UNIVERSIDAD POLITÉCNICA SALESIANA
SEDE QUITO
CARRERA DE INGENIERÍA BIOMÉDICA

EVALUACIÓN DE TÉCNICAS DE NAVEGACIÓN PARA COLONOSCOPIA VIRTUAL

Trabajo de titulación previo a la obtención del Título de:
INGENIERO BIOMÉDICO

AUTOR: PAULO CÉSAR RUBIO SALAZAR

TUTOR: PhD. FABÍAN RODRIGO NARVÁEZ ESPINOZA

Quito - Ecuador

2025

CERTIFICADO DE RESPONSABILIDAD Y AUTORÍA DEL TRABAJO DE TITULACIÓN

Yo, Paulo César Rubio Salazar con documento de identificación N° 1722509336 manifiesto que:

Soy el autor y responsable del presente trabajo; y, autorizo a que sin fines de lucro la Universidad Politécnica Salesiana pueda usar, difundir, reproducir o publicar de manera total o parcial el presente trabajo de titulación.

Quito, 15 de octubre del año 2025

Atentamente,

Paulo César Rubio Salazar
1722509336

CERTIFICADO DE CESIÓN DE DERECHOS DE AUTOR DEL TRABAJO DE TITULACIÓN A LA UNIVERSIDAD POLITÉCNICA SALESIANA

Yo, Paulo César Rubio Salazar con documento de identificación No. 1722509336 expreso mi voluntad y por medio del presente documento cedo a la Universidad Politécnica Salesiana la titularidad sobre los derechos patrimoniales en virtud de que soy autor del Proyecto técnico: EVALUACIÓN DE TÉCNICAS DE NAVEGACIÓN PARA COLONOSCOPIA VIRTUAL el cual ha sido desarrollado para optar por el título de: INGENIERO BIOMÉDICO, en la Universidad Politécnica Salesiana, quedando la Universidad facultada para ejercer plenamente los derechos cedidos anteriormente.

En concordancia con lo manifestado, suscribo este documento en el momento que hago la entrega del trabajo final en formato digital a la Biblioteca de la Universidad Politécnica Salesiana.

Quito, 15 de octubre del año 2025

Atentamente,

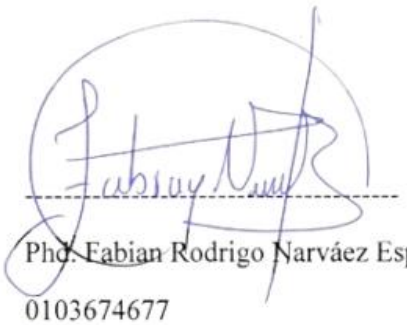
Paulo César Rubio Salazar
1722509336

CERTIFICADO DE DIRECCIÓN DEL TRABAJO DE TITULACIÓN

Yo, Fabián Rodrigo Narváez Espinoza con documento de identificación N° 0103674677, docente de la Universidad Politécnica Salesiana declaro que bajo mi tutoría fue desarrollado el trabajo de titulación: EVALUACIÓN DE TÉCNICAS DE NAVEGACIÓN PARA COLONOSCOPIA VIRTUAL, realizado por Paulo César Rubio Salazar con documento de identificación N° 1722509336, obteniendo como resultado final el trabajo de titulación bajo la opción Proyecto Técnico que cumple con todos los requisitos determinados por la Universidad Politécnica Salesiana.

Quito, 15 de octubre del año 2025

Atentamente,



Phd. Fabian Rodrigo Narváez Espinoza
0103674677

Dedicatoria

A mis padres, quienes con su amor inagotable y su ejemplo de trabajo y dedicación me han enseñado el valor del esfuerzo y la perseverancia. Gracias por ser mi mayor apoyo, por sus palabras de aliento en los momentos difíciles y por celebrar conmigo cada logro alcanzado. Este triunfo es tan suyo como mío. A mi familia, que con su cercanía y cariño ha sido un refugio en los días de incertidumbre. Su confianza en mí me ha dado la fuerza para continuar adelante, incluso cuando el camino parecía complicado. A mí mismo, por no rendirme, por enfrentar cada obstáculo con determinación y por mantenerme firme en la búsqueda de mis sueños. Este logro es el reflejo de noches de estudio, de días de esfuerzo y de la pasión por aprender y crecer. Con gratitud y alegría, cierro este capítulo con la certeza de que es solo el inicio de nuevas oportunidades y desafíos.

Agradecimientos

A la Universidad Politécnica Salesiana, mi sincero agradecimiento por brindarme el espacio y las herramientas necesarias para desarrollar esta investigación. En especial, extiendo mi reconocimiento al Ph.D. Fabián Rodrigo Narváez Espinoza, cuya guía y compromiso fueron fundamentales para la culminación de este trabajo. Su paciencia, conocimientos y orientación han sido un pilar esencial en este proceso.

Queridos padres, este logro es tanto mío como suyo. Su amor incondicional, su sacrificio y su apoyo inquebrantable han sido la base sobre la que he construido mis sueños. Gracias por enseñarme que el esfuerzo y la perseverancia siempre dan frutos, por levantarme en los momentos de caída y por celebrar conmigo cada pequeña victoria.

Cada consejo, cada regaño y cada enseñanza han sido semillas que hoy dan fruto en este importante paso de mi vida. No hay palabras suficientes para expresar la gratitud que siento por todo lo que han hecho por mí. Este logro es también suyo, porque sin su apoyo y amor, nada de esto habría sido posible.

Con todo mi cariño y admiración, les dedico este esfuerzo, con la certeza de que es solo el comienzo de nuevos desafíos y metas por alcanzar.

Con todo mi amor y gratitud,

Resumen

La colonoscopia virtual (CTC) es una técnica no invasiva que permite analizar el interior del colon a partir de imágenes de tomografía computarizada, reconstruir su estructura tridimensional y realizar una navegación virtual a través del modelo 3D generado, lo que posibilita una exploración interna sin procedimientos invasivos, apoyando la evaluación de la estructura del lumen del colón y el desarrollo de estrategias de navegación endoluminal. Este trabajo tiene como propósito comparar dos enfoques aplicados en la navegación dentro de entornos tridimensionales del colon: la navegación interactiva guiada por cobertura (ColNav) y la navegación automática basada en campos vectoriales libres de singularidades (SF-GVF). A partir de un estudio clínico extraído del repositorio TCIA (ACRIN 6664), se reconstruyó un modelo tridimensional del colon siguiendo un flujo metodológico que comprendió la segmentación del lumen mediante umbralización y operaciones morfológicas en 3D Slicer, la reconstrucción superficial y la extracción de la línea central con VMTK, la cual fue suavizada mediante interpolación B-spline para obtener una trayectoria continua de la anatomía interna del colón

Sobre este modelo anatómico se implementaron y evaluaron dos técnicas de navegación. En el caso de ColNav, se desarrolló un entorno interactivo con un mapa desplegado en dos dimensiones que facilito la visualización angular del lumen y el mapeo del movimiento según las regiones no observadas, lo que permitío un control de la cobertura del recorrido. Por otra parte, el método SF-GVF generó un campo vectorial continuo, libre de singularidades, que orientó el desplazamiento a través del lumen, manteniendo la dirección de avance de forma automática. Ambos enfoques fueron sometidos al mismo conjunto de condiciones experimentales con el fin de garantizar la comparabilidad de los resultados, utilizando una geometría idéntica, parámetros de simulación y un sistema de visualización.

La evaluación cuantitativa se realizó mediante métricas empleadas en estudios reportados en la literatura reciente sobre navegación virtual, lo que permitió comparar el desempeño de las técnicas implementadas, tales como la cobertura global, la desviación respecto a la línea central, la redundancia y el número total de pasos. Los resultados experimentales mostraron diferencias en el patrón de exploración, con ColNav orientado hacia decisiones locales dependientes de la visibilidad y SF-GVF priorizando trayectorias continuas con menor repetición del recorrido. Este estudio propone un protocolo reproducible de reconstrucción, implementación y medición que puede servir como referencia para la validación de nuevas estrategias de navegación en colonoscopia virtual y su extensión hacia otros algoritmos o conjuntos de datos en el análisis tridimensional del colon.

Palabras clave: colonoscopia virtual, navegación computarizada, tomografía computarizada, reconstrucción tridimensional, cobertura, ColNav, SF-GVF.

Abstract

Virtual colonoscopy (CTC) is a non-invasive technique that enables the analysis of the colon's interior using computed tomography images, allowing for the reconstruction of its three-dimensional structure and virtual navigation through the generated 3D model. This approach facilitates internal exploration without invasive procedures, supporting the evaluation of the colonic lumen structure and the development of endoluminal navigation strategies. The purpose of this work is to compare two approaches applied to navigation within three-dimensional colon environments: interactive coverage-guided navigation (ColNav) and automatic navigation based on singularity-free vector fields (SF-GVF). Using a clinical study extracted from the TCIA repository (ACRIN 6664), a 3D model of the colon was reconstructed following a methodological workflow that included lumen segmentation through thresholding and morphological operations in 3D Slicer, surface reconstruction, and centerline extraction with VMTK, which was smoothed using B-spline interpolation to obtain a continuous trajectory of the internal anatomy.

On this anatomical model, two navigation techniques were implemented and evaluated. In the case of ColNav, an interactive environment was developed with a two-dimensional unfolded map that facilitated angular visualization of the lumen and motion mapping according to unobserved regions, enabling control over the coverage of the traversal. In contrast, the SF-GVF method generated a continuous, singularity-free vector field that guided movement through the lumen while automatically maintaining the direction of advancement. Both approaches were tested under the same experimental conditions to ensure comparability of results, using identical geometry, simulation parameters, and visualization systems.

Quantitative evaluation was carried out using metrics reported in recent literature on virtual navigation, allowing the performance of the implemented techniques to be compared in terms of global coverage, deviation from the centerline, redundancy, and total number of steps. The experimental results revealed differences in exploration patterns, with ColNav oriented toward local visibility-dependent decisions, and SF-GVF favoring continuous trajectories with fewer repetitions. This study proposes a reproducible protocol for reconstruction, implementation, and measurement that may serve as a reference for validating new navigation strategies in virtual colonoscopy and for extending these approaches to other algorithms or datasets in three-dimensional colon analysis.

Keywords: virtual colonoscopy, computer-based navigation, computed tomography, three-dimensional reconstruction, coverage, ColNav, SF-GVF.

Contenido

Lista de símbolos

Abreviaturas

Abreviatura	Término
CCR	Cáncer colorrectal
CTC	Colonografía por tomografía computarizada
$DICOM$	Digital Imaging and Communications in Medicine
$VMTK$	Vascular Modeling Toolkit (Kit de Modelado Vascular)
CAD	Computer-Aided Detection (Detección asistida por computadora)
FPS	Frames Per Second (Cuadros por segundo)
$BiSPFPN$	Bidirectional Scale-Perceiving Feature Pyramid Network
IoU	Intersection over Union (Intersección sobre unión)
mAP	Mean Average Precision (Precisión promedio)
$PRISMA$	Ítems preferidos para revisiones sistemáticas
VTK	Visualization Toolkit (Biblioteca para visualización 3D)
$B\text{-}Splines$	Basis Splines (Interpolación por Splines)
CT	Computed Tomography (Tomografía Computarizada)
CNN	Convolutional Neural Network (Red Neuronal Convolucional)
ML	Machine Learning (Aprendizaje Automático)
$TCIA$	The Cancer Imaging Archive (Repositorio de Imágenes Médicas)
ROI	Region of Interest (Región de Interés)

1 Introducción

El cáncer de colón está considerado una de las principales causas de muerte a nivel mundial [1] y la colonoscopia óptica convencional sigue siendo el método más usado para su diagnóstico. Su detección temprana aumenta la expectativa de vida de los pacientes [1]. Sin embargo, este procedimiento es invasivo y puede presentar algunas complicaciones durante el proceso, como la perforación del colón, sangrado posterior a la extracción de alguna muestra de tejido maligno, entre otros. Para reducir estos riesgos y complicaciones, la colonoscopia virtual basada en tomografía computarizada (TC) se ha vuelto una alternativa menos invasiva, LA cual se basa en la reconstrucción de modelos tridimensionales del colon para su exploración interna [2]. Si bien, en sus inicios, la calidad de las imágenes y los algoritmos limitó su aplicación, el desarrollo de técnicas computacionales en los últimos años ha mejorado la exploración virtual y la precisión en el diagnóstico clínico [3]. En la actualidad, la colonografía por tomografía computarizada (CTC) permite visualizar la anatomía interna del colón de manera no invasiva, logrando así una alta sensibilidad para la detección de lesiones y reduciendo las complicaciones asociadas a la colonoscopia convencional [4]. Sin embargo, uno de los principales retos de la colonografía es controlar la rotación de la cámara durante la navegación dentro del modelo tridimensional. Este manejo es de gran importancia para lograr una visualización completa de la superficie interna del colón, ya que de esto depende cubrir la mayor cantidad posible del lumen.

A pesar de existir una gran variedad de algoritmos de navegación computacionales, actualmente no existe un consenso ni evidencia suficiente sobre cuáles son las técnicas más adecuadas para cada aplicación clínica o simulada, debido principalmente a la falta de criterios comparativos y métricas estandarizadas que permitan una imparcial entre los diferentes métodos. Ante esta situación, el presente trabajo compara de manera cuantitativa dos técnicas actuales y relevantes reportadas en la literatura: ColNav, un sistema de navegación en tiempo real que utiliza un desplegado en dos dimensiones del colón y un indicador tipo brújula para guiar la exploración [5], y el Singularity-Free Guiding Vector Field (SFGVF), un algoritmo de campo vectorial guía diseñado para eliminar los puntos singulares y garantizar una navegación global a lo largo de trayectorias complejas [6]. Ambas técnicas fueron implementadas y evaluadas usando modelos tridimensionales reconstruidos del colón obtenidos de bases de datos públicas. El desempeño de cada técnica se analizó empleando métricas cuantitativas como cobertura global, desviación respecto al eje central, redundancia en el recorrido y pasos totales [7].

Importancia y alcance

El desarrollo de técnicas de navegación computacional en la colonoscopia virtual ha permitido ampliar las opciones de diagnóstico para el cáncer colorrectal o cualquier otra patología que afecte la mucosa del colón, proporcionando así una alternativa no invasiva basada en la reconstrucción tridimensional del colón a partir de imágenes de tomografía computarizada [1]. Sin embargo, la calidad real de la CTC en la práctica clínica está muy relacionada con la capacidad de explorar el lumen del colón de forma completa. En este proceso, las técnicas de navegación computacional son de gran importancia, ya que ayudan a la exploración completa del modelo tridimensional del colón, incluyendo zonas de difícil acceso o no vistas mediante algoritmos específicos de trayectoria y orientación [8]. Actualmente, la mayoría de los estudios se centran en analizar cada método por separado o bajo condiciones experimentales diferentes, lo que dificulta establecer una comparación cuantitativa de su desempeño y aplicación clínica [5]. Por lo tanto, la evaluación cuantitativa de las técnicas de navegación se apoya en el uso de métricas reconocidas en la literatura, lo que permite establecer un enfoque que permite controlar mejor los experimentos, repetir las pruebas y comparar directamente los métodos. [7]. Este estudio se plantea en un entorno simulado mediante modelos 3D generados a partir de imágenes de CTC de acceso público. Esta aproximación facilita el control experimental, la repetibilidad y la comparación directa entre métodos, aunque también evidencia la necesidad de validar estos resultados en aplicaciones clínicas reales [5].

1.1. Objetivos

Para el desarrollo de este trabajo se dio cumplimiento a los siguientes objetivos:

1.1.1. Objetivo general

Evaluar técnicas de navegación más usadas en la colonoscopia virtual para apoyar la detección temprana de pólipos.

1.1.2. Objetivos específicos

- Revisar y seleccionar el estado del arte las mejores técnicas de navegación para colonoscopia virtual.
- Implementar las técnicas de navegación definidas en el estado del arte para la colonoscopia virtual.
- Definir e implementar las métricas de desempeño usadas para evaluar la efectividad y precisión de las técnicas de navegación en la colonoscopia virtual.

- Evaluar y comparar el desempeño de las técnicas de navegación en la colonoscopia virtual mediante métricas para este fin.

2 Marco teórico

2.1. Anatomía general del colón

El colon, también conocido como intestino grueso, es parte del sistema digestivo humano. Su estructura facilita la absorción de agua y la formación de heces compactas [9]. En los humanos, el colón tiene una longitud promedio de aproximadamente 1.5 metros y se divide en las siguientes regiones, donde cada una cumple una función:

- **Ciego:**

  Es la parte inicial del colón y se ubica en la parte inferior derecha del abdomen. El ciego funciona como el lugar de recepción del contenido que viene del intestino delgado a través de la válvula ileocecal, controlando así el paso del quimo y evitando el reflujo. Junto al ciego está el apéndice, una estructura tubular cuya función específica en los humanos aún se discute, aunque se le han atribuido funciones inmunológicas y en la microbiota intestinal [10].

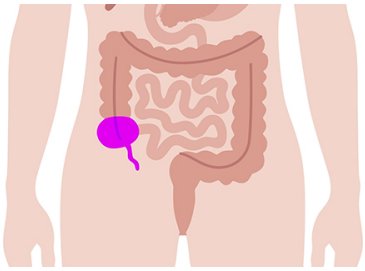


Figura 2-1: Ilustración del ciego en la parte izquierda, representado en color morado. [11]

- **Colón ascendente:**

  Se extiende hacia arriba desde el ciego hasta la flexura hepática, que está cerca del hígado. Esta área es importante para volver a absorber agua y sales, ayudando a compactar el contenido intestinal. El colón ascendente usa movimientos peristálticos para mover el material en contra de la fuerza de gravedad, un proceso ayudado por la coordinación del sistema nervioso entérico y las células musculares lisas [12].

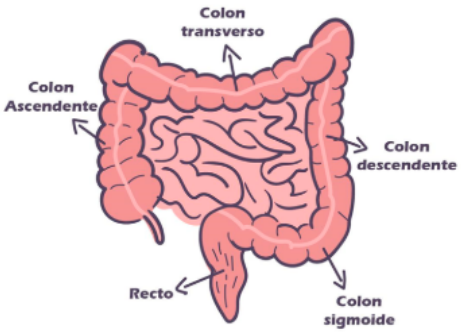


Figura 2-2: Ilustración del colón ascendente en la parte izquierda. [13]

- **Colón transverso:**

  Este segmento atraviesa la cavidad abdominal de lado a lado, desde la flexura hepática hasta la flexura esplénica, cerca del bazo. Es la parte más flexible del colón, sostenida por el mesocolon transverso. En esta sección se lleva a cabo una gran absorción de agua y nutrientes que quedan. También actúa como un espacio temporal para almacenar el contenido fecal mientras se solidifica [14].

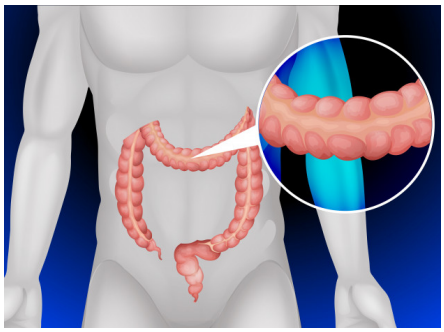


Figura 2-3: Ilustración del colón transverso. [15]

- **Colón descendente:**

  Este segmento desciende a lo largo del lado izquierdo del abdomen desde el ángulo esplénico hasta el colon sigmoide. Se caracteriza por una menor movilidad en comparación con el colón transverso y elimina el exceso de agua del contenido intestinal, reafirmando las heces antes de su almacenamiento final [16].

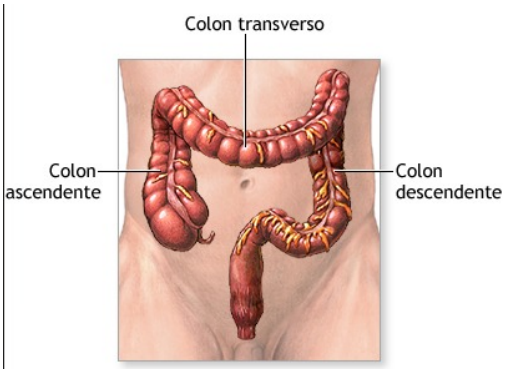


Figura 2-4: Ilustración del colón descendente. [17]

- **Colón sigmoideo:**

  Con forma de "S", une el colón descendente con el recto. Esta área tiene dos funciones: guardar temporalmente las heces y controlar su movimiento hacia el recto. Su forma curvada es esencial para manejar el flujo intestinal, funcionando como un depósito que ayuda a la evacuación en los momentos adecuados.[18].

Figura 2-5: Se puede observar en la inferior el colón sigmoideo. [19]

- **Recto:**

  Es la última parte del colon, que se une al canal anal. El recto funciona como un lugar donde se guardan las heces hasta que se expulsan. Este proceso es controlado por el esfínter anal interno, que es involuntario, y el esfínter anal externo, que es voluntario, lo que permite un control consciente sobre la eliminación de los residuos [20].

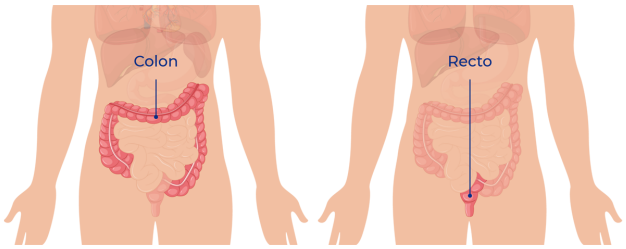


Figura 2-6: Ilustración de la parte inferior derecha del recto. [21]

2.1.1. Colonoscopia óptica o convencional

La colonoscopia óptica, también denominada colonoscopia convencional, es un procedimiento médico invasivo que ha sido considerado durante décadas como el principal método para el diagnóstico, tratamiento y seguimiento de enfermedades que afectan al colón y recto [22]. Este método consiste en la introducción de un colonoscopio, un tubo flexible equipado con una cámara y una fuente de luz en su extremo, a través del ano, permitiendo así la visualización directa y en tiempo real de la mucosa del intestino grueso. Las imágenes capturadas durante la navegación son proyectadas en un monitor, lo que facilita la identificación de anomalías como pólipos, tumores, lesiones inflamatorias, sangrados o cualquier alteración estructural de la pared del colon [23]. La colonoscopia no solo es una herramienta diagnóstica, sino que también es un importante método terapéutico, ya que durante el mismo procedimiento es posible realizar intervenciones como la extracción, toma de muestras o resección de lesiones sospechosas, contribuyendo a la prevención y manejo del cáncer colorrectal [24, 25].

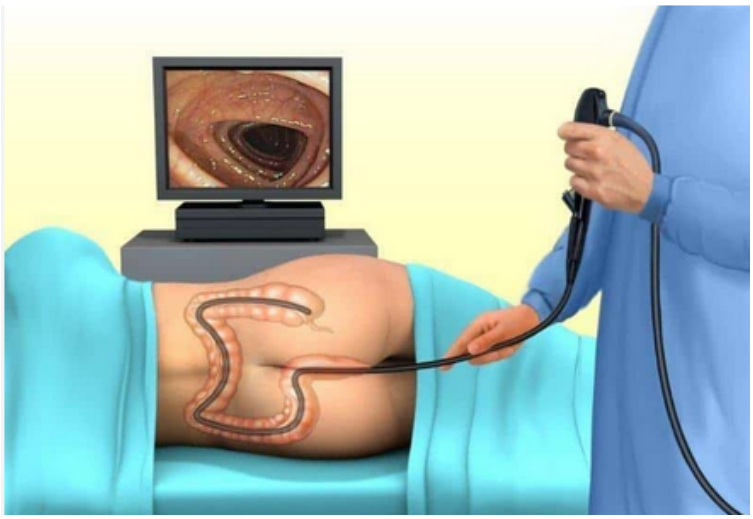


Figura 2-7: Ilustración del procedimiento de colonoscopia óptica. Se muestra la inserción del endoscopio flexible a través del recto, avanzando por el colón mientras transmite imágenes en tiempo real a un monitor.

