



Instituto Tecnológico de Buenos Aires

72.45 - PROYECTO FINAL - 2022

DESARROLLO DE UNA APLICACIÓN WEB  
PARA EL TRACKING DE FILAMENTOS  
INTRACELULARES A PARTIR DE IMÁGENES  
DE MICROSCOPÍA

**Tutora**

- Bruno, Luciana

**Autores**

- 59.527 - Brandy, Tobías Martín
- 59.244 - Sagüés, Ignacio
- 59.223 - Pannunzio, Faustino

## **Agradecimientos**

Agradecemos a nuestra tutora Luciana Bruno por guiarnos y ayudarnos durante todo el proyecto. A Valeria Soliani por acompañarnos en el proceso. A todos aquellos profesores cuyas enseñanzas nos permitieron llevar a cabo este trabajo, especialmente a María Juliana Gambini tanto por lo enseñado en materia de imágenes como por su atención ante nuestras dudas durante el proyecto. A Esteban Mocskos y Adrián Etchevarne por el soporte técnico. Por último, a nuestras familias, por su compañía y apoyo en este recorrido.

# Índice

<b>1. Introducción</b>	<b>1</b>
<b>2. Estado del arte</b>	<b>2</b>
2.1. Algoritmos de tracking de biofilamentos . . . . .	2
2.1.1. Umbralización . . . . .	2
2.1.2. Contornos activos . . . . .	3
2.1.3. Análisis de perfiles de intensidad . . . . .	3
2.2. AFTER . . . . .	4
2.3. Trabajo final previo . . . . .	5
<b>3. Extensión y alcance del proyecto</b>	<b>7</b>
<b>4. Solución</b>	<b>8</b>
4.1. Requisitos Funcionales . . . . .	8
4.2. Requisitos No Funcionales . . . . .	9
4.3. Relevamiento de la aplicación existente y mejoras implementadas . . . . .	9
4.4. Backend: Algoritmo de tracking . . . . .	14
4.4.1. Primera iteración: determinación aproximada de la forma del filamento . . . . .	15
4.4.2. Ajuste de los perfiles de intensidad en la dirección normal al filamento . . . . .	16
4.4.3. Obtención de las coordenadas del centro del filamento: ajuste gaussiano del perfil de intensidad . . . . .	17
4.4.4. Reposición de coordenadas longitudinales por interpolación . . . . .	18
4.4.5. Suavizado de la forma recuperada . . . . .	18
4.4.6. Complejidad temporal . . . . .	21
4.5. Backend: Implementación . . . . .	21
4.5.1. Implementación del algoritmo . . . . .	21
4.5.2. Arquitectura de la aplicación web . . . . .	22
4.6. Frontend: Interfaz web . . . . .	23
<b>5. Resultados</b>	<b>27</b>
5.1. Validación . . . . .	27
5.1.1. Metodología . . . . .	27
5.1.2. Análisis de precisión y rendimiento . . . . .	28
5.1.3. Intersección de filamentos lineales . . . . .	34
5.2. Casos de uso reales . . . . .	36
5.2.1. Tracking de filamentos en imágenes de microscopía confocal . . . . .	36
5.2.2. Tracking de filamentos en imágenes de microscopía Airyscan . . . . .	39
5.2.3. Tracking de filamentos macroscópicos tomados con cámara CCD . . . . .	43
<b>6. Manual de uso</b>	<b>44</b>

<b>7. Conclusiones y trabajo futuro</b>	<b>45</b>
<b>8. Bibliografia</b>	<b>47</b>
<b>9. Anexo</b>	<b>50</b>

## 1. Introducción

La microscopía de fluorescencia ha evolucionado durante los últimos años cambiando la forma en que exploramos y entendemos la función celular. Inicialmente utilizada para observar la distribución de componentes celulares en especímenes fijados, la microscopía brinda ahora información cuantitativa sobre procesos biológicos con mínima perturbación [1]. En particular, con el advenimiento de las sondas fluorescentes, el estudio del citoesqueleto (microtúbulos, filamentos intermedios y filamentos de actina, Fig. 1.1) en células vivas cobró impulso ya que fue posible visualizar los filamentos en su entorno natural de una forma muy poco invasiva [2].

