en GPGPU.* Obtenido de Simposio Argentino de Inteligencia Artificial.: http://www.electron.frba.utn.edu.ar/dplab

Dian Bah, M., Hafiane, A., & Canals, R. (May de 2018). Deep Learning with Unsupervised Data Labeling for Weeds Detection on UAV Images. *Research*. doi:10.20944/preprints201809.0088.v1

DJI. (19 de Octuber de 2021). *Mavic Pro - DJI.* Retrieved. Obtenido de (https://www.dji.com/mavic

Echevarría, J. (2008). *Tarjetas gráficas para acelerar el cómputo complejo.* Obtenido de C&T - Universidad de Palermo: http://www.palermo.edu/ingenieria/downloads/pdfwebc&T8/8CyT08.pdf

EduRed. (19 de Octuber de 2021). Ley de Amdahl - EcuRed. Obtenido de (https://www.ecured.cu/Ley_de_Amdahl

Flynn, M. J., & Rudd, K. W. (1996). *Parallel Architectures* (Vol. 28(1)). ACM Computing Survey (CSUR).

Frati, F. E., & De Giusti, A. (2015). *Software para arquitecturas basadas en procesadores de multiples núcleos.* Obtenido de Detección automática de errores de UBLP. : http://postgrado.info.unlp.edu.ar/Carreras/Doctorado/Tesis/2015_Fernando_Emmanuel_Frati.pdf

Fundation, E. (2021). *Eclipse IDE for Scientific Computing*. (Copyright © Eclipse Foundation) Obtenido de clipse.org/downloads/packages/release/photon/rc2/eclipse-ide-scientific-computing

García-Santillán, I., & Pajares, G. (November de 2018). On-line crop/weed discrimination through the Mahalanobis distance from images in maize fields. *Researches*. doi:10.1016/j.biosystemseng.2017.11.003

GNU. (01 de 10 de 2021). *GCC, the GNU Compiler Collection*. (Copyright (C) Free Software Foundation, Inc.) Obtenido de https://gcc.gnu.org/

Hassanein, M., & El-Sheimy, N. (September de 2018). AN EFFICIENT WEED DETECTION PROCEDURE USING LOW-COST UAV IMAGERY SYSTEM FOR PRECISION AGRICULTURE APPLICATIONS. doi:10.5194/isprs-archives-XLII-1-181-2018

Jackerndoff, R., & Audring, J. (May de 2019). 8. The Parallel Architecture. *ResearchGate*. doi:10.1515/9783110540253-008

Jiménez, L. A., Jiménez, L. F., & García, R. D. (2015). Software para el estudio de coberturas vegetales con conceptos de agricultura de precisión. *ResearchDate*.

Jiménez-López, A. F., Jiménez-López, F. R., & Pérez, E. F. (2013). PROCESAMIENTO DIGITAL DE IMÁGENES DE SENSORES REMOTOS PARA APLICACIONES DE AGRICULTURA DE PRECISIÓN. *Revista Colombiana de Tecnologías de Avanzada*. Obtenido de https://www.researchgate.net/publication/309374952_PROCESAMIENTO_DIGITAL_DE_IMAGENES_DE_SENSORES_REMOTOS_PARA_APLICACIONES_DE_AGRICULTURA_DE_PRECISION

Kerkech, M., Hafiane, A., & Canals, R. (October de 2018). Deep leaning approach with colorimetric spaces and vegetation indices for vine diseases detection in UAV images. *ResearchGate*. doi:10.1016/j.compag.2018.10.006

Leiva, F. (2003). LA AGRICULTURA DE PRECISIÓN: UNA PRODUCCIÓN MÁS SOSTENIBLE Y COMPETITIVA CON VISIÓN FUTURISTA. *VIII Congreso de la Sociedad Colombiana de Fitomejoramiento y Producción de Cultivos, Bogotá.* Bogotá. Obtenido de https://www.researchgate.net/publication/228425520_La_agricultura_de_precision_una_produccion_mas_sostenible_y_competitiva_con_vision_futurista

Li, Y., Qian, M., Liu, P., Cai, Q., Li, X., Guo, J., . . . Zhou, Z. (2019). The recognition of rice images by UAV based on capsule network. *Springer*. Obtenido de https://www.researchgate.net/publication/323811115_The_recognition_of_rice_images_by_UAV_based_on_capsule_network

Martinez-Rodríguez, A. (2016). *Sistema de procesamiento de imágenes RGB aéreas para agricultura de precisión.* Universidad Central "Marta Abreu" de las Villas, Santa Clara.