La capacidad diagnóstica de la colonoscopia convencional en la detección de patologías del colón está bien documentada, con tasas elevadas de sensibilidad y especificidad para la identificación de lesiones precancerosas o cancerosas. Sin embargo, el éxito de este procedimiento depende en gran medida de la experiencia del médico endoscopista, la calidad de la preparación intestinal y las características anatómicas del paciente. La visualización directa de la mucosa permite identificar lesiones de pequeño tamaño, lesiones planas o con localización atípica, aunque la detección puede verse limitada en presencia de pliegues profundos, segmentos colapsados o una limpieza intestinal insuficiente [24, 26]. Además, la exploración del colón puede ser complicada en pacientes que presenten variantes anatómicas, estenosis o alteraciones inflamatorias, situaciones en donde se requiere una mayor experiencia médica y, en ocasiones, apoyo de métodos complementarios como la colonografía por tomografía computarizada [27].

Para realizar una correcta colonoscopia óptica, es importante realizar una preparación intestinal adecuada, la cual consiste en la administración de laxantes y una dieta especial, con el fin de eliminar los restos y residuos fecales que puedan interferir en la visualización de la mucosa del colón [26]. Esta preparación, si bien es importante para el éxito del procedimiento, representa una de las principales molestias para los pacientes, quienes deben ayunar y cumplir con un régimen estricto el día antes del examen. El procedimiento generalmente se realiza bajo sedación o anestesia ligera, administrada por un médico anestesiólogo, con el fin de garantizar la tolerancia y minimizar las molestias o el dolor durante la inserción y manipulación del colonoscopio [28]. En condiciones normales, la duración de la colonoscopia está entre los 30 y los 45 minutos, aunque este tiempo puede variar en función de la anatomía del colón, la presencia de hallazgos clínicos o la necesidad de intervenciones adicionales.

A pesar de sus ventajas, la colonoscopia óptica no está libre de riesgos y limitaciones correspondientes al carácter invasivo del procedimiento. Entre las complicaciones más frecuentes se encuentran el dolor abdominal, la distensión, el sangrado y, en casos menos comunes, la perforación de la pared intestinal, sobre todo cuando se realizan polipectomías o biopsias extensas [26, 29]. Aunque la incidencia de complicaciones graves es baja, estas situaciones representan un desafío clínico relevante, ya que requieren resolverse en el mismo procedimiento endoscópico y, en ocasiones excepcionales, pueden hacer necesaria una intervención quirúrgica o un tiempo de hospitalización mayor [29]. Existen también riesgos asociados a la sedación, incluyendo alteraciones en la frecuencia cardíaca, la presión arterial y la saturación de oxígeno en la sangre, lo que puede suponer un riesgo elevado para pacientes que ya tienen otras enfermedades o problemas de salud [30]. Adicionalmente, un buen diagnóstico en la colonoscopia óptica se puede ver afectado por la calidad de la preparación intestinal y la experiencia del médico tratante, siendo fundamental la formación continua y la aplicación de protocolos estandarizados para optimizar los resultados y minimizar los riesgos [24].
Este procedimiento clínico es de gran importancia en la prevención y el diagnóstico temprano del cáncer colorrectal, el cual constituye una de las principales causas de mortalidad por cáncer a nivel mundial [1]. La detección temprana y la extirpación de pólipos posiblemente malignos mediante colonoscopia se han asociado con una reducción significativa de la incidencia y mortalidad por cáncer de colón [31, 32]. Durante el procedimiento, el médico puede identificar y extirpar pólipos o tomar muestras para biopsia, permitiendo no solo el diagnóstico, sino también la intervención terapéutica. Sin embargo, a pesar de su alta sensibilidad y de ser la técnica más usada, existen situaciones en las que la colonoscopia óptica no se puede completar, ya sea por obstrucciones, mala tolerancia o condiciones médicas del paciente. En estos casos, la colonografía por TC o colonoscopia virtual surge como una alternativa diagnóstica de gran importancia [27].

2.1.2. Colonoscopia virtual

La colonografía por tomografía computarizada (CTC), bien conocida como colonoscopia virtual, representa una innovación en métodos de diagnóstico por imágenes del colón, al utilizar tecnologías de adquisición y procesamiento de datos que permiten obtener una visión tridimensional y detallada de la anatomía interna del colón [3]. Este procedimiento se basa en la obtención de imágenes de diferentes cortes anatómicos (cortes axiales, coronales y sagitales), mediante tomografía computarizada, que posteriormente son procesadas para reconstruir modelos 3D de colón, lo cual ayuda a la inspección no invasiva del lumen intestinal sin la necesidad de insertar un endoscopio en el paciente. Este procedimiento consiste en la preparación intestinal del paciente, similar a la empleada en la colonoscopia óptica, con el objetivo de asegurar la eliminación de residuos y optimizar la calidad de las imágenes. Durante el procedimiento, se introduce aire o gas, principalmente dióxido de carbono, por vía rectal para expandir el colón, facilitando así la adquisición de imágenes en diferentes posiciones,

generalmente en decúbito prono y supino, lo que contribuye a mejorar la visualización de las paredes del colón y a diferenciar verdaderas lesiones de artefactos o residuos fecales [27, 33].

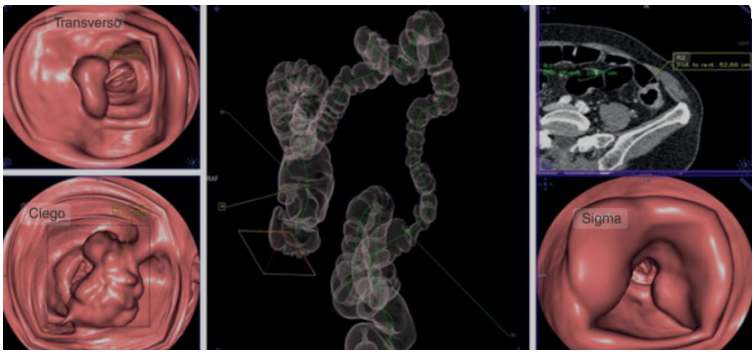


Figura 2-8: Representación visual del proceso de colonoscopia virtual. Se observan imágenes endoluminales correspondientes a distintas regiones del colón (ciego, sigma y colon transverso), cortes axiales obtenidos mediante tomografía computarizada y un modelo 3D reconstruido del lumen.

La adquisición de imágenes de CT se realiza en pocos minutos y no requiere sedación, permitiendo que el paciente retome sus actividades cotidianas de forma casi inmediata [34]. Además, el análisis de las imágenes puede repetirse las veces que sea necesario, ya que se almacenan en formato digital, generalmente DICOM, y pueden ser revisadas por diferentes especialistas. Este método resulta especialmente útil en casos donde la colonoscopia convencional está contraindicada o no pudo completarse por razones anatómicas o clínicas. La capacidad de generar modelos tridimensionales detallados permite la identificación de pólipos, masas y otras alteraciones de la estructura de la mucosa que podrían pasar desapercibidas con la colonoscopia convencional [35].
Sin embargo, la colonoscopia virtual tiene ciertas limitaciones técnicas que frenan su aplicación clínica, como por ejemplo su capacidad para detectar lesiones pequeñas, especialmente pólipos menores de 5 mm, que es inferior en comparación con la colonoscopia óptica, lo que puede disminuir la sensibilidad para lesiones tempranas. Además, no permite realizar intervenciones terapéuticas como la toma de biopsias o la extracción de pólipos durante el mismo estudio, por lo que si existen hallazgos sospechosos, resulta necesario complementar el diagnóstico con una colonoscopia óptica convencional [27, 34]. La calidad diagnóstica está relacionada con la calidad de la preparación intestinal y la resolución de las imágenes de tomografía, así como las técnicas de navegación empleadas durante el análisis de los datos. Por otro lado, la presencia variable de estructuras anatómicas, pliegues complejos o segmentos colapsados puede dificultar la reconstrucción completa del colón y así limitar la navegación diagnóstica completa [36–38].

En el ámbito del diagnóstico y la prevención del cáncer colorrectal, la colonografía ha demostrado ser una herramienta complementaria de gran utilidad [1], debido a que permite

analizar el lumen del colón desde diferentes ángulos y con herramientas de navegación avanzadas. Sin embargo, es importante señalar que, aunque la colonoscopia virtual representa un avance en accesibilidad y confort para el paciente, no sustituye la función terapéutica de la colonoscopia óptica y, por tanto, su uso debe ser considerado dentro de un enfoque completo de diagnóstico y manejo del paciente. Es así que el desarrollo de la colonoscopia virtual se ha visto mejorado por la integración de técnicas de procesamiento de imágenes, inteligencia artificial y nuevos algoritmos de navegación, que buscan mejorar la exploración del colón y la detección de lesiones importantes [8, 39].

Por otro lado, esta técnica confronta desafíos computacionales que incluyen el manejo de grandes volúmenes de datos, la segmentación precisa del lumen y la adaptación a diferentes estructuras del colón, circunstancias que requieren que las técnicas de navegación computacional se vayan perfeccionando. En este aspecto, la relación entre la anatomía compleja del colón y las soluciones computacionales desarrolladas constituye un campo de investigación activo, con impacto directo en la calidad y capacidad de la colonoscopia virtual como herramienta diagnóstica.

2.1.3. Técnicas de navegación guiada en colonoscopia virtual

El avance de la colonoscopia virtual ha desarrollado nuevas posibilidades en el diagnóstico de enfermedades del colón al permitir la exploración interna del órgano sin recurrir a procedimientos invasivos. Para lograr que esta exploración sea de calidad y comparable a la realizada en una colonoscopia óptica, es importante emplear técnicas de navegación computarizada para guiar el recorrido de la cámara virtual a lo largo de modelos tridimensionales generados a partir de imágenes de tomografía computarizada. Estas técnicas han ido evolucionando debido a los desafíos que presenta la anatomía del colón, como su longitud, los numerosos pliegues, las flexuras y los diversos tipos de pacientes que existen [8].
En las primeras apariciones, la navegación virtual se basaba en métodos geométricos que utilizaban la línea central del lumen del colón como guía para el movimiento de la cámara virtual. Este enfoque facilitaba la inspección continua y ordenada del interior del colón, permitiendo visualizar amplias superficies de la mucosa. Sin embargo, con el tiempo se observaron algunas limitaciones, sobre todo en la capacidad de cubrir regiones ocultas o en aquellas donde la estructura del colón dificultaba el seguimiento de una ruta central ideal [5, 40]. Para superar estos inconvenientes, la investigación avanzó hacia el diseño de algoritmos más avanzados, como los basados en campos vectoriales, que permiten orientar el movimiento dentro del lumen según diferentes criterios de cobertura y exploración. Estos algoritmos son capaces de adaptar la dirección de la cámara virtual a los cambios anatómicos y responder en tiempo real a la geometría irregular de cada colón, mejorando la cobertura de áreas complejas o de difícil acceso como son sus curvas y pliegues cerrados [6, 41].
En los últimos años, el desarrollo de técnicas de navegación asistidas por inteligencia artificial

ha dado un salto importante en este campo [5]. El aprendizaje automático y el análisis de grandes volúmenes de datos han hecho posible que los sistemas de navegación virtual puedan adaptarse a múltiples escenarios, predecir rutas óptimas y ajustar sus trayectorias ante cualquier problema, como cambios en la anatomía o la aparición de segmentos parcialmente incompletos [42, 43]. A esto se suman las técnicas de visualización avanzada, que facilitan el despliegue del colón en vistas planas o la simulación de recorridos virtuales que ayudan tanto en el análisis diagnóstico como en la formación de personal médico especializado [5, 44].
El objetivo principal de todas estas técnicas es lograr una exploración de calidad de la mucosa del colón, de modo que se reduzcan las regiones no visualizadas y aumente la posibilidad de detectar lesiones en la pared de la mucosa. La selección de la técnica de navegación más adecuada depende no solo de las características del modelo anatómico, sino también de las necesidades clínicas, los recursos computacionales disponibles y el contexto de uso, ya sea para diagnóstico, investigación o formación.
Con el fin de identificar y analizar las técnicas computacionales más importantes, una búsqueda sistemática de literatura ha sido realizada, para lo cual se consideraron bases de datos especializadas como IEEE Xplore, ScienceDirect, PubMed y SpringerLink. La búsqueda fue realizada mediante palabras clave como: virtual colonoscopy, colon navigation, endoluminal navigation, computed tomography colonography y combinaciones asociadas.
Para la selección de los trabajos incluidos en esta revisión, se consideraron únicamente artículos científicos publicados entre 2019 y 2024 que presentaran un enfoque explícito en la navegación virtual o la planificación de trayectorias. Se priorizaron aquellos estudios que describen o aplican técnicas de navegación computacional, ya sean geométricas, basadas en inteligencia artificial, volumétricas o vectoriales, y que además resulten aplicables de manera directa o adaptable a la colonoscopia virtual o a procedimientos endoscópicos en estructuras anatómicas tubulares similares, como vías digestivas, vasculares o respiratorias. Asimismo, se analizaron aplicaciones con métricas cuantificables para la evaluación del desempeño, tales como precisión, eficiencia computacional, tiempo de cómputo, errores angulares, cobertura del lumen, la desviación respecto a la línea central, la redundancia en el trayecto y el número total de pasos. Otro criterio importante fue la disponibilidad de información técnica suficiente, de modo que el artículo detallara el algoritmo implementado o la técnica utilizada, idealmente incluyendo ecuaciones, pseudocódigo o visualizaciones relevantes.
De la misma forma, se excluyeron de la revisión todos aquellos estudios publicados antes de 2019, así como trabajos cuyo enfoque estuviese exclusivamente en la detección de pólipos, clasificación o diagnóstico, sin abordar técnicas de navegación. También se descartaron revisiones narrativas que no presentaran propuestas concretas ni un análisis estructurado de técnicas de navegación, así como artículos que no incluyeran validación cuantitativa o experimental, es decir, que no tuvieran métricas de desempeño o pruebas que se pudieran reproducir. Finalmente, se excluyeron los estudios realizados en regiones anatómicas cuya morfología, topología o dinámica no sea extrapolable al colón, como ocurre en investigaciones centradas en estructuras cerebrales o cavidades abiertas [45].

Entre los trabajos más relevantes identificados se encuentra el de Frank et al. (2023), quienes desarrollaron **ColNav**, un sistema de navegación en tiempo real que combina localización simultánea y cartografía (SLAM) con un módulo de unfolding para desplegar zonas no visualizadas del colón [5]. Por otro lado, Zhao et al. (2023) propusieron una técnica basada en campos vectoriales sin singularidades (SF-GVF), orientada a entornos ramificados como el tracto intestinal, garantizando suavidad y direccionalidad continua [46].
Otro estudio significativo es el de Corsi et al. (2023), quienes implementaron una arquitectura de aprendizaje por refuerzo profundo con control visuomotor, entrenada en entornos sintéticos de colón, lo que permite prescindir del conocimiento geométrico previo y adaptarse solo mediante imagen [42]. Finalmente, Cabezón et al. (2021) utilizaron interpolación con B-splines sobre la línea central segmentada del colón para generar trayectorias suaves y continuar la exploración en trayectos de visualización limitados [3].
A continuación, se presenta un resumen organizado de las 21 técnicas computacionales más relevantes identificadas durante la revisión del estado del arte, que cumplen con los criterios definidos y han sido analizadas en función de sus características técnicas, métricas y datos utilizados.

Tabla 2-1: Resumen de las 21 técnicas computacionales de navegación virtual encontradas en la revisión sistemática del estado del arte

#	Título	Año	Técnica de Navegación	Métricas Cuantitativas	Dataset
1	ColNav: Real-Time Colon Navigation for Colonoscopy	2023	SLAM + unfolding dinámico	Recall: 88.9 %, Cobertura: 96.4 %, Kappa: 88.4 %	Colon impreso, clips reales
2	Singularity-Free Guiding Vector Field	2023	Campo vectorial sin singularidades	Error lateral: 0.19 mm, Suavidad: 0.97, Éxito: 100 %	Colon sintético 3D
3	Deep RL for Intraluminal Navigation	2022	Deep RL basado en imagen	Precisión, retorno medio	Simulador Unity3D
4	Template-Based 3D Reconstruction	2022	Interpolación sobre centroide	Curvatura media, cobertura	Segmentado CT
5	End-to-End Deep Visuomotor Control	2022	Control visuomotor directo	Error angular, eficiencia	Colon sintético/real
6	VNet-based Autonomous System	2021	IA segmentación + navegación	Dice, cobertura	TC público (NIH)
7	Attention-Guided Visual SLAM	2022	SLAM con atención visual	Desviación al eje, éxito	TC y endoscopía sintética
8	Anatomical-Aware Navigation	2021	Modelos anatómicos locales	Reproducibilidad, cobertura	Datos sintéticos
9	Exploration-Aware RL	2023	Deep RL + mapas	Área cubierta, pasos	Colon CAD
10	Learning-Based Path Planning	2021	Planificación guiada por IA	Tiempo computacional, coste	Modelos simulados
11	Adaptive Centerline-Following	2020	Geometría adaptativa	Curvatura, distancia eje	NIH
12	Smart Field Navigation	2019	Campos vectoriales adaptativos	Éxito, redundancia	Colon sintético
13	B-spline Guided Path	2020	Interpolación con B-splines	Longitud, curvatura, cobertura	TC clínico
14	Endoluminal Navigation via Optimization	2019	Campos + optimización	Precisión angular, cobertura	CTIA phantom
15	Polyp-Guided Navigation	2021	SLAM guiado por sospechas	Cobertura, recall	Olympus dataset
16	Visual Odometry with Depth Prediction	2020	VO con profundidad aprendida	Error pose, error profundidad	Imagen sintética
17	Multi-View Path Planning	2021	Planeación multivista	Área visible, éxito	Phantom virtual
18	Segment-wise Navigation Framework	2023	División segmentaria + heurísticas	Tiempo por segmento, cobertura	Dataset propio
19	Spline-Fitted Optical Navigation	2022	Ajuste óptico splineado	Suavidad, éxito	CT segmentado manual
20	Navigability Estimation Learning	2023	Estimación basada en redes	Precisión, recall, tiempo	Base sintética 3D
21	Coverage-Aware Visual Navigation	2022	Mapas de cobertura visual	Cobertura, redundancia	Colon real con marcadores

Es así que esta revisión no solo permitió identificar una amplia variedad de algoritmos computacionales aplicados a la navegación en colonoscopia virtual, sino también seleccionar aquellas técnicas que presentan un desarrollo más robusto, validación cuantitativa y adap-

tabilidad clínica a cualquier modelo o paciente. La clasificación aquí presentada proporciona una visión estructurada de los últimos avances en navegación computacional aplicada a la colonoscopia virtual. Al reunir varios enfoques, los cuales son validados bajo métricas cuantitativas y comparables para su desarrollo, este análisis permite establecer una base sólida para futuras comparaciones experimentales y la selección fundamentada de técnicas en función de sus características técnicas y clínicas.

2.1.4. Métricas para la evaluación de técnicas de navegación

La comparación y validación de las distintas técnicas de navegación computarizada requieren el uso de métricas que se puedan verificar y reconocer en la literatura científica. Estas métricas permiten establecer criterios iguales para analizar el desempeño de los algoritmos bajo condiciones similares, facilitando así el avance en la investigación y la aplicación clínica. En la Tabla **2-2**, se describen las métricas definidas luego de un análisis durante la revisión del estado del arte, las mismas que están determinadas en base a su frecuencia e importancia de uso en la literatura revisada, su aplicabilidad en entornos tridimensionales simulados y su relevancia clínica y computacional.

Tabla 2-2: Descripción de métricas utilizadas para evaluar técnicas de navegación computacional.

Métrica	**Descripción**
Cobertura global	Porcentaje de la superficie del lumen colónico que ha sido visualizada por la cámara virtual durante la navegación. Un valor alto reduce el riesgo de no detectar lesiones o anomalías en regiones ocultas [5, 39].
Desviación al trayecto ideal	Distancia promedio entre la trayectoria seguida por el algoritmo y el centro geométrico del lumen. Indica qué tan bien se ajusta la navegación a la anatomía interna del colon [44].
Redundancia	Número de veces que una misma región del lumen es visualizada repetidamente durante la navegación. Puede señalar bucles innecesarios o trayectorias no eficientes [5].
Pasos totales o longitud del recorrido	Representa la cantidad total de desplazamientos realizados por el sistema de navegación. Sirve como indicador de eficiencia computacional y complejidad del algoritmo [42].

Estas métricas constituyen un sustento importante para comparar el rendimiento de diferentes métodos de navegación computarizada, para orientar mejoras en el futuro y establecer estándares para su aplicación en la colonoscopia virtual. La adquisición de criterios de evaluación cuantitativos que se puedan replicar permite avanzar hacia sistemas de navegación más avanzados y útiles, tanto en la investigación como en la práctica clínica [5].
Es así que los estudios revisados han puesto empeño en el uso de estas métricas para analizar el desempeño de técnicas de navegación virtual. Como por ejemplo, Frank et al. [5] eligieron la cobertura global y la redundancia para comparar distintos recorridos, explicando que la cobertura es importante para valorar el porcentaje de exploración de la superficie y la redundancia permite identificar zonas que han sido revisadas en exceso o no han sido cubiertas. Zhao et al. [46] emplearon la desviación lateral que mide la distancia respecto a la línea central, lo que resulta útil para evaluar el ajuste anatómico de la navegación. Por su parte, Corsi et al. [42] y Cabezón et al. [3] incluyeron métricas similares al analizar la generación de trayectorias mediante interpolación o métodos visuales, coincidiendo en que estos parámetros ofrecen una visión completa del comportamiento de la navegación.
La selección de métricas en el presente estudio se sustenta en el uso recurrente de estos parámetros en la literatura y en la función que cada uno cumple dentro del proceso de evaluación. Estas métricas permiten observar diferentes aspectos del desempeño de las técnicas: la cobertura informa sobre la cantidad de superficie visualizada, la curvatura aporta datos sobre la suavidad de la trayectoria, la desviación lateral facilita el análisis del alineamiento respecto al eje del lumen, la redundancia indica posibles recorridos innecesarios, y el número de pasos computacionales brinda una idea de la eficiencia del método. Optar por métricas que han sido validadas y discutidas en otros trabajos contribuye a que los resultados sean comparables y que la evaluación tenga tanto los requerimientos computacionales como los retos clínicos y geométricos que plantea la navegación virtual en el colón.

3 Implementacíon de técnicas de navegación para colonoscopia virtual

En este capítulo se describe la metodología propuesta para la implementación, simulación y comparación de técnicas de navegación computarizada en colonoscopia virtual. La Figura **3-1** ilustra el flujo de trabajo propuesto.

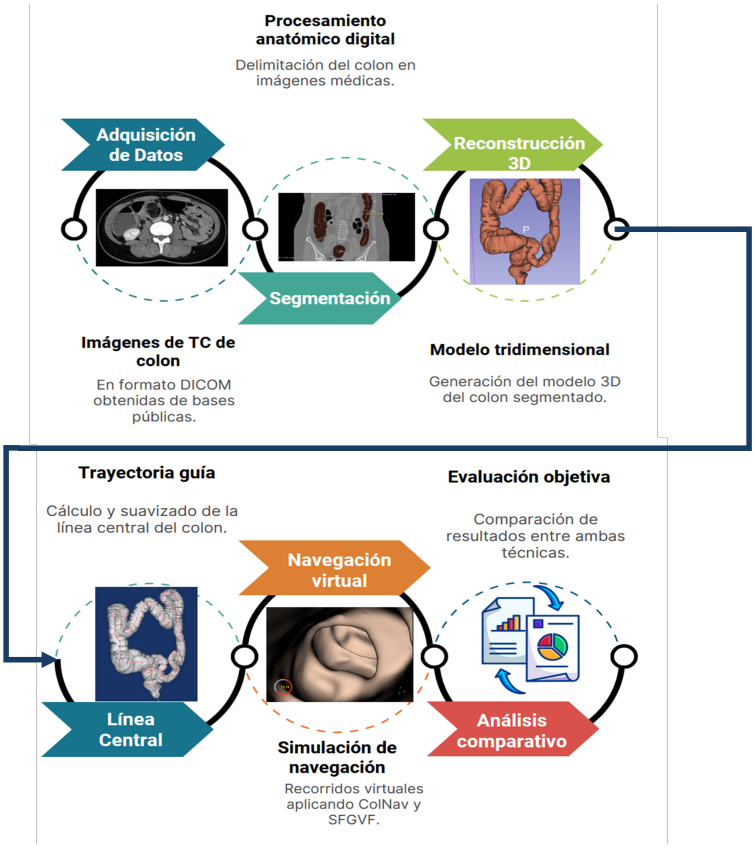


Figura 3-1: Flujo de trabajo propuesto para la evaluación y comparación de técnicas de navegación virtual en colonoscopia.

