UNIVERSIDAD TÉCNICA DEL NORTE

FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS

CARRERA DE INGENIERÍA EN MECATRÓNICA

Tema:

"DETERMINACIÓN DE LA DEFORESTACIÓN EN LA PROVINCIA DE SUCUMBÍOS A TRAVÉS DE IMÁGENES SATELITALES"

Trabajo de grado previo a la obtención de título de Ingeniero en Mecatrónica

Línea de investigación: Desarrollo Agropecuario y Forestal Sostenible

Autor:

Andrés Israel Grijalva Recalde

Director:

Ing. David Alberto Ojeda Peña, PhD.

Asesor:

Ing. Carlos Xavier Rosero Chandi, PhD.

Ibarra, febrero de 2026

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:	1004740054		
APELLIDOS Y NOMBRES:	Grijalva Recalde Andres Israel		
DIRECCIÓN:	Av. Eugenio Espejo, Reinaldo Chavez 17-53, Ibarra		
EMAIL:	aigrijalvar@utn.edu.ec		
TELÉFONO FIJO:	065013387	**TELÉFONO MÓVIL:**	0990620463

DATOS DE LA OBRA	
TÍTULO:	Determinación de la deforestación en la Provincia de Sucumbíos a través de Imágenes Satelitales
AUTOR (ES):	Andres Israel Grijalva Recalde
FECHA: DD/MM/AAAA	24/02/2026
PROGRAMA:	■ GRADO ☐ POSGRADO
TITULO POR EL QUE OPTA:	Ingeniero en Mecatrónica
ASESOR /DIRECTOR:	Ing. David Alberto Ojeda Peña, PhD. Ing. Carlos Xavier Rosero Chandi, PhD

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 que es (son) el (los) 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 24 días del mes de febrero de 2026

EL AUTOR:

Firmado electrónicamente por:
ANDRES ISRAEL
GRIJALVA RECALDE
Validar únicamente con FirmaEC

Andres Israel Grijalva Recalde

UNIVERSIDAD TÉCNICA DEL NORTE
FACULTAD DE INGENIERÍA EN CIENCIAS APLICAS
CARRERA DE INGENIERÍA EN MECATRÓNICA

CERTIFICACIÓN DEL DIRECTOR DEL TRABAJO DE INTEGRACIÓN CURRICULAR

Ibarra, 24 de febrero de 2026

Ing. David Alberto Ojeda Peña, PhD.
DIRECTOR DEL TRABAJO DE INTEGRACIÓN CURRICULAR

CERTIFICA:

Haber revisado el presente informe final del Trabajo de Integración Curricular, el mismo que se ajusta a las normas vigentes de la Universidad Técnica del Norte; en consecuencia, autorizo su presentación para los fines legales pertinentes.

Firmado electrónicamente por:
DAVID ALBERTO OJEDA PENA
Validar únicamente con FirmaEC

……

Ing. David Alberto Ojeda Peña, PhD.
C.C.: 1757898489

UNIVERSIDAD TÉCNICA DEL NORTE
FACULTAD DE INGENIERÍA EN CIENCIAS APLICAS
CARRERA DE INGENIERÍA EN MECATRÓNICA

APROBACIÓN DEL COMITÉ CALIFICADOR

El Comité Calificado del trabajo de Integración Curricular "DETERMINACIÓN DE LA DEFORESTACIÓN EN LA PROVINCIA DE SUCUMBÍOS A TRAVÉS DE IMÁGENES SATELITALES" elaborado por Andres Israel Grijalva Recalde, previo a la obtención del título de INGENIERO EN MECATRÓNICA, aprueba el presente informe de investigación en nombre de la Universidad Técnica del Norte:

Firmado electrónicamente por:
DAVID ALBERTO OJEDA PENA
Validar únicamente con FirmaEC

……

Ing. David Alberto Ojeda Peña, PhD.
C.C.: 1757898489

Firmado electrónicamente por:
CARLOS XAVIER ROSERO CHANDI
Validar únicamente con FirmaEC

……

Ing. Carlos Xavier Rosero Chandi, PhD
C.C.: 1002515821

DEDICATORIA

Quiero dedicar este triunfo a mis padres, a mi querida madre Diana Recalde y a mi querido padre Ermel Grijalva, ya que sin su esfuerzo nada de esto hubiera sido posible, por brindarme su apoyo incondicional, sus consejos de vida, y su sabiduría. Queridos padres, el sueño de ver a sus hijos como profesionales se ha complido, con años de trabajo, perseverancia, sacrificio, dedicación, nos apoyaron a mi y a mis hermanos, con lo mucho o poco que se tenía, pero sin dejar atrás los sueños. Este triunfo es también un reflejo de los valores que me inculcaron: la perseverancia, la honestidad, la humildad y la fe en los sueños. Hoy no solo celebro un logro académico, sino la herencia de su esfuerzo, y los valores, que me acompañará en cada paso de mi vida. Este título lleva sus nombres grabados, porque cada página escrita refleja su amor y sacrificio.

AGRADECIMIENTO

Quiero agradecer a Dios, por darme la vida, la salud, la capacidad y las aptitudes para poder culminar esta carrera universitaria, por siempre resguardar y guiar nuestros pasos, de mi y de mi familia. A mis padres, por todo su esfuerzo, su cariño, sus enseñanzas y todo lo que pude aprender de ellos, a mis hermanos por su apoyo y su compañía. A mis tíos, tías, primos y abuelitos que me apoyaron y creyeron en mí. Deseo expresar mi más sincero agradecimiento a mi tutor Ing. David Ojeda, quien con su guía, paciencia y rigurosidad académica orientó cada etapa de este trabajo de investigación. Su compromiso, disponibilidad y valiosas observaciones fueron fundamentales para el desarrollo y culminación de esta tesis. Reconozco en su acompañamiento no solo la excelencia profesional, sino también la calidad humana que me motivó a mantener la disciplina y el entusiasmo en este proceso. Expreso mi sincero agradecimiento a mi asesor Ing. Xavier Rosero, por su orientación y compromiso durante esta investigación. Sus observaciones fueron fundamentales para fortalecer la calidad académica. A todos los profesores de la carrera de mecatrónica que impartieron su conocimiento. A mis amigos que con su apoyo y esfuerzo pudimos salir adelante en cada proyecto por más difícil que parecía.

RESUMEN

La provincia de Sucumbíos, en Ecuador, presenta un proceso acelerado de deforestación que afecta la biodiversidad, altera el régimen hidrológico y compromete la estabilidad climática regional. El estudio aborda esta problemática mediante el análisis de imágenes satelitales con el fin de cuantificar la magnitud y la evolución de la pérdida de cobertura forestal. La investigación se sustenta en la necesidad de generar evidencia científica que apoye la gestión sostenible de los recursos naturales y contribuya al cumplimiento de los Objetivos de Desarrollo Sostenible relacionados con la conservación de ecosistemas terrestres. La metodología se estructura a partir de la recopilación de bases de datos satelitales, como Google Earth Engine y Landsat, junto con el desarrollo de un algoritmo en Python que integra librerías de visión por computador, entre ellas OpenCV, NumPy, Matplotlib y Pillow. Se aplican técnicas de segmentación, umbralización y cálculo de áreas deforestadas para cuantificar la cobertura vegetal en distintos periodos y delimitar zonas críticas. Los resultados evidencian una reducción significativa de la cobertura arbórea, con concentraciones mayores en sectores como Cuyabeno, Palma Roja y Lago Agrio, los cuales representan más del 60% de la deforestación registrada. Se identifican, además, impactos ambientales directos, como la alteración del régimen hídrico, la pérdida de biodiversidad y el incremento de la vulnerabilidad climática

Palabras clave: Deforestación, Imágenes Satelitales, Segmentación, Visión Artificial.

ABSTRACT

The province of Sucumbíos, in Ecuador, exhibits an accelerated deforestation process that affects biodiversity, alters the hydrological regime, and compromises regional climate stability. This study addresses the issue through the analysis of satellite images aimed at quantifying the magnitude and temporal evolution of forest cover loss. The research is grounded in the need to generate scientific evidence that supports the sustainable management of natural resources and contributes to the Sustainable Development Goals related to the conservation of terrestrial ecosystems. The methodology is based on the compilation of satellite datasets, including Google Earth Engine and Landsat, and the development of a Python algorithm that integrates computer vision libraries such as OpenCV, NumPy, Matplotlib, and Pillow. Techniques of segmentation, thresholding, and deforested-area calculation are applied to quantify vegetation cover across different periods and identify critical zones. The results indicate a significant reduction in tree cover, with higher concentrations in areas such as Cuyabeno, Palma Roja, and Lago Agrio, which account for more than 60% of the recorded deforestation. Additionally, the study identifies direct environmental impacts, including hydrological regime alteration, biodiversity loss, and increased climate vulnerability.

Keywords: Deforestation, Satellite Imagery, Segmentation, Computer Vision.

ÍNDICE DE CONTENIDOS

ÍNDICE DE FIGURAS

CAPÍTULO 1: INTRODUCCIÓN

1.1 Planteamiento del problema

La Amazonía, con una extensión aproximada de 7 millones de km² [1], es conocida como el "pulmón del mundo" debido a su capacidad para absorber millones de toneladas de dióxido de carbono y producir alrededor del 20% del oxígeno en la Tierra [2]. Esta extensa región comprende áreas de varios países, incluyendo Bolivia, Brasil, Colombia, Ecuador, Guyana, Perú, Venezuela, Surinam y la Guayana Francesa [3]. Según la Organización de las Naciones Unidas para la Alimentación y la Agricultura (FAO), en el mundo se deforestan anualmente cerca de 100.000 km² de bosques [4].

La deforestación global es una de las principales causas del cambio climático y la pérdida de biodiversidad. En 2022, se perdieron aproximadamente 66.000 km² de bosques debido a actividades como la tala ilegal, la expansión agrícola, la minería y los incendios descontrolados. La Amazonía es el ecosistema más afectado por la deforestación, lo que representa una amenaza crítica para el equilibrio climático global. Este fenómeno tuvo mayor impacto a inicios del siglo XX, principalmente en regiones tropicales [5].

Ecuador ocupa el quinto lugar en pérdida de cobertura forestal en la región, solo superado por Brasil, Bolivia, Perú y Colombia. En comparación, su pérdida forestal es mayor que la de Venezuela y Surinam, a pesar de que ambos países poseen importantes porciones de la Amazonía [6]. En particular, la provincia ecuatoriana de Sucumbíos enfrenta una rápida deforestación, con serias repercusiones para la biodiversidad, el clima local y los recursos naturales, generando pérdidas de hasta 1.840 km2 de bosques deforestados [7].

De acuerdo con el Ministerio de Ambiente y Agua de Ecuador (MAAE), cada año se pierden en promedio 943,53 km² de bosques en el país, una cifra significativa para su territorio en comparación con otras naciones de la región. Entre 1992 y 2018. Ecuador ha experimentado la

pérdida de más de 20.000 km² de bosques tropicales, lo que representa cerca del 7,8% de su superficie total. Esta pérdida de ecosistemas, como bosques, manglares y selvas tropicales, también pone en riesgo la existencia de miles de especies únicas, ya que Ecuador es uno de los países más biodiversos del planeta, con un alto número de especies endémicas [8].

La gran pérdida forestal podría alterar el ciclo hídrico, ya que la densidad de los bosques contribuye a atraer lluvias; su disminución podría llevar a sequías y crear condiciones de desertificación. Para evaluar y cuantificar las áreas deforestadas, en este trabajo de integración curricular se propone realizar un análisis satelital que permita determinar el impacto ambiental en la Amazonía ecuatoriana.

1.2 Objetivos

1.2.1 General

- Determinar la deforestación en la provincia de Sucumbíos mediante imágenes satelitales.

1.2.2 Específicos

- Seleccionar la base de datos de imágenes satelitales que contengan la zona en estudio.
- Desarrollar un programa computacional que procese las imágenes satelitales.
- Calcular las áreas deforestadas en la provincia de Sucumbíos.

1.3 Alcance

Se desarrolla un programa computacional que utilice visión artificial para procesar imágenes satelitales a partir de las imágenes seleccionadas. Además, este programa lleva a cabo un análisis que permita, al final, proporcionar información sobre la evolución de la deforestación, si hubiere, ocurrida en la provincia de Sucumbíos.

1.4 Justificación

Esta investigación se fundamenta en tres aspectos clave:

En el ámbito ambiental, se busca concientizar a la sociedad sobre la necesidad de controlar la deforestación en el "pulmón del mundo," con énfasis en la región amazónica de Ecuador. Además, se apoya el monitoreo de los Objetivos de Desarrollo Sostenible (ODS), con el fin de frenar la pérdida de biodiversidad y gestionar de manera sostenible los ecosistemas terrestres.

Desde el punto de vista tecnológico, se plantea desarrollar una herramienta computacional para el análisis de imágenes satelitales, centrada en detectar y evaluar la deforestación.

En el ámbito social, esta investigación aspira a proporcionar a la comunidad global una alerta que facilite el control y la implementación de medidas preventivas para proteger la Amazonía ecuatoriana.

CAPÍTULO 2: MARCO REFERENCIAL

2.1 Antecedentes

En 2022, en la Universidad Politécnica Salesiana se realizó un análisis multitemporal de imágenes satelitales para evaluar el estado de la deforestación en las provincias de Pastaza y Orellana, ubicadas en la región amazónica de Ecuador. Estas provincias contienen la mayor extensión de bosque primario de la región, por lo que se utilizó esta evaluación para entender mejor su conservación. El análisis incluyó el uso de imágenes satelitales de Landsat 7 y 8, las cuales fueron procesadas y clasificadas en cuanto a uso de suelo y cobertura vegetal mediante los programas ENVI y ArcMap. A partir de los datos obtenidos, se aplicó la fórmula de la FAO para calcular la tasa de deforestación, revelando cambios en la cobertura vegetal, así como el impacto de actividades mineras y petroleras en el área estudiada [9].

Por otro lado, Tudorescu elaboró un índice compuesto destinado a evaluar los recursos naturales, específicamente tierras con potencial para sustentar vegetación. Además, se implementó una solución de computación en la nube basada en una arquitectura de microservicios que permite monitorear recursos naturales como el agua, el suelo y la vegetación mediante imágenes satelitales provistas por SentinelHub. Estas imágenes son procesadas para obtener los valores de índices como NDWI (Índice de Diferencia Normalizada de Agua), BSI (Índice de Suelo Desnudo), NDVI (Índice de Diferencia Normalizada de Vegetación) y NDMI (Índice de Diferencia Normalizada de Humedad). A partir de los datos obtenidos, se realizó el cálculo del índice de calidad y se hicieron comparaciones para el área especificada en función de estos índices [10].

En otras latitudes, Yang, propuso métodos de predicción de la radiación solar utilizando imágenes satelitales y modelos de aprendizaje profundo para optimizar la previsión en sistemas de energía solar. La radiación solar es esencial para la gestión de energía en fuentes

fotovoltaicas; sin embargo, la predicción de nubes y la radiación presenta dificultades debido a la variabilidad de las condiciones atmosféricas. Este estudio propone el uso de dos modelos de aprendizaje profundo junto con un enfoque físico, aplicados a datos satelitales para estimar la irradiancia solar. Estos métodos fueron evaluados en los Países Bajos [11].

En el mismo contexto, Shuangcheng, propone un novedoso enfoque basado en aprendizaje profundo para identificar daños en carreteras a partir de imágenes satelitales de alta resolución. La detección de daños en vías es fundamental en situaciones de desastre, donde el acceso rápido a la infraestructura vial es crucial para las operaciones de rescate. En el estudio se introduce un modelo de segmentación, denominado RDSeg, que combina el codificador ViT-B, previamente entrenado con el modelo SAM (Segment Anything Model), y un codificador CNN ResNet-18 para mejorar la precisión en la segmentación de daños a nivel de píxel. Para el entrenamiento de este modelo, se desarrolló un conjunto de datos denominado CAU-RoadDamage, que incluye imágenes de áreas afectadas por terremotos en China [12].

De igual manera, Zhao analizó la calidad del agua en cuerpos interiores de Zhejiang (China) usando imágenes Sentinel-2 y aprendizaje automático. Con datos de más de 300 estaciones de monitoreo, desarrolló modelos para estimar CODMn, TN, TP, NH3-N y turbidez. Mediante XGBoost, obtuvo alta correlación con mediciones de campo, permitiendo observar la variación espacial y temporal de estos parámetros. [13].

2.2 Bases Teóricas

2.2.1 Deforestación

La deforestación es la eliminación total o parcial de los bosques, generalmente provocada por actividades humanas como la agricultura, la ganadería, la minería y la urbanización. Aunque la naturaleza puede causar la pérdida de árboles a través de incendios forestales o fenómenos climáticos extremos [14].

2.2.2 Causas de la deforestación

Agricultura intensiva y ganadería

La expansión de áreas agrícolas constituye uno de los factores más determinantes en los procesos de deforestación. De manera complementaria, la práctica de ganadería extensiva requiere amplias superficies de pastizales, generando la sustitución de coberturas boscosas nativas [14].

Tala indiscriminada de árboles

La explotación forestal se orienta a extraer madera utilizada en la construcción, la elaboración de mobiliario y la producción de papel. En numerosos países, esta práctica se lleva a cabo de forma ilegal y sin implementar procesos de reforestación. La explotación forestal ilegal en Ecuador representa una amenaza crítica para la biodiversidad y el equilibrio ecológico, especialmente en la Amazonía y los bosques secos. La extracción no autorizada de madera se realiza sin prácticas sostenibles, generando pérdida de hábitats y afectaciones a comunidades locales [14].

Urbanización y construcción de infraestructura

La expansión urbana ha provocado la desaparición de extensas áreas boscosas, generando cambios en los ecosistemas y afectando los servicios que estos ofrecen a la biodiversidad y a las personas. En Ecuador, la expansión urbana ha generado la desaparición de bosques en regiones como la Amazonía, alterando ecosistemas y reduciendo servicios ambientales esenciales, entre ellos la regulación hídrica, la captura de carbono y la protección del suelo. La pérdida de hábitats naturales afecta la biodiversidad y a las comunidades dependientes de los recursos forestales, mientras que la urbanización incrementa la contaminación y contribuye al cambio climático. Pese a los esfuerzos de conservación, el crecimiento urbano continúa siendo un desafío para compatibilizar desarrollo y sostenibilidad ecológica [14].

Minería y extracción de recursos

La minería a cielo abierto y la extracción de petróleo han arrasado vastas áreas de bosques en naciones como Brasil, Colombia y Perú. Más allá de la deforestación, estas prácticas generan contaminación en ríos y suelos debido al uso de compuestos tóxicos. En Ecuador, la minería y la extracción petrolera generan impactos ambientales severos, principalmente en la Amazonía. Estas actividades ocasionan deforestación y contaminación de ríos y suelos con compuestos tóxicos como mercurio e hidrocarburos, lo que afecta la biodiversidad y la salud de las comunidades locales [14].

2.2.3 Consecuencias de la deforestación

Pérdida de biodiversidad.