Estas técnicas de microscopía se basan en un fenómeno fotofísico conocido como fluorescencia. La fluorescencia es la propiedad que poseen ciertas moléculas, conocidas como fluoróforos, de emitir luz al ser excitadas con luz de una determinada longitud de onda [3]. Un microscopio de fluorescencia está equipado con fuentes de alta intensidad que emiten luz en el espectro visible. Utilizando filtros específicos se selecciona luz con la longitud de onda correspondiente a la de excitación del fluoróforo, y se la hace incidir sobre la muestra marcada con la sonda fluorescente. La fluorescencia emitida por la muestra es colectada por el objetivo y luego atraviesa un filtro de emisión que permite seleccionar el rangopectral de los fotones que serán colectados [4].

Existen distintos tipos de microscopía de fluorescencia. La más básica se conoce como *wide field* o microscopía de campo amplio en la que se ilumina una región amplia de la muestra y se colecta fluorescencia de las moléculas en el volumen excitado. Por otra parte, en un microscopio confocal se aumenta el contraste rechazando la luz que proviene de planos fuera de foco. Esto se logra agregando una apertura confocal o pinhole al camino óptico previo a la colección de la fluorescencia emitida [5].

En estas microscopías las imágenes se encuentran limitadas por difracción, que es el fenómeno que se presenta cuando la luz atraviesa una ranura pequeña. El patrón de difracción de una fuente puntual puede aproximarse a una forma gaussiana con un ancho finito. Por lo tanto, existe un límite en la resolución óptica que está dado por la configuración de las lentes del microscopio y la longitud de onda de la luz emitida. Para longitudes de onda en el espectro visible, la resolución es del orden de los 200 nm [4]. Aunque este valor resulta despreciable para la visualización de células enteras o grandes especímenes, puede ser una limitación cuando se estudian estructuras intracelulares, tales como los filamentos que componen el citoesqueleto (Fig. 1.1) cuyos diámetros característicos están por debajo del límite de difracción. Es por ello que el desarrollo de rutinas que permitan recuperar las coordenadas de estas estructuras con precisión subpixel a partir de las imágenes son fundamentales para el estudio cuantitativo de la dinámica y mecánica del citoesqueleto, entre otros.

En la próxima sección describimos los enfoques más utilizados para abordar este problema y los antecedentes al desarrollo propuesto en el presente trabajo.