Inicialmente, un conjunto de casos clínicos fue seleccionado de una base de datos de dominio público [47]. Una vez definido el caso a ser estudiado, un método de segmentación es propuesto para separar el colón del resto de órganos visualizados en los cortes transversales de CTC. Para ello, cada imagen en 2D es segmentada, delimitando únicamente las estructuras que

corresponden al colón. Seguido, estas segmentaciones son utilizadas para la reconstrucción del modelo tridimensional del colón, para lo cual un método de reconstrucción por enmallado es aplicado (renderización). Entonces, una vez que el modelo 3D del colón es extraído, se aplica una estrategia de definición de la línea central para guiar a los métodos de navegación bajo análisis. Finalmente, los métodos de navegación son implementados para su evaluación. A continuación, se describen los pasos propuestos para esta implementación.

3.1. Casos clínicos de CTC

La selección de casos clínicos constituye el primer paso para la reconstrucción del modelo tridimensional del colón mediante colonografía por tomografía computarizada. Para este trabajo, se seleccionó un estudio clínico completo del National CT Colonography Trial (ACRIN 6664), disponible en la base pública The Cancer Imaging Archive (TCIA), el cual tiene imágenes de tomografía computarizada (CT) de alta calidad específicamente destinadas para el desarrollo y validación de técnicas computacionales en colonoscopia virtual [48].
El caso clínico elegido corresponde a un varón adulto, explorado mediante un equipo Philips Brilliance 40 bajo protocolo de colonoscopia virtual en modalidad helicoidal. Las imágenes originales fueron adquiridas en posición supina, formato DICOM, con una resolución de 512 x 512 píxeles, tamaño de voxel de 0.902 mm $\times$ 0.902 mm, grosor de corte de 1.00 mm y distancia entre cortes de 0.80 mm. El diámetro de escaneo y reconstrucción fue de 462 mm, con una tensión de tubo de 140 kVp y corriente de 185 mA. Todos estos parámetros se detallan en los encabezados (headers) DICOM. El proceso de descarga de los archivos se realizó con la herramienta NBIA Data Retriever del National Cancer Institute (NCI), que nos ayuda en la gestión y organización de grandes volúmenes de imágenes médicas. En general, los archivos recuperados corresponden a estudios completos en posición supina y prono, lo que disminuye la compresión anatómica y facilita la segmentación de las estructuras del colon por la redistribución natural de líquidos y gases en el lumen, producida por efecto de la gravedad. La serie seleccionada incluyó un total de 575 cortes axiales para la posición supina, lo que aporta suficiente resolución espacial para la reconstrucción tridimensional.

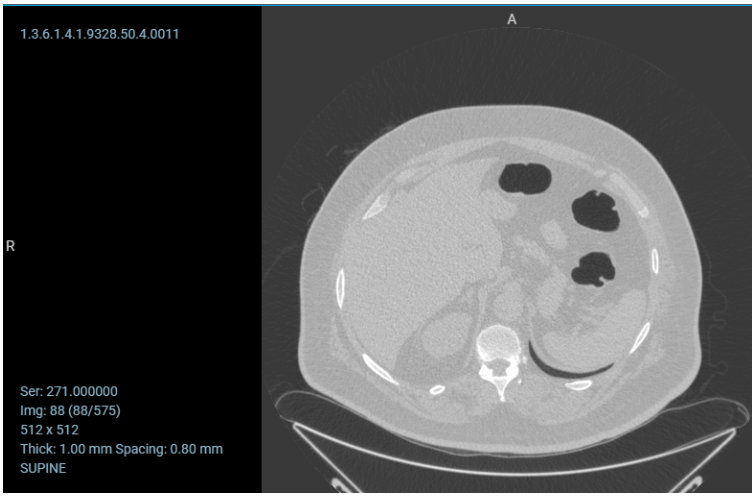


Figura 3-2: Corte axial representativo de la tomografía computarizada (TC) de colón seleccionada del dataset ACRIN 6664, con parámetros técnicos detallados en el texto.

La Figura **3-2** ilustra un corte axial representativo del caso clínico seleccionado, donde se aprecia la resolución y contraste característicos de la modalidad y parámetros técnicos descritos. La correcta organización y revisión visual previa de los estudios permitió mantener la integridad de los datos, sirviendo como base robusta para los pasos siguientes de segmentación, modelado y simulación de la navegación virtual.

3.1.1. Segmentación del colón

Después de la selección del caso clínico de CTC con imágenes DICOM, la segmentación del colón es el primer paso para reconstruir un modelo tridimensional del mismo. Este proceso se realizó en 3D Slicer, aprovechando su amplio conjunto de herramientas para la segmentación anatómica. En esta etapa, el objetivo es aislar el lumen del colón mediante la aplicación de métodos automáticos y semiautomáticos, priorizando la precisión geométrica y la integridad de la estructura segmentada [49]. La segmentación es conseguida por la aplicación de la técnica thresholding (umbralización), una técnica que clasifica los voxeles de la imagen en función de su valor de intensidad, facilitando la distinción entre el lumen del colón (aire) y los tejidos adyacentes. Formalmente, el procedimiento consiste en seleccionar todos los voxeles $I(x, y, z)$ cuya intensidad se encuentre dentro de un rango definido $[T_{min}, T_{max}]$:

$$S(x,y,z) = \begin{cases} 1, & \text{si } T_{min} \leq I(x,y,z) \leq T_{max} \\ 0, & \text{en caso contrario} \end{cases} \tag{3-1}$$

donde $S(x, y, z)$ representa la máscara binaria de segmentación. En este trabajo, se eligió el rango $T_{min} = -1024$ HU y $T_{max} = -650$ HU, que corresponde a la intensidad del aire y parte

de la mucosa colónica, permitiendo excluir estructuras de mayor densidad como: músculos y hueso [50]. Luego de la umbralización, el resultado presenta pequeñas discontinuidades y artefactos debido al ruido en la adquisición anatómica. Por corregir esto, se aplicó una estrategia de suavizado, usando principalmente un filtro de mediana, el cual reemplaza el valor de cada voxel por la mediana de sus vecinos en una ventana de tamaño $k \times k \times k$, definido por:

$$I'(x, y, z) = \text{median}\,\{I(i, j, k) \mid (i, j, k) \in \mathcal{N}(x, y, z)\} \tag{3-2}$$

donde $\mathcal{N}(x, y, z)$ representa el vecindario del voxel central. Este filtro es especialmente útil para preservar bordes y eliminar picos aislados de ruido. Además, se emplearon operaciones morfológicas como (opening) y (closing), aplicadas sobre la imagen binaria segmentada para eliminar artefactos pequeños y rellenar discontinuidades [50]. La apertura se define como una erosión seguida de una dilatación:

$$A \circ B = (A \ominus B) \oplus B \tag{3-3}$$

y el closing como una dilatación seguida de una erosión:

$$A \bullet B = (A \oplus B) \ominus B \tag{3-4}$$

donde A es la imagen segmentada y B es el elemento estructurante [50]. La erosión elimina objetos pequeños y la dilatación recupera la forma de los objetos grandes, permitiendo que solo estructuras anatómicas coherentes queden en la segmentación.

Para eliminar regiones desconectadas y garantizar que la máscara final represente exclusivamente el órgano completo (colón), se aplicó una estrategia conocida como Islands (Keep Largest Island). Formalmente, esta operación consiste en identificar todas las componentes anexas en la imagen segmentada y conservar únicamente aquella de mayor volumen, eliminando cualquier región pequeña o aislada que pueda representar ruido, restos de heces u otros artefactos [51]. Estos métodos de preprocesamiento están contenidos en el software 3D Slicer. Todo este flujo asegura que el modelo 3D obtenido sea anatómicamente estructurado y libre de errores que podrían afectar la reconstrucción del modelo final. La verificación final de la segmentación se realizó mediante inspección visual en los planos axial, sagital y coronal, confirmando la correcta estructura real del colón, tal como se ilustra en la Figura **3-3**.

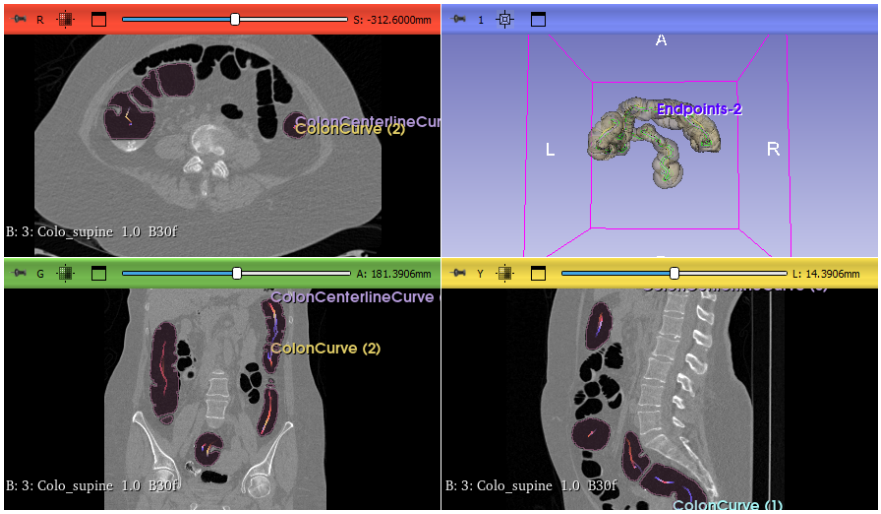


Figura 3-3: Resultado de la segmentación del colón en 3D Slicer. Se observa el lumen marcado en los planos axial (superior izquierdo), coronal (inferior izquierdo) y sagital (inferior derecho), junto con la reconstrucción tridimensional (superior derecho).

3.1.2. Reconstrucción tridimensional del colón

Una vez que el lumen del colon ha sido segmentado, se procede con la reconstrucción tridimensional del mismo, para ello, se transforman los cortes bidimensionales en un modelo volumétrico para su análisis. Este procedimiento es realizado mediante el enmallado de los voxeles contenidos en los contornos de la segmentación de cada imagen axial mediante una estrategia de renderización, la misma que está implementada en el software 3D Slicer y contenida en la herramienta *Volume Rendering*, permitiendo la generación y exportación de modelos 3D en formatos compatibles con VTK [52].
Este procedimiento utiliza la información de segmentación de los cortes axiales y resulta en el volumen anatómico del colón. Formalmente, el modelo segmentado se representa mediante una función indicadora $\chi(x, y, z)$ en el dominio tridimensional, donde χ toma valor 1 para los puntos dentro del lumen y 0 en el resto:

$$\chi(x, y, z) = \begin{cases} 1, & \text{si } (x, y, z) \text{ pertenece al lumen colónico} \\ 0, & \text{en caso contrario} \end{cases} \tag{3-5}$$

La superficie del colón se obtiene con la aplicación del algoritmo *Marching Cubes*, que recorre el volumen voxelado e identifica los puntos de transición entre la máscara segmentada. De esta forma, se genera una malla triangular que describe la superficie del lumen [49]. El modelo poligonal resultante se define como:

$$M = (V, F) \tag{3-6}$$

donde $V = \{v_i \in \mathbb{R}^3\}$ es el conjunto de vértices y F es el conjunto de caras (triángulos) que conectan estos vértices.

Entonces, el modelo generado es visualizado en el software 3D Slicer desde diferentes ángulos para comprobar la continuidad y estructura anatómica con la segmentación original [50, 52]. Finalmente, este modelo es exportado en formato (`.vtp`), como se ilustra en la Figura **3-4**, para su uso en entornos de programación con librerías VTK (Visualization Toolkit) y lenguaje de programación Python.

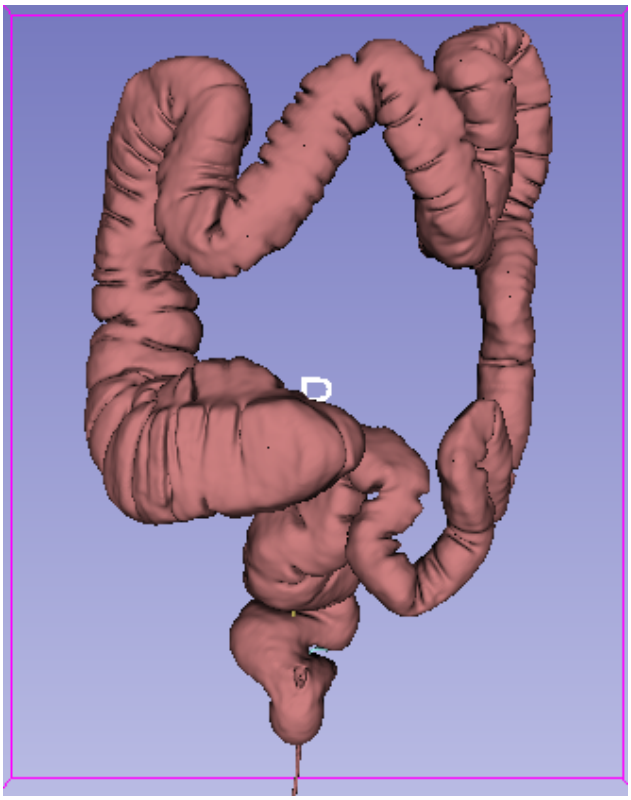


Figura 3-4: Ilustración del modelo tridimensional del colón generado a partir de la segmentación, visualizado en el entorno de reconstrucción volumétrica.

3.1.3. Extracción de la línea central del lumen del colón

Una vez definido el modelo 3D del colón, se procede a la extracción de la línea central que pasará por el centro del lumen, la cual sirve para guiar los algoritmos de navegación y establecer puntos de referencia espaciales en el interior del colon.

La extracción de la línea central se basa en el método de propagación de onda, en particular el algoritmo de *fast marching*, que simula la expansión de una onda desde un punto inicial a lo largo del volumen segmentado. Para cada punto p en el dominio segmentado, el algoritmo resuelve la ecuación de Eikonal, definida como:

$$|\nabla T(p)| = F(p) \tag{3-7}$$

El término $\nabla T(p)$ representa el gradiente espacial de $T(p)$, es decir, el vector que indica la dirección de mayor incremento del tiempo de llegada de la onda en el dominio tridimensional. Su magnitud describe la variación local de $T(p)$ respecto a la posición, por lo que el gradiente permite estimar la orientación y pendiente del frente de propagación dentro del lumen. De este modo, el gradiente actúa como guía para determinar el trayecto óptimo o camino mínimo entre dos puntos.
Donde $T(p)$ es el tiempo de llegada de la onda al punto p, y $F(p)$ es la velocidad local (habitualmente constante en el lumen). El camino de menor distancia o "geodésica" entre los extremos del colón se determina a partir del gradiente de $T(p)$, generando así una curva tridimensional que recorre el centro geométrico del lumen [53].
Una vez extraída, la línea central se representa como una curva paramétrica $\gamma(s)$, con s el parámetro de longitud de arco a lo largo de la trayectoria:

$$\gamma(s) : [0, L] \rightarrow \mathbb{R}^3 \tag{3-8}$$

Donde L es la longitud total de la línea central y $\gamma(s)$ proporciona las coordenadas tridimensionales a lo largo del recorrido interno del colón.

Este procedimiento se encuentra implementado en el software 3D Slicer, utilizando el módulo Vascular Modeling Toolkit (VMTK), el cual permite obtener la trayectoria central de estructuras tubulares a partir de modelos segmentados [52, 54]. La Figura **3-5** ilustra el resultado de este proceso, superponiendo la línea central sobre el modelo tridimensional y los cortes de los planos, verificando su correcta ubicación dentro del lumen y la continuidad a lo largo de todo el trayecto. Finalmente, la trayectoria extraída se exporta en formato FCSV, compatible con los algoritmos de navegación virtual implementados en Python y VTK [54, 55].

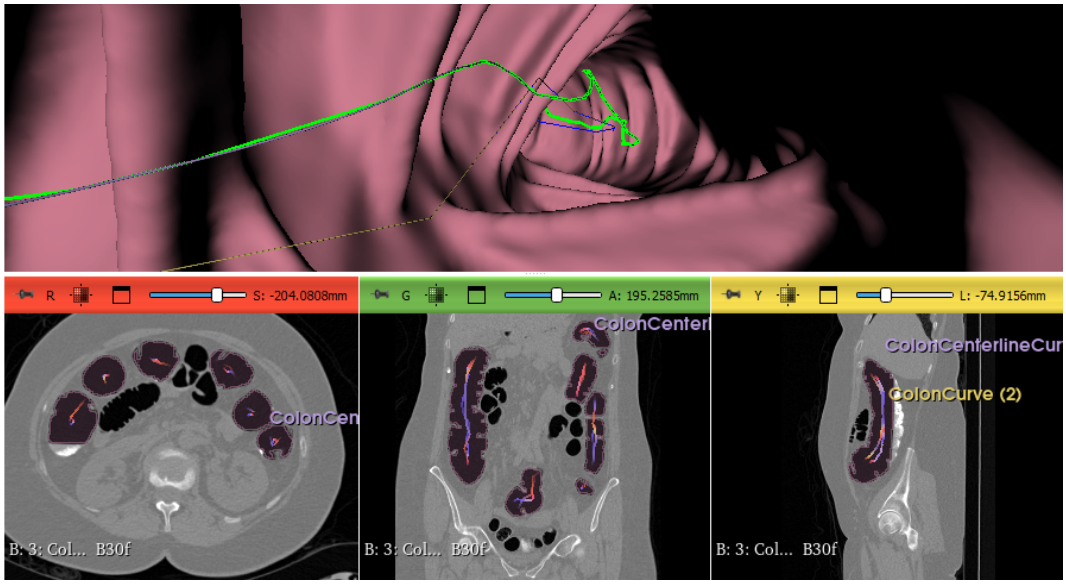


Figura 3-5: Ilustración del proceso de extracción de la línea central del colón utilizando el módulo VMTK en 3D Slicer. Se representa la curva central superpuesta tanto en el modelo tridimensional como en los planos axial, coronal y sagital.

3.1.4. Integración del entorno de desarrollo y visualización en VTK/Python

Luego de la extracción de la línea central y la reconstrucción tridimensional del colón, los archivos correspondientes al modelo 3D del colón (formato `.vtp`) y la trayectoria central (formato `.fcsv`) fueron integrados para su procesamiento y visualización en el entorno de programación Python, utilizando la biblioteca VTK.

El archivo de la malla del colón fue exportado en formato `.vtp` (VTK PolyData), que describe la geometría superficial mediante una lista de vértices y caras. La línea central es establecida como un archivo de coordenadas en formato `.fcsv`, donde cada fila representa un punto de coordenadas espaciales (x, y, z), siguiendo la continuidad de la trayectoria definida.

Para el entorno de visualización, se implementó un visor en Python 3.10, desarrollado en Visual Studio Code, con las librerías `vtk`, `numpy` y `pandas`. La función `cargar_modelo_vtp` lee el archivo `.vtp` empleando el lector `vtkXMLPolyDataReader`, cargando y almacenando la malla poligonal del colón como un objeto manipulable por VTK. La función `cargar_puntos_fcsv` procesa el archivo `.fcsv`, definiendo las coordenadas espaciales de la línea central y almacenándolas en un arreglo de `numpy`, lo que facilita operaciones geométricas y matemáticas para su posterior análisis.

La visualización interactiva es implementada mediante una configuración de mappers y actores en VTK, donde el modelo del colón se representa con una coloración neutra (gris) y la línea central se sobrepone en otro color (rojo), permitiendo comprobar la correspondencia

espacial entre ambos objetos, como se ilustra en la figura **3-6**. El renderizado se gestiona mediante la clase `vtkRenderer`, mientras que la interacción con la escena tridimensional se habilita con el estilo `vtkInteractorStyleTrackballCamera`, que permite rotar, acercar y desplazar la vista con acciones establecidas en el mouse [44, 52, 56].
El correcto funcionamiento de la importación y visualización es validado por el registro del modelo 3D y su línea central, bajo la consideración de alineamiento, que además permite la interacción rotacional, lo cual garantiza una buena geometría y una guía para la implementación de los algoritmos de navegación virtual.

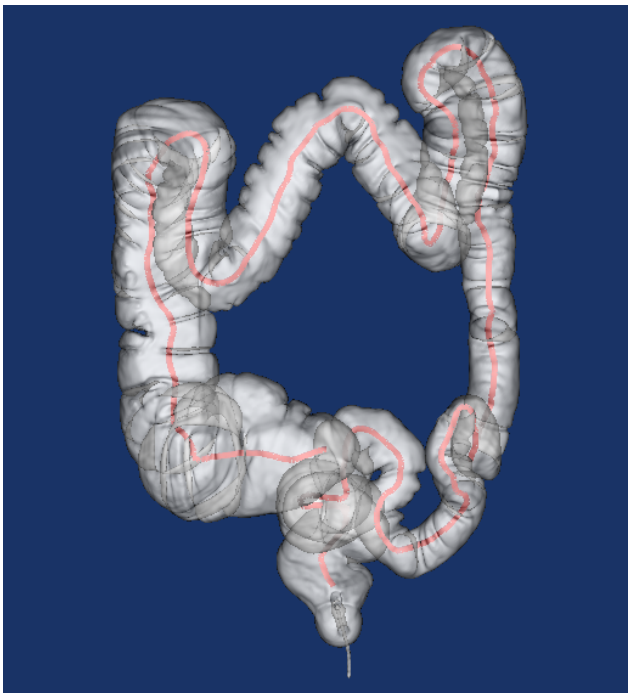


Figura 3-6: Integración del entorno de visualización en Python/VTK del modelo 3D del colon (gris) con la línea central (rojo) superpuesta.

3.1.5. Implementación de la navegación guiada en colonoscopia virtual

Se han reportado diferentes métodos propuestos en la literatura para la navegación computarizada en colonoscopia virtual, los cuales han sido analizados en este trabajo, cada uno con características particulares en cuanto a trayectoria, cobertura y adaptación a la anatomía del paciente. Luego de una revisión sistemática de la literatura reportada sobre navegación virtual en colonoscopia y basada en el modelo PRISMA, se seleccionaron dos técnicas ampliamente documentadas y empleadas en el ámbito de la navegación 3D del colón: ColNav y la navegación basada en campos vectoriales sin singularidades. Particularmente, ColNav utiliza el monitoreo de la cobertura de la superficie interna del colón y una guía visual basada

en parámetros de la superficie y el despliegue del modelo tridimensional. Por otro lado, la técnica de navegación por campos vectoriales sin singularidades implementa un enfoque basado en geometría diferencial mediante la construcción de campos de dirección que permiten el seguimiento de trayectorias en modelos con curvaturas cerradas como el colón, evitando la aparición de puntos singulares [5, 6].

La decisión de seleccionar estas técnicas se apoya, sobre todo, en la claridad con la que permiten medir y comparar resultados. Cuando se trata de navegación en modelos del colón, lo que realmente importa es conocer la eficacia de un método al recorrer y visualizar la superficie disponible del colon. Es así que en este campo, la cobertura no es solo una métrica más: se convierte en el corazón del problema [3, 5]. Una técnica puede ser novedosa o compleja, pero si no deja claro cuánta superficie queda realmente explorada, resulta difícil confiar en su utilidad real, ya sea para investigación o para la práctica clínica [5, 39].

Otra cuestión que fue importante en la selección tiene que ver con la reproducibilidad y la posibilidad de trasladar estas técnicas a diferentes escenarios, incluso cuando no se cuenta con infraestructura de alto nivel. Por ejemplo, los métodos de inteligencia artificial muy avanzados suelen requerir bases de datos masivas y procesos de entrenamiento largos, que en muchos casos no son factibles fuera de centros especializados [42, 43]. Elegir estrategias que ofrecen buenos resultados sin depender de recursos poco accesibles permite que los hallazgos y las mejoras puedan ser replicados por más equipos, en contextos variados y con herramientas estándar [5, 39].