Los bosques son el principal hábitat de la mayoría de las especies terrestres. La deforestación destruye estos ecosistemas, amenazando la supervivencia de flora y fauna esenciales y comprometiendo la estabilidad ecológica [14].

Cambio climático y calentamiento global

La cobertura arbórea actúa como sumidero de dióxido de carbono, contribuyendo a la regulación térmica del sistema climático global. La reducción de masas forestales intensifica el desequilibrio atmosférico y acelera el cambio climático, lo que incrementa la frecuencia y magnitud de eventos extremos tales como huracanes, incendios forestales y olas de calor [14].

Alteraciones en el ciclo del agua

La disminución de la cobertura forestal reduce la disponibilidad de humedad atmosférica, altera los patrones de precipitación y favorece la ocurrencia de sequías en distintas regiones. Este proceso genera desequilibrios hidrológicos que afectan la recarga de acuíferos y la estabilidad de ecosistemas dependientes del régimen hídrico [14].

Erosión del suelo y desertificación

La cobertura forestal contribuye a la estabilización del suelo y a la prevención de procesos de degradación. Su eliminación disminuye la disponibilidad de nutrientes, incrementa la erosión y reduce la capacidad productiva del terreno, favoreciendo la expansión de la desertificación [14].

Impacto en comunidades indígenas y locales

Diversas comunidades indígenas mantienen una relación de dependencia con los ecosistemas forestales para garantizar su seguridad alimentaria, provisión de materiales de vivienda y preservación cultural. La deforestación indiscriminada genera procesos de desplazamiento forzado y compromete la continuidad de sus sistemas de subsistencia, incrementando el riesgo de pérdida sociocultural y demográfica [14].

2.2.4 Estadísticas

La Fig 2.1 evidencia la pérdida de bosque primario en la provincia de Sucumbíos, Ecuador, durante el período 2002–2024. En este intervalo se registró una reducción de 7.0 kha de bosque húmedo primario, equivalente al 41% de la cobertura arbórea inicial en la zona. La línea discontinua indica una disminución acumulada del 5% en el porcentaje de bosque restante. Las barras verdes representan la magnitud de la pérdida anual, mientras que se advierte que los procedimientos metodológicos aplicados para la obtención de datos han variado a lo largo del tiempo, particularmente antes y después de 2015, lo que exige cautela en la comparación de resultados [15].

Fig. 2.1 Superficie de pérdida forestal dentro de la extensión de bosque primario, en Ecuador, entre 2002 y 2024 [15]

Entre 2001 y 2024, dos zonas de la provincia de Sucumbíos concentraron el 68% de la deforestación total registrada. Como se evidencia en la Fig. 2.2, la zona de Lago Agrio presentó la mayor pérdida de cobertura arbórea, con 76 mil hectáreas, valor significativamente superior al promedio de 27 mil hectáreas [16].

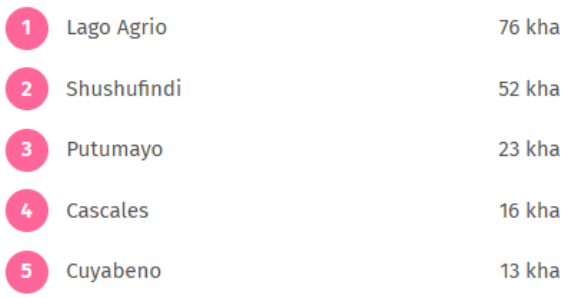

Fig. 2.2 Zonas de la provincia de Sucumbíos con mayor perdida forestal [16]

2.2.5 Métodos para determinar la deforestación

Teledetección satelital

Los satélites Landsat, Sentinel y MODIS ofrecen la posibilidad de analizar series temporales y observar variaciones en la cobertura de los bosques [17].

- Ventajas: brindan alcance mundial, disponibilidad sin costo y una frecuencia elevada de adquisición de imágenes.
- Desventajas: la presencia de nubes en regiones tropicales y la necesidad de aplicar técnicas avanzadas de procesamiento para obtener resultados precisos.

Fig. 2.3 Imagen Satelital [17]

Inventarios forestales y trabajo de campo

Los métodos clásicos de medición forestal se basan en la instalación de parcelas representativas, dentro de las cuales se registran el número de árboles, la diversidad de especies, el volumen de madera y la biomasa, junto con una valoración del nivel de conservación del área [18].

- Ventaja: Los inventarios forestales ofrecen una gran exactitud, ya que permiten recopilar información muy precisa y confiable acerca de las especies presentes, la biomasa y las condiciones del bosque.

- Desventaja: Su aplicación implica altos costos y una cobertura reducida, pues resultan complejos de implementar a gran escala debido al elevado consumo de tiempo, personal y recursos.

Fig. 2.4 Trabajo de Campo [18]

Modelos estadísticos y geoespaciales

La aplicación de Sistemas de Información Geográfica (GIS) permite integrar y superponer diferentes capas de información, como el uso del suelo, la red vial, las zonas protegidas y las concesiones agrícolas [19].

- Ventaja: La capacidad de predicción de estos métodos posibilita proyectar escenarios futuros de deforestación, gracias a la integración de diversas variables en el análisis.

- Desventaja: La fiabilidad de los resultados depende en gran medida de la calidad de los datos; cuando la información empleada es incompleta o está desactualizada, los análisis pueden perder precisión y generar conclusiones poco seguras.

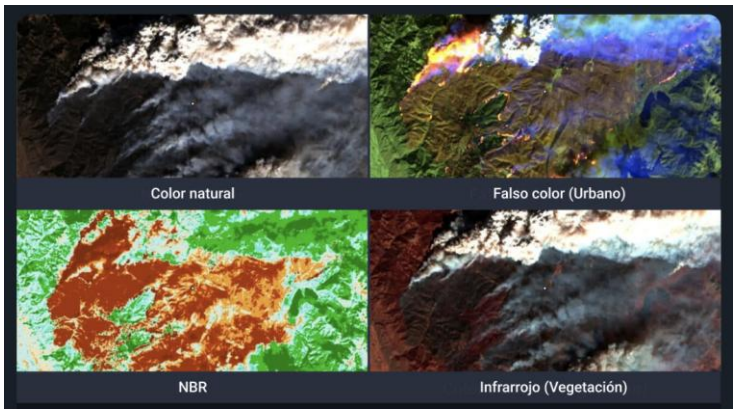

Fig. 2.5 Análisis espacial de humo intenso [19]

2.2.6 Imágenes satelitales

Una imagen satelital puede entenderse como una representación fotográfica de la Tierra obtenida desde el espacio mediante satélites. Estos dispositivos cuentan con sensores y cámaras capaces de registrar información en distintas longitudes de onda, lo que permite recopilar datos diversos sobre la superficie terrestre. Gracias a ello, las imágenes satelitales resultan

fundamentales para lograr una visión más precisa del planeta, su entorno y las transformaciones que en él se producen [20].

2.2.7 Técnicas de análisis de imágenes Satelitales.

El análisis de imágenes satelitales proporciona datos esenciales para los usuarios comerciales de distintos ámbitos, entre ellos la agricultura, la gestión forestal y las iniciativas de sostenibilidad [21].

Índices espectrales

Los índices espectrales permiten identificar de manera automática distintas características del paisaje en imágenes Landsat, tales como nieve y hielo, cobertura nubosa, vegetación y cuerpos de agua [22].

- **NDVI (Índice de vegetación de diferencia normalizada)**

  Es un índice que refleja el nivel de verdor, la densidad y el estado de salud de la vegetación en cada píxel de una imagen satelital, este índice de vegetación se ha consolidado como uno de los más empleados en el ámbito de la teledetección, y la agricultura digital figura entre los sectores que más se benefician de sus aportes [22].

- **NDWI (Índice de agua de diferencia normalizada)**

  Se emplea para realzar la presencia de cuerpos de agua en imágenes satelitales. Esto se logra al disminuir de forma significativa la reflectancia del suelo y de la vegetación, lo que permite que las superficies acuáticas se distingan con mayor claridad en la representación [22]

Análisis multiespectral

Comprende la captura de un mismo objeto a partir de la combinación de la luz reflejada o emitida en distintas bandas y rangos espectrales. En otras palabras, corresponde a un conjunto de imágenes de la misma escena registradas en diferentes longitudes de onda. Generalmente, una imagen destinada al análisis multiespectral se obtiene mediante la adquisición de entre tres y quince bandas espectrales [23].

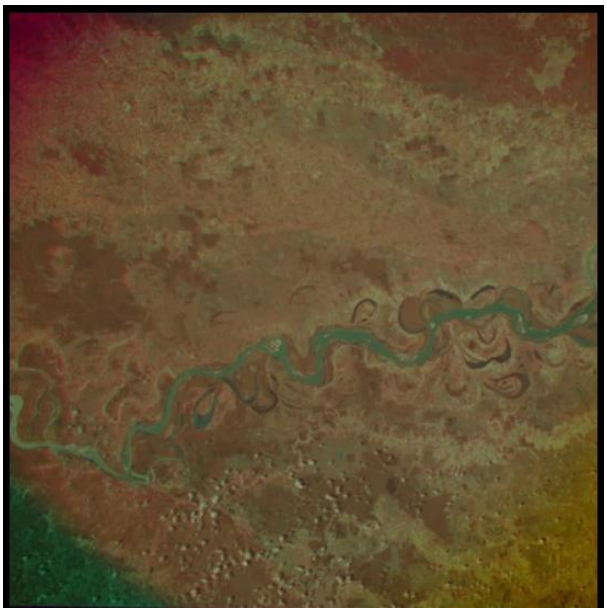

Fig. 2.6 Imagen Multiespectral del Rio Mississipi obtenida a través de la combinación de tres bandas [23].

Análisis hiperespectral

La imagen hiperespectral constituye una técnica que recopila y analiza información a lo largo del espectro electromagnético, permitiendo obtener el espectro correspondiente a cada píxel de una imagen. El espectro electromagnético abarca todas las formas de radiación lumínica, desde las ondas de radio de gran longitud, pasando por las microondas, la radiación infrarroja, la luz visible, los rayos ultravioletas y los rayos X, hasta llegar a los rayos gamma de longitud extremadamente corta, la mayoría de los cuales no son perceptibles por el ojo humano [24].

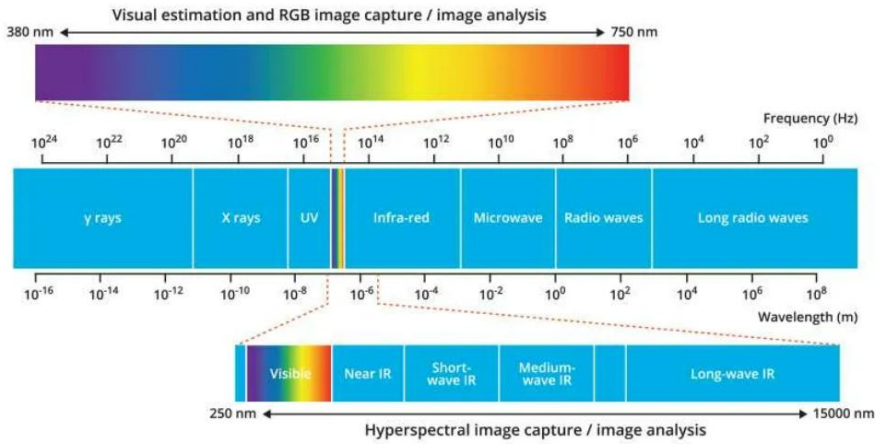

Fig. 2.7 La imagen hiperespectral capturan longitudes de onda de 250 nm a 15 000 nm e infrarrojo térmico [24].

Detección de cambios

Son los responsables habituales de realizar este tipo de análisis. En términos generales, la detección de cambios en teledetección y en los Sistemas de Información Geográfica (SIG) consiste en identificar las diferencias entre dos imágenes satelitales tomadas antes y después de un evento específico. Si bien esta técnica se aplica principalmente en el ámbito empresarial de diversos sectores, también posee usos no comerciales [25].

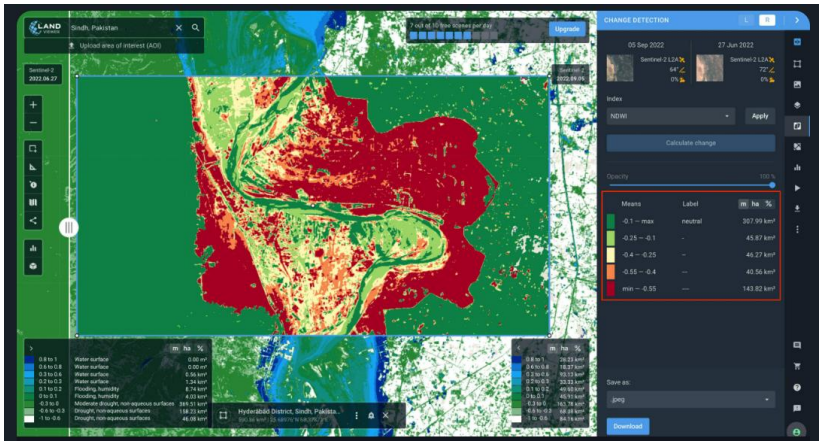

Fig. 2.8 Visualización de los cambios de los contornos del río después de la inundación en EOSDA LandViewer [25].

Segmentación

La segmentación de imágenes es una técnica de visión artificial que consiste en dividir una imagen digital en conjuntos definidos de píxeles (segmentos), con el fin de facilitar la detección de objetos y otras tareas asociadas. Al organizar la información visual compleja en segmentos con formas determinadas, este procedimiento posibilita un procesamiento más eficiente y sofisticado de las imágenes [26].

2.2.8 Segmentación

La segmentación de imágenes es un método de visión artificial que consiste en dividir una imagen digital en conjuntos diferenciados de píxeles (segmentos), con el objetivo de facilitar la identificación de objetos y otras tareas vinculadas. Al descomponer la información visual compleja en segmentos con formas definidas, esta técnica posibilita un procesamiento más eficiente y sofisticado de las imágenes [27].

Umbralización

La umbralización es una técnica específica de segmentación de imágenes que convierte una imagen en binaria, es decir, con solo dos valores posibles (0 y 1, o blanco y negro), en función de un valor de intensidad, en ciertos casos, de color dentro del espacio HSV. De este modo, los píxeles cuya intensidad supera el umbral se asignan a una clase (por ejemplo, vegetación), mientras que aquellos con valores inferiores se clasifican [28].

En términos generales, los métodos de umbralización generan imágenes binarias al clasificar los píxeles según si su intensidad se encuentra por encima o por debajo de un valor umbral establecido. Para determinar dicho valor de manera óptima, suele emplearse el método de Otsu, el cual busca minimizar la variación interna dentro de cada clase [28].

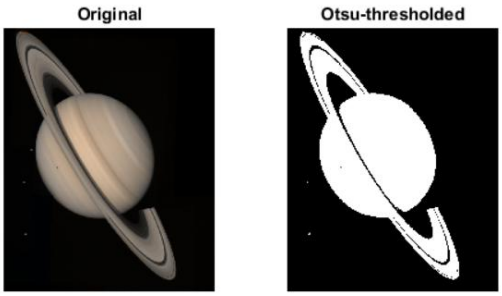

Fig. 2.9 Una imagen de Saturno antes y después de aplicar la umbralización [26].

Método de Otsu

El método de Otsu es una técnica ampliamente utilizada en visión artificial y procesamiento de imágenes para la determinación automática del umbral. Su propósito es establecer un valor óptimo que permita separar de manera efectiva los píxeles de una imagen en dos categorías: fondo y primer plano [29].

El procedimiento se basa en el análisis del histograma de la imagen, el cual refleja la distribución de intensidades de los píxeles. A partir de este análisis, Otsu busca el umbral que reduzca al mínimo la varianza intraclase o, de forma equivalente, que maximice la varianza interclase. Con ello, se obtiene el umbral que mejor distingue las regiones de interés del fondo, convirtiéndose en una herramienta clave para la segmentación de imágenes [29].

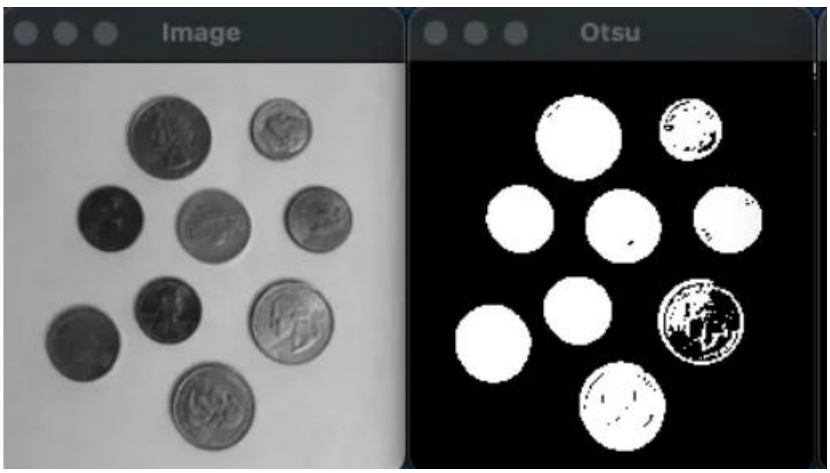

Fig. 2.10 Imagen utilizamos una técnica de limpieza de Otsu [27].

2.2.9 Lenguajes de programación para el tratamiento de imágenes

Python

Su versatilidad y simplicidad de uso, Python se ha consolidado como una opción destacada para el procesamiento y la manipulación de imágenes. Con el apoyo de bibliotecas especializadas como OpenCV y Pillow, este lenguaje se ha transformado en una herramienta esencial para el desarrollo de aplicaciones de visión por computadora y tratamiento de imágenes digitales [30].

- Ventajas:

    1. Facilidad de uso y aprendizaje: Python se distingue por su sintaxis sencilla y clara, diseñada para favorecer la legibilidad del código.

    2. Lenguaje versátil y ampliable: Su flexibilidad le permite adaptarse a múltiples escenarios, desde la creación de scripts básicos hasta la implementación de sistemas avanzados de inteligencia artificial.

    3. Amplio ecosistema de bibliotecas: Python cuenta con una extensa gama de bibliotecas y frameworks, entre los que destacan NumPy, SciPy y TensorFlow, consolidándolo como una pieza fundamental en el ámbito de la ciencia de datos y la inteligencia artificial.

- Desventajas:

    1. Rendimiento: Al ser un lenguaje interpretado, Python puede presentar ciertas limitaciones en cuanto a velocidad de ejecución, especialmente si se lo compara con lenguajes compilados como C++.

    2. Multiprocesamiento: Aunque dispone de módulos que permiten la concurrencia, Python enfrenta restricciones debido al Global Interpreter Lock (GIL), lo que puede convertirse en un impedimento para la optimización de aplicaciones que requieren múltiples hilos de ejecución.

Java

Java es un lenguaje ampliamente utilizado en el procesamiento de imágenes, sobre todo en el desarrollo de aplicaciones de escritorio y móviles. Gracias a bibliotecas como Java Advanced Imaging (JAI) y JavaFX, los programadores pueden diseñar aplicaciones gráficas robustas y escalables [31].