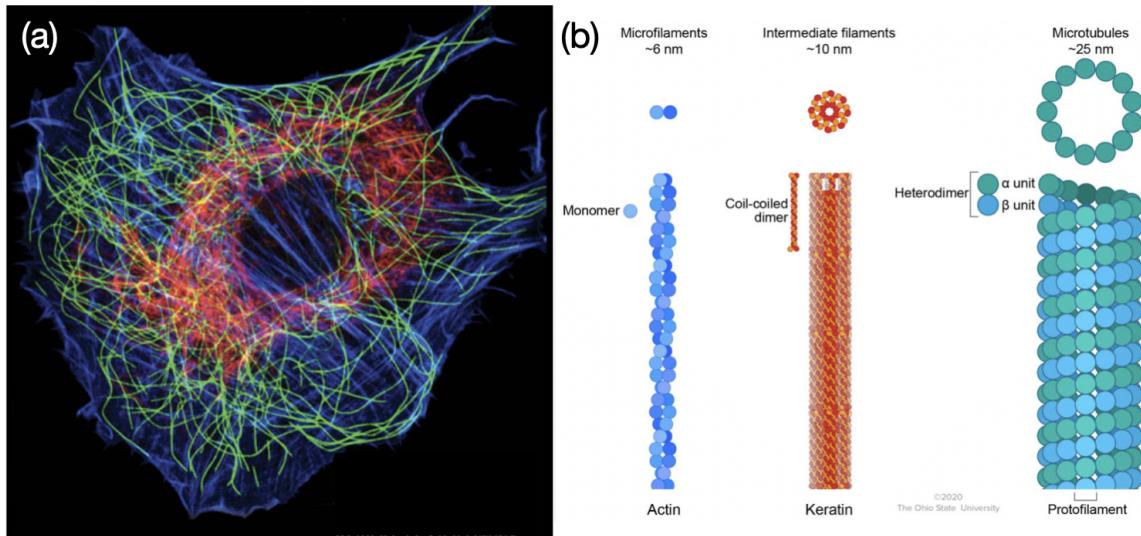


FIGURA 1.1: Filamentos del citoesqueleto. (a) Imagen confocal de una célula eucariota en la que observan las redes del citoesqueleto: microtúbulos (verde), filamentos intermedios (rojo) y filamentos de actina (azul) (adaptada de *The Cytoskeleton*, Pollard y Goldman, 2016). (b) Representación esquemática de los biopolímeros del citoesqueleto con sus dimensiones características (adaptado de *Veterinary Histology*, Jennings y Premanandan, 2020).

## 2. Estado del arte

### 2.1. Algoritmos de tracking de biofilamentos

Existen numerosos algoritmos para el seguimiento (tracking) con precisión nanométrica de partículas individuales de geometría tipo-esférica. Sin embargo, hacer el seguimiento de filamentos que se desplazan y transforman a lo largo del tiempo es un trabajo para el cual existen menos estrategias. Una forma de abordar a este problema es la presentada en Gittes et al. (1993) [6] donde se eligen manualmente 10-15 puntos pertenecientes al filamento fluorescente en cada cuadro de la película, y se realiza una interpolación lineal para obtener así una aproximación de la forma del mismo. Los problemas que surgen con este método son que la precisión no es sub-pixel y que requiere que el usuario ingrese una gran cantidad de datos. Para realizar el proceso de seguimiento de una forma más automatizada y con mayor precisión, existen tres enfoques diferentes que se discuten brevemente a continuación.

#### 2.1.1. Umbralización

Este enfoque busca, previo a realizar el análisis de las imágenes, umbralizar las mismas para intentar delimitar el contorno del filamento a trackear. Esto consiste, de manera simplificada, en definir un valor umbral de intensidad y a aquellos píxeles con intensidades por encima y por debajo del umbral asignales el valor 1 y 0, respectivamente. Una vez obtenida la máscara, se realizan diferentes procedimientos.

Brangwynne y colaboradores (2007) [7], por ejemplo, realizan una “esqueletización” de la máscara, que consiste en erosionar de forma tal que se reduzcan la cantidad de píxeles que representan

al filamento pero sin perder la conectividad de los mismos. Los puntos obtenidos son ajustados polinómicamente para obtener la forma del filamento.

El procedimiento propuesto por Ruhnnow et al. (2011) [8] está basado en pasos similares a los propuestos en [7]. Consiste nuevamente en umbralizar y esqueletizar la imagen. Luego se realiza una segmentación según el tipo de figura encontrada analizando si es un filamento corto, una partícula puntual, el extremo, centro o intersección de filamentos.

Esta técnica presenta varios inconvenientes. En primer lugar, dada la naturaleza de las imágenes con las que se trabaja y que pueden presentar altos niveles de ruido, nada garantiza que umbralizar facilite el seguimiento en lugar de complejizarlo aún más. En segundo lugar, este proceso se realiza en la totalidad de la imagen, lo cual tiene un costo computacional y agrega latencia a la respuesta. Al trabajar con secuencias de un gran número de imágenes, este costo escalará linealmente con la cantidad, lo cual perjudica a esta alternativa.

### 2.1.2. Contornos activos

El método de contornos activos es un algoritmo utilizado para poder seguir de manera rápida y precisa una figura a lo largo de una secuencia de imágenes [9]. En este algoritmo se debe considerar al filamento a seguir como una unidad.

Por ejemplo, Valdman et al. (2012) [10] considera la forma completa del filamento como un contorno expandido en una base ortogonal de polinomios. Este enfoque le brinda una mayor tolerancia a los cambios en intensidad de la fluorescencia. Sin embargo, esto requiere un bajo nivel de ruido en la imagen.

Otro ejemplo basado en contornos activos es el plugin JFilament para FIJI. Este es un paquete de procesamiento de imágenes de código abierto basado en ImageJ2 [11, 12].