Finalmente, priorizar métodos que utilicen métricas cuantitativas transparentes permite realizar evaluaciones objetivas y comparaciones directas entre técnicas distintas. Este aspecto permite disponer de resultados numéricos sobre el porcentaje de superficie del colón cubierta y las áreas no visualizadas, lo que facilita la identificación de mejoras, la formulación de ajustes metodológicos y el desarrollo de sistemas de navegación en diferentes entornos [3, 5, 39].

Por lo tanto, se selecciono métodos que puedan implementarse y evaluarse directamente sobre modelos tridimensionales generados a partir de casos clínicos de CTC reales, obtenidos de bases de datos de acceso público que incluyen anotación clínica documentada y métricas cuantitativas estandarizadas para su análisis. En particular, se seleccionaron la navegación ColNav y la navegación basada en campos vectoriales sin singularidades, las cuales fueron comparadas bajo las mismas condiciones experimentales y utilizando el mismo modelo anatómico tridimensional del colón, lo que permitió realizar un análisis cuantitativo y reproducible de su desempeño [5, 6].

A continuación se describe la implementación computacional de las técnicas aplicadas al modelo generado en este estudio.

Implementación de la técnica de navegación ColNav

La técnica ColNav, que viene de navegación sobre colonoscopia, es implementada sobre el modelo tridimensional del colón y la línea central, generando un entorno de navegación virtual interactivo que permite simular el recorrido de una cámara como si fuera un endoscopio real y evaluar la cobertura explorada de la superficie interna del colón.
La configuración computacional de ColNav combina procesos geométricos con visualización adaptativa en tiempo real. El sistema consta de cuatro módulos funcionales: (1) preprocesamiento del modelo 3D, (2) interpolación de la trayectoria, (3) mapeo desplegado y (4) navegación activa con retroalimentación visual. Cada módulo se comunica de forma secuencial, permitiendo la ejecución autónoma y continua del recorrido simulado [5].
El proceso inicia con el suavizado de la línea central. La secuencia discreta de puntos extraída del modelo se interpola mediante B-splines para obtener una curva continua y diferenciable. Matemáticamente, dados los puntos $\mathbf{P} = \{p_0, p_1, ..., p_N\}$, la curva suavizada $C(u)$ se genera resolviendo:

$$C(u) = \sum_{i=0}^{N} N_{i,k}(u)\, p_i \tag{3-9}$$

donde $N_{i,k}(u)$ son las funciones base B-spline de grado k y $u \in [0, 1]$ es el parámetro de la curva [3]. Este suavizado evita irregularidades y saltos bruscos durante la navegación de la cámara virtual dentro del colón, como se ilustra en la Figura **3-7**.

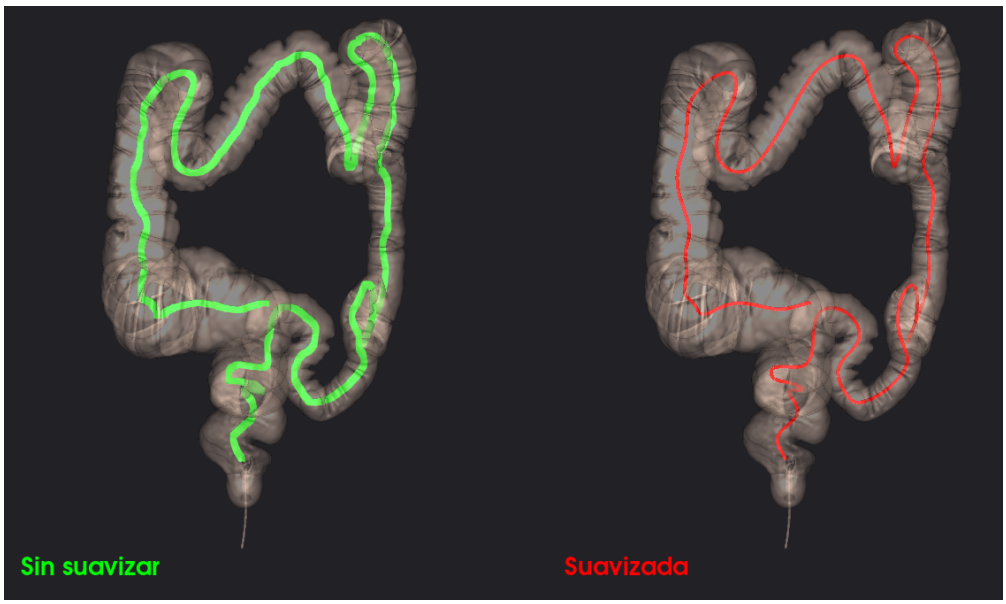


Figura 3-7: Comparación visual de la línea central extraída del modelo tridimensional del colón. A la izquierda se observa la trayectoria original (sin suavizar) y a la derecha la línea suavizada mediante interpolación B-spline.

Después, se realiza el unfolding o desplegado virtual del colón. El mismo que consiste en abrir cada corte axial del colón como información plana 2D (secciones transversales), las cuales son perpendiculares a la línea central del colón, esto se aplica a varios intervalos continuos. Así cada una de esas secciones se proyecta en un plano, y los puntos de esa sección se representan con coordenadas polares respecto al centro de la línea central:

$$\theta = \arctan 2(y, x) \tag{3-10}$$

$$r = \sqrt{x^2 + y^2} \tag{3-11}$$

donde (x, y) son las coordenadas de cada punto, las cuales son medidas desde el centroide de la sección. De este modo se desenrolla la superficie interna del colón y se coloca en un mapa bidimensional; en este mapa, el eje horizontal muestra el ángulo (θ) que indica hacia dónde está cada punto en la sección y el eje vertical indica la posición a lo largo de la línea central del colón. De esta forma es posible ver toda la superficie interna del intestino grueso como si fuera un plano, facilitando así el análisis y localización de cualquier zona, como ilustra la Figura **3-8**.

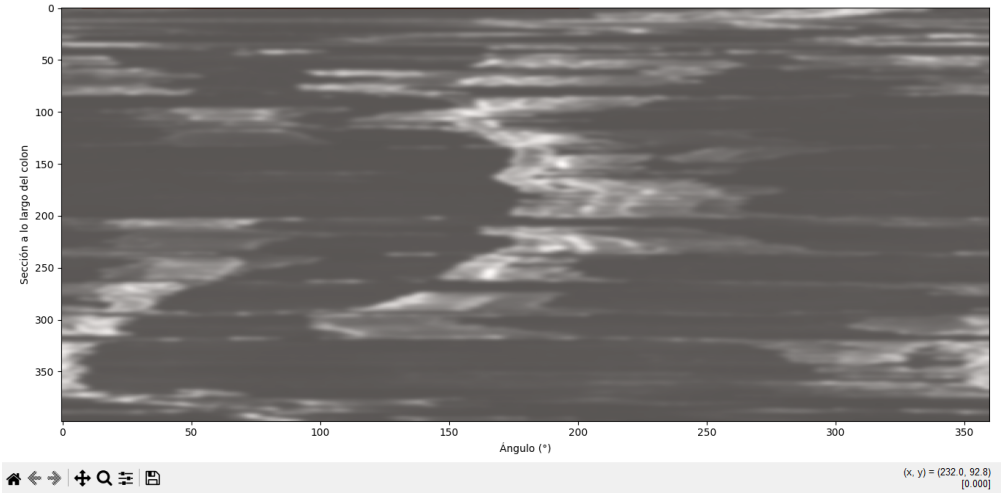


Figura 3-8: Ilustración del mapa desplegado ("unfolded") del colon. El eje horizontal corresponde al ángulo alrededor de la línea central (de 0° a 360°) y el eje vertical a la posición a lo largo del colón desde el recto hasta el ciego. Las intensidades reflejan la presencia y densidad de la superficie interna del colon en cada sección transversal.

Una vez generado el mapa desplegado, este actúa como representación auxiliar para el análisis y control de cobertura. En cada momento de navegación, se calcula el campo de visibilidad

angular de la cámara y se proyecta sobre la sección transversal correspondiente en el mapa. Esta información se almacena en una matriz binaria que representa las celdas del lumen vistas o no vistas. Así, ColNav no solo permite guiar el desplazamiento, sino también controlar su rendimiento a nivel espacial.
Durante la navegación, la cámara virtual avanza a lo largo de la línea central y puede rotar sobre su propio eje para explorar diferentes direcciones. La cobertura de la mucosa se monitorea en tiempo real sobre el mapa desplegado. Como ilustra la figura **3-9**, el sistema implementa un feedback visual, compuesto por un mini-mapa y una brújula digital. La brújula muestra los sectores ya visualizados (por ejemplo, en gris) y los pendientes (en rojo), ayudando a orientar la cámara hacia regiones no cubiertas y pasadas por alto.

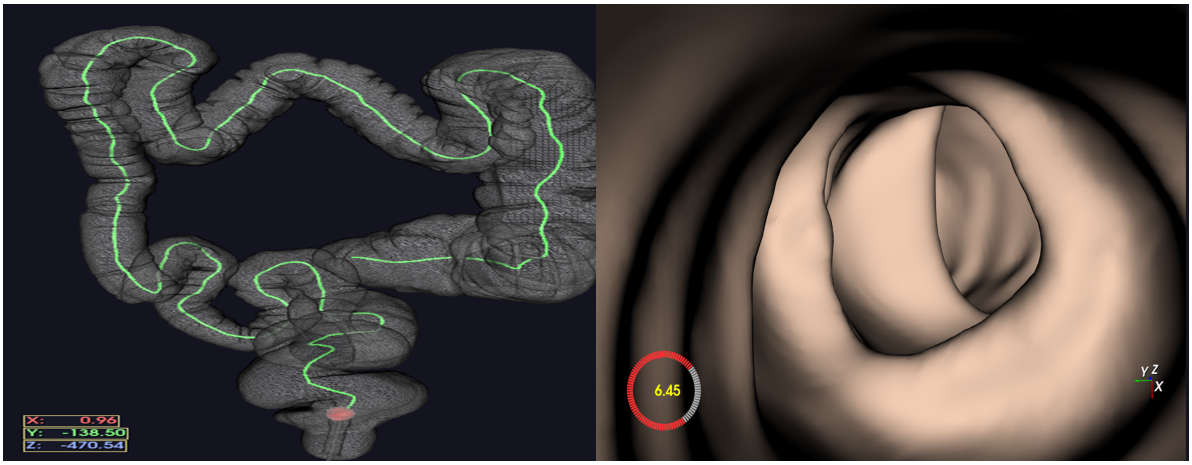


Figura 3-9: Interfaz de navegación ColNav. Izquierda: mini-mapa tridimensional del colon con la línea central marcada y la posición de la cámara virtual. Derecha: Visión endoluminal mostrando la guía visual tipo brújula digital, que indica los sectores explorados y pendientes.

Es así que al terminar la navegación, la cobertura global del lumen, la desviación de la cámara respecto al centro y la curvatura media de la trayectoria son registradas durante la simulación. Estos valores permiten comparar cuantitativamente el desempeño de la técnica ColNav frente a otros métodos, evaluando su capacidad para recorrer la totalidad del lumen y evitar zonas sin explorar.
Adicionalmente, el sistema almacena la secuencia de vectores tangentes de la trayectoria, con los cuales se calcula la curvatura local utilizando la fórmula:

$$\kappa_i = \frac{\|\mathbf{v}_{i+1} - \mathbf{v}_i\|}{\|\mathbf{p}_{i+1} - \mathbf{p}_i\|} \tag{3-12}$$

donde $\mathbf{v}_i$ representa el vector tangente en el punto i, y $\mathbf{p}_i$ la posición correspondiente en la curva. Este valor permite obtener la curvatura media de toda la trayectoria y evaluar la suavidad del desplazamiento de forma cuantitativa [42].

Implementación de navegación por campos vectoriales sin Singularidades

La técnica de navegación por campos vectoriales sin singularidades (SF-GVF, por sus siglas en inglés) constituye un método reciente que busca superar los problemas y limitantes generales de los métodos convencionales de campos vectoriales, en particular la aparición de puntos singulares en trayectorias cerradas, auto-intersectadas o complejas [41]. Esta implementación parte del mismo modelo tridimensional del colón y de la línea central suavizada descrita en la sección anterior.
El objetivo principal es guiar la posición de la cámara virtual a lo largo de una trayectoria central parametrizada, evitando puntos donde el campo vectorial se anule (singularidades). Para esto, se extiende la representación del campo de n a $n+1$ dimensiones, agregando una coordenada virtual w al espacio tridimensional físico. Sea la curva central parametrizada como $\mathbf{r}(w) = (x(w), y(w), z(w))$ para $w \in [0, 1]$.
Para ello se definió el siguiente sistema de funciones implícitas, cuyas superficies de nivel describen la trayectoria deseada en el espacio extendido:

$$\phi_1(\mathbf{x}, w) = x - x(w), \qquad \phi_2(\mathbf{x}, w) = y - y(w), \qquad \phi_3(\mathbf{x}, w) = z - z(w) \tag{3-13}$$

El conjunto solución $\{\mathbf{x}, w : \phi_i(\mathbf{x}, w) = 0\}$ representa la curva central integrada en el espacio extendido. El campo vectorial para guiar la navegación sobre esta curva se define como

$$\chi(\mathbf{x}, w) = \nabla \times \boldsymbol{\phi} - \sum_{i=1}^{n} k_i \phi_i \nabla \phi_i \tag{3-14}$$

donde $\phi_i(\mathbf{x}, w) = x_i - f_i(w)$ para $i = 1, \ldots, n$, k_i son constantes positivas, y $\nabla \times \boldsymbol{\phi}$ corresponde al producto cruzado generalizado entre los gradientes de las funciones ϕ_i. El primer término, $\nabla \times \boldsymbol{\phi}$, impulsa el movimiento a lo largo de la curva central, mientras que el segundo término atrae el punto de navegación hacia la curva.
Para el caso tridimensional, el usado en la navegación 3D del colón, ($n = 3$), las funciones implícitas son

$$\begin{cases} \phi_1(x_1, x_2, x_3, w) = x_1 - f_1(w) \\ \phi_2(x_1, x_2, x_3, w) = x_2 - f_2(w) \\ \phi_3(x_1, x_2, x_3, w) = x_3 - f_3(w) \end{cases}$$

El producto cruzado generalizado se expresa como

$$\nabla \times \boldsymbol{\phi} = \begin{pmatrix} \frac{df_1(w)}{dw} \\ \frac{df_2(w)}{dw} \\ \frac{df_3(w)}{dw} \\ 1 \end{pmatrix}$$

Por lo tanto, el campo vectorial en el espacio extendido es

$$\chi(x_1, x_2, x_3, w) = \begin{pmatrix} \frac{df_1(w)}{dw} \\ \frac{df_2(w)}{dw} \\ \frac{df_3(w)}{dw} \\ 1 \end{pmatrix} - \sum_{i=1}^{3} k_i \phi_i \nabla \phi_i$$

donde cada gradiente $\nabla\phi_i$ tiene un 1 en la i-ésima posición, $-f_i'(w)$ en la última posición, y ceros en las demás.
Este campo vectorial hace que el punto de navegación avance siguiendo la curva central y se mantenga cerca de ella. La coordenada adicional w permite evitar puntos singulares, lo cual es importante en trayectorias con auto-intersecciones o curvas cerradas [41].
El siguiente paso consiste en calcular, en cada instante de la navegación, la posición de la cámara virtual en el espacio extendido. La evolución de la posición se describe por:

$$\frac{d}{dt}\begin{pmatrix} \mathbf{x} \\ w \end{pmatrix} = s\,\frac{\chi(\mathbf{x}, w)}{\|\chi(\mathbf{x}, w)\|} \tag{3-15}$$

Dónde s es un escalar de velocidad. La proyección de la trayectoria generada en el subespacio tridimensional recupera el recorrido físico dentro del colón.
En la práctica, la implementación parte de la misma línea central suavizada obtenida previamente y genera una versión interpolada como curva paramétrica. El parámetro w representa la posición normalizada a lo largo de la curva central, variando entre 0 y 1, y permite identificar cada punto de la trayectoria en función de su progreso dentro del lumen. Para cada paso de la navegación, se calcula la posición actual y el parámetro w más cercano sobre la curva central, se evalúa el campo χ en el punto actual usando las derivadas de la curva y el desplazamiento respecto a la línea central, y el desplazamiento de la cámara se determina por la dirección del campo, normalizada.
Es así que, durante la navegación, el sistema, al igual que la técnica COlNav, visualiza de forma interactiva un mini-mapa con la posición de la cámara y la línea central del colón, lo que permite la referencia espacial y el monitoreo del avance, como se ilustra en la figura **3-10**.

Figura 3-10: Minimapa 3D del modelo de colón con la trayectoria central, útil para la referencia espacial durante la navegación SF-GVF.

De manera complementaria, la navegación incorpora una brújula digital que muestra el ángulo relativo del campo visual de la cámara respecto al eje local del colón, permitiendo al usuario mantener el sentido de orientación a lo largo del trayecto, como se ilustra en la figura **3-11**. Es decir, el ángulo que aparece en el centro del círculo indica hacia dónde está mirando la cámara virtual dentro del colón, medido en grados. Este valor muestra la diferencia entre la orientación actual de la cámara y la dirección que marca el campo vectorial de navegación. Por ejemplo, si el valor es 337°, significa que la cámara está apuntando en esa dirección específica respecto al sistema de referencia interno. Esta información ayuda a saber si la cámara está alineada con la trayectoria que se debe seguir durante la navegación virtual.

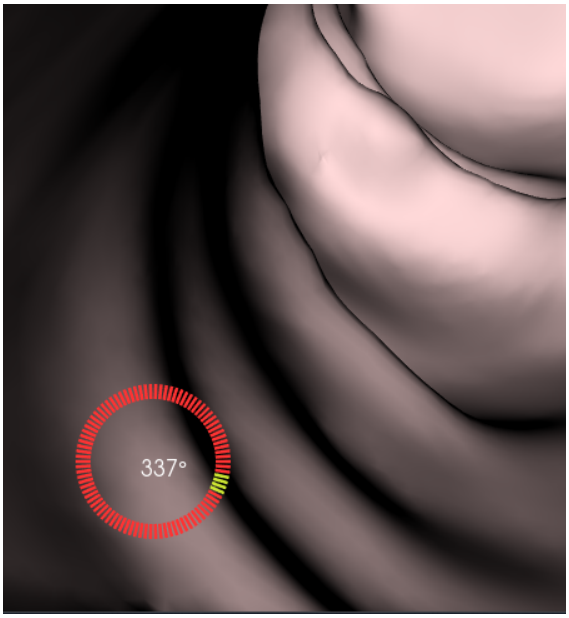


Figura 3-11: Brújula interactiva que indica la orientación relativa del campo visual respecto al eje local del colon en la técnica SF-GVF.

Además, se puede activar la visualización del vector guía SF-GVF como una flecha amarilla superpuesta en el entorno tridimensional, la cual indica la dirección instantánea calculada localmente por el campo extendido. Esto facilita la interpretación visual del avance y la verificación del funcionamiento matemático del algoritmo, como ilustra la Figura **3-12**.

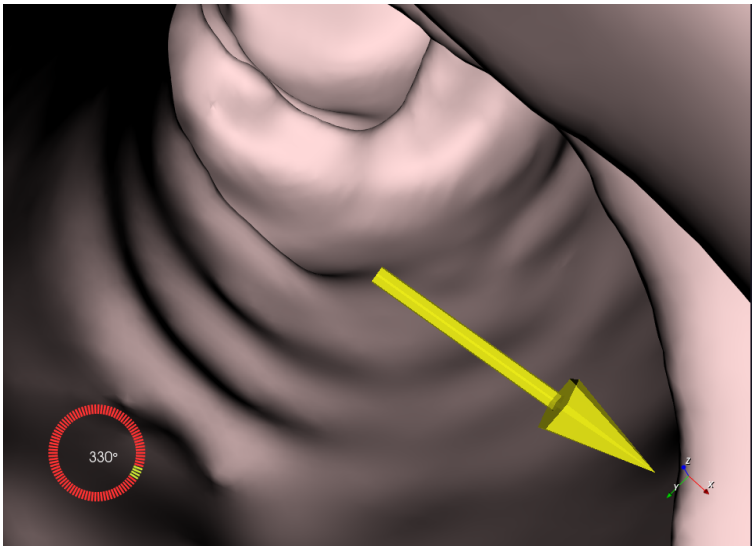


Figura 3-12: Visualización del vector guía SF-GVF (flecha amarilla), que representa la dirección calculada localmente para avanzar a lo largo del campo vectorial extendido sin singularidades.

Esta técnica implementada proporciona una interacción continua hacia la trayectoria central, sin quedarse detenido en regiones de intersección o alta curvatura, lo que la hace aplicable a la navegación virtual en modelos complejos como el colón [41].

3.1.6. Interfaz gráfica de usuario y control de parámetros de navegación

Con el propósito de asegurar que los datos se puedan replicar en todos los experimentos realizados, se implementó una interfaz gráfica de usurio, la misma que une la selección, configuración y ejecución de las técnicas de navegación bajo estudio. Esta interfaz es desarrollada en Python y basada en PySide6, y permite: cargar los archivos de modelo tridimensional del colon y de la línea central, seleccionar el algoritmo de navegación (ColNav o SF-GVF), definir el modo de operación (manual o automático) y ajustar parámetros experimentales clave como el número de cortes para ColNav para así navegar de forma más suave, recordando que un mayor número de cortes representa un tiempo mayor de cómputo, pero más suavidad en la navegación virtual.
En la figura **3-13** se ilustra la estructura de la aplicación. El usuario selecciona el modelo .vtp y la curva central .fcsv, define la técnica a evaluar y especifica el modo de navegación. La interfaz controla de forma programática la inicialización de los scripts de cada técnica, transmitiendo los parámetros seleccionados por el usuario como variables de entorno, de modo que todas las ejecuciones se realicen bajo condiciones iguales y medibles.

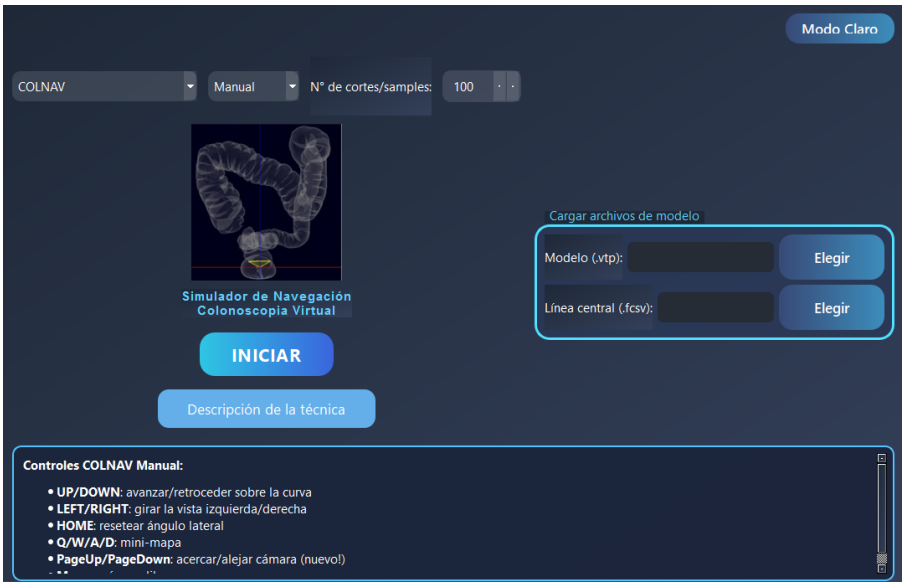


Figura 3-13: Interfaz experimental desarrollada para la ejecución sistemática y parametrizada de los algoritmos de navegación. Permite la selección de técnica, modo y parámetros, así como la carga de archivos y el monitoreo del progreso experimental.