Moran, M. (2016). *Industría, Innovación e Infraestructura* . Obtenido de https://www.un.org/sustainabledevelopment/es/infrastructure/

Moran, M. (2016). *Vida de Ecosistemas Terrestres* . Obtenido de https://www.un.org/sustainabledevelopment/es/biodiversity/

Moreano, S. G., Cajamarca, V. J., & Tenicota, G. A. (diciembre de 10 de 2019). Agricultura de Precisión: Preprocesamiento y Segmentación de Imágenes para Obtención de una Ruta de Navegación Autónoma Terrestre. *SciELO*. Obtenido de http://scielo.senescyt.gob.ec/scielo.php?script=sci_arttext&pid=S1390-01292019000500043

Moreno Díaz, A. L. (2020). *Análisis comparativo de arquitecturas de redes neuronales para la clasificación de imágenes.* Universidad Internacional de la Rioja (UNIR), Madrid. Obtenido de https://reunir.unir.net/bitstream/handle/123456789/10008/Moreno%20D%C3%ADaz-Alejo%2C%20Lara.pdf?sequence=1&isAllowed=y

Naiouf, M., De Giusti, A., De Giusti, L., Chichizola, F., Sanz, V., Pousa, A., . . . Balladini, J. (2012). *Algoritmos Paralelos y Distribuidos.* Obtenido de Fundamentos, Modelos y Aplicaciones. WICC 2012: http://sedici.unlp.edu.ar/bitstream/handle/10915/19392/Documento_completo.Fundamentos, Modelos y Aplicaciones.pdf?sequence=1

Oliveira, H. C., Guizilini, V., Nunes P., I., & Souza, J. R. (April de 2018). Failure Detection in Row Crops From UAV Images Using Morphological Operators. *ResearchGate*. doi:10.1109/LGRS.2018.2819944

Otero Barrera, G. (2019). *Precision Agriculture Management System.* Universidad Central "Marta Abreu" de las Villas, Department of Automation and Computational Systems. Santa Clara: DSpace. Obtenido de https://dspace.uclv.edu.cu/bitstream/handle/123456789/11497/Gustavo_Otero_Barrera.pdf?sequence=1&isAllowed=y

Pajares-Martinsanz, G., & De La Cruz-García, J. M. (2008). *Vision por Computador Imágenes Digitales y Aplicaciones.* Madrid: Paracuellos del Jarama (Madrid); Ra-Ma, 2 ed.

Pan, W., Gong, Y., & Qiu, G. (2019). Parallel curvature filter for high performance image processing. *International Workshop on Advanced Image Technology.* Shenzhhen, China: ResearchGate. doi:10.1117/12.2520823

Pérez- Ortiz, M., Gutiérrez, P. A., Peña- Barragán, J. M., Torres - Sánchez, J., López-Granados, F., & Hervás- Martínez, C. (December de 2016). Machine learning paradigms for weed mapping via unmanned aerial vehicles. *2016 IEEE Symposium Series on Computational.* doi:10.1109/SSCI.2016.7849987

Piccoli, M. F. (2011). *Computación de alto desempeño en GPU.* Obtenido de http://sedici.unlp.edu.ar/bitstream/handle/10915/18404/Documento_completo__.pdf?sequence=1

Printista, M., Gil-Costa, V., Alaniz, M., & Bustos, F. (2011). *Optimización de Algoritmos utilizando Sistemas de Cómputo Híbridos.* Obtenido de XIII Workshop de Investigadores En Ciencias de La Computación, 715–718.: http://sedici.unlp.edu.ar/bitstream/handle/10915/19882/Documento_completo.pdf?sequence=1

Pusdá-Chulde, M. R., Salazar-Fierro, F. A., Sandoval-Pillajo, A. L., Herrera-Granda, E. P., García-Santillán, I. D., & De-Giusti, A. (2019). Image Analysis Based on Heterogeneous Architectures for Precision Agriculture: A Systematic Literature Review. *International Conference on Computer Science, Electronics and Industrial Engineering (CSEI).* ResearchGate. Obtenido de https://www.researchgate.net/publication/336777050_Image_Analysis_Based_on_Heterogeneous_Architectures_for_Precision_Agriculture_A_Systematic_Literature_Review

Ricardo, D. (08 de 2018). *Agricultura de Precisión (AP): Tecnologías para mejorar la competitividad del sector agrario*. Obtenido de ainia: https://www.ainia.es/tecnoalimentalia/tecnologia/agricultura-precision/