Si bien esta técnica pareciera ser ideal para el problema, las imágenes con las que se trabaja presentan graves dificultades para el método. En primer lugar, debido a que los filamentos se comportan casi como rectas, teniendo muy pocos píxeles de grosor en la mayoría de los casos, no se llegan a poder delimitar contornos y por lo tanto el algoritmo no puede llevarse a cabo satisfactoriamente. A su vez, este método presenta mejores resultados en entornos donde el objeto a seguir está claramente delimitado del fondo, lo cual no sucede en muchos de los casos a tratar.

### 2.1.3. Análisis de perfiles de intensidad

Esta técnica consiste en trazar perfiles de intensidad en la dirección transversal al filamento a lo largo del mismo. Considerando que la intensidad será máxima en las zonas centrales del filamento, se utilizan diferentes métodos para determinar las coordenadas de estos puntos centrales.

Por ejemplo, el método propuesto por Kas et al. (1996) [13] consiste en delimitar el filamento con un punto en cada extremo y un punto en el centro para comenzar el análisis. Luego se traza un perfil de intensidad en torno al punto a analizar y se hace un ajuste cuadrático. En base a la pendiente de este ajuste se puede determinar la dirección perpendicular al filamento. Esta dirección luego es utilizada para obtener el siguiente punto de análisis.

El método propuesto por Brangwynne (2007) [7] previamente mencionado, también utiliza esta técnica. Una vez terminado el proceso de ajuste, se refinan las aproximaciones tomando perfiles de intensidad perpendiculares y aplicando una deconvolución gaussiana.

El procedimiento de Janson et al. (2004) [14] está basado en microscopía DIC. Estas imágenes presentan un gradiente de grises perpendicular al filamento. Las mismas son recorridas desde el extremo izquierdo del filamento (marcado por el usuario) en forma horizontal. Se realiza un análisis vertical de la intensidad, se aplica una convolución y se determina el centro del filamento. Este proceso tiene la desventaja de que requiere que el filamento sea relativamente recto.

Por ultimo, el algoritmo AFTER [15] adopta este enfoque, pero a diferencia de los trabajos mencionados, utiliza redes neuronales para interpolar la posición en la que se encuentra el máximo de intensidad de cada perfil. La principal ventaja de esta técnica es que permite realizar un análisis sobre cada punto individual del filamento teniendo en cuenta su entorno y los valores recuperados pueden tener precisión sub-pixel. Dado que este trabajo utiliza a AFTER como punto de partida, a continuación describimos brevemente este algoritmo.

## 2.2. AFTER

AFTER (Automated Filament Tracking and Evaluation Routine) es una herramienta implementada en MATLAB que permite realizar el seguimiento de filamentos a lo largo de múltiples imágenes de forma semi-automática. Fue desarrollada en el Laboratorio de Dinámica Intracelular de la Facultad de Ciencias Exactas y Naturales, UBA durante la Tesis Doctoral de la Dra. Pallavicini y publicado en 2014 [15]. Mediante una interfaz gráfica se cargan las imágenes, se seleccionan múltiples puntos a lo largo del filamento y se realiza el seguimiento. La implementación es semi-automática ya que necesita que el usuario seleccione múltiples puntos del filamento que se desea analizar en el primer fotograma de la secuencia. Estos puntos serán los únicos datos ingresados por el usuario, aparte de algunos parámetros que permiten personalizar el algoritmo según el tamaño del píxel y el tiempo entre cuadros.

El primer paso del procedimiento consiste en la rotación de la imagen de forma tal que los puntos que se desean analizar describan una curva similar a una recta horizontal. Luego, estos puntos son interpolados para definir la forma inicial del filamento. A partir de entonces, comienza un proceso iterativo. Se toma un perfil de intensidad perpendicular al filamento (en la dirección vertical, gracias a la rotación), se ajustan los puntos obtenidos con una red neuronal GRNN (General Regression Neural Network), se determina el punto de mayor intensidad, y se actualiza la posición del filamento en ese fotograma. El propósito de utilizar una red neuronal es aumentar la robustez de la determinación del centro del filamento frente a ruido inhomogéneo. Al igual que un ajuste gaussiano del perfil de intensidad, la red neuronal permite detectar los máximos con precisión sub-pixel, y posee mayor robustez ante ruido o filamentos entrecruzados. La curva formada por los máximos encontrados será luego suavizada (otra vez mediante la red neuronal mencionada), suponiendo que los filamentos no poseen curvas afiladas.