El diseño estructurado de la interfaz permite alternar de forma fácil e intuitiva entre las distintas técnicas implementadas, asegurando que la comparación cuantitativa de resultados se base en entradas, modelos y configuraciones iguales. Adicionalmente, todos los resultados y métricas generados por la navegación de cada técnica quedan registrados y pueden ser inspeccionados al finalizar cada simulación, facilitando en gran medida el proceso de validación cruzada entre ambas técnicas de navegación virtual.

3.1.7. Métricas de evaluación

Con el fin de evaluar el comportamiento de las técnicas de navegación implementadas: ColNav y SF-GVF, se definió un conjunto de métricas cuantitativas; estas se encargan de estimar las propiedades geométricas y espaciales del recorrido dentro del modelo tridimensional reconstruido del colón. Estas métricas han sido reportadas en trabajos previos que analizan algoritmos de navegación asistida por computadora aplicados a la colonoscopia virtual, donde señalan que un sistema de navegación de calidad debe maximizar la cobertura, mantener trayectorias suaves y centradas, evitar redundancias innecesarias y trabajar con eficiencia en el tiempo [5, 57].

Cobertura global

Indica el porcentaje del lumen del colón que fue visualizado durante la navegación. En el caso de la técnica ColNav, esta información es obtenida desde la imagen desplegada del colon, donde los píxeles visibles son marcados durante la simulación. En la técnica SF-GVF, la visibilidad se evalúa mediante proyección direccional (raycasting) desde la cámara virtual hacia la superficie del modelo 3D. El cálculo se realiza así:

$$\text{Cobertura} = \frac{\sum_{i,j} \delta(F(i,j))}{M \times N} \times 100 \tag{3-16}$$

donde $F(i,j)$ representa el valor del píxel en la imagen bidimensional desplegada de tamaño $M \times N$, y $\delta(F(i,j)) = 1$ si el píxel fue marcado como cubierto, o 0 en caso contrario [5].

Desviación media al centro

La desviación al centro corresponde a la distancia promedio entre cada punto de la trayectoria y el punto más cercano sobre la línea central del modelo del colón. Sea $C = \{c_j\}$ el conjunto de puntos que componen la línea central, y p_i un punto de la trayectoria; la distancia mínima se calcula como:

$$d_i = \min_{c_j \in C} \|p_i - c_j\| \tag{3-17}$$

y la desviación media se expresa como:

$$\delta = \frac{1}{n} \sum_{i=1}^{n} d_i \tag{3-18}$$

Este valor es obtenido para cada trayectoria completa generada por los algoritmos [44].

Redundancia

La redundancia corresponde al número de veces que una misma región del lumen es recorrida más de una vez. Para el cálculo, el modelo se divide en zonas definidas y se contabiliza la cantidad de entradas repetidas a dichos sectores. Esta métrica se obtiene a partir del trazado completo de la trayectoria sobre el modelo 3D [5].
Matemáticamente, se puede expresar como:

$$R = \sum_{j=1}^{M} \text{máx}(0, n_j - 1) \tag{3-19}$$

donde R es el valor de la redundancia total, M es el número total de sectores (o bins) en los que se ha fragmentado el lumen, y n_j es el número de veces que la trayectoria visita el sector j. Si un sector solo es visitado una vez, no contribuye a la redundancia; si es visitado más de una vez, solo las visitas adicionales se cuentan. Esta métrica refleja el recorrido innecesario de regiones ya exploradas y se calcula comparando la trayectoria de la cámara virtual sobre la representación aplanada del colon, tal como se implementa en ColNav [5].

Pasos totales

Esta métrica representa la cantidad total de posiciones registradas desde el inicio hasta el final de la navegación. Se conecta directamente con la resolución temporal del desplazamiento de la cámara virtual. El valor se obtiene como el número total de pasos computados durante la simulación del algoritmo correspondiente. Matemáticamente, se expresa como:

$$N_{\text{pasos}} = \sum_{i=1}^{T} 1 \tag{3-20}$$

donde N_{pasos} es el número total de pasos, T es el número total de iteraciones o movimientos realizados por la cámara virtual a lo largo de la trayectoria (línea central), y la sumatoria recorre cada paso registrado desde el punto inicial hasta el punto final. Cada incremento corresponde a una posición distinta de la cámara durante la navegación.
Esta métrica ha sido utilizada en estudios recientes de navegación virtual para cuantificar la duración y granularidad de las trayectorias generadas [42].

4 Evaluación y resultados

En este capítulo se presentan los resultados obtenidos de las técnicas de navegación implementadas para la colonoscopia virtual, utilizando métricas que permiten ver qué tanto logran explorar el colón y cómo se comportan en distintas condiciones. No se trata solo realizar un análisis numérico; se busca entender lo que ocurre detrás de cada gráfico, cada trayecto recorrido y cada porcentaje alcanzado. Para comparar de manera eficiente estas técnicas, fue necesario partir de un mismo entorno de simulación (el mismo modelo). Así, tanto ColNav como la navegación por campos vectoriales sin singularidades (SF-GVF) fueron probadas sobre el mismo modelo 3D del colón, bajo parámetros iguales y condiciones totalmente controladas. En la técnica ColNav, la intervención del usuario permitió que el usuario siguiera las señales visuales y decidiera en tiempo real por dónde avanzar y observar. En cambio, SF-GVF, la cámara se movió guiada por el campo vectorial, indicando por dónde sería la navegación, especialmente en las curvas cerradas, sin dudas ni desvíos voluntarios. Para el análisis se seleccionaron métricas como la cobertura del lumen, la desviación respecto al centro, la redundancia y el número de pasos, ya que reflejan si una técnica realmente explora todo lo necesario y si el procedimiento se lleva a cabo correctamente. Esta decisión no fue arbitraria: en la literatura reciente, estos parámetros son los más empleados para evaluar de manera completa el desempeño de sistemas de navegación en colonoscopia virtual [58].
A continuación, la tabla **4-1** resume los resultados más representativos de ambas técnicas.

Tabla 4-1: Comparación de métricas cuantitativas entre ColNav y SF-GVF

Métrica	**ColNav (Manual)**	**SF-GVF (Automática)**
Cobertura Global	99.38 %	99.94 %
Desviación media al centro	4.942 unidades	10.000 unidades
Redundancia (repeticiones)	41	0
Pasos Totales	198	3498

Sin embargo, estas métricas por sí solas no reflejan completamente el comportamiento de la técnica; por eso, en la Figura **4-1**, se ilustra una vista obtenida durante la navegación con ColNav. En la parte inferior izquierda de la imagen, se aprecia la brújula-guía, donde el segmento gris indica las zonas del lumen que ya han sido exploradas, mientras que el segmento rojo señala las áreas que aún permanecen sin cubrir. En este caso, la cámara virtual se encuentra orientada hacia una región todavía no explorada, lo que exige al usuario

girar la vista hacia arriba y ajustar la trayectoria para incrementar la cobertura. Este tipo de retroalimentación visual es de gran importancia en la navegación, ya que permite al usuario identificar en tiempo real hacia dónde debe dirigir la cámara para mejorar el recorrido y evitar que queden zonas sin explorar. Así, el proceso se vuelve mucho más interactivo y exige una participación activa, donde la toma de decisiones es continua y afecta directamente a la calidad de la exploración. En comparación, la Figura **4-2** ilustra la navegación con SF-GVF, donde el recorrido es guiado de manera continua e igual por el campo vectorial. En la imagen se observa una flecha amarilla, que representa el vector de guiado en ese punto del trayecto, señalando la dirección hacia donde debe avanzar la cámara virtual, lo cual resulta especialmente útil al enfrentar curvas pronunciadas o cambios abruptos en la geometría del colón. En la parte inferior izquierda se encuentra la brújula circular, que indica el ángulo de orientación respecto al eje central y ayuda a visualizar el alineamiento con el campo vectorial en tiempo real. Gracias a estos elementos, la navegación logra recorrer cada rincón del lumen con gran exactitud y minimiza la posibilidad de dejar zonas sin explorar o tener repeticiones innecesarias.

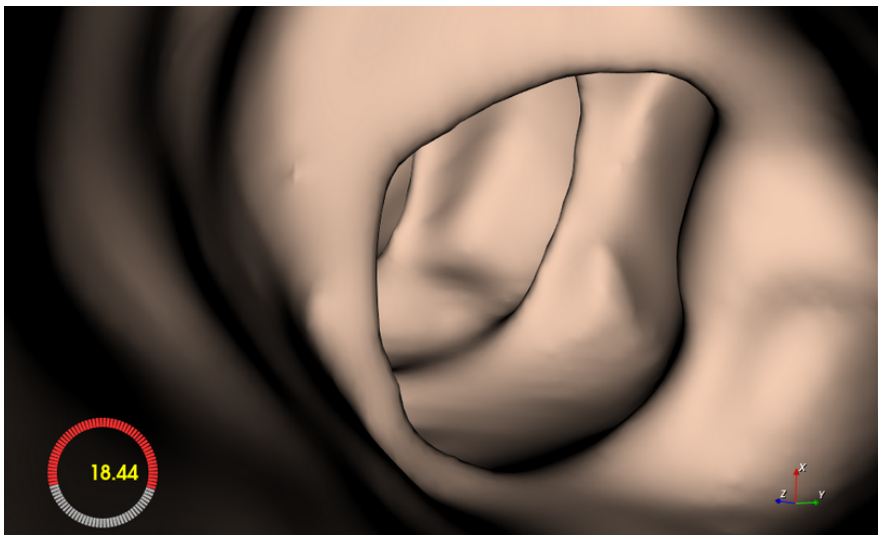


Figura 4-1: Vista endoluminal capturada durante la navegación con la técnica ColNav. En la esquina inferior izquierda se observa la brújula-guía. En la parte inferior derecha se encuentra el sistema de ejes tridimensionales (x, y, z), que sirve como orientación espacial para ubicar la posición y dirección actual dentro del modelo tridimensional del colon.

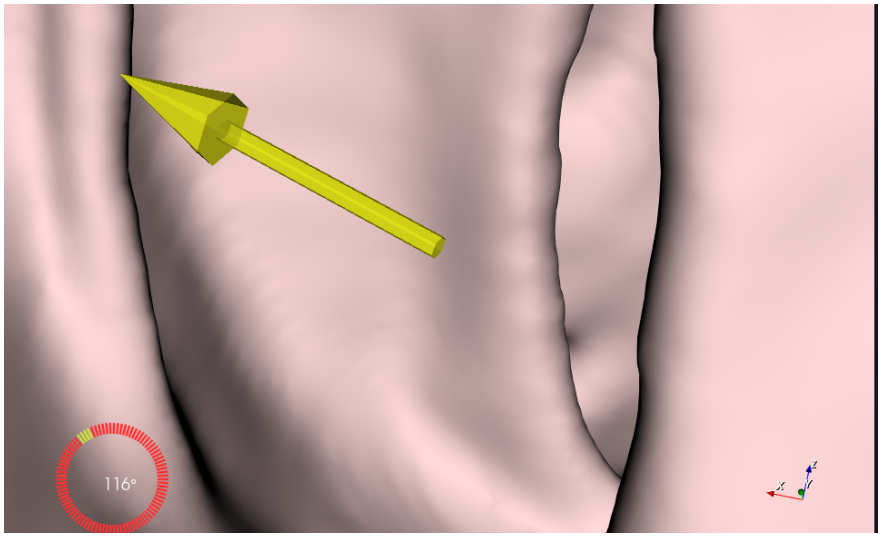


Figura 4-2: Visualización del recorrido automático con SF-GVF. La flecha amarilla representa el vector de guiado del campo vectorial.

Además, en la Figura **4-3** se ilustra el gráfico de barras que compara los valores de las métricas principales, facilitando la identificación de los puntos fuertes de cada método. Los puntos fuertes corresponden a las métricas en las que cada técnica obtiene los valores más altos o un desempeño más consistente, como mayor cobertura, menor desviación o menor redundancia en el recorrido.

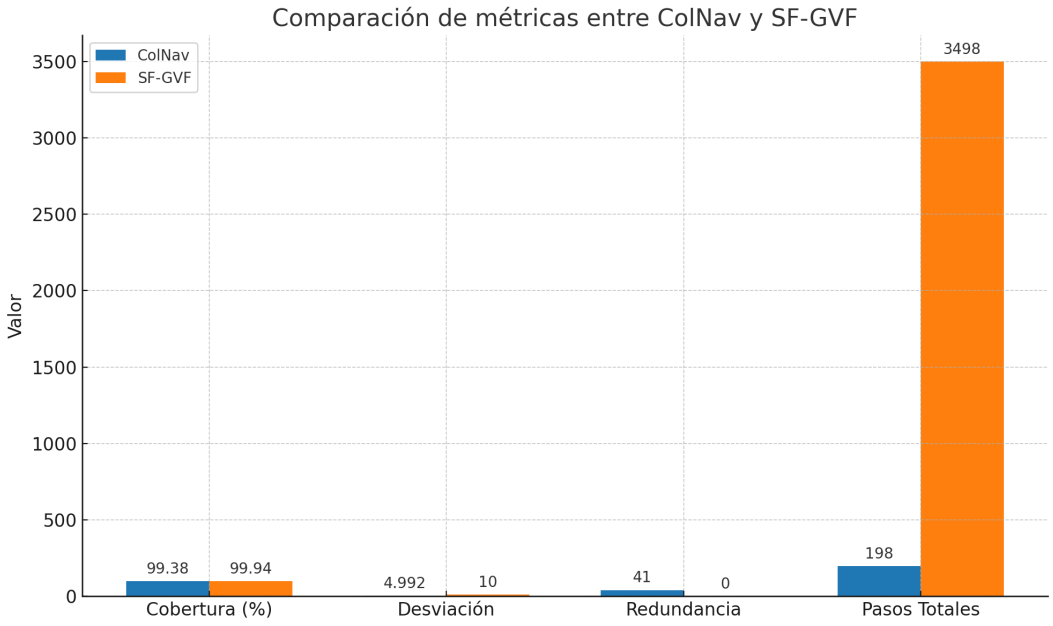


Figura 4-3: Visualización comparativa de métricas cuantitativas entre ColNav y SF-GVF.

Es importante tener en cuenta una comparación con los valores reportados por los autores

de cada técnica. Frank et al., con ColNav, documentan una cobertura promedio del 96.4 % usando modelos impresos y reales [5]. En el caso de SF-GVF, Yao et al. describen un error lateral promedio de 0.19 mm, una suavidad de trayectoria de 0.97 y garantizan convergencia global y ausencia de redundancia, incluso en caminos autointersectados o de geometría compleja [58]. En las pruebas locales, las coberturas alcanzadas (99.38 % y 99.94 %) resultan casi similares, aunque se debe tener en cuenta que las condiciones de simulación pueden diferir y que la ausencia de lesiones o variabilidad anatómica simplifica el escenario. El hecho de que SF-GVF no presente redundancia coincide con lo reportado en el artículo original y resalta su potencial para recorridos óptimos.

El análisis de las métricas resulta pertinente, ya que cada una aporta información específica sobre el comportamiento de la navegación. La cobertura representa el mapa de las regiones efectivamente exploradas; un área no recorrida puede corresponder a una zona anatómica relevante para la detección de lesiones o pólipos [5]. El hecho de que SF-GVF alcance una cobertura más alta se relaciona con su carácter automático, que garantiza una trayectoria continua sin omitir secciones del recorrido. No obstante, el control manual en ColNav mantiene importancia, pues permite ajustar la dirección de la navegación ante situaciones imprevistas y explorar zonas de interés definidas por el operador.

En cuanto a la redundancia, aunque pueda parecer una limitación, en entornos de simulación es una característica esperada. El operador puede repetir trayectos al revisar regiones de interés o validar observaciones previas. En contraste, la navegación automática mantiene una trayectoria continua y evita recorrer zonas ya visitadas. Finalmente, el número total de pasos, además de ser un parámetro técnico, determina la resolución espacial de la exploración: un mayor número de pasos implica un recorrido más detallado y un incremento del tiempo de cómputo. En ColNav, este parámetro puede ajustarse según el propósito de la simulación, permitiendo recorridos más rápidos o más densos.

Es importante saber reconocer las limitaciones del estudio. Este análisis se realizó sobre un único modelo tridimensional, en condiciones ideales y sin incluir: pólipos, lesiones ni otros factores clínicos. Tampoco se evaluó la diferencia anatómica ni la presencia de artefactos que suelen encontrarse en escenarios reales. Por lo tanto, los resultados obtenidos deben interpretarse como una primera aproximación. Para validarlos en la práctica, sería necesario probar estas técnicas sobre conjuntos de datos reales con esas alteraciones, con morfologías diversas y situaciones más complejas.

Observar las métricas y comprender cómo se relacionan con los estudios originales genera confianza en la calidad de la implementación y abre la puerta a futuras mejoras, tanto en investigación como en la práctica clínica [5, 58]. El análisis cuantitativo proporciona criterios objetivos que facilitan la evaluación y comparación de los métodos estudiados.

5 Conclusiones y recomendaciones

Conclusiones

En este estudio se comparó el desempeño de las técnicas de navegación ColNav y SF-GVF en colonoscopia virtual mediante métricas cuantitativas que describen distintos aspectos del proceso de navegación. En la cobertura global, ColNav alcanzó un 99.38 % y SF-GVF un 99.94 %, valores que indican que ambas técnicas lograron recorrer casi la totalidad del lumen y minimizar las zonas no observadas, lo que es fundamental para garantizar una exploración completa del modelo anatómico. En la desviación media al centro, ColNav presentó un valor de 4.942 unidades, mientras que SF-GVF alcanzó 10.000 unidades; esta diferencia sugiere que la trayectoria manual mantuvo una alineación más cercana al eje geométrico del colon, reduciendo el desplazamiento lateral durante la navegación. La redundancia, medida como la cantidad de repeticiones sobre las mismas regiones, fue de 41 para ColNav y 0 para SF-GVF; esto muestra que la navegación automática evita el retorno a zonas ya exploradas, lo que permite una cobertura más continua. En cuanto a los pasos totales, ColNav registró 198 movimientos y SF-GVF 3498; esta diferencia se explica porque en ColNav el número de pasos puede ser manipulado manualmente por el usuario, permitiendo ajustar la densidad de la navegación según el propósito del recorrido. Un mayor número de pasos implica una resolución espacial más fina y un aumento del tiempo de cómputo, mientras que un número menor de pasos reduce la carga computacional pero limita el nivel de detalle obtenido.

Las diferencias en el comportamiento de ambas técnicas se explican por sus principios de navegación. ColNav permite realizar ajustes manuales y correcciones durante el trayecto, lo que facilita una mejor alineación con la línea central y, por ende, menor desviación, aunque esto conlleva una mayor probabilidad de recorrer zonas ya visitadas, incrementando la redundancia. En contraste, SF-GVF utiliza un campo vectorial que dirige automáticamente la navegación hacia adelante, priorizando el recorrido y minimizando los accesos repetidos, aunque puede comprometer la alineación precisa con la curva central, sobre todo en regiones de mayor complejidad geométrica. De este modo, la selección de la técnica más adecuada dependerá de si se prioriza la precisión en la alineación o la eficiencia en la cobertura sin redundancia.

Es así que, de acuerdo con los resultados obtenidos, la técnica ColNav es ideal para escenarios donde se requiere mayor control y precisión en la alineación con la línea central, como en procesos de simulación clínica o entrenamiento. Por otro lado, la técnica SF-GVF resulta más adecuada para aplicaciones donde se busca eficiencia en el recorrido y menor redundancia,

como en evaluaciones automatizadas o comparativas de algoritmos.
Cabe señalar que, si bien ambas técnicas ofrecen distintos métodos de navegación: una basada en el control manual y otra en el desplazamiento automático mediante campos vectoriales, que permiten comparar la interacción del operador frente al cálculo autónomo del recorrido, la evaluación realizada en este estudio se basó en un único modelo tridimensional del colon, lo que constituye una limitación para generalizar los resultados a otras morfologías o condiciones clínicas. Esta decisión se fundamentó en la necesidad de mantener un entorno controlado que garantice la comparabilidad entre ambos métodos, evitando variaciones anatómicas o de segmentación que puedan alterar las métricas analizadas. El uso de un solo modelo permitió replicar las pruebas en condiciones estables y con parámetros constantes, asegurando que las diferencias observadas respondan únicamente al comportamiento de cada técnica.

Recomendaciones

De acurdo a los resultados obtenidos, se recomienda ampliar la evaluación de las técnicas ColNav y SF-GVF a un mayor número de modelos, incluyendo colones con características anatómicas más complejas, como mayor longitud, curvaturas acentuadas, segmentos parcialmente colapsados o alteraciones derivadas de patologías o intervenciones previas. Esto permitirá analizar la robustez y limitaciones de los algoritmos en escenarios más complejos en todos los ámbitos clínicos.

Asimismo, resulta oportuno validar el sistema en contextos reales mediante la participación de médicos especialistas y estudiantes de gastroenterología, con el objetivo de recoger información sobre la usabilidad, comodidad e intuición del sistema desde la perspectiva de los usuarios finales. Esta aproximación responde a la necesidad de integrar la experiencia clínica en el desarrollo y perfeccionamiento de herramientas de simulación.

Finalmente, se sugiere explorar el desarrollo de sistemas híbridos que combinen los enfoques manual y automático, permitiendo alternar el control según las necesidades del procedimiento o las preferencias del usuario. Este tipo de interacción adaptativa puede facilitar tanto el aprendizaje como la planificación clínica, y representa una evolución lógica respecto a los sistemas actuales, que en su mayoría separan estrictamente ambas modalidades. Continuar investigando en esta dirección contribuirá a la generación de herramientas más versátiles para la simulación y la práctica clínica en colonoscopia virtual.

Referencias

(1) R. Siegel, K. Miller, and A. Jemal. Colorectal cancer statistics, 2020. *CA: A Cancer Journal for Clinicians*, 70(3):145–164, 2020.

(2) Lázaro. Utilidad de la colonografía-tc o colonoscopia virtual en la patología colorrectal. *CORE*, 2017.

(3) C. Cabezón and J. Pérez. Colonografía por tomografía computada tuvo buen rendimiento para la pesquisa de cáncer colorrectal en pacientes de alto riesgo. *Revista Médica de Chile*, 2021.

(4) M. Shaabani and S. Ghafari. Advancements in ct colonography and electronic colon cleansing for enhanced colon cancer detection. *ResearchGate*, 2024.

(5) N. Frank, E. Posner, E. Muhlethaler, and A. Zholkover. Colnav: Real-time colon navigation for colonoscopy. In *MICCAI Workshop on Colonoscopy Navigation*, 2023.

(6) Weijia Yao, Héctor García de Marina, Bohuan Lin, and Ming Cao. Singularity-free guiding vector field for robot navigation. *IEEE Transactions on Robotics*, 2021.

(7) Wallapak Tavanapong, Jung Hwan Oh, Michael A. Riegler, Mohammed Khaleel, Bhuvan Mittal, and Piet C. De Groen. Artificial intelligence for colonoscopy: Past, present, and future. *IEEE Journal of Biomedical and Health Informatics*, 26:3950–3965, 8 2022.

(8) Javier Morlana, Juan D. Tardós, and J. M. M. Montiel. Colonmapper: topological mapping and localization for colonoscopy. *arXiv preprint*, 2023.

(9) A Cazelles, G Manceau, and L Maggiori. Anatomía quirúrgica del colon. *EMC-Técnicas Quirúrgicas-Aparato Digestivo*, 39(1):1–9, 2023.

(10) Jaime Ruiz-Tovar, Purificación Calero García, Vicente Morales Castiñeiras, and Enrique Martínez Molina. Vólvulo de ciego: presentación de 18 casos y revisión de la literatura. *Cirugía Española*, 85(2):110–113, 2009.

(11) Pikovit. ¿qué es el ciego y cuál es su papel en el sistema digestivo?, 2023.

(12) Jaime Szereszwski. Anatomía quirúrgica del colon. *Cirugía Digestiva*, 3(301):1–6, 2019.

(13) Freepik. Ascending colon. `https://tinyurl.com/freepik-asc-colon`, 2023. Accedido el 2025-01-11.

(14) Rolando Cuevas. Caracterización del cáncer de colon. *Cirugía paraguaya*, 41(1):8–13, 2017.