- Ventajas:
    1. Portabilidad: Uno de los principios fundamentales de Java es “write once, run anywhere”, lo que garantiza una amplia portabilidad en distintas plataformas.
    2. Madurez y estabilidad: Tras varias décadas de evolución, Java dispone de un ecosistema consolidado de bibliotecas, frameworks y herramientas de desarrollo que le otorgan gran solidez y confiabilidad.

3. Multitarea y seguridad: El lenguaje incorpora un modelo de concurrencia robusto junto con mecanismos de seguridad avanzados, lo que lo hace especialmente adecuado para entornos empresariales y aplicaciones que involucran transacciones críticas.

- Desventajas:

1. Verbosidad: A menudo se señala que Java requiere una cantidad considerable de código repetitivo o "boilerplate" incluso para la ejecución de tareas sencillas.

2. Rendimiento en interfaces gráficas: Las aplicaciones de escritorio desarrolladas en Java suelen mostrar un desempeño menos fluido y responsivo en comparación con las aplicaciones nativas, particularmente cuando se trata de interfaces gráficas complejas.

Lenguaje C++

C++ es un lenguaje de programación de alto rendimiento, ampliamente empleado en escenarios donde se demanda gran velocidad y eficiencia en el procesamiento de imágenes, particularmente en el desarrollo de sistemas embebidos y en aplicaciones de tiempo real [32].

- Ventajas:

1. Ofrece gran rendimiento y posibilidades de optimización.

2. Es ampliamente utilizado en aplicaciones industriales y en sistemas de tiempo real.

- Desventajas:

    1. Presenta una sintaxis más compleja y requiere una gestión manual de la memoria.
    2. Su curva de aprendizaje resulta más exigente en comparación con lenguajes como Python.

Matlab

Es una herramienta muy empleada en ámbitos académicos y de investigación en ingeniería, ya que su Image Processing Toolbox proporciona un entorno integrado que facilita la visualización, segmentación y optimización de imágenes, evitando la necesidad de desarrollar infraestructuras complejas desde cero [33].

- Ventajas:
    1. Proporciona un marco integrado que facilita tanto la exploración visual como el estudio de datos.
    2. Destaca por su eficiencia en la creación de prototipos rápidos, favoreciendo la experimentación inmediata.
    3. Ofrece una extensa colección de herramientas específicas para el procesamiento de imágenes, lo que simplifica el desarrollo sin necesidad de implementar recursos adicionales desde cero.

- Desventajas:
    1. Costo elevado: Requiere una licencia de pago, la cual suele ser costosa.

2. No ofrece la misma adaptabilidad que Python en el desarrollo de proyectos de código abierto.

2.2.10 Librerías para el procesamiento de imágenes en Python

Estas librerías se basan en la conversión de datos visuales en matrices numéricas con el fin de aplicar filtros, reconocer objetos u optimizar la calidad de una imagen. Proporciona un lenguaje de programación de uso extendido en áreas como el desarrollo web, la creación de software, la ciencia de datos y el aprendizaje automático (ML) [34].

NumPy

NumPy es una de las librerias fundamentales de Python, ya que proporciona soporte para trabajar con arrays. En este contexto, una imagen puede entenderse como un array de NumPy compuesto por píxeles representados como puntos de datos. Gracias a operaciones básicas como segmentación, enmascaramiento e indexación, es posible modificar los valores de dichos píxeles. Posteriormente, la imagen puede ser cargada con skimage y visualizada mediante Matplotlib [35].

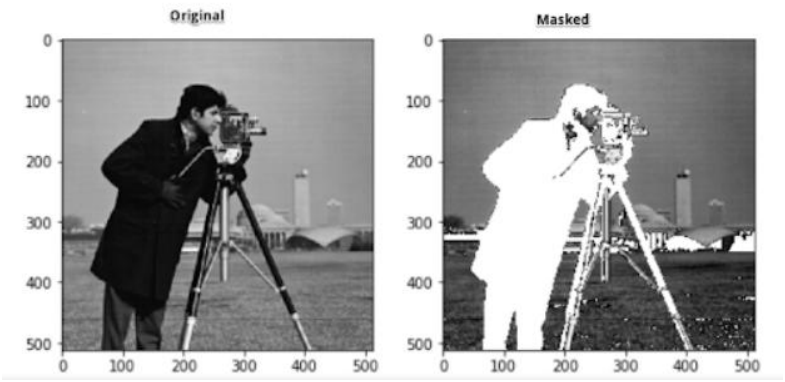

Fig. 2.11 Segmentación por Umbralización [35]

Open CV

OpenCV, una reconocida biblioteca de visión artificial de código abierto, es ampliamente utilizada en el desarrollo de aplicaciones de este campo. Su implementación en Python, conocida como OpenCV-Python, combina la rapidez derivada de su núcleo escrito en C/C++ con la simplicidad de programación que ofrece el contenedor en Python. Se usa para realizar tareas de procesamiento de imágenes, segmentación, y operaciones morfológicas en tu código. Se encarga de manipular las imágenes de forma eficiente y permite aplicar varios métodos de análisis [36].

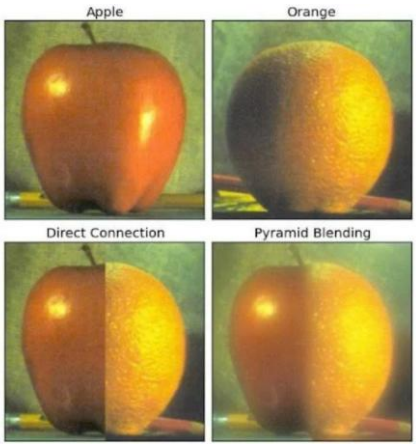

Fig. 2.12 Combinación de imágenes utilizando pirámides para crear una nueva fruta llamada manzana [36].

Pillow (PIL - Biblioteca de imágenes de Python)

PIL (Python Imaging Library) es una biblioteca gratuita para Python que permite abrir, manipular y almacenar imágenes en distintos formatos. No obstante, su desarrollo quedó detenido, siendo la última actualización publicada en 2009. En respuesta, surgió Pillow, una bifurcación de PIL que se mantiene en desarrollo activo, resulta más sencilla de instalar, es compatible con los principales sistemas operativos y funciona con Python 3. Esta biblioteca ofrece funciones esenciales de procesamiento de imágenes, como

operaciones de puntos, filtrado mediante núcleos de convolución predefinidos y conversiones entre espacios de color [37].

Fig. 2.13 Ajuste de Niveles de Contraste y Color [37].

Matplotlib

Sestinada a la creación de visualizaciones de datos de alta calidad. Permite generar gráficos como líneas, barras, dispersión, histogramas, entre otros, ofreciendo opciones de personalización y exportación en diversos formatos [38].

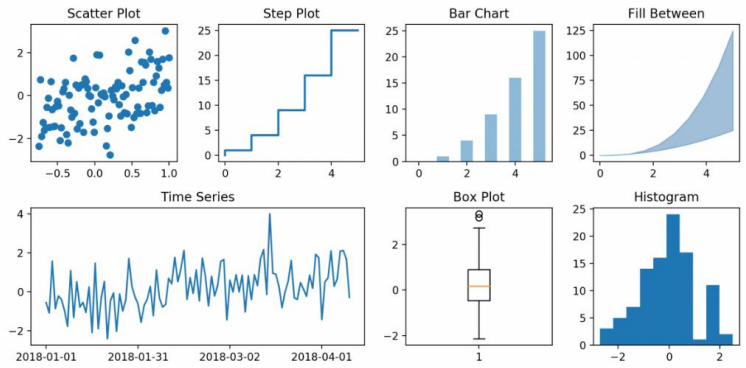

Fig. 2.14 Ejemplos de tipos de gráficos [48].

CAPÍTULO 3: MARCO METODOLÓGICO

3.1 Modelo de investigación

En este trabajo de titulación se emplea una investigación aplicada, ya que su propósito es resolver un problema específico relacionado con el impacto ambiental en la provincia de Sucumbíos. El objetivo principal es determinar el grado de deforestación en esta región mediante el análisis de imágenes satelitales, proporcionando información útil para la toma de decisiones en políticas ambientales y conservación.

Además, la investigación es de tipo documental, pues incluye la recopilación y análisis de diversas fuentes de información, tales como estudios previos sobre deforestación, bases de datos satelitales (Landsat, Sentinel-2) y literatura científica relacionada con el impacto ambiental y los métodos de análisis satelital.

Estas fuentes sirven para contextualizar y fundamentar el marco teórico de la investigación. Se detallarán las características del fenómeno de la deforestación en términos de su distribución espacial y temporal. Esto permitirá cuantificar y visualizar la pérdida de cobertura forestal en diferentes periodos de tiempo.

Por último, se incluye una investigación experimental, ya que se prueban diferentes técnicas de análisis satelital, como el cálculo de índices de vegetación y la detección de cambios multitemporales, con el fin de garantizar la precisión en la medición del área deforestada y analizar los resultados obtenidos.

Esta etapa permite evaluar la efectividad de los métodos empleados para determinar el impacto ambiental.

3.2 Diseño de la Investigación

3.2.1. Fase 1: Adquisición la base de datos de imágenes satelitales que contengan la zona en estudio

I. Investigación de las bases de datos que contienen imágenes satelitales, específicamente en el Ecuador.

Usando el internet se investigan las plataformas que tienen las bases de datos de imágenes satelitales que las tengan disponibles. El propósito es identificar fuentes fiables de imágenes con alta resolución. Se da prioridad a las bases de datos que contengan imágenes satelitales con temporalidad adecuada y que sean de acceso público o por acuerdo con entidades de investigación. Se buscan fuentes que proporcionen acceso a imágenes satelitales, tales como:

- USGS Earth Explorer: Ofrece imágenes de satélites como Landsat con alta resolución espacial y temporal.
- Copernicus Open Access Hub: Proporciona datos de satélites Sentinel, que tienen una resolución temporal diaria y son especialmente útiles para el monitoreo ambiental.
- NASA Earth Observing System Data and Information System (EOSDIS): Para acceder a imágenes de satélites de la NASA con diversos periodos de observación.
- Google Earth Engine: Proporciona acceso a un gran conjunto de datos satelitales, con herramientas de análisis integradas.

Este proceso se lleva a cabo en un entorno de trabajo adecuado que cuente con acceso a internet y herramientas para acceder a las bases de datos mencionadas.

II. Selección de la base de datos para la investigación

Se selecciona la base de datos después de la investigación realizada en la primera parte. Ahora se elige la plataforma que proporcione la mayor resolución temporal y espacial, así como el tipo de datos que mejor se adapten a las necesidades de este proyecto. Para lograr esta selección se mantienen reuniones constantes con el grupo de profesionales que dominan el tema, de tal manera de garantizar una escogencia eficiente.

III. Definición del rango de análisis.

Dependiendo de los datos disponibles, se determina el rango de análisis de las imágenes, de acuerdo con las investigaciones realizadas en el capítulo I. Según el Ministerio de Ambiente y Agua del Ecuador, se tiene evidencia de deforestación desde el año 2019, por lo tanto, el interés de esta investigación es realizar el estudio, tentativamente, desde esa época hasta la actualidad.

IV. Obtención de la base de datos de imágenes en estudio.

Una vez definido el rango de análisis se procede a obtener y organizar la base de datos de Google Earth Engine que proporciona imágenes satelitales correspondientes a la zona amazónica del Ecuador, específicamente aquellas que pertenezcan a la provincia de Sucumbíos. Para ello, se deben establecer los límites de las coordenadas geográficas, tanto latitud como longitud, tomando en consideración dos puntos estratégicos, la esquina superior derecha y la esquina inferior izquierda, de esos dos puntos trazamos una diagonal y obtenemos la dimensión de la imagen y procedemos a descargarla.

3.2.2. Fase 2: Implementación de un software para el análisis de imágenes

I. Investigación de los algoritmos de programación para el análisis de imágenes satelitales.

Se realiza una investigación de los algoritmos de programación que permitan el análisis de imágenes satelitales. Preferiblemente, los desarrollados con software libre. Seguidamente, se hace un análisis del algoritmo de cada programa y se evalúa la posibilidad de modificación de este.

II. Selección del algoritmo de análisis de imágenes.

Una vez se hayan investigado los algoritmos, se selecciona el más adecuado. Se realiza una reunión con especialistas para tomar esta decisión.

La selección del algoritmo podría constar de las siguientes características:

- Mayor precisión en la detección de grafos o bordes
- Facilidad de cargar la base de datos
- Eficiente procesamiento de imágenes.
- Fácil cálculo de los índices espectrales para detectar vegetación.
- Implementar técnicas de detección de cambios para identificar áreas donde la vegetación ha disminuido en períodos específicos.

Una vez seleccionado el algoritmo, Visual Studio en su entorno de Python fue la mejor opción para la programación, ya que proporciona una amplia gama de librerías adecuadas para el procesamiento de imágenes satelitales.

III. Programación del algoritmo seleccionado para el procesamiento de imágenes satelitales.

Se realiza la programación del algoritmo usando las librerías necesarias para el procesamiento de imágenes, luego se realizan pruebas para medir el funcionamiento del programa.

IV. Comprobación del algoritmo.

Para la comprobación de algoritmo se pueden identificar y corregir errores lógicos, evaluar el desempeño y garantizar que el algoritmo cumpla con los requisitos del problema, asegurarse de que el algoritmo sea eficiente, preciso y funcional antes de su implementación final.

Realizar un control del programa como:

1. Leer el código, línea por línea para encontrar errores evidentes, como mal uso de variables, operadores o estructuras condicionales.
2. Dividir el algoritmo en módulos o funciones y probar cada uno por separado.
3. Ejecutar el algoritmo completo con datos reales o simulados.

   Este proceso garantizará que el algoritmo sea comprobado de manera exhaustiva, reduciendo al mínimo los errores y optimizando su desempeño. Además, se documentará cada paso, facilitando futuras modificaciones.

3.2.3. Fase 3: Cuantificación del área de deforestación en la provincia de Sucumbíos a través del análisis de imágenes satelitales.

I. Determinación de las zonas que evidencien deforestación.

Una vez que el programa computacional se le ha comprobado su funcionamiento, se carga la imagen de referencia para que a partir de esa imagen determinar la evolución deforestación. Luego se comprueba el

funcionamiento cargando la base de datos para que el programa analice y muestre un histograma del avance de la deforestación. Además, está conectado con un Api que me permite visualizar un documento detallado con toda la información de las imágenes satelitales y la evolución de la deforestación.

II. Selección de las zonas afectadas por deforestación.

Para la selección de las zonas afectadas por la deforestación, es necesario todos los pasos anteriores. Una vez que se han determinado las zonas que evidencian deforestación, se calculan las áreas de las pérdidas de vegetación.

III. Cálculo de la evolución de las áreas deforestadas.

Para calcular la evolución de las áreas deforestadas, se propone una solución estructurada utilizando imágenes satelitales procesadas a lo largo del tiempo, combinando técnicas de procesamiento de imágenes, análisis temporal y ecuaciones matemáticas.

Para cada imagen, se calcula la cantidad de píxeles que corresponden a áreas deforestadas sobre la cantidad de pixeles totales de la imagen y multiplicando por cien. Esto da un porcentaje de forestación que es visualizado en el histograma.

Para determinar el área en kilómetros cuadrados, se divide el porcentaje antes mencionado sobre cien por el área total en kilómetros. Con estos datos se puede visualizar en el documento generado.

CAPÍTULO 4: ANÁLISIS DE RESULTADOS

4.1 Selección de la base de datos de imágenes satelitales que contengan la zona en estudio

4.1.1. Investigación y selección de bases de datos de imágenes satelitales en el Ecuador

Se realiza un análisis de las plataformas que proveen bases de datos de imágenes satelitales, considerando su disponibilidad en acceso libre o restringido. Se priorizaron aquellas que ofrecen amplia cobertura temporal, indispensable para los objetivos de la investigación. Posteriormente, se selecciona la plataforma con mayor resolución temporal y espacial, así como con el tipo de datos más adecuados para el proyecto. La elección correspondió a Google Earth Engine, dado que integra una extensa colección de imágenes históricas y, al ser de libre acceso, facilita tanto el procesamiento como la reproducibilidad de los resultados.

4.1.2. Definición del rango de análisis.

El rango de análisis de las imágenes se establece en función de la disponibilidad de datos, considerando que la resolución y la calidad varían anualmente. La presencia de nubosidad dificulta el procesamiento, por lo que resulta necesario seleccionar las imágenes con mejores condiciones. Se determina que el periodo comprendido entre 2019 y 2024 corresponde a los años de mayor impacto, de acuerdo con un reporte del Ministerio de Ambiente. Adicionalmente, se incorpora como referencia la imagen del año 2000. Con esta información se define el rango de análisis utilizado en la investigación

4.1.3. Obtención de la base de datos de imágenes en estudio.

En la plataforma Google Earth Engine se selecciona el satélite que provee la base de datos. El satélite USGS Landsat 7 se considera el más adecuado para la obtención de imágenes, dado que ofrece amplia cobertura histórica con calidad aceptable. Una vez definido el satélite, el área de trabajo se delimita mediante la identificación de

dos puntos de referencia y el trazado de una línea. A partir de dichos puntos se registran las coordenadas de latitud y longitud, lo que permite establecer las dimensiones de la

imagen y, posteriormente, realizar la captura del área seleccionada

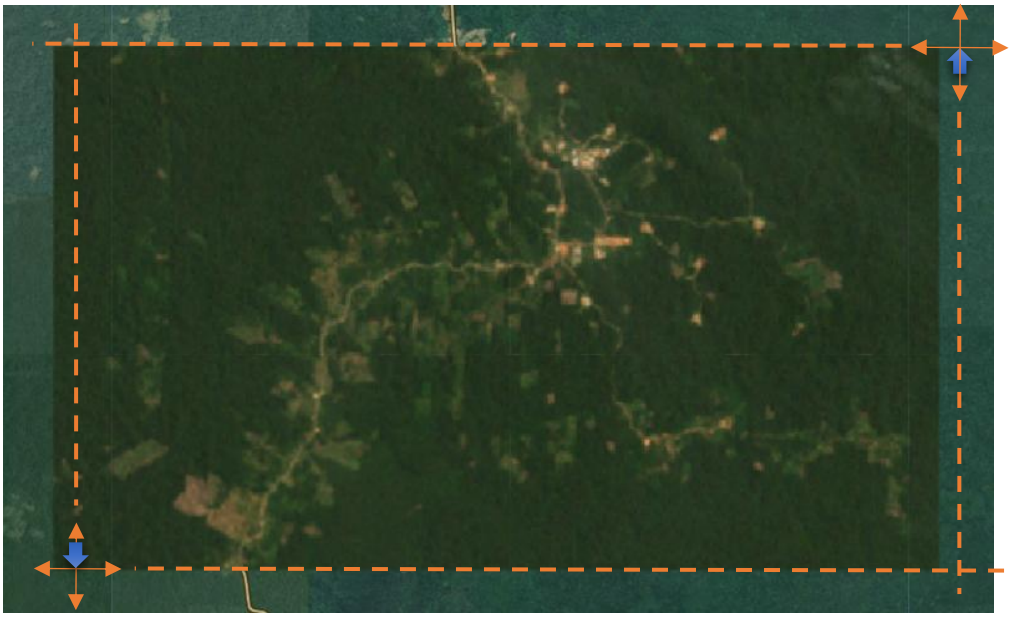

Fig. 4.1 Delimitación de la imagen

Mediante el uso de la herramienta Gemini se determina el área total, a partir de la colocación de los puntos de latitud y longitud correspondientes. El sistema procesa dichas coordenadas y proporciona los datos requeridos para la delimitación espacial.