A pesar de su capacidad para la recuperación de filamentos en imágenes de microscopía de fluorescencia, este algoritmo presenta algunas desventajas. La principal es que está implementada

en un lenguaje propietario, lo que reduce el espectro de posibles usuarios de la herramienta. Por otro lado, la rotación de la imagen puede generar errores de pixelación y tratar con las imágenes rotadas puede llegar a ser complejo para el usuario. Además, el mismo requiere de un procesamiento de rotación inversa para reconstruir la forma original del filamento. A su vez, la rutina está implementada como un único script de MATLAB que debe ser ejecutado en un entorno adecuado, aumentando el conocimiento técnico requerido por el usuario para utilizar esta herramienta. Por último, dada las limitaciones que presenta MATLAB a la hora de diseñar e implementar interfaces gráficas, la misma no se caracteriza por la simplicidad para el usuario. Por estos motivos, en un trabajo previo, estudiantes de proyecto final del ITBA implementaron en 2021 una aplicación web con el fin de subsanar estos obstáculos.

### 2.3. Trabajo final previo

La solución planteada en el proyecto previo [16] se compone por una aplicación web desarrollada en Flask (framework web de Python), que se despliega mediante un contenedor Docker, y que aprovecha librerías disponibles en el lenguaje, como Numpy, OpenCV y Scikit-Learn, todas herramientas open-source. En la figura 2.1 se puede visualizar esta arquitectura.

Mediante un navegador, el usuario puede subir al servidor los frames de la película del filamento a analizar en forma de imágenes de formato png, jpeg, tiff, archivos tiff y multi-tiff. Posteriormente debe seleccionar el segmento del filamento a trackear y, por último, visualizar y descargar las imágenes con la forma del filamento marcada.

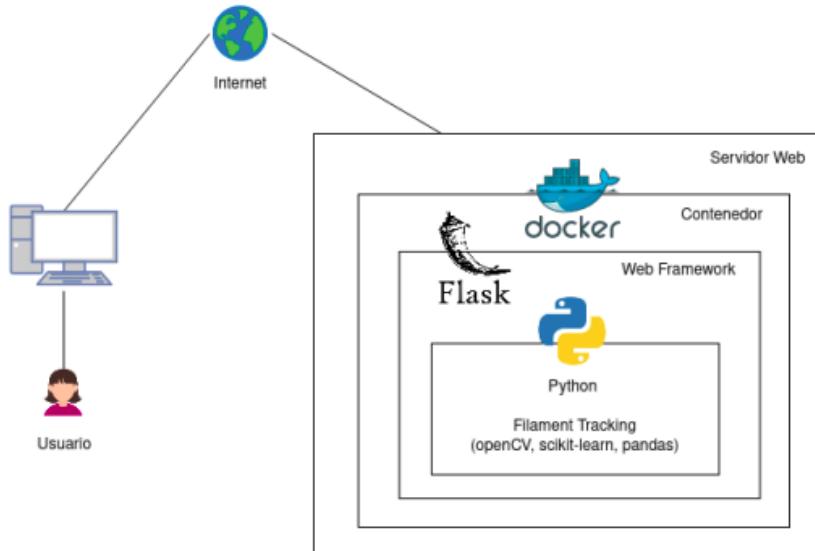


FIGURA 2.1: Arquitectura proyecto final existente

Algunos detalles importantes de esta implementación son la estrategia seguida para la identificación inicial de los filamentos, la heurística utilizada para el ajuste de los puntos a lo largo del tiempo y el manejo de las imágenes.

Antes de realizar cualquier tipo de análisis sobre las imágenes, éstas son pre-procesadas me-

diente distintas rutinas. Primero, son truncadas a 8 bits de profundidad de color. Luego, se les aplica un filtro de eliminación de ruido basado en la mediana. Por último, se umbraliza la imagen utilizando la técnica de *adaptative threshold* utilizando la media ofrecida por openCV [17], lo cual da como resultado una imagen binaria.