(15) Instituto Quirúrgico Lacy. Colón Transverso. En línea, 2023. Disponible en: https://www.iqlacy.com/que-operamos/cancer-de-colon/anatomia-del-colon-y-recto/ (Acceso: 11-jun-2024).

(16) C Castellón Pavón, Rey Pérez, M Hidalgo Pascual, H Ruiz, E Tomás Moro, and E Moreno Gotlzílez. del colon descendente. asimismo, el ciego, al ir situándose. *Surgery*, 103:496–498, 1988.

(17) Kenhub. Colón Descendente. En línea, 2023. Disponible en: https://www.kenhub.com/es/library/anatomia-es/colon-es (Acceso: 11-jun-2024).

(18) TE Madiba and SR Thomson. The management of sigmoid volvulus. *Journal of the Royal College of Surgeons of Edinburgh*, 45(2), 2000.

(19) Sigmoideo. Colon sigmoideo. `https://tinyurl.com/colon-sigmoideo-freepik`, 2023. Accedido el 2025-01-11.

(20) Laura L Tirado-Gómez and A Mohar-Betancourt. Epidemiología del cáncer de colon y recto. *Gaceta Sociedad Española de Oncología Médica*, 7(S4):3–11, 2008.

(21) Clínica Ciudad del Mar. Recto. En línea, 2023. Disponible en: https://www.ccdm.cl/cancer-de-colon-2/que-es-el-cancer-de-colon/ (Acceso: 11-ene-2025).

(22) G. Leifman, I. Kligvasser, R. Goldenberg, M. Elad, and E. Rivlin. Colonoscopy coverage revisited: Identifying scanning gaps in real-time. *arXiv preprint arXiv:2305.10026*, 2023.

(23) CB Williams and PD Fairclough. Colonoscopy. *Current Opinion in Gastroenterology*, 6(1):54–64, 2020.

(24) Giulia Pagano, Joan Carles Balboa Solbes, Agnes Soriano Varela, Miquel Urpi Ferreruella, and Xavier Bessa i Caserras. El cribado poblacional del cáncer colorrectal. evaluación de los resultados. *FMC - Formación Médica Continuada en Atención Primaria*, 29(4):174–181, April 2022.

(25) Aasma Shaukat, Charles J. Kahi, Carol A. Burke, Linda Rabeneck, Brian G. Sauer, and Douglas K. Rex. Long-term mortality after screening for colorectal cancer. *New England Journal of Medicine*, 385(23):2218–2229, 2021.

(26) Josefa Ibáñez, Mercedes Vanaclocha-Espí, Elena Pérez-Sanz, María José Valverde, Isabel Sáez-Lloret, Ana Molina-Barceló, Dolores Salas, and Grupo de Trabajo del Programa de Prevención de Cáncer Colorrectal de la Comunitat Valenciana. Complicaciones graves en las colonoscopias de cribado del cáncer colorrectal en la comunidad valencia-

na. *Gastroenterología y Hepatología*, 41(9):553–561, November 2018.

(27) Ignite Healthwise. Colonografía por tomografía computarizada. https://www.cigna.com/es-us/knowledge-center/hw/colonografia-por-tomografia-computarizada-abr3865, 2021. Accedido el 2025-01-12.

(28) Zhengrong Liang and Robert J Richards. Virtual colonoscopy versus optical colonoscopy. *Expert opinion on medical diagnostics*, 4(2):159–169, 2010.

(29) Xabier García-Albéniz, José Luis Hurtado, Francisco Carballo, Francisco Pérez-Riquelme, Margarita Peris, Montserrat Andreu, Luis Bujanda, Joaquín Cubiella, and Rodrigo Jover. Complicaciones graves en las colonoscopias de cribado del cáncer colorrectal en el programa de prevención de cáncer colorrectal de la comunidad valenciana. *Gastroenterología y Hepatología*, 41(9):559–567, 2018. Accedido el 2025-01-12.

(30) William A. Bye, Christopher Ma, Tran M. Nguyen, Claire E. Parker, Vipul Jairath, and James E. East. Strategies for detecting colorectal cancer in patients with inflammatory bowel disease: A cochrane systematic review and meta-analysis. *American Journal of Gastroenterology*, 113:1801–1809, 2018.

(31) Pedro Del Valle Llufrío, Sandra Rocío Romero Bareiro, and Yenia Santana Fuentes. Colon lesion diagnosed by colonoscopy in patients with hidden positive blood. *Revista Médica Electrónica*, 36(S1):692–699, 2017.

(32) Karla Carina Robelo-Arango, Rubén Gutiérrez-Alvarado, Isaías Garduño Hernández, Oscar Govea-González, and Fernando E Torre-Rendón. Correlación entre el diagnóstico óptico endoscópico e histopatológico de pólipos colónicos. *Endoscopia*, 31:346–356, 2019.

(33) Douglas K Rex, David Vining, and Kenyon K Kopecky. An initial experience with screening for colon polyps using spiral ct with and without ct colography (virtual colonoscopy). *Gastrointestinal endoscopy*, 50(3):309–313, 2019.

(34) A. Kumar and N. K. Ghosh. Ultra-minimally invasive endoscopic techniques and colorectal diseases: Current status and its future. *Artificial Intelligence in Gastrointestinal Endoscopy*, 5(2):1–12, 2024.

(35) S. Laghmati, B. Cherradi, and S. Hamida. Enhancing colorectal polyps detection using transfer learning on dicom metadata. *Engineering, Technology, Applied Science Research*, 15(1):1–10, 2025.

(36) A. E. Kaufman, S. Lakare, K. Kreeger, and I. Bitter. Virtual colonoscopy: Advances in 3d imaging and ai integration. *Communications of the ACM*, 66(4):125–138, 2023.

(37) T. Y. Lee, C. H. Lin, and Y. N. Sun. Interactive 3d virtual colonoscopy system: A review of recent developments. *IEEE Transactions on Medical Imaging*, 40(9):2150–2162, 2021.

(38) P. J. Pickhardt. Three-dimensional endoluminal ct colonography (virtual colonoscopy): Comparison of commercially available systems. *American Journal of Roentgenology*, 181(6):1599–1605, 2022.

(39) Sierra Bonilla, Shuai Zhang, Dimitrios Psychogyios, Danail Stoyanov, Francisco Vasconcelos, and Sophia Bano. Gaussian pancakes: Geometrically-regularized 3d gaussian splatting for realistic endoscopic reconstruction. *arXiv preprint*, 2024.

(40) Soraia F. Paulo, Daniel Medeiros, Daniel Lopes, and Joaquim Jorge. Controlling camera movement in vr colonography. *Virtual Reality*, 26:1079–1088, 9 2022.

(41) Weijia Yao, Héctor Garcia de Marina, Bohuan Lin, and Ming Cao. Singularity-free guiding vector field for robot navigation. *IEEE Transactions on Robotics*, 2020.

(42) Matteo Corsi, Alessandro Cosentino, Elena Di Martino, Marco Calì, Roberto Orlando, and Giovanni Bianchi. Colonoscopy navigation using end-to-end deep visuomotor control. *IEEE Transactions on Medical Robotics and Bionics*, 5(2):112–123, 2023.

(43) A. Pore, Z. Li, D. Dall'Alba, A. Hernansanz, E. De Momi, A. Menciassi, A. Casals, J. Dankelman, P. Fiorini, and E. Vander Poorten. Autonomous navigation for robot-assisted intraluminal and endovascular procedures: A systematic review. *IEEE Transactions on Robotics*, 39(4):2529–2542, 2023.

(44) Soraia F. Paulo, Daniel Medeiros, Pedro Borges, Joaquim Jorge, and Daniel Simões Lopes. Camera travel for immersive colonography. *arXiv preprint arXiv:2010.07798*, 2020.

(45) Katharina Lucas, Nathaniel Melling, Anastasios D. Giannou, Matthias Reeh, Oliver Mann, Thilo Hackert, Jakob R. Izbicki, Daniel Perez, and Julia K. Grass. Lymphatic mapping in colon cancer depending on injection time and tracing agent: A systematic review and meta-analysis of prospective designed studies. *Cancers*, 15, 6 2023.

(46) Xinxin Zhao, Jonghoek Park, Meng Li, and Hyo-Sang Choi. Singularity-free guiding vector field for robot navigation. *IEEE Transactions on Robotics*, 39(2):1323–1337, 2023.

(47) The Cancer Imaging Archive. The cancer imaging archive (tcia). `https://www.cancerimagingarchive.net/`, 2024. Accessed: 2024-07-22.

(48) The Cancer Imaging Archive (TCIA). CT COLONOGRAPHY Dataset, 2025. Acces-

sed: Jan. 30, 2025.

(49) Yuxuan Wang, Han Wang, Keqin Shen, Jincai Chang, and Jianzhong Cui. Brain ct image segmentation based on 3d slicer. *Journal of Complexity in Health Sciences*, 3:34–42, 6 2020.

(50) Gilang Argya Dyaksa, Nur Arfian, Lina Choridah, and Yosef Agung Cahyanta. Smoothing module for optimization cranium segmentation using 3d slicer. *International Journal of Applied Sciences and Smart Technologies*, 5:75, 2023.

(51) Fu Chang, Chun-Jen Chen, and Chi-Jen Lu. A component-labeling algorithm using contour tracing technique. In *Proceedings of the Seventh International Conference on Document Analysis and Recognition*, pages 849–853, 2003.

(52) 3D Slicer. Documentation 3d slicer — read the docs. `https://slicer.readthedocs.io/en/latest/user_guide/image-segmentation.html`, 2023. Accedido el 2025-01-12.

(53) Nicolas Makaroff, Théo Bertrand, and Laurent D. Cohen. Fast marching energy cnn. *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2023*, 14233:221–231, 6 2023.

(54) Alexandra Catasta, Chiara Martini, Arianna Mersanne, Ruben Foresti, Claudio Bianchini Massoni, Antonio Freyrie, and Paolo Perini. Systematic review on the use of 3d-printed models for planning, training and simulation in vascular surgery, 8 2024.

(55) Numi Sveinsson Cepero and Shawn C Shadden. Seqseg: Learning local segments for automatic vascular model construction. *Preprint*, 2025.

(56) Inc. Kitware. *vtkInteractorStyleTrackballCamera Class Reference*, 2025. Available: `https://vtk.org/doc/nightly/html/classvtkInteractorStyleTrackballCamera.html`. Accessed: 13-May-2025.

(57) Daniel Freedman, George Gaitanis, Siva Rajan, Gonen Barash, Yoel Shkolnisky, Michael Moshkovitz, and Shai Ben-Haim. Detecting and visualizing missing coverage in colonoscopy videos. *Medical Image Analysis*, 61:101653, 2020.

(58) Jianhua Yao, Binsheng Li, and Ronald M. Summers. Softennet: Symbiotic monocular depth estimation and lumen segmentation for colonoscopy endorobots. *arXiv preprint arXiv:2301.08157*, 2023.

6 Anexos

6.1. Código para la navegación COLNAV

Listing 6.1: Código completo de navegación COLNAV Manual

```
import os
import csv
import numpy as np
from scipy.interpolate import splprep, splev
from scipy.ndimage import gaussian_filter
import matplotlib.pyplot as plt
import threading
import time
import vtk
from vtkmodules.vtkRenderingAnnotation import vtkAxesActor
from vtkmodules.vtkInteractionWidgets import vtkOrientationMarkerWidget

# ====== CONFIGURACIÓN DINÁMICA DESDE LA INTERFAZ ======
file_path = os.environ.get("COLON_MODEL_PATH", r"C:\Users\T_User\Downloads
    \ColonModel.vtp")
ruta_fcsv = os.environ.get("COLON_CURVE_PATH", r"D:\TESIS 2024-2025 DATA
    SETS\ColonCenterlineCurve (0).fcsv")
num_samples = int(os.environ.get("COLON_NUM_SAMPLES", "100"))
fov_deg = 15

SENSIBILIDAD = 2
FOV_VIRTUAL = 60 if SENSIBILIDAD == 1 else fov_deg
HALF_FOV_FIX = 60 if SENSIBILIDAD == 2 else int(round(FOV_VIRTUAL / 2))
MAX_DESVIO_ANG = np.deg2rad(60)
vel_giro = 5

def calcular_tangente_interp(puntos, idx):
    idx0 = int(np.floor(idx))
    idx1 = int(np.ceil(idx))
    t = idx - idx0
    anterior = puntos[max(0, idx0 - 1)]
    siguiente = puntos[min(len(puntos) - 1, idx1 + 1)]
    tangente0 = siguiente - anterior
    tangente1 = puntos[min(len(puntos) - 1, idx1 + 1)] - puntos[max(0,
        idx1 - 1)]
```

```
33      tangente = (1 - t) * tangente0 + t * tangente1
34      return tangente / np.linalg.norm(tangente)
35
36  def calcular_view_up_interp(puntos, idx):
37      tangente = calcular_tangente_interp(puntos, idx)
38      z_ref = np.array([0, 0, 1])
39      if np.abs(np.dot(tangente, z_ref)) > 0.98:
40          ref = np.array([1, 0, 0])
41      else:
42          ref = z_ref
43      side = np.cross(tangente, ref)
44      up = np.cross(side, tangente)
45      return up / np.linalg.norm(up)
46
47  def calcular_tangente(puntos, indice):
48      anterior = puntos[max(0, indice - 1)]
49      siguiente = puntos[min(len(puntos) - 1, indice + 1)]
50      tangente = siguiente - anterior
51      return tangente / np.linalg.norm(tangente)
52
53  def calcular_view_up(tangente):
54      z_ref = np.array([0, 0, 1])
55      if np.abs(np.dot(tangente, z_ref)) > 0.98:
56          ref = np.array([1, 0, 0])
57      else:
58          ref = z_ref
59      side = np.cross(tangente, ref)
60      up = np.cross(side, tangente)
61      return up / np.linalg.norm(up)
62
63  def cargar_modelo_y_linea(file_path, ruta_fcsv, num_samples=1000):
64      reader = vtk.vtkXMLPolyDataReader()
65      reader.SetFileName(file_path)
66      reader.Update()
67      modelo_colon = reader.GetOutput()
68      puntos = []
69      with open(ruta_fcsv, 'r') as archivo:
70          lector = csv.reader(archivo)
71          for linea in lector:
72              if linea[0].startswith('#') or len(linea) < 4:
73                  continue
74              puntos.append([float(linea[1]), float(linea[2]), float(linea
                    [3])])
75      puntos = np.array(puntos)
76      tck, _ = splprep(puntos.T, s=5.0)
77      u_fino = np.linspace(0, 1, num_samples)
78      puntos_suavizados = np.array(splev(u_fino, tck)).T
79      return modelo_colon, puntos_suavizados
```

```
80
81 def generar_unfolded(modelo_colon, puntos_suavizados):
82     num_cortes = len(puntos_suavizados) - 2
83     resolucion_angular = 360
84     imagen_unfolded = np.zeros((num_cortes, resolucion_angular), dtype=np.
       float32)
85     for i in range(1, len(puntos_suavizados) - 1):
86         if i == 1 or i % max(1, num_cortes // 100) == 0 or i == num_cortes
           :
87             pct = int(100 * i / num_cortes)
88             print(f"{pct} %", flush=True)
89         p0 = puntos_suavizados[i - 1]
90         p1 = puntos_suavizados[i + 1]
91         centro = puntos_suavizados[i]
92         tangente = (p1 - p0)
93         if np.linalg.norm(tangente) == 0:
94             continue
95         tangente = tangente / np.linalg.norm(tangente)
96         plane = vtk.vtkPlane()
97         plane.SetOrigin(centro)
98         plane.SetNormal(tangente)
99         cutter = vtk.vtkCutter()
100        cutter.SetCutFunction(plane)
101        cutter.SetInputData(modelo_colon)
102        cutter.Update()
103        corte = cutter.GetOutput()
104        puntos_corte = corte.GetPoints()
105        if not puntos_corte or puntos_corte.GetNumberOfPoints() == 0:
106            continue
107        puntos = [puntos_corte.GetPoint(j) for j in range(puntos_corte.
               GetNumberOfPoints())]
108        puntos = np.array(puntos) - centro
109        for punto in puntos:
110            x, y, z = punto
111            angulo = (np.degrees(np.arctan2(y, x)) + 360) % 360
112            idx_col = int(angulo)
113            imagen_unfolded[i - 1, idx_col] += 1
114    imagen_normalizada = np.zeros_like(imagen_unfolded)
115    for i in range(imagen_unfolded.shape[0]):
116        fila = imagen_unfolded[i]
117        if fila.max() > 0:
118            imagen_normalizada[i] = fila / fila.max()
119    imagen_suavizada = gaussian_filter(imagen_normalizada, sigma=1.8)
120    return imagen_suavizada
121
122 class FeedbackCobertura:
123    def __init__(self, shape):
124        self.cobertura = np.zeros(shape, dtype=bool)
```

```
125         self.historial_vistas = []
126         self.lock = threading.Lock()
127     def marcar_vista(self, fila, angulo_centro, fov_deg=fov_deg):
128         fila = int(np.clip(fila, 0, self.cobertura.shape[0] - 1))
129         cols = self.cobertura.shape[1]
130         if SENSIBILIDAD == 1:
131             half_fov = int(round(FOV_VIRTUAL / 2))
132         elif SENSIBILIDAD == 2:
133             half_fov = HALF_FOV_FIX
134         else:
135             half_fov = int(round(fov_deg / 2))
136         with self.lock:
137             for delta in range(-half_fov, half_fov + 1):
138                 col = int((angulo_centro + delta) % cols)
139                 self.cobertura[fila, col] = True
140             self.historial_vistas.append((fila, angulo_centro))
141     def porcentaje_cobertura(self, mask_colon):
142         con_mask = np.sum(self.cobertura & mask_colon)
143         total_mask = np.sum(mask_colon)
144         return (con_mask / total_mask) * 100 if total_mask > 0 else 0.0
145
146 # (Aquí sigue TODO el resto del código, incluyendo la clase
      NavegacionColonDualMiniMapa y todo el main...)
147 # Si el código es muy largo para Overleaf, puedes partirlo en dos bloques
      lstlisting.
148 # No elimines NINGUNA LÍNEA.
149
150 # ========== MAIN ==========
151 if __name__ == "__main__":
152     modelo_colon, puntos_suavizados = cargar_modelo_y_linea(file_path,
          ruta_fcsv, num_samples=num_samples)
153     imagen_unfolded = generar_unfolded(modelo_colon, puntos_suavizados)
154     feedback = FeedbackCobertura(imagen_unfolded.shape)
155     navegacion = NavegacionColonDualMiniMapa(modelo_colon,
          puntos_suavizados, imagen_unfolded, feedback)
156     navegacion.iniciar()
```