4.2 Desarrollo de un programa computacional que procese las imágenes satelitales

4.2.1 Evaluación y selección de algoritmos de programación para el análisis de imágenes satelitales.

El procedimiento para el análisis de imágenes satelitales se desarrolla en tres fases principales. En la primera fase se investigan los algoritmos, librerías y plataformas disponibles para el procesamiento digital de imágenes. Se emplean herramientas como OpenCV para segmentación y detección de bordes, NumPy para operaciones matriciales sobre píxeles, Matplotlib para la visualización de

resultados y Pillow para manipulaciones básicas de imágenes; adicionalmente, Pandas se utiliza para la integración de datos espaciales. La combinación de estas herramientas conforma un entorno robusto y eficiente, indispensable para la identificación y monitoreo de procesos de deforestación en la provincia de Sucumbíos.

En la segunda fase se realiza la selección de algoritmos de análisis, destacando la segmentación como técnica fundamental para garantizar la precisión en la interpretación de imágenes satelitales. Se priorizan métodos que permitan separar de manera confiable las áreas vegetadas de las no vegetadas, siendo el método de Otsu para umbralización un referente por su capacidad de optimizar la clasificación de píxeles. La segmentación se complementa con índices espectrales que facilitan la identificación de vegetación y cuerpos de agua, así como con técnicas de detección de cambios multitemporales que permiten evaluar la evolución de la deforestación en distintos periodos. La correcta aplicación de segmentación constituye el eje central del análisis, ya que asegura la validez de los resultados y la eficiencia del monitoreo ambiental.

Finalmente, en la tercera fase se implementa la programación del algoritmo seleccionado. Para este propósito se adopta un enfoque de segmentación basado en el espacio de color HSV (Tono, Saturación, Valor), el cual resulta adecuado para discriminar áreas de vegetación respecto a zonas no vegetadas en imágenes satelitales, garantizando un procesamiento eficiente y técnicamente validado.

4.2.2 Programación del algoritmo seleccionado para el procesamiento de imágenes satelitales.

El algoritmo se basa en el siguiente diagrama de flujo:

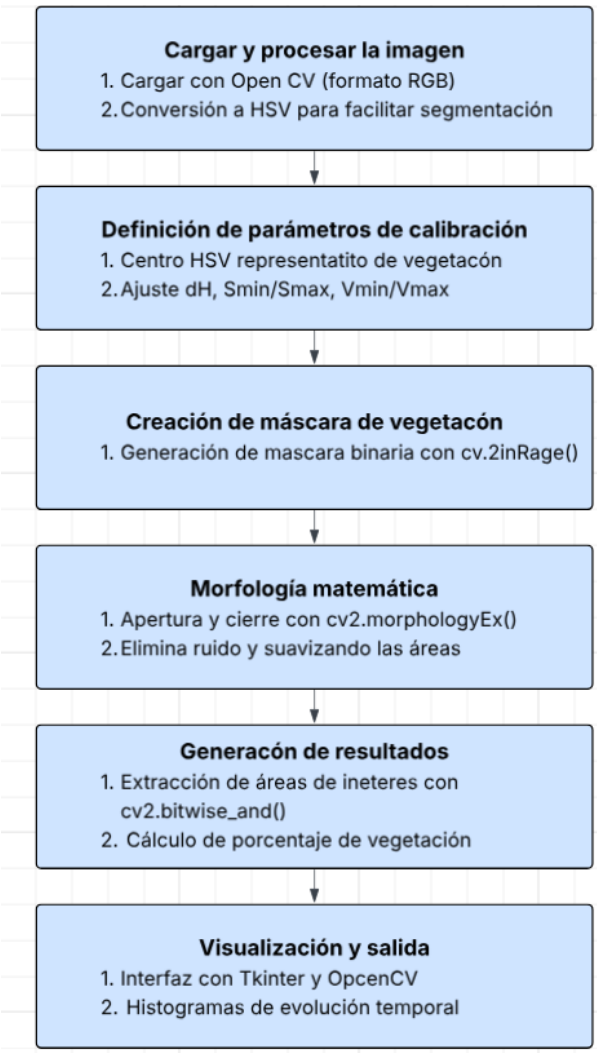

Fig. 4.2 Diagrama de flujo del algoritmo

A continuación, se detalla cada proceso del diagrama de flujo:

1. Cargar y procesar la imagen.

   La carga de la imagen base desde una carpeta en la PC se realiza mediante la librería OpenCV en su formato original (PNG, JPG u otros), almacenándola en una variable para su posterior procesamiento. Las imágenes deben estar codificadas e identificadas con la fecha correspondiente en el formato dd/mm/aaaa, con el fin de garantizar trazabilidad en el análisis.

Posteriormente, se efectúa la conversión del espacio de color RGB a HSV (Matiz, Saturación, Valor). Este espacio resulta más adecuado para procesos de segmentación, ya que permite una discriminación más eficiente de los colores. La transformación a HSV facilita la separación de la vegetación respecto a otros elementos presentes en la escena, optimizando la detección y clasificación de áreas de interés

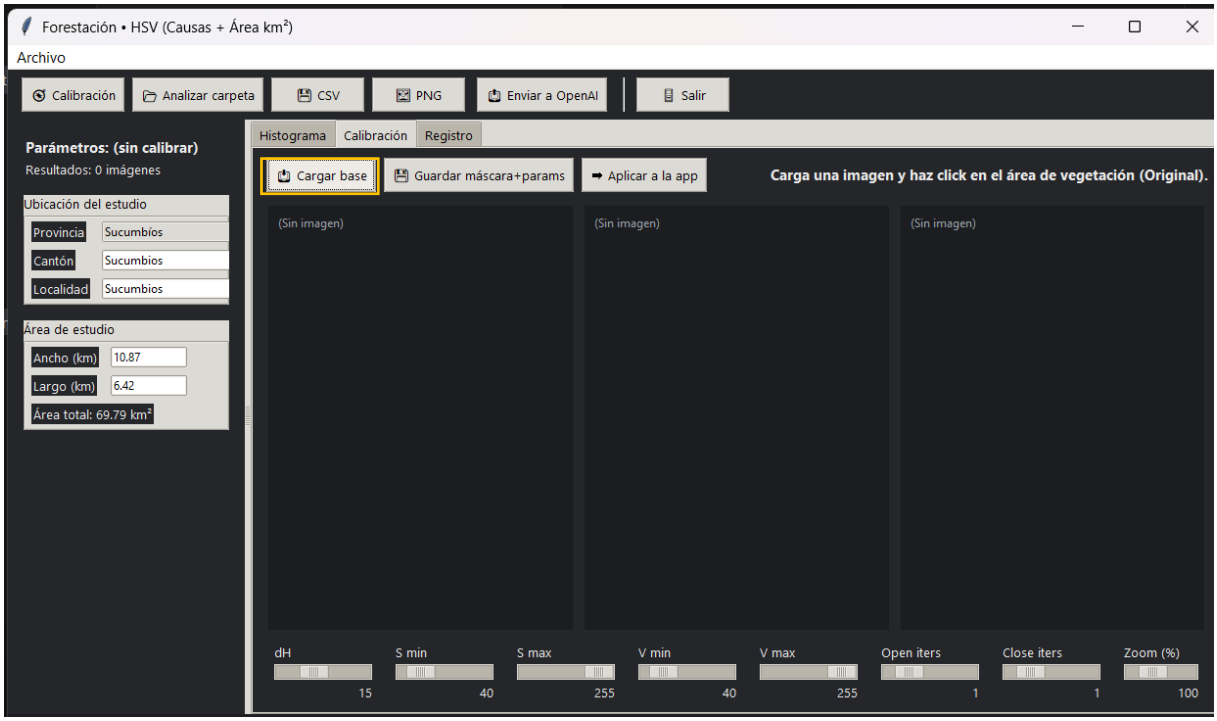

Fig 4.3 Interfaz principal

2. Definición de parámetros de calibración.

   La determinación del centro HSV representativo de la vegetación consiste en seleccionar un valor cromático central dentro del espacio de color HSV (Matiz, Saturación, Valor) que caracterice de manera óptima la vegetación presente en la imagen. Este procedimiento puede realizarse mediante la identificación de un punto específico en la escena que corresponda a un área homogénea de vegetación, de modo que el valor obtenido se utilice como referencia para los procesos de segmentación y clasificación posteriores.

La selección del centro HSV permite establecer un criterio objetivo para diferenciar la vegetación de otros elementos, garantizando mayor precisión en la delimitación de áreas de interés y reduciendo la ambigüedad en la interpretación espectral.

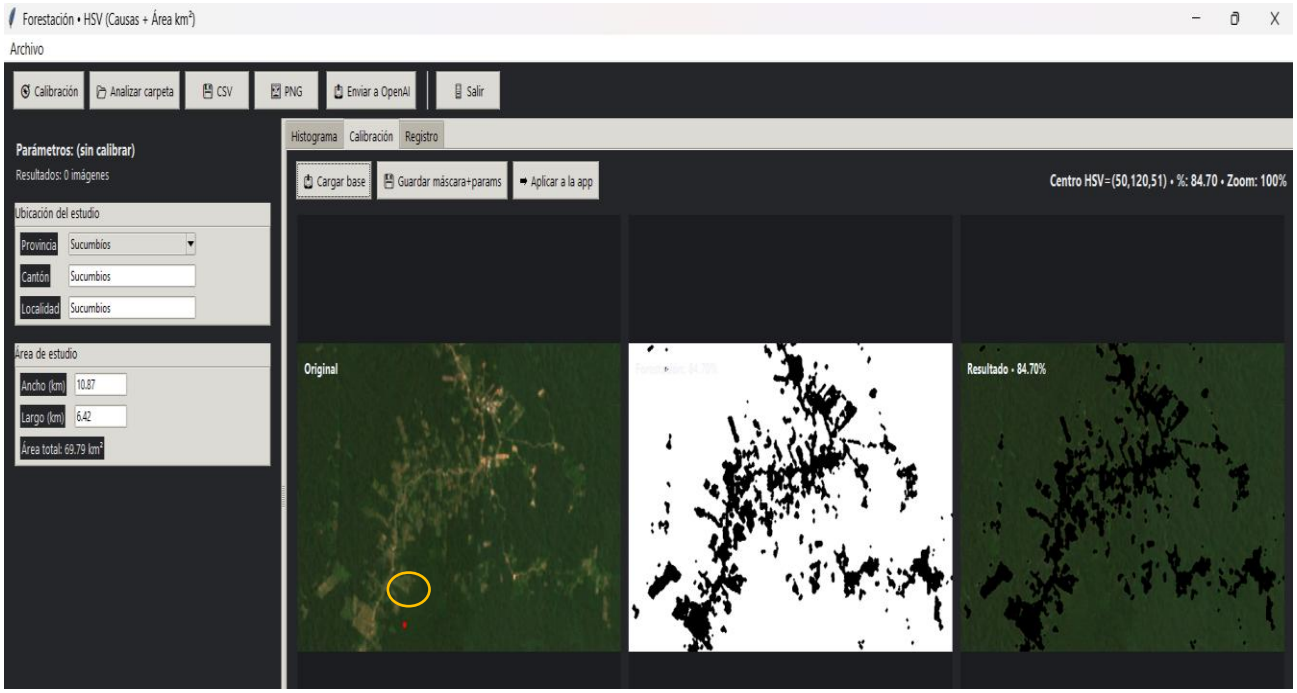

Fig 4.4 Determinación del centro HSV

El ajuste de parámetros en el espacio de color HSV se realiza mediante la definición de rangos específicos para el matiz (dH), la saturación (S_min/S_max) y el valor o brillo (V_min/V_max). Estos límites permiten generar una máscara de vegetación que discrimina las áreas vegetales respecto al resto de la imagen.

Rango específico para segmentación de vegetación verde:

Matiz (H): 35 a 85 (verde claro a verde oscuro).

Saturación (S): 60 a 255 (verde saturado, colores intensos).

Valor (V): 40 a 255 (colores con buena luminosidad).

La definición del rango de verde en el espacio HSV constituye un paso crítico en la segmentación de vegetación. Si el intervalo seleccionado resulta demasiado estrecho, se

generan falsos negativos, es decir, áreas de vegetación que no son detectadas. En contraste, un rango excesivamente amplio puede producir falsos positivos, clasificando como vegetación zonas que no corresponden a dicha cobertura.

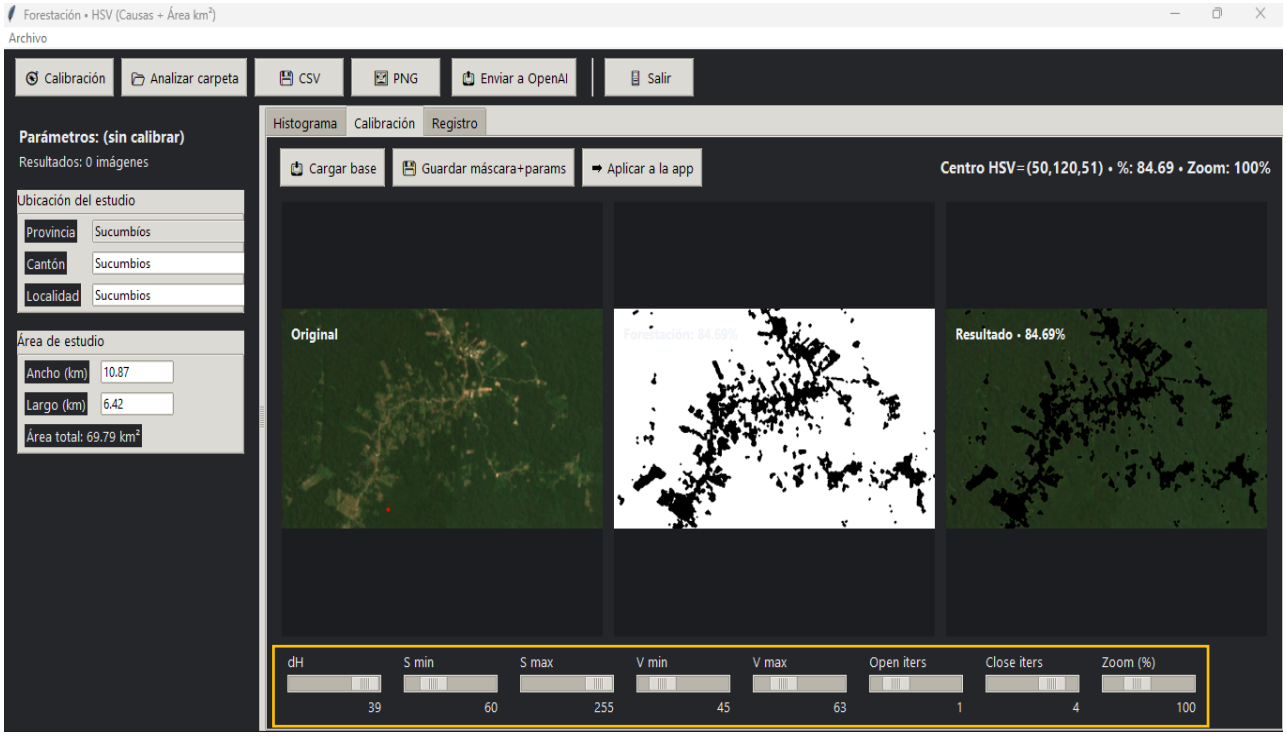

Fig 4.5 Definición del rango en el espacio HSV

3. Creación de máscara de vegetación

La generación de la máscara binaria se realiza mediante la función cv2.inRange(), utilizando los rangos previamente ajustados en el espacio HSV. Esta operación produce una imagen binaria en la que los píxeles que cumplen con los criterios establecidos son asignados al valor 1 (vegetación), mientras que el resto se clasifica con el valor 0 (no vegetación). La máscara resultante constituye una representación espacial de las áreas vegetales dentro de la escena, permitiendo su diferenciación respecto a otros elementos.

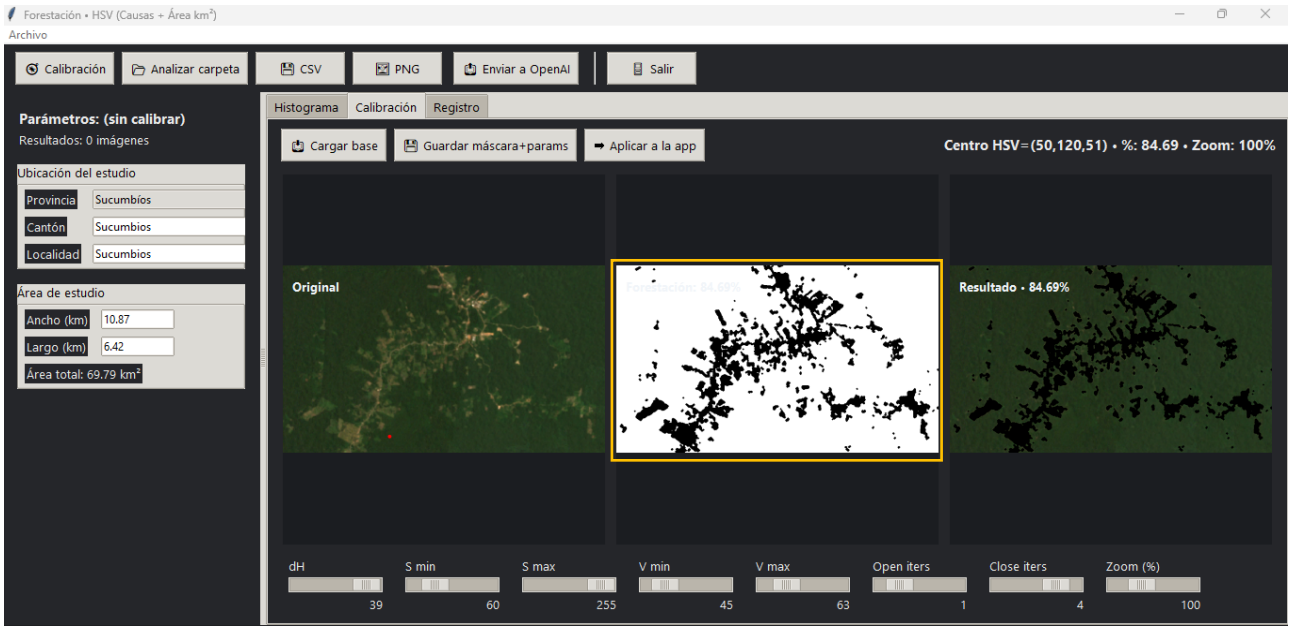

Fig 4.6 Máscara Binaria

4. Morfología matemática

   La depuración de la máscara binaria se realiza mediante operaciones morfológicas aplicadas con la función cv2.morphologyEx(). Estas operaciones permiten eliminar imperfecciones y mejorar la continuidad de las áreas clasificadas como vegetación.