Para la selección inicial de filamentos esta aplicación utiliza una técnica llamada *raycasting* que consiste en encontrar un camino que une 2 puntos, uno de partida y uno de llegada. Para ello, ambos puntos deben formar parte de un figura que los contenga, en particular un filamento. Estos puntos son seleccionados por el usuario realizando click sobre la imagen. El algoritmo consiste en un proceso iterativo donde se toma el último punto aproximado y se proyectan una serie de rayos que representan las direcciones candidatas para elegir el próximo punto. Una vez generadas las posibles direcciones se las evalúa en busca de la mejor opción. Para determinar la alternativa a seleccionar se analiza que tan centrada está la aproximación respecto a los bordes del filamento. Al evaluar las opciones no se toma el mínimo global (mejor aproximación) sino los mínimos locales, lo que permite contemplar bifurcaciones. Esto se puede apreciar en la imagen superior derecha de la figura 2.2.

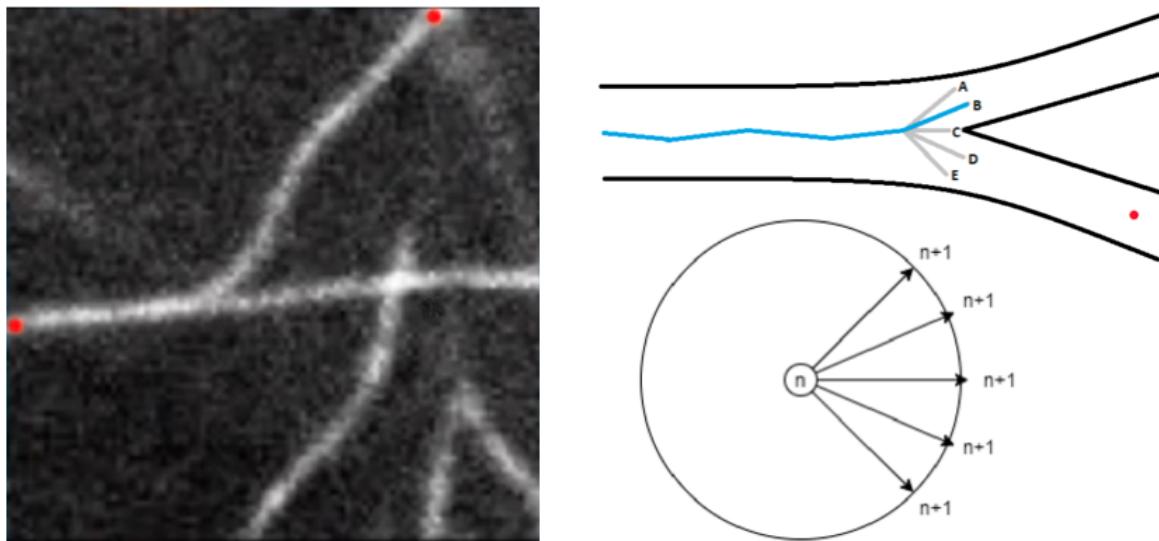


FIGURA 2.2: Figuras esquemáticas del proceso de *raycasting*. A la izquierda ejemplo en uso donde se seleccionan los extremos. A la derecha dos visualizaciones del paso a paso, estrategia de proyección de rayos. Imágenes obtenidas del Trabajo: Análisis de imágenes de microscopía de fluorescencia para segmentación y tracking de filamentos celulares [16]

Este procedimiento es utilizado en la primera iteración para generar una aproximación inicial del filamento. Luego, este debe ser seguido a lo largo del tiempo. Para ello, se trazan perfiles de intensidad normales al filamento y se ajusta el punto en función de su entorno. A cada uno de estos puntos  $p$  pertenecientes al filamento se les asigna un puntaje  $P(p) = \frac{3}{4} \frac{\text{Brillo}(p)}{255} + \frac{1}{4} (1 - \frac{\text{Distancia}(p, p_{ant})}{(L/2)})$  en función de su valor actual  $p$  y en la iteración anterior  $p_{ant}$ .

La aplicación desarrollada presenta avances positivos respecto de AFTER, como ser la implementación web. Sin embargo, existen varios aspectos que requieren mejoras y que llevaron a la realización de este trabajo. En las próximas secciones se describe en detalle el relevamiento realizado de la aplicación existente y las mejoras introducidas.