6.2. Código para la navegación por Campos Vectoriales SF-GVF

Listing 6.2: Código completo de navegación SF-GVF

```
1 import csv
2 import vtk
3 import numpy as np
4 from scipy.interpolate import splprep, splev, interp1d
```

```
import time
from vtkmodules.vtkRenderingAnnotation import vtkAxesActor
from vtkmodules.vtkInteractionWidgets import vtkOrientationMarkerWidget

# ================================
# ==== PARÁMETROS AJUSTABLES ====
# ================================
SUAVIDAD_CURVA = 8.0
KE_SF_GVF = 2.0
DISTANCIA_CAMARA = 10
ANGULO_VISION = 135
ADELANTO_CURVA = 40

# ---- CONTROL DE CÁMARA ORIENTADA (YAW/PITCH)
INICIAL_YAW = 0.0
INICIAL_PITCH = 0.0

def cargar_puntos_fcsv(ruta_archivo):
    puntos = []
    with open(ruta_archivo, 'r') as archivo:
        lector = csv.reader(archivo)
        for linea in lector:
            if linea[0].startswith('#') or len(linea) < 4:
                continue
            puntos.append([float(linea[1]), float(linea[2]), float(linea
                [3])])
    return np.array(puntos)
    def suavizar_linea_global(puntos, num_puntos=3500, suavidad=
        SUAVIDAD_CURVA):
    tck, _ = splprep(puntos.T, s=suavidad)
    u_fino = np.linspace(0, 1, num_puntos)
    return np.array(splev(u_fino, tck)).T, tck

def calcular_tangente(puntos, indice):
    anterior = puntos[max(0, indice - 1)]
    siguiente = puntos[min(len(puntos) - 1, indice + 1)]
    tangente = siguiente - anterior
    return tangente / np.linalg.norm(tangente)

def calcular_view_up(puntos, indice):
    tangente = calcular_tangente(puntos, indice)
    z_ref = np.array([0, 0, 1])
    if np.allclose(np.abs(np.dot(tangente, z_ref)), 1.0, atol=1e-2):
        ref = np.array([1, 0, 0])
    else:
        ref = z_ref
    side = np.cross(tangente, ref)
    up = np.cross(side, tangente)
```

```
51      return up / np.linalg.norm(up)
52
53  def crear_curva_parametrica_suave(puntos):
54      N = len(puntos)
55      w_samples = np.linspace(0, 1, N)
56      spl_x = interp1d(w_samples, puntos[:, 0], kind='cubic')
57      spl_y = interp1d(w_samples, puntos[:, 1], kind='cubic')
58      spl_z = interp1d(w_samples, puntos[:, 2], kind='cubic')
59      def curva_parametrizada(w):
60          return np.array([spl_x(w), spl_y(w), spl_z(w)])
61      def derivada_curva(w):
62          eps = 1e-5
63          wp = np.clip(w + eps, 0, 1)
64          wm = np.clip(w - eps, 0, 1)
65          dx = (spl_x(wp) - spl_x(wm)) / (2 * eps)
66          dy = (spl_y(wp) - spl_y(wm)) / (2 * eps)
67          dz = (spl_z(wp) - spl_z(wm)) / (2 * eps)
68          return np.array([dx, dy, dz])
69      return curva_parametrizada, derivada_curva
70
71  def compute_sf_gvf_vector_4d(pos3d, w, curva_parametrizada, derivada_curva
        , k_e=KE_SF_GVF):
72      x, y, z = pos3d
73      xc, yc, zc = curva_parametrizada(w)
74      dx, dy, dz = derivada_curva(w)
75      e = np.array([x - xc, y - yc, z - zc])
76      grad_e1 = np.array([1, 0, 0, -dx])
77      grad_e2 = np.array([0, 1, 0, -dy])
78      grad_e3 = np.array([0, 0, 1, -dz])
79      tau = np.array([dx, dy, dz, 1.0])
80      vec = tau - k_e * (e[0]*grad_e1 + e[1]*grad_e2 + e[2]*grad_e3)
81      norm = np.linalg.norm(vec)
82      if norm == 0:
83          return tau / np.linalg.norm(tau)
84      return vec / norm
85
86  class MetricasNavegacion:
87      def __init__(self, n_total):
88          self.cobertura = np.zeros(n_total, dtype=bool)
89          self.trayectoria = []
90          self.indices_curva = []
91          self.distancias_a_centro = []
92          self.curvaturas = []
93          self.n_total = n_total
94          self.redudancia = 0
95
96      def registrar_paso(self, pos, indice_curva):
97          self.trayectoria.append(pos.copy())
```

```
98          self.indices_curva.append(indice_curva)
99          if self.cobertura[indice_curva]:
100             self.redudancia += 1
101         self.cobertura[indice_curva] = True
102
103     def calcular_metricas(self, puntos_central):
104         for pos, idx in zip(self.trayectoria, self.indices_curva):
105             self.distancias_a_centro.append(np.linalg.norm(pos -
                    puntos_central[idx]))
106         for i in range(1, len(self.trayectoria) - 1):
107             v1 = self.trayectoria[i] - self.trayectoria[i - 1]
108             v2 = self.trayectoria[i + 1] - self.trayectoria[i]
109             if np.linalg.norm(v1) > 0 and np.linalg.norm(v2) > 0:
110                 dot = np.dot(v1 / np.linalg.norm(v1), v2 / np.linalg.norm(
                        v2))
111                 self.curvaturas.append(np.arccos(np.clip(dot, -1.0, 1.0)))
112         return self
113
114     def mostrar_metricas_finales(self, nombre_tecnica=""):
115         cobertura = np.sum(self.cobertura) / self.n_total * 100
116         curvatura_media = np.mean(self.curvaturas) if self.curvaturas else
                0
117         desviacion_media = np.mean(self.distancias_a_centro) if self.
                distancias_a_centro else 0
118         print(f"\n=== MÉTRICAS: {nombre_tecnica} ===")
119         print(f"Cobertura estimada:         {cobertura:.2f}% de la curva
                central")
120         print(f"Curvatura media (rad):      {curvatura_media:.3f}")
121         print(f"Desviación media al centro: {desviacion_media:.3f}
                unidades")
122         print(f"Redundancia (repeticiones): {self.redudancia}")
123         print(f"Pasos totales:              {len(self.trayectoria)}")
124
125 class NavegacionColonProMiniMapa:
126     def __init__(self, puntos, file_path):
127         self.puntos_originales = puntos
128         self.puntos, self.tck = suavizar_linea_global(puntos)
129         self.indice_actual = 0
130
131         self.curva_parametrizada, self.derivada_curva =
                crear_curva_parametrica_suave(self.puntos)
132         self.k_e = KE_SF_GVF
133         self.metricas = MetricasNavegacion(len(self.puntos))
134         self.mini_zoom = 2.5
135         self.mini_rot_angle = 1.8
136
137         self.yaw = INICIAL_YAW
138         self.pitch = INICIAL_PITCH
```

```
139
140         self.renderer = vtk.vtkRenderer()
141         self.mini_renderer = vtk.vtkRenderer()
142         self.renderer.SetBackground(0.09, 0.10, 0.15)
143         self.renderer.SetViewport(0.0, 0.0, 0.7, 1.0)
144         self.mini_renderer.SetBackground(0.08, 0.08, 0.13)
145         self.mini_renderer.SetViewport(0.7, 0.0, 1.0, 1.0)
146
147         self.render_window = vtk.vtkRenderWindow()
148         self.render_window.SetSize(1400, 900)
149         self.render_window.AddRenderer(self.renderer)
150         self.render_window.AddRenderer(self.mini_renderer)
151
152         self.render_window_interactor = vtk.vtkRenderWindowInteractor()
153         self.render_window_interactor.SetRenderWindow(self.render_window)
154
155         self.interactor_style = vtk.vtkInteractorStyleTrackballCamera()
156         self.render_window_interactor.SetInteractorStyle(self.
               interactor_style)
157
158         self.camera = self.renderer.GetActiveCamera()
159         self.camera.SetViewAngle(ANGULO_VISION)
160         self.mini_camera = self.mini_renderer.GetActiveCamera()
161
162         self.modelo_colon, self.modelo_colon_mapper = self.
               cargar_modelo_colon(file_path)
163         self.renderer.AddActor(self.crear_actor_superficie(self.
               modelo_colon_mapper))
164         self.mini_renderer.AddActor(self.crear_actor_wireframe(self.
               modelo_colon_mapper))
165         self.actor_linea_central = self.crear_actor_linea_central(self.
               puntos)
166         self.mini_renderer.AddActor(self.actor_linea_central)
167         self.mini_esfera_actor, self.mini_esfera_source = self.
               crear_actor_esfera(self.puntos[0])
168         self.mini_renderer.AddActor(self.mini_esfera_actor)
169         self.posicion_real = self.puntos[0].copy()
170
171         # Texto XYZ en mini-mapa
172         self.textActorX = vtk.vtkTextActor()
173         self.textActorY = vtk.vtkTextActor()
174         self.textActorZ = vtk.vtkTextActor()
175         for actor, color, yoff in zip(
176             [self.textActorX, self.textActorY, self.textActorZ],
177             [(1, 0.45, 0.45), (0.55, 1, 0.55), (0.6, 0.7, 1)],
178             [72, 54, 36]
179         ):
180             tp = actor.GetTextProperty()
```

```
181             tp.SetFontSize(13)
182             tp.SetBold(True)
183             tp.SetColor(*color)
184             tp.SetShadow(True)
185             tp.SetShadowOffset(1, -1)
186             tp.SetBackgroundColor(0.10, 0.11, 0.17)
187             tp.SetBackgroundOpacity(0.65)
188             tp.SetFrame(True)
189             tp.SetFrameColor(0.90, 0.85, 0.50)
190             tp.SetJustificationToLeft()
191             tp.SetVerticalJustificationToTop()
192             actor.SetPosition(18, yoff)
193             self.mini_renderer.AddActor2D(actor)
194         self._actualizar_texto_xyz()
195 
196         # Flecha SF-GVF
197         self.arrow_source = vtk.vtkArrowSource()
198         self.arrow_mapper = vtk.vtkPolyDataMapper()
199         self.arrow_mapper.SetInputConnection(self.arrow_source.
                GetOutputPort())
200         self.arrow_actor = vtk.vtkActor()
201         self.arrow_actor.SetMapper(self.arrow_mapper)
202         self.arrow_actor.GetProperty().SetColor(1, 1, 0.1)
203         self.arrow_actor.GetProperty().SetOpacity(0.9)
204         self.arrow_actor.SetVisibility(False)
205         self.renderer.AddActor(self.arrow_actor)
206         self.mostrar_arrow = False
207         self.max_rango = max(np.ptp(self.puntos, axis=0))
208 
209         # --- BRÚJULA/COLNAV ----
210         self.compass_lines = []
211         self.compass_radius = 62
212         self.compass_center = (120, 120)
213         compass_resolution = 90
214         for i in range(compass_resolution):
215             angle_deg = i * 4
216             angle_rad = np.deg2rad(angle_deg)
217             x1 = self.compass_center[0] + (self.compass_radius - 12) * np.
                    cos(angle_rad)
218             y1 = self.compass_center[1] + (self.compass_radius - 12) * np.
                    sin(angle_rad)
219             x2 = self.compass_center[0] + self.compass_radius * np.cos(
                    angle_rad)
220             y2 = self.compass_center[1] + self.compass_radius * np.sin(
                    angle_rad)
221             line = vtk.vtkLineSource()
222             line.SetPoint1(x1, y1, 0)
223             line.SetPoint2(x2, y2, 0)
```

```
224             line.Update()
225             mapper = vtk.vtkPolyDataMapper2D()
226             mapper.SetInputData(line.GetOutput())
227             actor2d = vtk.vtkActor2D()
228             actor2d.SetMapper(mapper)
229             actor2d.GetProperty().SetLineWidth(2.5)
230             actor2d.GetProperty().SetColor(1, 0.2, 0.2)
231             self.compass_lines.append((line, actor2d))
232             self.renderer.AddActor2D(actor2d)
233 
234         self.texto_brjula = vtk.vtkTextActor()
235         self.texto_brjula.SetInput("0\u00B0")
236 
237         self.texto_brjula.GetTextProperty().SetFontSize(18)
238         self.texto_brjula.GetTextProperty().SetColor(1, 1, 1)
239         self.texto_brjula.SetPosition(self.compass_center[0]-10, self.
                compass_center[1]-16)
240         self.renderer.AddActor2D(self.texto_brjula)
241 
242         self.configurar_camara_inicial()
243         self._actualizar_mini_mapa()
244         self.render_window_interactor.AddObserver("KeyPressEvent", self.
                controlar_movimiento)
245         self.render_window_interactor.AddObserver("CharEvent", self.
                controlar_movimiento)
246         self.render_window_interactor.AddObserver("ExitEvent", self.
                finalizar_navegacion)
247 
248         # -------------- ORIENTATION MARKER WIDGET GRANDE (esquina
                inferior derecha del mini-mapa) ---------------
249         self.axes_marker = vtkAxesActor()
250         self.axes_marker.SetTotalLength(800, 80, 80)  # Tamaño de los ejes
251         self.orientation_widget = vtkOrientationMarkerWidget()
252         self.orientation_widget.SetOutlineColor(0.8, 0.8, 0.8)
253         self.orientation_widget.SetOrientationMarker(self.axes_marker)
254         self.orientation_widget.SetInteractor(self.
                render_window_interactor)
255         self.orientation_widget.SetViewport(0.89, 0.01, 0.995, 0.35)  # Á
                REA DEL WIDGET (ajustable)
256         self.orientation_widget.SetEnabled(1)
257         self.orientation_widget.InteractiveOff()
258         #
                ------------------------------------------------------------------------
259 
260     # --------- MODELO COLON -----------
261     def cargar_modelo_colon(self, file_path):
262         reader = vtk.vtkXMLPolyDataReader()
```

```
263         reader.SetFileName(file_path)
264         reader.Update()
265         mapper = vtk.vtkPolyDataMapper()
266         mapper.SetInputConnection(reader.GetOutputPort())
267         return reader.GetOutput(), mapper
268 
269     def crear_actor_wireframe(self, mapper):
270         actor = vtk.vtkActor()
271         actor.SetMapper(mapper)
272         actor.GetProperty().SetColor(1, 1, 1)
273         actor.GetProperty().SetOpacity(0.13)
274         actor.GetProperty().SetRepresentationToWireframe()
275         return actor
276 
277     def crear_actor_superficie(self, mapper):
278         actor = vtk.vtkActor()
279         actor.SetMapper(mapper)
280         actor.GetProperty().SetColor(1, 0.85, 0.85)
281         actor.GetProperty().SetOpacity(1.0)
282         return actor
283 
284     def crear_actor_linea_central(self, puntos):
285         puntos_vtk = vtk.vtkPoints()
286         for p in puntos:
287             puntos_vtk.InsertNextPoint(*p)
288         linea = vtk.vtkPolyLine()
289         linea.GetPointIds().SetNumberOfIds(len(puntos))
290         for i in range(len(puntos)):
291             linea.GetPointIds().SetId(i, i)
292         celdas = vtk.vtkCellArray()
293         celdas.InsertNextCell(linea)
294         polydata = vtk.vtkPolyData()
295         polydata.SetPoints(puntos_vtk)
296         polydata.SetLines(celdas)
297         mapper = vtk.vtkPolyDataMapper()
298         mapper.SetInputData(polydata)
299         actor = vtk.vtkActor()
300         actor.SetMapper(mapper)
301         actor.GetProperty().SetColor(0, 1, 0)
302         actor.GetProperty().SetLineWidth(2.5)
303         return actor
304 
305     def crear_actor_esfera(self, posicion):
306         esfera = vtk.vtkSphereSource()
307         esfera.SetCenter(*posicion)
308         esfera.SetRadius(6.0)
309         esfera.SetThetaResolution(20)
310         esfera.SetPhiResolution(20)
```

```
311         mapper = vtk.vtkPolyDataMapper()
312         mapper.SetInputConnection(esfera.GetOutputPort())
313         actor = vtk.vtkActor()
314         actor.SetMapper(mapper)
315         actor.GetProperty().SetColor(1, 0, 0)
316         return actor, esfera
317
318     # --------- CÁMARA Y CONTROLES -----------
319
320     def _actualizar_camara(self, indice):
321         if indice >= len(self.puntos) - 2:
322             indice = len(self.puntos) - 2
323         if indice < 0:
324             indice = 0
325         self.indice_actual = indice
326
327         punto_actual = self.puntos[indice]
328         if indice + ADELANTO_CURVA < len(self.puntos) - 1:
329             indice_adelantado = indice + ADELANTO_CURVA
330         else:
331             indice_adelantado = len(self.puntos) - 2
332
333         punto_futuro = self.puntos[indice_adelantado]
334         tangente = punto_futuro - punto_actual
335         if np.linalg.norm(tangente) < 1e-6 or np.any(np.isnan(tangente)):
336             tangente = np.array([1, 0, 0])
337         else:
338             tangente = tangente / np.linalg.norm(tangente)
339
340         up = calcular_view_up(self.puntos, indice)
341
342         eje_z = tangente
343         eje_x = np.cross(up, eje_z)
344         if np.linalg.norm(eje_x) < 1e-6 or np.any(np.isnan(eje_x)):
345             eje_x = np.array([1, 0, 0])
346         else:
347             eje_x = eje_x / np.linalg.norm(eje_x)
348         eje_y = np.cross(eje_z, eje_x)
349
350         R_yaw = self._rot_matrix(eje_y, np.deg2rad(self.yaw))
351         R_pitch = self._rot_matrix(eje_x, np.deg2rad(self.pitch))
352         direccion = R_yaw @ R_pitch @ eje_z
353
354         if np.linalg.norm(direccion) < 1e-6 or np.any(np.isnan(direccion))
                :
355             direccion = np.array([1, 0, 0])
356
357         pos = punto_actual - direccion * DISTANCIA_CAMARA
```

```
358         focal = punto_actual + direccion * 60
359 
360         self.camera.SetPosition(*pos)
361         self.camera.SetFocalPoint(*focal)
362         self.camera.SetViewUp(*up)
363         self.mini_esfera_source.SetCenter(*punto_actual)
364         self.metricas.registrar_paso(pos, indice)
365         self._actualizar_arrow_sfgvf()
366         self._actualizar_texto_xyz()
367         self.actualizar_brjula()
368 
369     def _rot_matrix(self, axis, theta):
370         axis = np.asarray(axis)
371         axis = axis / np.linalg.norm(axis)
372         a = np.cos(theta / 2.0)
373         b, c, d = -axis * np.sin(theta / 2.0)
374         return np.array([
375             [a*a+b*b-c*c-d*d, 2*(b*c-a*d),   2*(b*d+a*c)],
376             [2*(b*c+a*d),     a*a+c*c-b*b-d*d, 2*(c*d-a*b)],
377             [2*(b*d-a*c),     2*(c*d+a*b),   a*a+d*d-b*b-c*c]
378         ])
379 
380     def configurar_camara_inicial(self):
381         self._actualizar_camara(self.indice_actual)
382         self._actualizar_mini_mapa()
383 
384     def controlar_movimiento(self, obj, event):
385         key = obj.GetKeySym().lower()
386         need_camara_update = False
387         if key == "z" and self.indice_actual < len(self.puntos) - 2:
388             self.indice_actual += 1
389             need_camara_update = True
390         elif key == "x" and self.indice_actual > 0:
391             self.indice_actual -= 1
392             need_camara_update = True
393         elif key == "left":
394             self.yaw -= 10
395             need_camara_update = True
396         elif key == "right":
397             self.yaw += 10
398             need_camara_update = True
399         elif key == "up":
400             self.pitch += 10
401             self.pitch = np.clip(self.pitch, -89, 89)
402             need_camara_update = True
403         elif key == "down":
404             self.pitch -= 10
405             self.pitch = np.clip(self.pitch, -89, 89)
```

```
406             need_camara_update = True
407         elif key == "home":
408             self.yaw = INICIAL_YAW
409             self.pitch = INICIAL_PITCH
410             need_camara_update = True
411         elif key == "w":
412             self.mini_zoom = min(5.0, self.mini_zoom * 1.15)
413             self._actualizar_mini_mapa()
414         elif key == "s":
415             self.mini_zoom = max(0.2, self.mini_zoom * 0.85)
416             self._actualizar_mini_mapa()
417         elif key == "a":
418             self.mini_rot_angle -= np.pi / 24
419             self._actualizar_mini_mapa()
420         elif key == "d":
421             self.mini_rot_angle += np.pi / 24
422             self._actualizar_mini_mapa()
423         elif key == "g":
424             self.mostrar_arrow = not self.mostrar_arrow
425             self.arrow_actor.SetVisibility(self.mostrar_arrow)
426             self._actualizar_arrow_sfgvf()
427         if need_camara_update:
428             self._actualizar_camara(self.indice_actual)
429             self._actualizar_texto_xyz()
430         self.render_window.Render()
431         time.sleep(0.01)
432 
433     def actualizar_brjula(self):
434         cam_pos = np.array(self.camera.GetPosition())
435         cam_fp = np.array(self.camera.GetFocalPoint())
436         v_dir = cam_fp - cam_pos
437         v_dir = v_dir / np.linalg.norm(v_dir)
438         tangente = calcular_tangente(self.puntos, self.indice_actual)
439         up = calcular_view_up(self.puntos, self.indice_actual)
440         eje_x = np.cross(up, tangente)
441         eje_x = eje_x / np.linalg.norm(eje_x)
442         eje_y = np.cross(tangente, eje_x)
443         eje_y = eje_y / np.linalg.norm(eje_y)
444         rel_view = np.dot(v_dir, eje_y) + 1j * np.dot(v_dir, eje_x)
445         angulo = (np.angle(rel_view, deg=True) + 360) % 360
446         self.texto_brujula.SetInput(f"{angulo:3.0f}\u00B0")
447 
448         compass_resolution = len(self.compass_lines)
449         centro_linea = int((angulo / 360.0) * compass_resolution)
450         fov_lines = int(self.camera.GetViewAngle() / 360 *
                compass_resolution // 2)
451         for i, (line, actor) in enumerate(self.compass_lines):
```

```
452             if abs((i - centro_linea + compass_resolution) %
                    compass_resolution) <= fov_lines:
453                 actor.GetProperty().SetColor(0.8, 0.9, 0.2)
454             else:
455                 actor.GetProperty().SetColor(1, 0.2, 0.2)
456
457     def _actualizar_arrow_sfgvf(self):
458         if not self.mostrar_arrow:
459             self.arrow_actor.SetVisibility(False)
460             return
461         punto_actual = self.puntos[self.indice_actual]
462         w = self.indice_actual / (len(self.puntos) - 1)
463         sf_vec4d = compute_sf_gvf_vector_4d(
464             punto_actual, w,
465             self.curva_parametrizada, self.derivada_curva,
466             self.k_e)
467         sf_vec = sf_vec4d[:3]
468         start = punto_actual
469         scale_arrow = 0.01 * self.max_rango
470         end = punto_actual + sf_vec * scale_arrow
471         direction = end - start
472         length = np.linalg.norm(direction)
473         if length < 1e-3:
474             self.arrow_actor.SetVisibility(False)
475             return
476         self.arrow_actor.SetVisibility(True)
477         transform = vtk.vtkTransform()
478         transform.Translate(*start)
479         norm_dir = direction / length
480         z_axis = np.array([0, 0, 1])
481         axis = np.cross(z_axis, norm_dir)
482         angle = np.arccos(np.clip(np.dot(z_axis, norm_dir), -1.0, 1.0)) *
                180 / np.pi
483         if np.linalg.norm(axis) > 1e-5:
484             transform.RotateWXYZ(angle, *axis)
485         transform.Scale(length, length, length)
486         self.arrow_actor.SetUserTransform(transform)
487         self.arrow_actor.Modified()
488
489     def _actualizar_mini_mapa(self):
490         centro = np.mean(self.puntos, axis=0)
491         centro[2] -= 23
492         rango = np.ptp(self.puntos, axis=0)
493         max_rango = max(rango)
494         distancia = max_rango * self.mini_zoom
495         ang = self.mini_rot_angle
496         offset = np.array([
497             distancia * np.cos(ang),
```

```
498             distancia * np.sin(ang),
499             0
500         ])
501         cam_pos = centro + offset
502         up_vector = np.array([0, 0, 1])
503         self.mini_camera.SetPosition(*cam_pos)
504         self.mini_camera.SetFocalPoint(*centro)
505         self.mini_camera.SetViewUp(*up_vector)
506         self._actualizar_texto_xyz()
507 
508     def _actualizar_texto_xyz(self):
509         x, y, z = self.puntos[self.indice_actual]
510         self.textActorX.SetInput(f"X: {x:9.2f}")
511         self.textActorY.SetInput(f"Y: {y:9.2f}")
512         self.textActorZ.SetInput(f"Z: {z:9.2f}")
513 
514     def finalizar_navegacion(self, obj, event=None):
515         self.metricas.calcular_metricas(self.puntos)
516         self.metricas.mostrar_metricas_finales("SF-GVF")
517 
518     def iniciar(self):
519         self.renderer.ResetCamera()
520         self.render_window.Render()
521         self.render_window_interactor.Start()
522 
523 #
524 # ==== MAIN ====
525 ruta_fcsv = r"D:\\TESIS 2024-2025 DATA SETS\\ColonCenterlineCurve (0).fcsv
    "
526 file_path = r"C:\\Users\\T_User\\Downloads\\ColonModel.vtp"
527 
528 puntos_centro = cargar_puntos_fcsv(ruta_fcsv)
529 navegacion = NavegacionColonProMiniMapa(puntos_centro, file_path)
530 navegacion.iniciar()
```