   La apertura (Open iters) elimina pequeños objetos no deseados presentes en la máscara, reduciendo la presencia de píxeles aislados que no corresponden a vegetación. El cierre (Close iters) rellena huecos de tamaño reducido dentro de las áreas vegetales, asegurando una representación más uniforme y continua de la cobertura.

5. Cargar la base de datos

   Para proceder a cargar la base de datos es necesario hacer clic en “Guardar máscara + parámetros” y después en “Aplicar a la app”, ya que permite guardar los parámetros de HSV. Dicho eso, se procede a cargar la base de datos de imágenes satelitales de la zona en estudio en el apartado de Analizar carpeta.

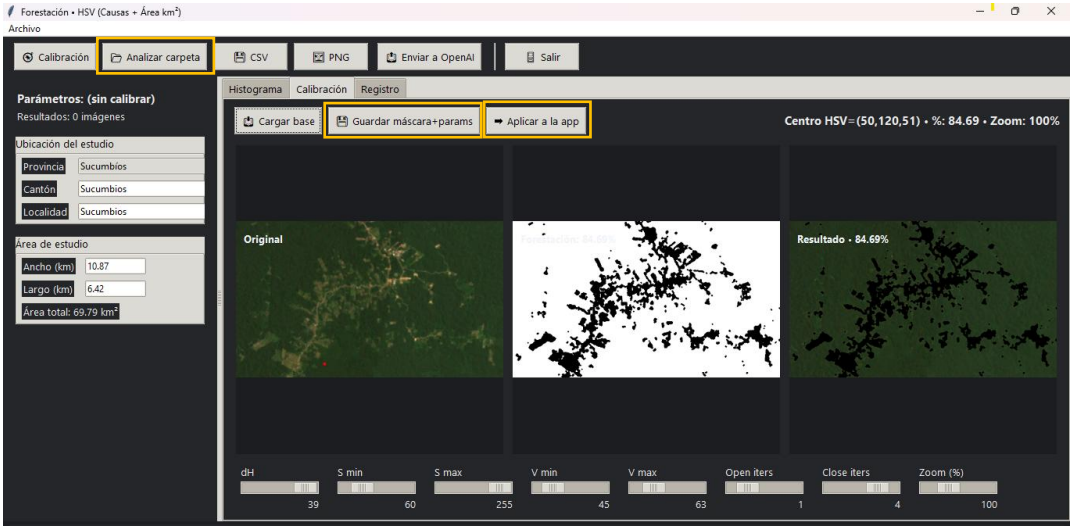

Fig 4.7 Guardar los parámetros

Se debe tener las imágenes en una sola carpeta con la misma codificación antes mencionada.

6. Generación de resultados

La extracción de áreas de interés se realiza mediante la función cv2.bitwise_and(), aplicando la máscara binaria previamente generada sobre la imagen original. Este procedimiento permite conservar únicamente los píxeles correspondientes a la vegetación, descartando el resto de los elementos presentes en la escena. De esta manera, se obtiene una representación visual precisa de las zonas clasificadas como vegetación. Posteriormente, se calcula el porcentaje de cobertura vegetal en la imagen. Este indicador se obtiene dividiendo el número de píxeles clasificados como vegetación entre el total de píxeles de la imagen por cien para obtener el porcentaje, lo que proporciona una medida cuantitativa de la proporción de vegetación respecto al área total analizada. Dicho cálculo constituye un parámetro fundamental para la evaluación de cambios en la cobertura forestal y para el monitoreo de procesos de deforestación en análisis multitemporales.

7. Visualización y salida

La implementación de la interfaz gráfica se desarrolla mediante la integración de Tkinter y OpenCV, permitiendo la visualización de los resultados del análisis.

Adicionalmente, se generan histogramas de evolución temporal con el fin de representar gráficamente la variación del porcentaje de vegetación a lo largo de diferentes periodos o conjuntos de imágenes. Estos histogramas constituyen una herramienta analítica que permite identificar tendencias, evaluar la dinámica de la cobertura vegetal y detectar posibles procesos de deforestación o regeneración en la zona de estudio.

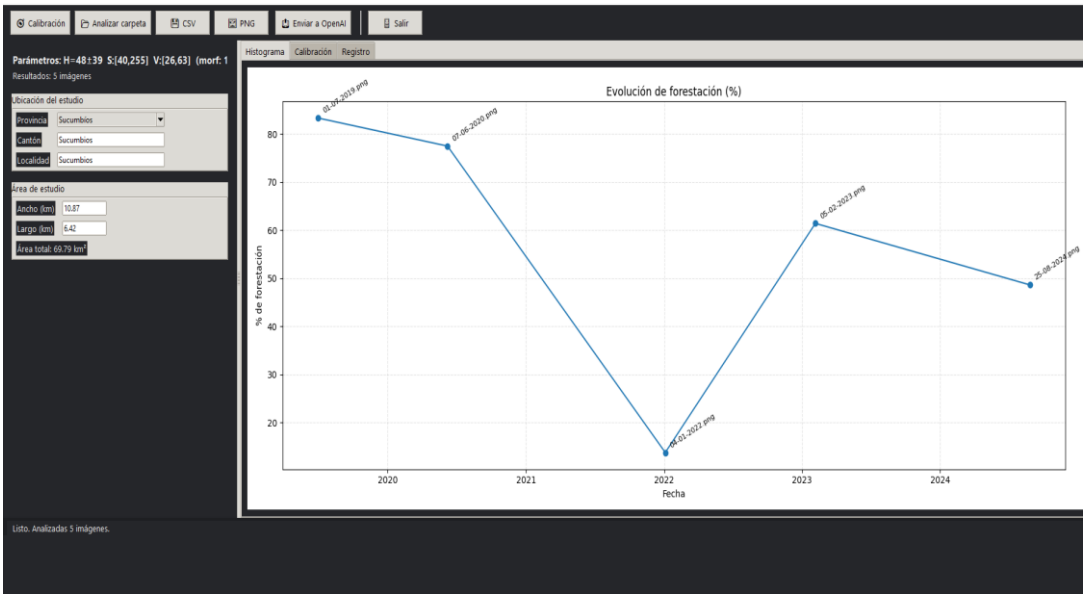

Fig 4.8 Histograma de la evolución de la forestación

Además, se puede generar un PDF con todos los resultados e información detallada de todas las imágenes, ya que el programa tiene conexión con una API.

4.2.3 Comprobación del algoritmo.

El algoritmo se somete a diversas pruebas y validaciones con el fin de garantizar su correcto funcionamiento, eficiencia y precisión en la detección de vegetación y áreas deforestadas; para ello se verifica su capacidad de distinguir entre zonas de vegetación y no vegetación mediante la comparación con imágenes de

referencia previamente clasificadas, así como la exactitud de las máscaras binarias generadas, comprobando la adecuada calibración de parámetros de umbralización, saturación y valor. Se evalúa la precisión de la segmentación contrastando los resultados con registros históricos de cobertura vegetal y deforestación; se analizaron variaciones en los parámetros de calibración (tono, saturación y valor) para determinar su influencia en la robustez del algoritmo frente a distintas condiciones de las imágenes satelitales, incluyendo variaciones de nubosidad e iluminación, constatándose que en presencia de nubes se reduce la capacidad de identificación de áreas deforestadas, lo que exige imágenes limpias para mayor exactitud. Se verifica la consistencia de los resultados en diferentes periodos, confirmando la correcta identificación de zonas deforestadas, conservadas y regeneradas, así como la precisión en el cálculo de superficies expresadas en kilómetros cuadrados mediante comparación con áreas previamente registradas. Finalmente, se revisa la interfaz de usuario, asegurando que permita la carga de imágenes, ajuste de parámetros, visualización de resultados y generación de gráficos e informes de manera intuitiva, corroborando que los histogramas y reportes producidos fueran coherentes con los datos de entrada y los resultados esperados.

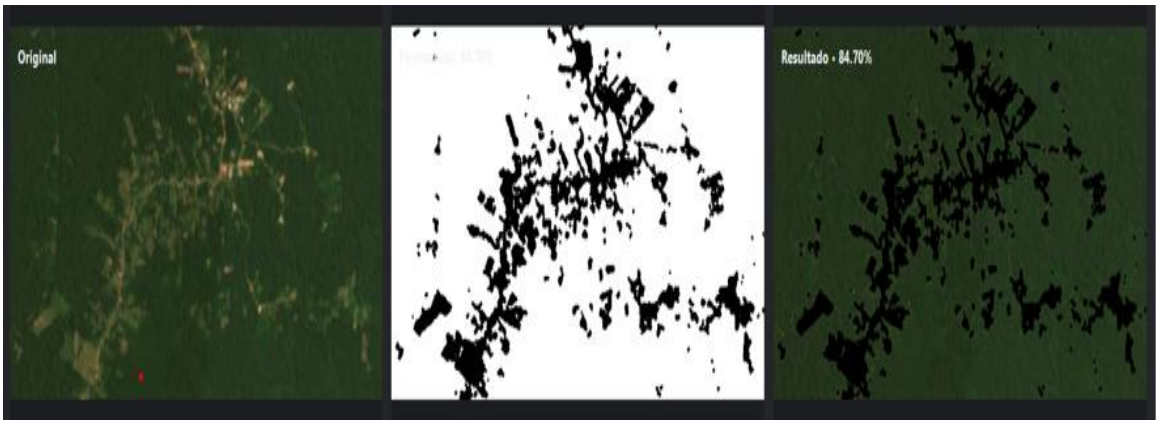

Fig. 4.9 Imagen de la derecha evidencia zonas de deforestación

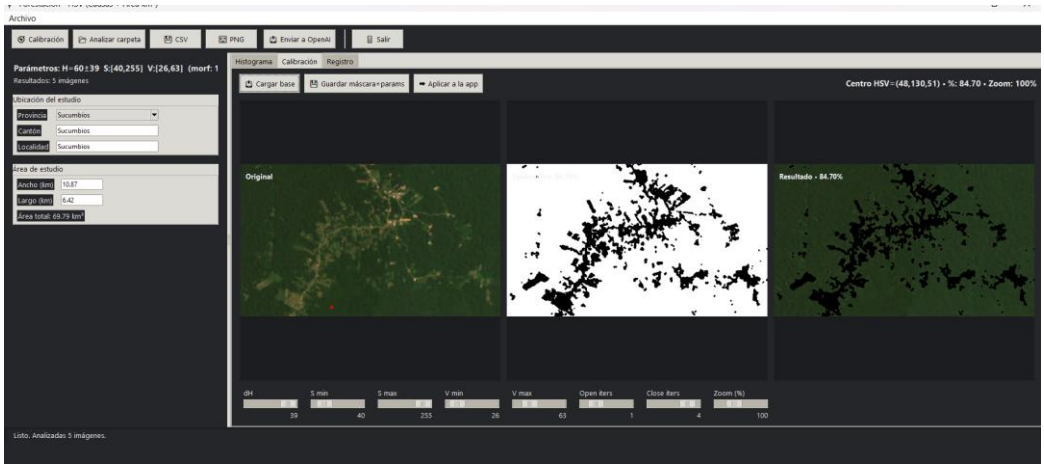

Fig. 4.10 Interfaz de usuario fácil de usar

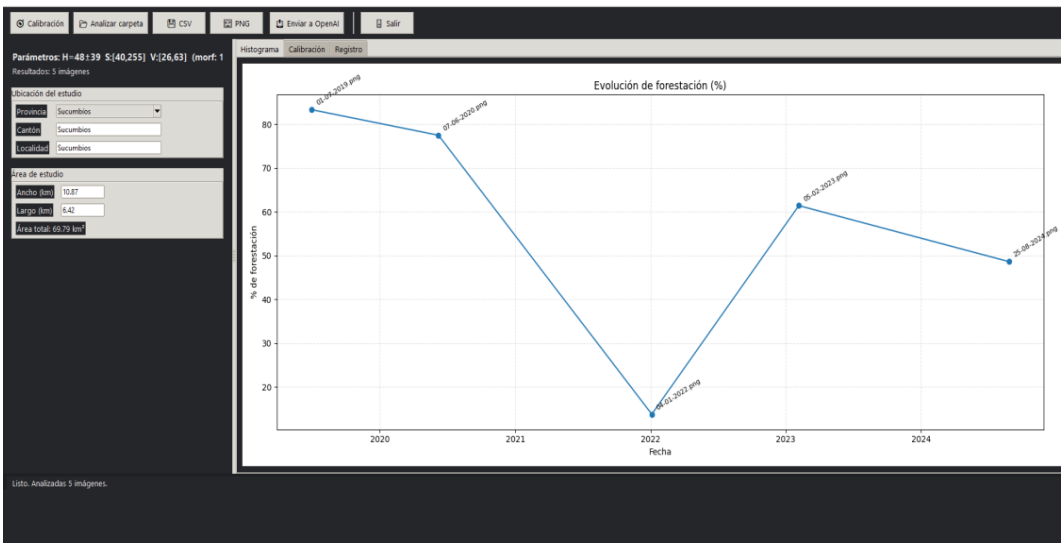

Fig. 4.11 Histograma con datos reales y funcional

4.3 Cálculo las áreas deforestadas en la provincia de Sucumbíos.

4.3.1 Determinación y selección de las zonas que evidencien deforestación.

La selección de la localidad de Cuyabeno como área de estudio se realiza mediante un proceso en dos etapas. En primer lugar, se revisaron antecedentes e investigaciones previas enfocadas en la provincia de Sucumbíos, donde se identifica a Cuyabeno como un territorio con actividades humanas relevantes sobre la cobertura forestal, asociadas principalmente a tala indiscriminada de bosques, apertura de vías, expansión agropecuaria y minería ilegal. En segundo lugar, se efectuó una inspección exploratoria multitemporal con imágenes satelitales para comparar visualmente cambios de cobertura entre distintos años, priorizando zonas con evidencia clara de pérdida de vegetación y continuidad espacial del cambio. Como resultado de esta verificación preliminar, el sector Palma Roja dentro de Cuyabeno presentó la mayor presencia de patrones compatibles con deforestación además de contar con escenas adecuadas para el análisis.

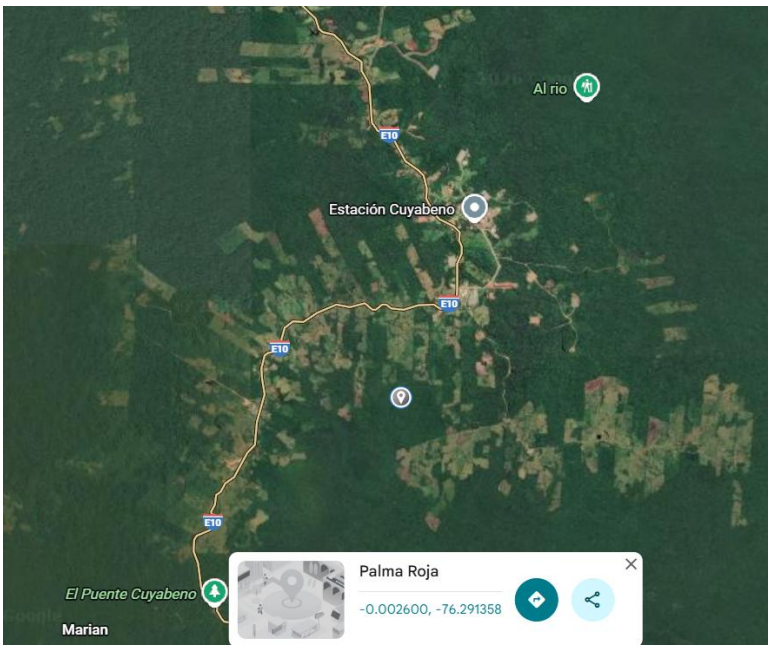

Fig. 4.12 Zona de mayor perdida forestal

4.3.2 Cálculo de la evolución de las áreas deforestadas.

La deforestación se calcula generalmente a partir de la segmentación de la imagen en áreas de vegetación (bosque) y no vegetación tal como indica en la Ec 4.1.

$$\%\ de\ vegetación = \left(\frac{Píxeles\ de\ vegetación}{Total\ de\ Píxeles}\right) * 100 \qquad \text{Ec 4.1}$$

Píxeles de Vegetación: Es el número de píxeles que han sido clasificados como vegetación en la imagen segmentada. Este valor se obtiene a partir de la máscara binaria generada en el algoritmo.

Total de Píxeles: Es el número total de píxeles de la imagen original.

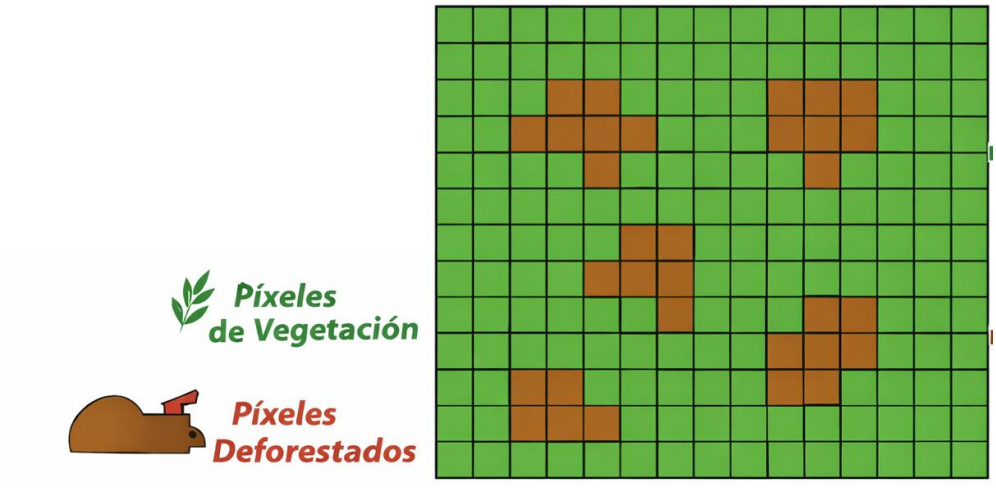

Fig. 4.13 Vista micro de los pixeles en las imágenes

Para el cálculo de área deforestada se usa la siguiente fórmula tal como indica la Ec 4.2 y Ec 4.3.

$$\%\ de\ deforestación = 100 - \%\ de\ vegetación \qquad \text{Ec 4.2}$$

$$Área\ deforestada(km^2) = \left(\frac{\%\ de\ deforestación}{100}\right) * Área\ total(km^2) \qquad \text{Ec 4.3}$$

% de Deforestación: Representa el complemento del porcentaje de vegetación, es decir, la proporción de píxeles que pertenecen a zonas sin cobertura vegetal (deforestadas).

Área Total (km²): Es el área total del área de estudio en kilómetros cuadrados (obtenida a partir de las dimensiones de la imagen o las coordenadas geográficas del área).