Rodríguez González, L. J. (2020). *AGRICULTURA DE PRECISIÓN EN EL MUNDO Y EN COLOMBIA: REVISIÓN BIBLIOGRÁFICA.* Universidad del Valle. Santiago de Cali: Biblioteca Digital. Obtenido de https://bibliotecadigital.univalle.edu.co/bitstream/handle/10893/19416/Agricultura-Precision-Mundo-Rodriguez-Leydi-3745-R696a.pdf?sequence=1&isAllowed=y

Rovira-Más, F., Zhang, Q., Reid, J. F., & Will, J. D. (2003). *Machine Vision Based Automated Tractor Guidance.* Obtenido de International Journal of Smart Engineering System Design, 5(4), 467–480: https://doi.org/10.1080/10255810390445300

Sanz, V. M., De Giusti, A., & Naiouf, M. (2014). *ANÁLISIS DE RENDIMIENTO Y OPTIMIZACIÓN DE ALGORITMOS PARALELOS BEST-FIRST SEARCH SOBRE MULTICORE Y CLUSTER DE MULTICORE.* Obtenido de http://postgrado.info.unlp.edu.ar/Carreras/Doctorado/Tesis/2015_Victoria_Maria_Sanz.pdf

Schiaffino, G. (2018). *DIVISIONES TERRITORIALES DEL TRABAJO Y CIRCUITOS DE LA ECONOMÍA URBANA: LAS EMPRESAS DE SERVICIOS TÉCNICO-CIENTÍFICOS DE AGRICULTURA DE PRECISIÓN EN EL ÁREA CONCENTRADA DE ARGENTINA.* Universidad de Buenos Aires. Directora: Dra. María Laura Silveira. Obtenido de http://157.92.88.55/bitstream/handle/filodigital/11349/uba_ffyl_t_2018_40464.pdf?sequence=1&isAllowed=y

SEMPLADES. (2017). *Toda una Vida 1.* Semplades, 1_148.

Shah, J., Prajapati, H. B., & Dabhi, V. K. (Marzo de 2016). Survey on Detection and Classification of Rice Plant Diseases. *Researchgate*. doi:10.1109 / ICCTAC.2016.7567333

Shanker-Tiwari, P., Kumar-Sahni, R., Prakash-Kumar, S., & Kumar, V. (April de 2019). Precision agriculture application in horticulture. *ResearchGate*. Obtenido de https://www.researchgate.net/publication/341079105_Precision_agriculture_application_in_horticulture

Shi, Z., Dang, H., Liu, Z., & Zhou, X. (Enero de 2020). Detection and Identification of Stored-Grain Insects Using Deep Learning: A More Effective Neural Network. *ResearchGate*. doi:10.1109 / ACCESS.2020.3021830

Shrivastava, V., Pradhan, M., Minz, S., & Thakur, M. (July de 2019). RICE PLANT DISEASE CLASSIFICATION USING TRANSFER LEARNING OF DEEP CONVOLUTION NEURAL NETWORK. *ResearchGate*. doi:10.5194/isprs-archives-XLII-3-W6-631-2019

Stpiczynski, P. (April de 2018). Language-based vectorization and parallelization using intrinsics, OpenMP, TBB and Cilk Plus. *ResearchGate - Springer, The Journal of Supercomputing 74(6):1-12*. doi:10.1007/s11227-017-2231-3

Sucar, L. E., & Gómez, G. (2015). *Visión Computacional*. Obtenido de Instituto Nacional de Astrofísica, Óptica y Electrónica: http://ccc.inaoep.mx/~esucar/Libros/vision-sucar-gomez.pdf

Tenhunen, H., Pahikkala, T., Nevalainen, O., Nevalainen, O., Teuhola, J., Mattila, H., & Tyystjarvi, E. (July de 2019). Automatic detection of cereal rows by means of pattern recognition techniques. doi:10.1016/j.compag.2019.05.002

Trujillo Rasúa, R. A. (2009). *Algoritmos paralelos para la solución de problemas de optimización discretos aplicados a la decdificación de señales.* Tesis.

Universidad de Lleila. (17 de 10 de 2018). Obtenido de Agricultura de Precisión: http://www.grap.udl.cat/es/presentacion/ap.html

Wang, J., Ma, X., Zhu, Y., & Sun, J. (2014). *Efficient parallel implementation of active appearance model fitting algorithm on GPU.* Obtenido de TheScientificWorldJournal, 2014, 528080.: https://doi.org/10.1155/2014/528080