6.3. Código para la interfaz gráfica de simulación

Listing 6.3: Código completo de la interfaz gráfica

```
1 import sys
2 import os
3 import shutil
4 import time
5 import threading
6 from PySide6 import QtWidgets, QtGui, QtCore
7 
```

```
# ===== RUTAS FIJAS (pueden quedar vacías, ya que ahora se suben los archivos) =====
SCRIPTS_DIR = r"D:\TESIS 2024-2025 DATA SETS\ColonNavigation"
SCRIPTS = {
    ("COLNAV", "Manual"):      "COLNAV6.py",
    ("COLNAV", "Automático"):  "COLNAVAu.py",
    ("Campos", "Manual"):      "CAMPOm.py",
    ("Campos", "Automático"):  "CAMPOaut.py"
}

# ===== IMÁGENES =====
ASSETS_DIR = os.path.join(os.path.dirname(__file__), "assets")
if not os.path.exists(ASSETS_DIR): os.makedirs(ASSETS_DIR)
SPLASH_PATH = os.path.join(ASSETS_DIR, "colon1.png")
LOGO_PATH = os.path.join(ASSETS_DIR, "colon2.png")

for src, dst in [
    (r"C:\Users\T_User\Desktop\colon1.png", SPLASH_PATH),
    (r"C:\Users\T_User\Desktop\colon2.png", LOGO_PATH)
]:
    if not os.path.exists(dst):
        try: shutil.copyfile(src, dst)
        except Exception as e: print(f"No se pudo copiar imagen {src}: {e}")

# ====== ESTILOS PROFESIONALES ======
DARK_STYLE = """
QWidget {
    background: qlineargradient(x1:0, y1:0, x2:1, y2:1, stop:0 #1e2837, stop:1 #34405a);
    color: #fff;
    font-size: 15px;
    font-family: 'Segoe UI', Arial;
}
QPushButton {
    background: qlineargradient(x1:0, y1:0, x2:1, y2:0, stop:0 #384a77, stop:1 #3c8dbc);
    border-radius: 16px;
    padding: 12px 18px;
    font-weight: 600;
    font-size: 17px;
    color: #fff;
}
QPushButton:hover {
    background: qlineargradient(x1:0, y1:0, x2:1, y2:0, stop:0 #5dafff, stop:1 #3c8dbc);
    color: #f2fffd;
}
```

```
QPushButton:pressed {
    background: #222;
}
QComboBox, QTextEdit, QProgressBar, QSpinBox, QLineEdit {
    background: rgba(255,255,255,0.12);
    color: #fff;
    border-radius: 10px;
    padding: 7px;
}
QComboBox QAbstractItemView {
    background: #34405a;
    selection-background-color: #3c8dbc;
    color: #fff;
}
QProgressBar {
    height: 26px;
    font-size: 17px;
    border: 2px solid #4468a7;
    border-radius: 12px;
    text-align: center;
}
QProgressBar::chunk {
    background: qlineargradient(x1:0, y1:0, x2:1, y2:0, stop:0 #33ffd1,
        stop:1 #397ad6);
    border-radius: 10px;
}
QLabel#logoTitle {
    font-size: 2.3em;
    font-family: 'Montserrat', Arial, sans-serif;
    color: #60d0ff;
    font-weight: bold;
    margin-bottom: 8px;
    letter-spacing: 1px;
}
QTextEdit {
    font-size: 1.09em;
    background: rgba(22, 28, 38, 0.92);
    border: 1px solid #4661a5;
}
QDialog, QMessageBox {
    background: qlineargradient(x1:0, y1:0, x2:1, y2:1, stop:0 #243047,
        stop:1 #222f42);
    color: #d9f7ff;
    border-radius: 15px;
}
QGroupBox {
    border: 2px solid #51e0ff;
    border-radius: 7px;
```

```
97      margin-top: 8px;
98  }
99  QGroupBox:title {
100     subcontrol-origin: margin;
101     left: 10px;
102     top: -9px;
103     padding: 0 6px 0 6px;
104     background: #1e2837;
105     color: #51e0ff;
106     font-size: 1.02em;
107 }
108 """
109
110 LIGHT_STYLE = """
111 QWidget {
112     background: qlineargradient(x1:0, y1:0, x2:1, y2:1, stop:0 #e2f1fa,
            stop:1 #d3e3fc);
113     color: #233;
114     font-size: 15px;
115     font-family: 'Segoe UI', Arial;
116 }
117 QPushButton {
118     background: qlineargradient(x1:0, y1:0, x2:1, y2:0, stop:0 #a6d8ff,
            stop:1 #68a6ff);
119     border-radius: 16px;
120     padding: 12px 18px;
121     font-weight: 600;
122     font-size: 17px;
123     color: #1c3b5a;
124 }
125 QPushButton:hover {
126     background: qlineargradient(x1:0, y1:0, x2:1, y2:0, stop:0 #d1f5ff,
            stop:1 #89c6ff);
127     color: #183959;
128 }
129 QPushButton:pressed {
130     background: #ccc;
131 }
132 QComboBox, QTextEdit, QProgressBar, QSpinBox, QLineEdit {
133     background: rgba(240,252,255,0.91);
134     color: #233;
135     border-radius: 10px;
136     padding: 7px;
137 }
138 QComboBox QAbstractItemView {
139     background: #f7fcff;
140     selection-background-color: #a6d8ff;
141     color: #123;
```

```
142 }
143 QProgressBar {
144     height: 26px;
145     font-size: 17px;
146     border: 2px solid #98b4e6;
147     border-radius: 12px;
148     text-align: center;
149 }
150 QProgressBar::chunk {
151     background: qlineargradient(x1:0, y1:0, x2:1, y2:0, stop:0 #33ffd1,
            stop:1 #6aa2ff);
152     border-radius: 10px;
153 }
154 QLabel#logoTitle {
155     font-size: 2.2em;
156     font-family: 'Montserrat', Arial, sans-serif;
157     color: #257ce6;
158     font-weight: bold;
159     margin-bottom: 8px;
160     letter-spacing: 1px;
161 }
162 QTextEdit {
163     font-size: 1.09em;
164     background: rgba(255, 255, 255, 0.95);
165     border: 1px solid #abc5e7;
166 }
167 QDialog, QMessageBox {
168     background: qlineargradient(x1:0, y1:0, x2:1, y2:1, stop:0 #ecf6ff,
            stop:1 #d3e3fc);
169     color: #16406b;
170     border-radius: 15px;
171 }
172 QGroupBox {
173     border: 2px solid #257ce6;
174     border-radius: 7px;
175     margin-top: 8px;
176 }
177 QGroupBox:title {
178     subcontrol-origin: margin;
179     left: 10px;
180     top: -9px;
181     padding: 0 6px 0 6px;
182     background: #e2f1fa;
183     color: #257ce6;
184     font-size: 1.02em;
185 }
186 """
187
```

```
INSTRUCTIONS = {
    ("COLNAV", "Manual"): """
<b>Controles COLNAV Manual:</b>
<ul>
<li><b>UP/DOWN</b>: avanzar/retroceder sobre la curva</li>
<li><b>LEFT/RIGHT</b>: girar la vista izquierda/derecha</li>
<li><b>HOME</b>: resetear ángulo lateral</li>
<li><b>Q/W/A/D</b>: mini-mapa</li>
<li><b>PageUp/PageDown</b>: acercar/alejar cámara (nuevo!)</li>
<li><b>Mouse</b>: cámara libre</li>
</ul>
""",
    ("COLNAV", "Automático"): """
<b>Controles COLNAV Automático:</b>
<ul>
<li><b>UP</b>: colocar la cámara en el primer punto (modo túnel)</li>
<li><b>SPACE</b>: pausar/reanudar modo automático</li>
<li><b>ESC</b>: cerrar simulador</li>
<li><b>LEFT/RIGHT</b>: girar vista en manual</li>
<li><b>Q/W/A/D</b>: mini-mapa</li>
<li><b>PageUp/PageDown</b>: acercar/alejar cámara</li>
<li><b>Mouse</b>: cámara libre</li>
</ul>
""",
    ("Campos", "Manual"): """
<b>Controles SF-GVF Manual:</b>
<ul>
<li><b>Z/X</b>: avanzar/retroceder sobre la curva</li>
<li><b>LEFT/RIGHT</b>: girar vista (yaw)</li>
<li><b>UP/DOWN</b>: mirar arriba/abajo (pitch)</li>
<li><b>HOME</b>: resetear ángulo</li>
<li><b>W/S/A/D</b>: mini-mapa</li>
<li><b>G</b>: mostrar/ocultar vector SF-GVF</li>
<li><b>PageUp/PageDown</b>: acercar/alejar cámara</li>
<li><b>Mouse</b>: cámara libre</li>
</ul>
""",
    ("Campos", "Automático"): """
<b>Controles SF-GVF Automático:</b>
<ul>
<li><b>Z/X</b>: avanzar/retroceder manual</li>
<li><b>SPACE/P</b>: activar/desactivar automático</li>
<li><b>LEFT/RIGHT</b>: girar vista (yaw)</li>
<li><b>UP/DOWN</b>: mirar arriba/abajo (pitch)</li>
<li><b>HOME</b>: resetear ángulo</li>
<li><b>W/S/A/D</b>: mini-mapa</li>
<li><b>G</b>: mostrar/ocultar vector SF-GVF</li>
<li><b>PageUp/PageDown</b>: acercar/alejar cámara</li>
```

```
<li><b>Mouse</b>: cámara libre</li>
</ul>
"""
}

EXPLANATIONS = {
    "COLNAV": """
<b>Que es COLNAV</b><br>
Es una técnica que guía la cámara a lo largo de la línea central del colon para lograr explorar toda la superficie interna.<br><br>
<b>Como ayuda</b><br>
Muestra un mapa desplegado del colon, indicando las zonas no vistas en negro y usando una brújula para señalar hacia dónde avanzar.<br><br>
Funciona en modo manual (control total del usuario) o automático (la cámara avanza sola por la trayectoria principal).<br><br>
<i>Su meta es facilitar la exploración completa y aumentar la detección de lesiones.</i>
""",
    "Campos": """
<b>Que es SF-GVF</b><br>
Es una técnica de navegación que utiliza campos vectoriales (flechas matemáticas) para guiar la cámara de manera continua y sin atascos.<br><br>
<b>Como ayuda</b><br>
Genera una flecha guía en cada punto que muestra la mejor dirección a seguir, incluso en trayectorias curvas o cerradas.<br><br>
Puede usarse de forma manual (viendo la flecha) o automática (la cámara sigue el campo sola).<br><br>
<i>Permite una navegación suave, cubriendo bien todas las zonas.</i>
"""
}

class ProgressPipeReader(QtCore.QObject):
    progressChanged = QtCore.Signal(int)
    textChanged = QtCore.Signal(str)
    def __init__(self, process):
        super().__init__()
        self._process = process
        self._running = True
        self._thread = threading.Thread(target=self._read)
        self._thread.daemon = True
        self._thread.start()
    def _read(self):
        import re
        prog_re = re.compile(r"(\d+(\.\d+)?)\s*%")
        while self._running:
            line = self._process.stdout.readline()
            if not line: break
            try:
```

```
277                 txt = line.decode("utf-8").strip()
278             except UnicodeDecodeError:
279                 txt = line.decode("latin-1").strip()
280             self.textChanged.emit(txt)
281             m = prog_re.search(txt)
282             if m:
283                 pct = float(m.group(1))
284                 self.progressChanged.emit(int(pct))
285         self._running = False
286     def stop(self):
287         self._running = False
288
289 class ExplanationDialog(QtWidgets.QDialog):
290     def __init__(self, tecnica, parent=None):
291         super().__init__(parent)
292         self.setWindowTitle("Explicación de la técnica")
293         self.setMinimumSize(430, 300)
294         layout = QtWidgets.QVBoxLayout()
295         text = QtWidgets.QTextBrowser()
296         text.setHtml(EXPLANATIONS[tecnica])
297         layout.addWidget(text)
298         btn = QtWidgets.QPushButton("Cerrar")
299         btn.clicked.connect(self.accept)
300         layout.addWidget(btn, alignment=QtCore.Qt.AlignRight)
301         self.setLayout(layout)
302
303 class LogoProgressDialog(QtWidgets.QDialog):
304     def __init__(self, parent=None, logo_path=None, mensaje="Generando
        cortes, por favor espere..."):
305         super().__init__(parent)
306         self.setWindowTitle("Preparando simulación")
307         self.setModal(True)
308         self.setFixedSize(360, 220)
309         layout = QtWidgets.QVBoxLayout()
310         if logo_path and os.path.exists(logo_path):
311             logo = QtWidgets.QLabel()
312             logo.setPixmap(QtGui.QPixmap(logo_path).scaledToWidth(110,
                QtCore.Qt.SmoothTransformation))
313             logo.setAlignment(QtCore.Qt.AlignCenter)
314             layout.addWidget(logo)
315         label = QtWidgets.QLabel(mensaje)
316         label.setAlignment(QtCore.Qt.AlignCenter)
317         label.setStyleSheet("font-size: 1.16em; font-weight: 600; margin:
            10px 0;")
318         layout.addWidget(label)
319         self.bar = QtWidgets.QProgressBar()
320         self.bar.setRange(0, 100)
321         layout.addWidget(self.bar)
```

```
322         self.pct_label = QtWidgets.QLabel("0%")
323         self.pct_label.setAlignment(QtCore.Qt.AlignCenter)
324         self.pct_label.setStyleSheet("font-size: 1.3em; font-weight: bold;
                color: #51e0ff;")
325         layout.addWidget(self.pct_label)
326         self.text = QtWidgets.QLabel()
327         self.text.setAlignment(QtCore.Qt.AlignCenter)
328         layout.addWidget(self.text)
329         self.setLayout(layout)
330     def set_progress(self, pct):
331         self.bar.setValue(pct)
332         self.pct_label.setText(f"{pct}%")
333     def set_text(self, txt):
334         self.text.setText(txt)
335
336 class ModelGroupBox(QtWidgets.QGroupBox):
337     """Zona para subir/cargar archivos del modelo y línea central."""
338     def __init__(self, parent=None):
339         super().__init__("Cargar archivos de modelo", parent)
340         self.setMinimumSize(390, 110)
341         layout = QtWidgets.QVBoxLayout()
342         # Modelo VTP
343         vtp_row = QtWidgets.QHBoxLayout()
344         self.vtp_label = QtWidgets.QLabel("Modelo (.vtp):")
345         self.vtp_path = QtWidgets.QLineEdit()
346         self.vtp_path.setReadOnly(True)
347         vtp_btn = QtWidgets.QPushButton("Elegir")
348         vtp_btn.clicked.connect(self.cargar_vtp)
349         vtp_row.addWidget(self.vtp_label)
350         vtp_row.addWidget(self.vtp_path)
351         vtp_row.addWidget(vtp_btn)
352         # Línea central
353         fcsv_row = QtWidgets.QHBoxLayout()
354         self.fcsv_label = QtWidgets.QLabel("Línea central (.fcsv):")
355         self.fcsv_path = QtWidgets.QLineEdit()
356         self.fcsv_path.setReadOnly(True)
357         fcsv_btn = QtWidgets.QPushButton("Elegir")
358         fcsv_btn.clicked.connect(self.cargar_fcsv)
359         fcsv_row.addWidget(self.fcsv_label)
360         fcsv_row.addWidget(self.fcsv_path)
361         fcsv_row.addWidget(fcsv_btn)
362         # Layouts
363         layout.addLayout(vtp_row)
364         layout.addLayout(fcsv_row)
365         self.setLayout(layout)
366     def cargar_vtp(self):
367         path, _ = QtWidgets.QFileDialog.getOpenFileName(self, "Selecciona
              el modelo (.vtp)", "", "Modelo VTP (*.vtp)")
```

```
368         if path: self.vtp_path.setText(path)
369     def cargar_fcsv(self):
370         path, _ = QtWidgets.QFileDialog.getOpenFileName(self, "Selecciona
                la línea central (.fcsv)", "", "Curva central (*.fcsv)")
371         if path: self.fcsv_path.setText(path)
372     def get_paths(self):
373         return self.vtp_path.text(), self.fcsv_path.text()
374
375 class MainWindow(QtWidgets.QWidget):
376     def __init__(self):
377         super().__init__()
378         self.setWindowTitle("Simulador de Navegación Colonoscopia Virtual"
                )
379         self.setWindowIcon(QtGui.QIcon(LOGO_PATH))
380         self.resize(1040, 700)
381         self.is_dark = True
382
383         # ------- Logo y título --------
384         logo = QtWidgets.QLabel()
385         if os.path.exists(LOGO_PATH):
386             logo.setPixmap(QtGui.QPixmap(LOGO_PATH).scaledToWidth(180,
                    QtCore.Qt.SmoothTransformation))
387         logo.setAlignment(QtCore.Qt.AlignCenter)
388         title = QtWidgets.QLabel("Simulador de Navegación\nColonoscopia
                Virtual")
389         title.setObjectName("logoTitle")
390         title.setAlignment(QtCore.Qt.AlignCenter)
391
392         # ------- Barra superior -------
393         self.modo_btn = QtWidgets.QPushButton("Modo Claro")
394         self.modo_btn.setFixedWidth(130)
395         self.modo_btn.setMinimumHeight(36)
396         self.modo_btn.clicked.connect(self.toggle_style)
397         barra_sup = QtWidgets.QHBoxLayout()
398         barra_sup.addStretch()
399         barra_sup.addWidget(self.modo_btn, alignment=QtCore.Qt.AlignRight)
400
401         # ------- Controles superiores -------
402         controles_top = QtWidgets.QHBoxLayout()
403         controles_top.setSpacing(13)
404         self.tecnica_combo = QtWidgets.QComboBox()
405         self.tecnica_combo.addItems(["COLNAV", "Campos Vectoriales (SF-GVF
                )"])
406         self.tecnica_combo.setMinimumHeight(34)
407         self.modo_combo = QtWidgets.QComboBox()
408         self.modo_combo.addItems(["Manual", "Automático"])
409         self.modo_combo.setMinimumHeight(34)
410
```

```
411         self.samples_spin = QtWidgets.QSpinBox()
412         self.samples_spin.setMinimum(20)
413         self.samples_spin.setMaximum(2000)
414         self.samples_spin.setValue(100)
415         self.samples_spin.setFixedWidth(94)
416         self.samples_spin.setMinimumHeight(30)
417         controles_top.addWidget(self.tecnica_combo)
418         controles_top.addWidget(self.modo_combo)
419         controles_top.addWidget(self.samples_label)
420         controles_top.addWidget(self.samples_spin)
421         controles_top.addStretch(2)
422 
423         # ------- Zona de archivos tipo "card" -------
424         self.model_box = ModelGroupBox(self)
425         self.model_box.setMinimumWidth(410)
426         self.model_box.setStyleSheet("""
427             QGroupBox {
428                 background: rgba(41,56,87,0.79);
429                 border: 2.5px solid #51e0ff;
430                 border-radius: 18px;
431                 margin-top: 16px;
432                 margin-bottom: 0px;
433                 box-shadow: 0px 3px 20px #19335560;
434             }
435             QGroupBox:title {
436                 subcontrol-origin: margin;
437                 left: 12px;
438                 top: -5px;      /* antes -17px, ahora -5px o incluso 0px
                       si lo quieres pegado */
439                 background: #1e2837;
440                 color: #60d0ff;
441                 font-size: 1.13em;
442                 font-weight: bold;
443                 padding-left: 4px;
444                 padding-right: 4px;
445               }
446 
447             QLineEdit { font-size: 15px; border-radius: 7px; background:
                  #262c36; color: #d2f8ff;}
448             QPushButton { min-width: 90px; min-height: 30px; }
449         """)
450 
451         # ------- Botón iniciar y descripción -------
452         self.iniciar_btn = QtWidgets.QPushButton("INICIAR")
453         self.iniciar_btn.setMinimumHeight(46)
454         self.iniciar_btn.setMaximumWidth(195)
455         self.iniciar_btn.setStyleSheet("""
456             QPushButton {
```

```
457                 background: qlineargradient(x1:0, y1:0, x2:1, y2:0, stop:0
                        #2ec6e3, stop:1 #3a65dd);
458                 color: #fff;
459                 font-size: 21px;
460                 border-radius: 18px;
461                 padding: 12px 38px;
462                 font-weight: bold;
463                 box-shadow: 0px 2px 12px #60c0ff70;
464                 letter-spacing: 0.5px;
465             }
466             QPushButton:hover {
467                 background: qlineargradient(x1:0, y1:0, x2:1, y2:0, stop:0
                        #59f7e0, stop:1 #518ffd);
468                 color: #f2fffd;
469                 font-size: 22px;
470             }
471         """)
472         self.iniciar_btn.clicked.connect(self.lanzar_simulador)
473 
474         self.explicacion_btn = QtWidgets.QPushButton("Descripción de la té
               cnica")
475         self.explicacion_btn.setFixedWidth(260)
476         self.explicacion_btn.setMinimumHeight(36)
477         self.explicacion_btn.clicked.connect(self.mostrar_explicacion)
478         self.explicacion_btn.setStyleSheet("""
479             QPushButton {
480                 background: #64afea;
481                 color: #fff;
482                 border-radius: 13px;
483                 font-size: 17px;
484                 font-weight: 500;
485                 margin-top: 6px;
486             }
487             QPushButton:hover {
488                 background: #47c8f7;
489                 color: #132340;
490             }
491         """)
492 
493         # ------- Instrucciones / Ayuda -------
494         self.instrucciones = QtWidgets.QTextEdit()
495         self.instrucciones.setReadOnly(True)
496         self.instrucciones.setMinimumHeight(116)
497         self.instrucciones.setMaximumHeight(200)
498         self.actualizar_instrucciones()
499         self.tecnica_combo.currentIndexChanged.connect(self.
               actualizar_instrucciones)
```

```
500         self.modo_combo.currentIndexChanged.connect(self.
               actualizar_instrucciones)
501         self.instrucciones.setStyleSheet("""
502             QTextEdit {
503                 font-size: 1.16em;
504                 background: rgba(27, 36, 59, 0.97);
505                 border: 1.8px solid #54c4ff;
506                 border-radius: 11px;
507                 color: #fff;
508                 margin-top: 11px;
509             }
510         """)
511 
512         # ------- Distribución general -------
513         grid = QtWidgets.QGridLayout()
514         grid.setHorizontalSpacing(16)
515         grid.setVerticalSpacing(10)
516         grid.addLayout(controles_top, 0, 0, 1, 2)
517         # Espaciador de 2cm (aprox. 70px) arriba del ModelGroupBox:
518         spacer = QtWidgets.QSpacerItem(20, 70, QtWidgets.QSizePolicy.
               Minimum, QtWidgets.QSizePolicy.Fixed)
519         grid.addItem(spacer, 0, 2, 1, 1)
520         grid.addWidget(self.model_box, 1, 2, 5, 1, alignment=QtCore.Qt.
               AlignVCenter)
521         grid.addWidget(logo, 1, 0, 2, 2, alignment=QtCore.Qt.AlignCenter)
522         grid.addWidget(title, 3, 0, 1, 2, alignment=QtCore.Qt.AlignCenter)
523         grid.addWidget(self.iniciar_btn, 4, 0, 1, 2, alignment=QtCore.Qt.
               AlignCenter)
524         grid.addWidget(self.explicacion_btn, 5, 0, 1, 2, alignment=QtCore.
               Qt.AlignCenter)
525         grid.addWidget(self.instrucciones, 6, 0, 2, 3)
526         grid.setColumnStretch(0, 1)
527         grid.setColumnStretch(1, 1)
528         grid.setColumnMinimumWidth(2, 430)
529         grid.setRowMinimumHeight(1, 95)
530         grid.setRowStretch(7, 1)
531 
532         main = QtWidgets.QVBoxLayout()
533         main.addLayout(barra_sup)
534         main.addSpacing(10)
535         main.addLayout(grid)
536         self.setLayout(main)
537         self.setStyleSheet(DARK_STYLE)
538 
539     def toggle_style(self):
540         self.is_dark = not self.is_dark
541         self.setStyleSheet(DARK_STYLE if self.is_dark else LIGHT_STYLE)
```

```
542         self.modo_btn.setText("Modo Claro" if self.is_dark else "Modo
                Oscuro")
543
544     def mostrar_explicacion(self):
545         tecnica = "COLNAV" if "COLNAV" in self.tecnica_combo.currentText()
                else "Campos"
546         dlg = ExplanationDialog(tecnica, self)
547         dlg.exec()
548
549     def actualizar_instrucciones(self):
550         tecnica = "COLNAV" if "COLNAV" in self.tecnica_combo.currentText()
                else "Campos"
551         modo = self.modo_combo.currentText()
552         html = INSTRUCTIONS.get((tecnica, modo), "Controles no disponibles
                .")
553         self.instrucciones.setHtml(html)
554
555     def lanzar_simulador(self):
556         tecnica = "COLNAV" if "COLNAV" in self.tecnica_combo.currentText()
                else "Campos"
557         modo = self.modo_combo.currentText()
558         samples = self.samples_spin.value()
559         vtp_path, fcsv_path = self.model_box.get_paths()
560
561         if not (os.path.isfile(vtp_path) and os.path.isfile(fcsv_path)):
562             QtWidgets.QMessageBox.critical(self, "Error", "Debes
                    seleccionar ambos archivos:\n\n- Modelo .vtp\n- Línea
                    central .fcsv")
563             return
564
565         key = (tecnica, modo)
566         script_path = os.path.join(SCRIPTS_DIR, SCRIPTS[key])
567
568         # Mostrar explicación SIEMPRE antes de iniciar
569         dlg = ExplanationDialog(tecnica, self)
570         dlg.exec()
571
572         import subprocess
573         import re
574
575         progress = None
576         outputs = []
577         use_progress = tecnica == "COLNAV"
578         if use_progress:
579             progress = LogoProgressDialog(self, LOGO_PATH)
580             progress.show()
581             QtWidgets.QApplication.processEvents()
582         env = os.environ.copy()
```

```
583         env["COLON_MODEL_PATH"] = vtp_path
584         env["COLON_CURVE_PATH"] = fcsv_path
585         env["COLON_NUM_SAMPLES"] = str(samples)
586         process = subprocess.Popen(
587             [sys.executable, script_path],
588             stdout=subprocess.PIPE, stderr=subprocess.STDOUT,
589             env=env
590         )
591         def handle_text(txt):
592             outputs.append(txt)
593             if progress:
594                 progress.set_text(txt)
595         reader = ProgressPipeReader(process)
596         if progress:
597             reader.progressChanged.connect(progress.set_progress)
598         reader.textChanged.connect(handle_text)
599 
600         while process.poll() is None:
601             QtWidgets.QApplication.processEvents()
602             time.sleep(0.03)
603         reader.stop()
604         if progress:
605             progress.set_progress(100)
606             progress.set_text("Completado.")
607             QtWidgets.QApplication.processEvents()
608             time.sleep(0.8)
609             progress.close()
610 
611         salida = '\n'.join(outputs).strip()
612         lineas_finales = []
613         for linea in salida.splitlines():
614             if not re.match(r'^\s*\d+\s*%$', linea):
615                 lineas_finales.append(linea)
616         salida_final = '\n'.join([l for l in lineas_finales if l.strip()])
617         if salida_final:
618             QtWidgets.QMessageBox.information(self, "Métricas o salida
                    final", salida_final)
619 
620 def main():
621     app = QtWidgets.QApplication(sys.argv)
622     if os.path.exists(SPLASH_PATH):
623         splash_pix = QtGui.QPixmap(SPLASH_PATH)
624         splash = QtWidgets.QSplashScreen(splash_pix)
625         splash.showMessage(
626             "Simulador de Colonoscopia Virtual\nCargando...",
627             alignment=QtCore.Qt.AlignBottom | QtCore.Qt.AlignCenter,
628             color=QtCore.Qt.white
629         )
```

```
630         splash.show()
631         app.processEvents()
632         time.sleep(2.5)
633     else:
634         splash = None
635 
636     window = MainWindow()
637     window.show()
638     if splash: splash.finish(window)
639     sys.exit(app.exec())
640 
641 
642 
643 if __name__ == "__main__":
644     main()
```