- Cálculo de la evolución temporal de la deforestación mostrada en la Ec 4.4.

$$\Delta Deforestación = \%\ de\ deforestación_{t2} - \%\ de\ deforestación_{t1} \quad \text{Ec 4.4}$$

$\%\ de\ deforestación_{t2}$= % de deforestación en el tiempo t2 (fecha posterior)

$\%\ de\ deforestación_{t1}$= % de deforestación en el tiempo t1 (fecha anterior)

El cambio de deforestación puede ser positivo (indicado un aumento en la deforestación) o negativo (indicado una recuperación de la vegetación).

- Cálculo del Delta de deforestación en Área específicas en kilómetros cuadrados mostrada en la Ec 4.5.

$$\Delta Área\ deforestada\ (km^2) = \left(\frac{\Delta\ Deforestación}{100}\right) * Área\ total\ (km^2) \quad \text{Ec 4.5}$$

$\Delta\ Deforestación$ = Es la diferencia en el porcentaje de deforestación entre dos fechas.

$Área\ total\ (km^2)$ = Es el área total de la región que se está analizando.

- Cálculo de la Deforestación Relativa (Comparación con la Base) tal como indica la Ec 4.6.

Para medir la deforestación relativa, es decir, cómo varía en relación con la imagen base (la primera imagen del periodo de análisis)

$$\Delta Deforestación\ Relativa = \%\ de\ Deforestación - \%\ de\ Deforestación\ Base \quad \text{Ec 4.6}$$

$\%\ de\ Deforestación$ = El porcentaje de deforestación calculado en una imagen específica.

$\%\ de\ Deforestación\ Base$ = El porcentaje de deforestación calculado en la imagen base (la primera imagen del periodo).

Esto proporciona una medida relativa de cuánto ha aumentado o disminuido la deforestación en comparación con la imagen base.

En la Fig 4.7 puede observarse que existen 15x13 pixeles. Realizando los cálculos con las ecuaciones 4.1, 4.2, 4.3, se obtiene el porcentaje de vegetación y el área deforestada.

El total de pixeles = 195

Pixeles de no vegetación = 32

Pixeles de vegetación = 163

$$\%\ de\ vegetación = \left(\frac{163}{195}\right) * 100 = 83.58\%$$

$$\%\ de\ deforestación = 100 - 83.58\% = 16.42\%$$

$$Área\ deforestada(km^2) = \left(\frac{16.42}{100}\right) * 69.79km^2$$

$$Área\ deforestada = 11.45\ km^2$$

Esto representa un ejemplo, ya que con datos reales los datos varían.

CONCLUSIONES

Se seleccionó una base de datos satelital con cobertura histórica adecuada para el estudio, priorizando criterios de disponibilidad temporal y accesibilidad. En particular, se emplearon imágenes procesadas en la plataforma Google Earth Engine, con énfasis en el sensor Landsat 7, lo que permitió establecer una serie de análisis consistente para la provincia de Sucumbíos.

Se desarrolló un software en Python utilizando librerías de visión artificial (OpenCV, NumPy, Matplotlib, Pillow y Tkinter), el cual implementa un flujo integral de procesamiento que incluye carga de imágenes, calibración, segmentación en espacio HSV, generación de máscaras y visualización de resultados mediante interfaz gráfica e histogramas. La funcionalidad del sistema fue validada mediante pruebas con imágenes reales.

La deforestación fue estimada a partir de la relación entre píxeles clasificados como vegetación y no vegetación, convertida posteriormente a unidades de área (km^2) en función del área total del sitio de estudio. La georreferenciación mediante coordenadas de latitud y longitud permitió evaluar la evolución temporal de la cobertura forestal mediante el cálculo de deltas respecto a una imagen base.

El análisis identificó al sector Palma Roja, en la zona de Cuyabeno, como uno de los sectores con mayor pérdida de cobertura forestal dentro del área evaluada. El fenómeno se asocia principalmente a factores antrópicos descritos en el estudio, tales como expansión agrícola, asentamientos humanos y actividades extractivas, lo que convierte a esta área en un punto crítico para el monitoreo ambiental.

La precisión del cálculo se encuentra condicionada por factores inherentes a la teledetección en la Amazonía, particularmente la presencia de nubosidad, sombras y la variabilidad radiométrica entre escenas. En consecuencia, la selección de imágenes con menor

interferencia atmosférica y la calibración adecuada de los parámetros de segmentación resultan elementos críticos para la reducción de errores de clasificación.

La segmentación mediante el método de Otsu constituye una opción óptima para la determinación de la deforestación a partir de imágenes satelitales, al permitir la clasificación de píxeles mediante una máscara binaria y facilitar el cálculo de variaciones en la cobertura forestal.

La deforestación en la provincia de Sucumbíos fue cuantificada mediante el análisis de imágenes satelitales, utilizando una base de datos multitemporal integrada con un algoritmo computacional orientado a la segmentación de la cobertura vegetal y a la estimación de su variación temporal. Este procedimiento permitió la identificación de zonas críticas de pérdida forestal.

RECOMENDACIONES

La utilización de plataformas que suministren imágenes de alta resolución y mínima interferencia atmosférica, libres de nubosidad, resulta esencial para evitar restricciones en la precisión del cálculo.

Una calibración adecuada de los parámetros de saturación, intensidad y brillo incrementa la precisión de los resultados, dado que dichos parámetros constituyen la base fundamental del proceso de segmentación.

El empleo de software especializado de licencia comercial, como ArcGIS, permite verificar los resultados obtenidos mediante la medición y contraste de los mismos parámetros utilizados en el análisis.

La ampliación del número de imágenes procesadas favorece la efectividad de los resultados y posibilita una evaluación más precisa de la deforestación.

Durante el envío de solicitudes a la API para la generación de archivos PDF, es indispensable que el equipo disponga de conexión a internet, condición necesaria para la correcta ejecución del proceso.

La extensión del estudio hacia otras zonas de la provincia con evidencia de deforestación, y eventualmente hacia provincias adicionales, fortalece la representatividad y el alcance del análisis.

La optimización del algoritmo resulta clave para detectar deforestación en zonas fronterizas afectadas por minería ilegal y ausencia de control gubernamental, donde el monitoreo convencional se ve limitado.

Se recomienda, que el algoritmo sea capaz de diferenciar de zonas donde existe urbanización y donde no existe urbanización, de igual manera donde hay carreteras primer y segundo orden.

Es conveniente que el algoritmo incorpore la capacidad de diferenciar entre zonas urbanizadas y no urbanizadas, así como identificar la presencia de carreteras de primer y segundo orden, con el fin de mejorar la precisión en la detección de deforestación.

REFERENCIAS BIBLIOGRAFICAS

[1] N. y C. Internacional, "La Amazonía," Naturaleza y Cultura Internacional, 2024. [En línea]. Disponible en: https://www.natureandculture.org/es/amazon-rainforest/

[2] Get Up and Goals, "Amazonia: El pulmón del planeta," Get Up and Goals, 2024. [En línea]. Disponible en: https://getupandgoals.es/amazonia-el-pulmon-del-planeta/. [Accedido: 6-nov-2024].

[3] Universidad Externado de Colombia, "El pulmón del mundo necesita un respiro para su preservación," Medio Ambiente y Sostenibilidad, 2024. [En línea]. Disponible en: https://medioambiente.uexternado.edu.co/el-pulmon-del-mundo-necesita-un-respiro-para-su-preservacion/. [Accedido: 6-nov-2024].

[4] "Mongabay Latam," Nuevo estudio: en los últimos 26 años, Ecuador ha perdido más de 2 millones de hectáreas de bosque, 2021. [En línea]. Disponible en: https://es.mongabay.com/2021/03/nuevo-estudio-en-los-ultimos-26-anos-ecuador-ha-perdido-mas-de-2-millones-de-hectareas-de-bosque/. [Accedido: 6-nov-2024].

[5] WWF Ecuador, "¿Qué es la deforestación?" WWF Ecuador, 2024. [En línea]. Disponible en: https://www.wwf.org.ec/?389690/que-es-la-deforestacion. [Accedido: 8-nov-2024].

[6] "Mongabay Latam," Amazonía ecuatoriana ha perdido más de 623 mil hectáreas en dos décadas, 2022. [En línea]. Disponible en: https://es.mongabay.com/2022/11/amazonia-ecuatoriana-ha-perdido-mas-de-623-mil-hectareas-en-dos-decadas/. [Accedido: 8-nov-2024].

[7] El Oriente, "Sucumbíos es la provincia amazónica más afectada por la deforestación," El Oriente, 2024. [En línea]. Disponible en: https://www.eloriente.com/articulo/sucumbios-es-la-provincia-amazonica-mas-afectada-por-la-deforestacion/36739. [Accedido: 8-nov-2024].

[8] "Mongabay Latam," Nuevo estudio: en los últimos 26 años, Ecuador ha perdido más de 2 millones de hectáreas de bosque, 2021. [En línea]. Disponible en: https://es.mongabay.com/2021/03/nuevo-estudio-en-los-ultimos-26-anos-ecuador-ha-perdido-mas-de-2-millones-de-hectareas-de-bosque/. [Accedido: 8-nov-2024].

[9] A. S. Quezada, J. D. Sevilla Tapia, y E. C. Avilés Sacoto, "Estimación de la tasa de deforestación en Pastaza y Orellana- Ecuador mediante el análisis multitemporal de imágenes satelitales durante el período 2000-2020," ALFA. Revista de Investigación en Ciencias Agronómicas y Veterinarias, vol. 6, no. 17, pp. 282-299, mayo-agosto 2022. [En línea]. Disponible: https://doi.org/10.33996/revistaalfa.v6i17.168

[10] A.-M. Tudorescu, C. Negru, B.-C. Mocanu, y F. Pop, "Quality sustaining vegetation index for natural resources monitoring using satellite images," Engineering Science and Technology, an International Journal, vol. 59, art. no. 101847, pp. 1-12, 2024. [En línea]. Disponible: https://doi.org/10.1016/j.jestch.2024.101847

[11] Y. Cui, P. Wang, J. F. Meirink, N. Ntantis, y J. S. Wijnands, "Solar radiation nowcasting based on geostationary satellite images and deep learning models," Solar Energy, vol. 282, art. no. 112866, pp. 1-15, 2024. [En línea]. Disponible: https://doi.org/10.1016/j.solener.2024.112866

[12] S. Zhang, X. He, B. Xue, T. Wu, K. Ren, y T. Zhao, "Segment-anything embedding for pixel-level road damage extraction using high-resolution satellite images," International Journal of Applied Earth Observation and Geoinformation, vol. 131, art. no. 103985, 2024. [En línea]. Disponible: https://doi.org/10.1016/j.jag.2024.103985
[13] Y. Zhao, X. He, S. Pan, Y. Bai, D. Wang, T. Li, F. Gong, y X. Zhang, "Satellite retrievals of water quality for diverse inland waters from Sentinel-2 images: An example from Zhejiang Province, China," Int. J. Appl. Earth Observ. Geoinformation, vol. 132, p. 104048, 2024. [En línea]. Disponible en: https://doi.org/10.1016/j.jag.2024.104048
[14] "Deforestación: Causas, Consecuencias y Soluciones," Evaluación de Impacto Ambiental, [En línea]. Disponible en: https://evaluaciondeimpactoambiental.com/deforestacion-ambiental/
[15] Global Forest Watch, "Panel del país: Ecuador (ECU)," [En línea]. Disponible en: https://www.globalforestwatch.org/dashboards/country/ECU/22/
[16] Global Forest Watch, "Panel del país: Ecuador (ECU)," [En línea]. Disponible en: https://www.globalforestwatch.org/dashboards/country/ECU/22/
[17] "Teledetección, una manera para redescubrir el mundo," Geoinnova, 27-abr.-2016. [En línea]. Disponible en: https://geoinnova.org/blog-territorio/teledeteccion-una-manera-redescubrir-mundo/
[18] FAO (Organización de las Naciones Unidas para la Alimentación y la Agricultura), "Inventario forestal," Conjunto de Herramientas para la Gestión Forestal Sostenible (GFS), [En línea]. Disponible en: https://www.fao.org/sustainable-forest-management-toolbox/modules/forest-inventory/es
[19] EOS Data Analytics, "Análisis espacial de datos: tipos, prácticas y usos," EOS Data Analytics Blog, 2-jun.-2022. [En línea]. Disponible en: https://eos.com/es/blog/analisis-espacial/
[20] "IMÁGENES SATELITALES: DESCUBRE SUS USOS Y …," Aprendix, [En línea]. Disponible en: https://aprendix.org/para-que-sirve-una-imagen-satelital/
[21] P. Kogut, "¿Cómo se interpretan las imágenes satelitales?," EOS Data Analytics Blog, 6-nov.-2024. [En línea]. Disponible en: https://eos.com/es/blog/interpretacion-de-imagenes-satelitales/
[22] EOS Data Analytics, "Apilamiento de índices (NDVI, NDWI, NDSI)," EOS Data Analytics (EOSDA), 27-sep.-2021 (última actualización: 06-nov.-2023). [En línea]. Disponible en: https://eos.com/es/make-an-analysis/index-stack/
[23] GeoSpectral, "¿Qué es el análisis multiespectral?," GeoSpectral Blog (Enterprise), 19-may.-2023. [En línea]. Disponible en: https://geospectral.com.mx/blogs/enterprise/que-es-el-analisis-multiespectral
[24] Specim, "What is hyperspectral imaging? A comprehensive guide," Specim, 27-jun.-2024. [En línea]. Disponible en: https://www.specim.com/technology/what-is-hyperspectral-imaging/
[25] K. Sergieieva, "Detección de cambios: cuándo usarla y ejemplos prácticos," EOS Data Analytics Blog, 27-feb.-2023. [En línea]. Disponible en: https://eos.com/es/blog/deteccion-de-cambios/
[26] IBM, "¿Qué es la segmentación de imágenes?," IBM Think (Topics), [En línea]. Disponible en: https://www.ibm.com/mx-es/think/topics/image-segmentation

[27] IBM, "¿Qué es la segmentación de imágenes?," IBM Think (Topics), [En línea]. Disponible en: https://www.ibm.com/mx-es/think/topics/image-segmentation
[28] Ultralytics, "Thresholding in Image Processing Explained," Ultralytics Blog, 12-ago.-2025. [En línea]. Disponible en: https://www.ultralytics.com/es/blog/thresholding-in-image-processing
[29] V. Gopalakrishnan, "Image segmentation using otsu threshold selection method," Medium, 2-jun.-2023. [En línea]. Disponible en: https://medium.com/@vignesh.g1609/image-segmentation-using-otsu-threshold-selection-method-856ccdacf22
[30] itcwebs, "Ventajas y desventajas de los principales lenguajes de programación," ITC Web Solutions, 6-ene.-2024. [En línea]. Disponible en: https://itcwebsolutions.com/desarrollo-y-soporte-web/programacion-y-herramientas/lenguajes-de-programacion/ventajas-y-desventajas-de-los-principales-lenguajes-de-programacion/
[31] BM, "¿Qué es Java?," IBM Think (Topics), [En línea]. Disponible en: https://www.ibm.com/mx-es/think/topics/java
[32] Fundamentos de Programación en C++, "El lenguaje C++," Fundamentos de Programación en C++ (Curso 2020/2021), 2020. [En línea]. Disponible en: https://www2.eii.uva.es/fund_inf/cpp/temas/1_introduccion/introduccion.html
[33] Formadores IT, "¿Qué es Matlab y para qué sirve?," Formadores IT, 8-ene.-2024. [En línea]. Disponible en: https://formadoresit.es/que-es-matlab-y-para-que-sirve/
[34] P. Pandey, "10 Python Image Manipulation Tools You Can Try Today," Built In, actualiz. por B. Whitfield, 28-abr.-2025. [En línea]. Disponible en: https://builtin.com/data-science/python-image-processing
[35] NumPy Developers, "What is NumPy?," NumPy v2.5.dev0 Manual, [En línea]. Disponible en: https://numpy.org/devdocs/user/whatisnumpy.html
[36] G. Boesch, "What is OpenCV? The Complete Guide (2025)," viso.ai, 1-oct.-2024. [En línea]. Disponible en: https://viso.ai/computer-vision/opencv/
[37] S. Gruppetta, "Image Processing With the Python Pillow Library," Real Python, [En línea]. Disponible en: https://realpython.com/image-processing-with-the-python-pillow-library/
[38] DataScientest, "Matplotlib: todo lo que tienes que saber sobre la librería Python de Dataviz," DataScientest Blog, 25-abr. [En línea]. Disponible en: https://datascientest.com/es/todo-sobre-matplotlib

ANEXOS

Anexo del algoritmo de programación

```
import os, re, glob, json, datetime, threading, tempfile
import numpy as np
import cv2
import tkinter as tk
from tkinter import ttk, filedialog, messagebox

from PIL import Image, ImageTk
import matplotlib
matplotlib.use("TkAgg")
import matplotlib.pyplot as plt
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg

# .env opcional
try:
    from dotenv import load_dotenv # type: ignore
    load_dotenv()
except Exception:
    pass

# OpenAI client
try:
    from openai import OpenAI
    _OPENAI_AVAILABLE = True
except Exception:
    _OPENAI_AVAILABLE = False

def ensure_openai_key_interactive(root=None) -> bool:
    if os.getenv("OPENAI_API_KEY"):
        return True
    try:
        import tkinter.simpledialog as sd
        parent = root if root else tk.Tk()
        if not root:
            parent.withdraw()
        key = sd.askstring("OpenAI API Key", "Pega tu OPENAI_API_KEY (no se guardará en
disco):", show='*')
        if key and key.strip():
            os.environ["OPENAI_API_KEY"] = key.strip()
            return True
    except Exception:
        pass
    return False

APP_PREFS = os.path.join(os.path.expanduser("~"), ".forest_gui_prefs.json")

EC_PROVINCES = [
    "Azuay","Bolívar","Cañar","Carchi","Chimborazo","Cotopaxi","El
Oro","Esmeraldas","Galápagos",
    "Guayas","Imbabura","Loja","Los Ríos","Manabí","Morona
Santiago","Napo","Orellana","Pastaza",
    "Pichincha","Santa Elena","Santo Domingo de los
Tsáchilas","Sucumbíos","Tungurahua","Zamora Chinchipe"
]

def load_prefs():
    prefs = {"max_width": 1800,"date_pattern_priority": "ISO",
             "province": "Imbabura","canton": "","localidad": "","dark_mode": True,
             "width_km": 1.0, "length_km": 1.0}
    try:
        if os.path.isfile(APP_PREFS):
            with open(APP_PREFS, "r", encoding="utf-8") as f:
                prefs.update(json.load(f))
    except Exception:
        pass
    return prefs

def save_prefs(prefs):
    try:
        with open(APP_PREFS, "w", encoding="utf-8") as f:
            json.dump(prefs, f, ensure_ascii=False, indent=2)
    except Exception:
        pass
```

```
def try_parse_date_from_name(name, mode="ISO"):
    base = os.path.basename(name)
    if mode == "LATAM":
        patterns = [r'(\d{2})[-_\.](\d{2})[-_\.](\d{4})', r'(\d{4})[-_\.](\d{2})[-
_\.](\d{2})']
    else:
        patterns = [r'(\d{4})[-_\.](\d{2})[-_\.](\d{2})', r'(\d{2})[-_\.](\d{2})[-
_\.](\d{4})']
    for pat in patterns:
        m = re.search(pat, base)
        if m:
            g = m.groups()
            try:
                if len(g[0]) == 4:
                    y, mth, d = int(g[0]), int(g[1]), int(g[2])
                else:
                    d, mth, y = int(g[0]), int(g[1]), int(g[2])
                return datetime.date(y, mth, d)
            except:
                pass
    for block in re.findall(r'(\d{8})', base):
        try:
            return datetime.date(int(block[:4]), int(block[4:6]), int(block[6:8]))
        except:
            try:
                return datetime.date(int(block[4:8]), int(block[2:4]), int(block[0:2]))
            except:
                pass
    return None

def compute_mask_pct(bgr, hsv_center, dH, smin, smax, vmin, vmax, open_iters=1,
close_iters=1):
    hsv_img = cv2.cvtColor(bgr, cv2.COLOR_BGR2HSV)
    centerH, centerS, centerV = hsv_center
    if centerH - dH < 0:
        low_wrap = np.array([0, smin, vmin], dtype=np.uint8)
        up_wrap  = np.array([max(0, min(centerH + dH, 179)), smax, vmax], dtype=np.uint8)
        low2     = np.array([max(0, min(centerH - dH + 180, 179)), smin, vmin],
dtype=np.uint8)
        up2      = np.array([179, smax, vmax], dtype=np.uint8)
        mask = cv2.inRange(hsv_img, low_wrap, up_wrap) | cv2.inRange(hsv_img, low2, up2)
    elif centerH + dH > 179:
        low3 = np.array([max(0, min(centerH - dH, 179)), smin, vmin], dtype=np.uint8)
        up3  = np.array([179, smax, vmax], dtype=np.uint8)
        low4 = np.array([0, smin, vmin], dtype=np.uint8)
        up4  = np.array([max(0, min(centerH + dH - 180, 179)), smax, vmax],
dtype=np.uint8)
        mask = cv2.inRange(hsv_img, low3, up3) | cv2.inRange(hsv_img, low4, up4)
    else:
        low1 = np.array([max(0, min(centerH - dH, 179)), smin, vmin], dtype=np.uint8)
        up1  = np.array([max(0, min(centerH + dH, 179)), smax, vmax], dtype=np.uint8)
        mask = cv2.inRange(hsv_img, low1, up1)

    kernel = cv2.getStructuringElement(cv2.MORPH_ELLIPSE, (3,3))
    if open_iters > 0:
        mask = cv2.morphologyEx(mask, cv2.MORPH_OPEN, kernel, iterations=open_iters)
    if close_iters > 0:
        mask = cv2.morphologyEx(mask, cv2.MORPH_CLOSE, kernel, iterations=close_iters)

    pct = (np.count_nonzero(mask) / mask.size) * 100.0
    return mask, pct

# -------- Markdown -> PDF con histograma --------
def save_markdown_to_pdf_with_plot(markdown_text:str, pdf_path:str, plot_png:str=None):
    try:
        from reportlab.lib.pagesizes import A4
        from reportlab.lib.styles import getSampleStyleSheet, ParagraphStyle
        from reportlab.lib.enums import TA_LEFT
        from reportlab.platypus import SimpleDocTemplate, Paragraph, Spacer, Image as
RLImage, PageBreak
        from reportlab.lib.units import cm
    except Exception:
        raise RuntimeError("Falta reportlab. Instala con: pip install reportlab")

    styles = getSampleStyleSheet()
    styles.add(ParagraphStyle(name="H1", parent=styles["Heading1"], fontName="Helvetica-
Bold", fontSize=16, leading=20, spaceAfter=8))
```

```
    styles.add(ParagraphStyle(name="H2", parent=styles["Heading2"], fontName="Helvetica-
Bold", fontSize=13, leading=16, spaceAfter=6))
    styles.add(ParagraphStyle(name="Body", parent=styles["BodyText"],
fontName="Helvetica", fontSize=10.5, leading=14, spaceAfter=4, alignment=TA_LEFT))

    doc = SimpleDocTemplate(pdf_path, pagesize=A4, leftMargin=2*cm, rightMargin=2*cm,
topMargin=2*cm, bottomMargin=2*cm)
    story = []

    # Conversión ligera de Markdown
    for raw in markdown_text.splitlines():
        line = raw.strip()
        if not line:
            story.append(Spacer(1, 6))
            continue
        if line.startswith("## "):
            story.append(Paragraph(line[3:], styles["H2"]))
        elif line.startswith("# "):
            story.append(Paragraph(line[2:], styles["H1"]))
        elif line.startswith("- "):
            story.append(Paragraph("• " + line[2:], styles["Body"]))
        else:
            story.append(Paragraph(line, styles["Body"]))

    # Añadir histograma (si existe)
    if plot_png and os.path.isfile(plot_png):
        story.append(PageBreak())
        story.append(Paragraph("Figura 1. Evolución del porcentaje de forestación (%)",
styles["H2"]))
        img = RLImage(plot_png)
        max_w = A4[0] - 4*cm  # ancho máximo (márgenes)
        # Escalar manteniendo proporción
        iw, ih = img.imageWidth, img.imageHeight
        scale = min(max_w / iw, 1.0)
        img.drawWidth = iw * scale
        img.drawHeight = ih * scale
        story.append(img)

    doc.build(story)

class ForestApp(ttk.Frame):
    def __init__(self, master):
        super().__init__(master)
        self.master: tk.Tk = master
        self.prefs = load_prefs()
        self.params = None
        self.results = []
        self.folder = None

        self.calib_bgr = None
        self.calib_hsv = None
        self.calib_base_name = None
        self.calib_center = [60,150,150]  # corregido
        self.calib_last_click = None

        self.meta = {"province": self.prefs.get("province","Imbabura"),
                     "canton": self.prefs.get("canton",""),
                     "localidad": self.prefs.get("localidad","")}

        self._build_style()
        self._build_menu()
        self._build_toolbar()
        self._build_main()
        self._build_status()
        self._bind_shortcuts()

    def _apply_theme(self, dark=True):
        style = ttk.Style(self.master)
        try:
            style.theme_use("clam")
        except Exception:
            pass
        base_fg = "#eaeaea" if dark else "#222"
        base_bg = "#1e1f22" if dark else "#ffffff"
        panel_bg= "#25262a" if dark else "#f6f7fb"
        self.master.configure(bg=base_bg)
        style.configure(".", font=("Segoe UI", 10))
```

```
        style.configure("TFrame", background=panel_bg)
        style.configure("TLabel", background=panel_bg, foreground=base_fg)
        style.configure("Header.TLabel", font=("Segoe UI", 11, "bold"),
foreground=base_fg)
        style.configure("TButton", padding=6)

    def _build_style(self):
        self.master.title("Forestación • HSV (Causas + Área km²)")
        self.master.geometry("1240x800"); self.master.minsize(1080, 660)
        self._apply_theme(self.prefs.get("dark_mode", True))

    def _build_menu(self):
        menubar = tk.Menu(self.master)
        m_file = tk.Menu(menubar, tearoff=0)
        m_file.add_command(label="Exportar CSV…", command=self.action_export_csv,
accelerator="Ctrl+E")
        m_file.add_command(label="Guardar gráfico PNG…",
command=self.action_save_plot_png, accelerator="Ctrl+G")
        m_file.add_separator()
        m_file.add_command(label="Enviar a OpenAI…", command=self.action_send_to_openai,
accelerator="Ctrl+O")
        m_file.add_separator()
        m_file.add_command(label="Salir", command=self.action_quit, accelerator="Ctrl+Q")
        menubar.add_cascade(label="Archivo", menu=m_file)
        self.master.config(menu=menubar)

    def _build_toolbar(self):
        bar = ttk.Frame(self.master, padding=(10,8)); bar.pack(fill="x")
        ttk.Button(bar, text="🎯 Calibración", command=lambda:
self.tabs.select(self.tab_calib)).pack(side="left", padx=4)
        ttk.Button(bar, text="🗁 Analizar carpeta",
command=self.action_analyze_folder).pack(side="left", padx=4)
        ttk.Button(bar, text="💾 CSV", command=self.action_export_csv).pack(side="left",
padx=4)
        ttk.Button(bar, text="🖼 PNG", command=self.action_save_plot_png).pack(side="left",
padx=4)
        ttk.Button(bar, text="📤 Enviar a OpenAI",
command=self.action_send_to_openai).pack(side="left", padx=8)
        ttk.Separator(bar, orient="vertical").pack(side="left", fill="y", padx=8)
        ttk.Button(bar, text="🔋 Salir", command=self.action_quit).pack(side="left",
padx=4)

    def _build_main(self):
        body = ttk.Panedwindow(self.master, orient="horizontal"); body.pack(fill="both",
expand=True)
        left = ttk.Frame(body, padding=16); body.add(left, weight=1)
        self.params_var = tk.StringVar(value="Parámetros: (sin calibrar)");
ttk.Label(left, textvariable=self.params_var, style="Header.TLabel").pack(anchor="w")
        self.results_var = tk.StringVar(value="Resultados: 0 imágenes"); ttk.Label(left,
textvariable=self.results_var).pack(anchor="w", pady=(0,12))

        # Ubicación
        geo = ttk.LabelFrame(left, text="Ubicación del estudio"); geo.pack(fill="x",
pady=(4,12))
        ttk.Label(geo, text="Provincia").grid(row=0, column=0, sticky="w", padx=6, pady=4)
        self.cmb_prov = ttk.Combobox(geo, values=EC_PROVINCES, state="readonly", width=28)
        self.cmb_prov.set(self.prefs.get("province","Imbabura"))
        self.cmb_prov.grid(row=0, column=1, sticky="we", padx=6, pady=4)
        ttk.Label(geo, text="Cantón").grid(row=1, column=0, sticky="w", padx=6, pady=4)
        self.ent_canton = ttk.Entry(geo, width=30); self.ent_canton.insert(0,
self.prefs.get("canton",""))
        self.ent_canton.grid(row=1, column=1, sticky="we", padx=6, pady=4)
        ttk.Label(geo, text="Localidad").grid(row=2, column=0, sticky="w", padx=6, pady=4)
        self.ent_localidad = ttk.Entry(geo, width=30); self.ent_localidad.insert(0,
self.prefs.get("localidad",""))
        self.ent_localidad.grid(row=2, column=1, sticky="we", padx=6, pady=4)

        # Área de estudio
        area = ttk.LabelFrame(left, text="Área de estudio"); area.pack(fill="x",
pady=(4,12))
        ttk.Label(area, text="Ancho (km)").grid(row=0, column=0, sticky="w", padx=6,
pady=4)
        self.ent_width_km = ttk.Entry(area, width=12)
        self.ent_width_km.insert(0, str(self.prefs.get("width_km", 1.0)))
        self.ent_width_km.grid(row=0, column=1, sticky="w", padx=6, pady=4)
```

```
        ttk.Label(area, text="Largo (km)").grid(row=1, column=0, sticky="w", padx=6,
pady=4)
        self.ent_length_km = ttk.Entry(area, width=12)
        self.ent_length_km.insert(0, str(self.prefs.get("length_km", 1.0)))
        self.ent_length_km.grid(row=1, column=1, sticky="w", padx=6, pady=4)
        self.area_var = tk.StringVar(value="Área total: 1.00 km²")
        ttk.Label(area, textvariable=self.area_var).grid(row=2, column=0, columnspan=2,
sticky="w", padx=6, pady=4)

        def _update_area_label(*_):
            try:
                w = float(self.ent_width_km.get()); l = float(self.ent_length_km.get())
                self.area_var.set(f"Área total: {w*l:.2f} km²")
            except:
                self.area_var.set("Área total: (valor inválido)")
        self.ent_width_km.bind("<KeyRelease>", _update_area_label)
        self.ent_length_km.bind("<KeyRelease>", _update_area_label)
        _update_area_label()

        # Tabs
        self.tabs = ttk.Notebook(body); body.add(self.tabs, weight=3)
        tab_plot = ttk.Frame(self.tabs, padding=10); self.tabs.add(tab_plot,
text="Histograma")
        self.fig, self.ax = plt.subplots(figsize=(7.8,4.4), dpi=100)
        self.canvas = FigureCanvasTkAgg(self.fig, master=tab_plot)
        self.canvas.get_tk_widget().pack(fill="both", expand=True)
        self.ax.set_title("Evolución de forestación (%)", fontsize=12)
        self.ax.set_xlabel("Fecha", fontsize=10); self.ax.set_ylabel("% de forestación",
fontsize=10)
        self.ax.grid(True, linestyle="--", alpha=0.35)
        self.fig.tight_layout()

        self.tab_calib = ttk.Frame(self.tabs, padding=(10,10));
self.tabs.add(self.tab_calib, text="Calibración")
        self._build_calib_big(self.tab_calib)

        self.tab_log = ttk.Frame(self.tabs, padding=10); self.tabs.add(self.tab_log,
text="Registro")
        self.txt = tk.Text(self.tab_log, height=14, wrap="word", bg="#0f1115",
fg="#eaeaea",
                           insertbackground="#eaeaea", relief="flat")
        self.txt.pack(fill="both", expand=True, pady=(2,10))
        self.progress = ttk.Progressbar(self.tab_log, mode="determinate", maximum=100);
self.progress.pack(fill="x")

    def _build_status(self):
        self.status = ttk.Label(self.master, text="Listo.", anchor="w", relief="flat");
self.status.pack(fill="x", pady=(4,0), padx=8)

    def _bind_shortcuts(self):
        self.master.bind("<Control-q>", lambda e: self.action_quit())
        self.master.bind("<Control-e>", lambda e: self.action_export_csv())
        self.master.bind("<Control-g>", lambda e: self.action_save_plot_png())
        self.master.bind("<Control-o>", lambda e: self.action_send_to_openai())

    def set_status(self, msg): self.status.configure(text="  " + msg)
    def log(self, msg):
        try:
            self.txt.insert("end", msg + "\n"); self.txt.see("end")
        except Exception:
            pass

    def action_quit(self):
        if messagebox.askyesno("Salir", "¿Deseas cerrar la aplicación?"):
            self.master.destroy()

    def _build_calib_big(self, parent):
        topbar = ttk.Frame(parent); topbar.pack(fill="x", pady=(0,8))
        ttk.Button(topbar, text="⬇ Cargar base",
command=self.calib_load_image).pack(side="left", padx=4)
        ttk.Button(topbar, text="💾 Guardar máscara+params",
command=self.calib_save).pack(side="left", padx=4)
        ttk.Button(topbar, text="➡ Aplicar a la app",
command=self.calib_apply).pack(side="left", padx=4)
        self.calib_info = tk.StringVar(value="Carga una imagen y haz click en el área de
vegetación (Original).")
```

```
        ttk.Label(topbar, textvariable=self.calib_info,
style="Header.TLabel").pack(side="right")

        mid = ttk.Frame(parent); mid.pack(fill="both", expand=True)
        self.canvas_orig = tk.Canvas(mid, bd=0, highlightthickness=0,
background="#1b1d21")
        self.canvas_mask = tk.Canvas(mid, bd=0, highlightthickness=0,
background="#1b1d21")
        self.canvas_res  = tk.Canvas(mid, bd=0, highlightthickness=0,
background="#1b1d21")
        self.canvas_orig.grid(row=0, column=0, sticky="nsew", padx=6, pady=6)
        self.canvas_mask.grid(row=0, column=1, sticky="nsew", padx=6, pady=6)
        self.canvas_res.grid(row=0, column=2, sticky="nsew", padx=6, pady=6)
        mid.columnconfigure(0, weight=1); mid.columnconfigure(1, weight=1);
mid.columnconfigure(2, weight=1)
        mid.rowconfigure(0, weight=1)
        self.canvas_orig.bind("<Configure>", lambda e: self.calib_update_previews())
        self.canvas_mask.bind("<Configure>", lambda e: self.calib_update_previews())
        self.canvas_res.bind("<Configure>",  lambda e: self.calib_update_previews())
        self.canvas_orig.bind("<Button-1>", self.calib_on_click_original)

        bottom = ttk.Frame(parent); bottom.pack(fill="x", pady=(6,0))
        def add_slider(label, from_, to_, init):
            frm = ttk.Frame(bottom); frm.pack(side="left", padx=10)
            ttk.Label(frm, text=label).pack(anchor="w")
            var = tk.IntVar(value=init)
            s = ttk.Scale(frm, from_=from_, to=to_, orient="horizontal", variable=var,
command=lambda e: self.calib_update_previews())
            s.pack(fill="x", padx=2, pady=2)
            val_label = ttk.Label(frm, text=str(init)); val_label.pack(anchor="e")
            def on_var(*_): val_label.config(text=str(var.get()))
            var.trace_add("write", on_var)
            return var

        self.var_dH   = add_slider("dH", 0, 40, 15)
        self.var_smin = add_slider("S min", 0, 255, 40)
        self.var_smax = add_slider("S max", 0, 255, 255)
        self.var_vmin = add_slider("V min", 0, 255, 40)
        self.var_vmax = add_slider("V max", 0, 255, 255)
        self.var_open = add_slider("Open iters", 0, 5, 1)
        self.var_close= add_slider("Close iters", 0, 5, 1)

        zfrm = ttk.Frame(bottom); zfrm.pack(side="left", padx=10)
        ttk.Label(zfrm, text="Zoom (%)").pack(anchor="w")
        self.var_zoom = tk.IntVar(value=100)
        ttk.Scale(zfrm, from_=50, to=200, orient="horizontal", variable=self.var_zoom,
command=lambda e: self.calib_update_previews()).pack(fill="x", padx=2, pady=2)
        self.lbl_zoom = ttk.Label(zfrm, text="100"); self.lbl_zoom.pack(anchor="e")
        self.var_zoom.trace_add("write", lambda *_:
self.lbl_zoom.config(text=str(self.var_zoom.get())))

    def calib_load_image(self):
        path_img = filedialog.askopenfilename(title="Selecciona imagen base",
                                              filetypes=[("Imágenes","*.jpg *.jpeg *.png
*.bmp *.tif *.tiff")])
        if not path_img: return
        bgr = cv2.imread(path_img)
        if bgr is None:
            messagebox.showerror("Error", "No se pudo abrir la imagen base."); return
        self.calib_bgr = bgr
        self.calib_hsv = cv2.cvtColor(self.calib_bgr, cv2.COLOR_BGR2HSV)
        self.calib_base_name = os.path.splitext(os.path.basename(path_img))[0]
        self.calib_last_click = None
        self.calib_update_previews()
        self.set_status("Imagen base cargada")

    def calib_on_click_original(self, event):
        if self.calib_bgr is None: return
        cw = self.canvas_orig.winfo_width(); ch = self.canvas_orig.winfo_height()
        zoom = max(0.5, min(2.0, self.var_zoom.get()/100.0))
        ih, iw = self.calib_bgr.shape[:2]
        scale = min(cw/(iw*zoom), ch/(ih*zoom))
        if scale <= 0: return
        draw_w = int(iw*zoom*scale); draw_h = int(ih*zoom*scale)
        offx = (cw - draw_w)//2; offy = (ch - draw_h)//2
        x = event.x - offx; y = event.y - offy
        if x < 0 or y < 0 or x >= draw_w or y >= draw_h: return
```

```
        ix = int(x / (zoom*scale)); iy = int(y / (zoom*scale))
        ix = max(0, min(ix, iw-1)); iy = max(0, min(iy, ih-1))
        H,S,V = self.calib_hsv[iy, ix].tolist()
        self.calib_center = [H,S,V]; self.calib_last_click = (ix, iy)
        self.calib_update_previews()

    def calib_get_mask(self):
        if self.calib_bgr is None: return None, 0.0
        dH   = int(self.var_dH.get())
        smin = int(min(self.var_smin.get(), self.var_smax.get()))
        smax = int(max(self.var_smin.get(), self.var_smax.get()))
        vmin = int(min(self.var_vmin.get(), self.var_vmax.get()))
        vmax = int(max(self.var_vmin.get(), self.var_vmax.get()))
        open_iters  = int(self.var_open.get())
        close_iters = int(self.var_close.get())
        centerH, centerS, centerV = self.calib_center
        hsv = self.calib_hsv
        if centerH - dH < 0:
            low_wrap = np.array([0, smin, vmin], dtype=np.uint8)
            up_wrap  = np.array([max(0, min(centerH + dH, 179)), smax, vmax],
dtype=np.uint8)
            low2     = np.array([max(0, min(centerH - dH + 180, 179)), smin, vmin],
dtype=np.uint8)
            up2      = np.array([179, smax, vmax], dtype=np.uint8)
            mask = cv2.inRange(hsv, low_wrap, up_wrap) | cv2.inRange(hsv, low2, up2)
        elif centerH + dH > 179:
            low3 = np.array([max(0, min(centerH - dH, 179)), smin, vmin], dtype=np.uint8)
            up3  = np.array([179, smax, vmax], dtype=np.uint8)
            low4 = np.array([0, smin, vmin], dtype=np.uint8)
            up4  = np.array([max(0, min(centerH + dH - 180, 179)), smax, vmax],
dtype=np.uint8)
            mask = cv2.inRange(hsv, low3, up3) | cv2.inRange(hsv, low4, up4)
        else:
            low1 = np.array([max(0, min(centerH - dH, 179)), smin, vmin], dtype=np.uint8)
            up1  = np.array([max(0, min(centerH + dH, 179)), smax, vmax], dtype=np.uint8)
            mask = cv2.inRange(hsv, low1, up1)
        kernel = cv2.getStructuringElement(cv2.MORPH_ELLIPSE, (3,3))
        if open_iters > 0:
            mask = cv2.morphologyEx(mask, cv2.MORPH_OPEN, kernel, iterations=open_iters)
        if close_iters > 0:
            mask = cv2.morphologyEx(mask, cv2.MORPH_CLOSE, kernel, iterations=close_iters)
        pct = (np.count_nonzero(mask) / mask.size) * 100.0
        return mask, pct

    def calib_update_previews(self, *_):
        if self.calib_bgr is None:
            for c in (self.canvas_orig, self.canvas_mask, self.canvas_res):
                c.delete("all"); c.create_text(10, 10, text="(Sin imagen)",
fill="#9aa0a6", anchor="nw")
            return
        mask, pct = self.calib_get_mask()
        orig = self.calib_bgr.copy()
        if self.calib_last_click is not None:
            cv2.circle(orig, (self.calib_last_click[0], self.calib_last_click[1]), 8,
(0,0,255), -1)
        mask3 = cv2.cvtColor(mask, cv2.COLOR_GRAY2BGR)
        res = cv2.bitwise_and(self.calib_bgr, self.calib_bgr, mask=mask)
        self._render_to_canvas(self.canvas_orig, orig, annotate="Original")
        self._render_to_canvas(self.canvas_mask, mask3, annotate=f"Forestación:
{pct:.2f}%")
        self._render_to_canvas(self.canvas_res,  res,   annotate=f"Resultado •
{pct:.2f}%")
        H,S,V = self.calib_center
        self.calib_info.set(f"Centro HSV=({H},{S},{V}) • %: {pct:.2f} • Zoom:
{self.var_zoom.get()}%")

    def _render_to_canvas(self, canvas, bgr_img, annotate=None):
        if bgr_img is None: return
        cw = canvas.winfo_width(); ch = canvas.winfo_height()
        if cw < 10 or ch < 10: return
        ih, iw = bgr_img.shape[:2]
        zoom = max(0.5, min(2.0, self.var_zoom.get()/100.0))
        scale = min(cw/(iw*zoom), ch/(ih*zoom))
        if scale <= 0: return
        draw_w = int(iw*zoom*scale); draw_h = int(ih*zoom*scale)
        resized = cv2.resize(bgr_img, (draw_w, draw_h), interpolation=cv2.INTER_AREA)
        rgb = cv2.cvtColor(resized, cv2.COLOR_BGR2RGB)
```

```
        pil = Image.fromarray(rgb)
        photo = ImageTk.PhotoImage(pil)
        canvas.delete("all")
        offx = (cw - draw_w)//2; offy = (ch - draw_h)//2
        canvas.create_image(offx, offy, image=photo, anchor="nw")
        if annotate:
            canvas.create_text(offx+10, offy+22, text=annotate, fill="#f1f5f9",
                               anchor="w", font=("Segoe UI", 10, "bold"))
        canvas.image = photo

    def calib_save(self):
        if self.calib_bgr is None:
            messagebox.showwarning("Aviso", "Carga una imagen primero."); return
        mask, pct = self.calib_get_mask()
        out_mask = f"{self.calib_base_name}_mask.png"
        cv2.imwrite(out_mask, mask)
        dH   = int(self.var_dH.get())
        smin = int(min(self.var_smin.get(), self.var_smax.get())); smax =
int(max(self.var_smin.get(), self.var_smax.get()))
        vmin = int(min(self.var_vmin.get(), self.var_vmax.get())); vmax =
int(max(self.var_vmin.get(), self.var_vmax.get()))
        open_iters  = int(self.var_open.get()); close_iters = int(self.var_close.get())
        with open(f"{self.calib_base_name}_params.txt","w", encoding="utf-8") as f:
            for k,v in
[("centerH",self.calib_center[0]),("centerS",self.calib_center[1]),("centerV",self.calib_c
enter[2]),
                         ("dH",dH),("smin",smin),("smax",smax),("vmin",vmin),("vmax",vmax),
                         ("open_iters",open_iters),("close_iters",close_iters)]:
                f.write(f"{k}={v}\n")
        self.set_status("Máscara y parámetros guardados")

    def calib_apply(self):
        if self.calib_bgr is None:
            messagebox.showwarning("Aviso", "Carga una imagen y ajusta los parámetros.");
return
        self.params = {
            "centerH": self.calib_center[0], "centerS": self.calib_center[1], "centerV":
self.calib_center[2],
            "dH": int(self.var_dH.get()),
            "smin": int(min(self.var_smin.get(), self.var_smax.get())),
            "smax": int(max(self.var_smin.get(), self.var_smax.get())),
            "vmin": int(min(self.var_vmin.get(), self.var_vmax.get())),
            "vmax": int(max(self.var_vmin.get(), self.var_vmax.get())),
            "open_iters": int(self.var_open.get()),
            "close_iters": int(self.var_close.get())
        }
        self.refresh_labels()
        self.set_status("Parámetros aplicados")

    def action_analyze_folder(self):
        if not self.params:
            messagebox.showwarning("Aviso", "Primero calibra o carga parámetros HSV.");
            self.tabs.select(self.tab_calib);
            return
        folder = filedialog.askdirectory(title="Selecciona carpeta con imágenes")
        if not folder: return
        self._update_geo_and_area()
        self.folder = folder
        files = []
        for ext in ("*.jpg","*.jpeg","*.png","*.bmp","*.tif","*.tiff"):
            files += glob.glob(os.path.join(folder, ext))
        files.sort()
        if not files:
            messagebox.showwarning("Aviso", "No se encontraron imágenes en la carpeta.");
return

        self.results = []; self.progress["value"] = 0; self.progress["maximum"] =
len(files)
        try:
            self.txt.delete("1.0", "end")
        except Exception:
            pass
        self.log(f"Ubicación: Provincia={self.meta['province']} |
Cantón={self.meta['canton']} | Localidad={self.meta['localidad']}")
        self.log(f"Área total: {self.meta['area_km2']:.4f} km²
(Ancho={self.meta['width_km']} km, Largo={self.meta['length_km']} km)")
        self.set_status("Analizando imágenes…")
```

```
        mode = self.prefs.get("date_pattern_priority","ISO")

        def worker():
            count = 0
            total_area = float(self.meta["area_km2"])
            for fp in files:
                try:
                    img = cv2.imread(fp)
                    if img is None:
                        self.log(f"Saltado: {os.path.basename(fp)}"); continue
                    h, w = img.shape[:2]
                    if max(h, w) > self.prefs["max_width"]:
                        scale = float(self.prefs["max_width"]) / float(max(h, w))
                        img = cv2.resize(img, None, fx=scale, fy=scale,
interpolation=cv2.INTER_AREA)
                    mask, pct = compute_mask_pct(
                        img, (self.params["centerH"], self.params["centerS"],
self.params["centerV"]),
                        self.params["dH"], self.params["smin"], self.params["smax"],
                        self.params["vmin"], self.params["vmax"],
                        self.params["open_iters"], self.params["close_iters"]
                    )
                    dt = try_parse_date_from_name(fp, mode=mode)
                    if dt is None: dt =
datetime.datetime.fromtimestamp(os.path.getmtime(fp)).date()
                    cover_km2 = (float(pct) / 100.0) * total_area
                    self.results.append({
                        "file": os.path.basename(fp),
                        "date": dt,
                        "pct": pct,
                        "cover_km2": cover_km2,
                        "province": self.meta["province"],
                        "canton": self.meta["canton"],
                        "localidad": self.meta["localidad"]
                    })
                    self.log(f"{os.path.basename(fp)} -> {pct:.2f}% ({cover_km2:.4f}
km²)")
                except Exception as e:
                    self.log(f"Error con {os.path.basename(fp)}: {e}")
                finally:
                    count += 1;
                    try:
                        self.progress["value"] = count; self.master.update_idletasks()
                    except Exception:
                        pass

            # Ordenar y calcular delta vs base (primera por fecha)
            self.results.sort(key=lambda x: x["date"])
            if self.results:
                base_pct = float(self.results[0]["pct"])
                for r in self.results:
                    r["delta_pct_vs_base"] = float(r["pct"]) - base_pct
                    r["delta_km2_vs_base"] = (r["delta_pct_vs_base"] / 100.0) * total_area
            self.refresh_labels()
            self.set_status(f"Listo. Analizadas {len(self.results)} imágenes.")
            self.render_histogram_embedded()
            try:
                messagebox.showinfo("Listo", f"Analizadas {len(self.results)} imágenes.")
            except Exception:
                pass
        threading.Thread(target=worker, daemon=True).start()

    def render_histogram_embedded(self):
        self.ax.clear()
        self.ax.set_title("Evolución de forestación (%)", fontsize=12)
        self.ax.set_xlabel("Fecha", fontsize=10); self.ax.set_ylabel("% de forestación",
fontsize=10)
        self.ax.grid(True, linestyle="--", alpha=0.35)
        if self.results:
            dates = [r["date"] for r in self.results]
            pcts  = [r["pct"] for r in self.results]
            labels= [r["file"] for r in self.results]
            self.ax.plot(dates, pcts, marker='o')
            for x,y,lab in zip(dates,pcts,labels):
                self.ax.annotate(lab, (x,y), xytext=(5,5), textcoords="offset points",
fontsize=8, rotation=30)
        else:
```

```
            self.ax.text(0.5, 0.5, "Sin datos aún", ha="center", va="center",
transform=self.ax.transAxes, alpha=0.6)
        self.fig.tight_layout()
        self.canvas.draw_idle()

    def action_save_plot_png(self):
        if not self.results:
            messagebox.showwarning("Aviso", "No hay resultados para graficar."); return
        out = filedialog.asksaveasfilename(title="Guardar gráfico",
defaultextension=".png",
                                           filetypes=[("PNG","*.png")],
initialfile="hist_forestacion.png")
        if not out: return
        self.fig.savefig(out, dpi=180)
        self.log(f"Gráfico guardado en: {out}")
        self.set_status("Gráfico guardado")

    def action_export_csv(self):
        if not self.results:
            messagebox.showwarning("Aviso", "No hay resultados para exportar."); return
        out = filedialog.asksaveasfilename(title="Guardar CSV", defaultextension=".csv",
                                           filetypes=[("CSV","*.csv")],
initialfile="resultados_forestacion.csv")
        if not out: return
        try:
            with open(out, "w", encoding="utf-8") as f:
                f.write("file,date,forest_pct,cover_km2,delta_pct_vs_base,delta_km2_vs_bas
e,province,canton,localidad\n")
                for r in self.results:
                    f.write(f"{r['file']},{r['date'].isoformat()},{r['pct']:.4f},{r['cover
_km2']:.6f},{r.get('delta_pct_vs_base',0):.4f},{r.get('delta_km2_vs_base',0):.6f},{r['prov
ince']},{r['canton']},{r['localidad']}\n")
            self.log(f"CSV guardado en: {out}")
            self.set_status("CSV guardado")
        except Exception as e:
            messagebox.showerror("Error", f"No se pudo guardar el CSV:\n{e}")
            self.set_status("Error guardando CSV")

    def _update_geo_and_area(self):
        self.prefs["province"] = self.meta["province"] = self.cmb_prov.get()
        self.prefs["canton"] = self.meta["canton"] = self.ent_canton.get().strip()
        self.prefs["localidad"] = self.meta["localidad"] =
self.ent_localidad.get().strip()
        # área
        try:
            w = float(self.ent_width_km.get()); l = float(self.ent_length_km.get())
        except Exception:
            w, l = 1.0, 1.0
        self.prefs["width_km"] = self.meta["width_km"] = w
        self.prefs["length_km"] = self.meta["length_km"] = l
        self.meta["area_km2"] = float(w*l)
        save_prefs(self.prefs)

    def refresh_labels(self):
        if self.params:
            p = self.params
            self.params_var.set(
                f"Parámetros: H={p['centerH']}±{p['dH']}  "
                f"S:[{p['smin']},{p['smax']}]  V:[{p['vmin']},{p['vmax']}]  "
                f"(morf: {p['open_iters']}/{p['close_iters']})"
            )
        else:
            self.params_var.set("Parámetros: (sin calibrar)")
        self.results_var.set(f"Resultados: {len(self.results)} imágenes")

    def action_send_to_openai(self):
        if not _OPENAI_AVAILABLE:
            messagebox.showerror("OpenAI", "Falta el paquete 'openai'. Instala con:\n\npip
install openai")
            return
        if not self.results:
            messagebox.showwarning("Aviso", "Primero analiza una carpeta para generar
resultados.")
            return
        if not ensure_openai_key_interactive(self.master):
            messagebox.showerror("OpenAI", "No hay OPENAI_API_KEY. Cancela o vuelve a
intentarlo.")
```

```
            return

        self._update_geo_and_area()
        compact = [
            {
                "file": r["file"],
                "date": r["date"].isoformat(),
                "forest_pct": round(float(r["pct"]), 4),
                "cover_km2": round(float(r["cover_km2"]), 6),
                "delta_pct_vs_base": round(float(r.get("delta_pct_vs_base",0.0)), 4),
                "delta_km2_vs_base": round(float(r.get("delta_km2_vs_base",0.0)), 6)
            }
            for r in self.results
        ]
        payload = {
            "ubicacion": self.meta,
            "area_km2_total": round(float(self.meta["area_km2"]), 6),
            "mediciones": compact
        }
        prompt = (
            "Eres un analista ambiental especializado en deforestación/reforestación en
Ecuador. "
            "Con el JSON siguiente, redacta un informe técnico (máx. 2 páginas, español)
que priorice: "
            "1) Identificación de tendencias (% y km²) y cambios vs la línea base. "
            "2) Posibles **motores/causas** de cambio en cobertura vegetal (contextualiza
para Ecuador): "
            "   expansión agrícola/ganadera, apertura de vías, tala legal/ilegal,
incendios forestales/ENSO, "
            "   minería (legal/ilegal), hidrocarburos, expansión urbana, y **factores de
forestación** "
            "   (programas de restauración, regeneración natural, plantaciones
forestales). "
            "3) Advertencias metodológicas: fenología estacional, nubosidad/sombras,
diferencias de sensor/ángulo, "
            "   umbral HSV y errores de clasificación. "
            "4) Recomendaciones accionables: monitoreo periódico, verificación en campo,
cruce con capas oficiales "
            "   (cobertura/uso de suelo), alertas tempranas, y métricas adicionales (NDVI
si aplica). "
            "Incluye: Resumen ejecutivo; Metodología (HSV, área km² = ancho×largo);
Resultados con tabla "
            "(Fecha, % Forestación, km², Δ% vs base, Δkm² vs base); Discusión de causas
probables; Limitaciones; "
            "y Recomendaciones. Usa **Markdown** con subtítulos claros. JSON:\n\n"
            + json.dumps(payload, ensure_ascii=False)
        )

        self.set_status("Generando informe con OpenAI…")
        self.log("Enviando datos (texto)…")
        try:
            client = OpenAI()
            res = client.responses.create(
                model="gpt-5",
                input=prompt
            )
            report_md = res.output_text
        except Exception as e:
            messagebox.showerror("Error API", f"No se pudo generar el informe:\n{e}")
            self.set_status("Error al generar informe")
            return

        # Guardar como MD o PDF (si PDF, incrustar histograma actual)
        out = filedialog.asksaveasfilename(
            title="Guardar informe",
            defaultextension=".pdf",
            filetypes=[("PDF", "*.pdf"), ("Markdown", "*.md"), ("Texto", "*.txt")],
            initialfile="informe_forestacion.pdf"
        )
        if not out:
            return

        try:
            if out.lower().endswith(".pdf"):
                tmp_png = tempfile.mktemp(suffix=".png")
                self.fig.savefig(tmp_png, dpi=180)
                save_markdown_to_pdf_with_plot(report_md, out, plot_png=tmp_png)
```

```
                try:
                    os.remove(tmp_png)
                except Exception:
                    pass
            else:
                with open(out, "w", encoding="utf-8") as f:
                    f.write(report_md)
            self.log(f"Informe guardado: {out}")
            messagebox.showinfo("Listo", "Informe generado.")
            self.set_status("Informe generado")
        except RuntimeError as e:
            messagebox.showerror("PDF", str(e))
            self.set_status("Falta reportlab")
        except Exception as e:
            messagebox.showerror("Error", f"No se pudo guardar el informe:\n{e}")
            self.set_status("Error guardando informe")

def main():
    root = tk.Tk()
    app = ForestApp(root); app.pack(fill="both", expand=True)
    root.mainloop()

if __name__ == "__main__":
    main()
```