### 3. Extensión y alcance del proyecto

El principal objetivo del proyecto es proveer una herramienta que permita realizar el seguimiento (tracking) de filamentos intracelulares a partir de imágenes de microscopía de fluorescencia. Se buscará que el proceso de seguimiento pueda ser llevado a cabo de manera eficiente y replicable.

El seguimiento de filamentos intracelulares cuenta con ciertas limitaciones y complejidades específicas. Entre ellas, destacamos la baja relación señal-ruido que estas imágenes pueden presentar, los posibles solapamientos y cruces entre distintos filamentos, las grandes curvaturas y los rápidos desplazamientos que se observan habitualmente en las películas.

Es por ello que hay algunos requisitos que las secuencias de imágenes deben cumplir para el éxito de la aplicación. En primer lugar, se analizarán películas donde los movimientos de los filamentos a analizar no sean demasiado bruscos, es decir, películas de filamentos con movimientos de alta velocidad deberán contar con una cantidad de cuadros por segundo que muestren ese movimiento suavemente y sin saltos demasiado pronunciados. En segundo lugar, si bien las imágenes pueden contar con una baja señal-ruido, será necesario que el filamento sea distinguible del fondo durante toda la película o fracción de la misma que se deseé analizar. Por último, el filamento a analizar no deberá yuxtaponérse con otro para evitar errores de identificación. Un detalle importante es que, en las imágenes de filamentos en su entorno celular, los filamentos no están aislados, por lo que sólo se podrán recuperar segmentos de los mismos. Esta limitación no es exclusiva de este trabajo, sino de todas las rutinas mencionadas en el capítulo anterior.

Por otro lado, se buscará obtener una interfaz que presente las funcionalidades disponibles de manera clara y práctica, ya que esta aplicación será utilizada como una herramienta de trabajo por investigadores para los que la recuperación de las formas de los filamentos representa el primer paso para un análisis cuantitativo posterior, y no un fin en sí mismo.

Tomaremos como punto de partida el proyecto final mencionado [16], y se realizarán las modificaciones que se consideren pertinentes para poder alcanzar estos objetivos y construir la mejor herramienta posible. Sin embargo, no se considerará como limitación al desarrollo mantener el algoritmo ni la interfaz del proyecto en caso de no poder adaptarse a los objetivos deseados.

## 4. Solución

### 4.1. Requisitos Funcionales

Los requisitos funcionales representan todas las funcionalidades o acciones que se espera que puedan ser realizadas utilizando la herramienta. Al contar con futuros usuarios de la plataforma, se pudieron definir claramente aspectos como formatos de imagen soportados, tolerancia al ruido necesario y formas de presentar y obtener los resultados una vez finalizado el análisis. Durante el proceso de desarrollo de la aplicación se fueron discutiendo estos aspectos funcionales, necesarios para que la aplicación cumpliera con los casos de uso planteados por los investigadores del Grupo de Dinámica Intracelular. Entre estas funcionalidades se destacan:

- Soporte de los siguientes formatos de imagen de entrada: png, jpeg, tiff. En el caso de los archivos tiff, se deben soportar tanto secuencias de imágenes individuales como archivos multi-tiff.
- Obtención de las coordenadas del filamento con localización sub-píxel y descarga de las mismas en un formato de texto estructurado.
- Capacidad de tolerar zonas en las que momentáneamente no se pueda trackear el filamento mediante estrategias secundarias, modificables por configuración.

Uno de los principales requisitos es la implementación de una interfaz gráfica que permita interactuar con la rutina de seguimiento de manera sencilla, pudiendo aprovechar las funcionalidades brindadas sin agregar complejidad adicional a la hora de utilizarlas. A su vez, se definieron aspectos adicionales a la interfaz para simplificar el proceso de trabajo con el algoritmo. Los requisitos definidos finalmente fueron:

- Permitir subir las imágenes a utilizar desde la computadora del usuario.
- Visualizar el primer cuadro de la secuencia y utilizarlo de referencia para la selección del filamento a ser utilizado por la rutina y los parámetros de configuración.
- Poder deshacer y rehacer los puntos al realizar la selección.
- Permitir hacer zoom sobre la imagen al seleccionar los puntos, pudiéndose mover a través de la misma.
- Presentar un manual de uso de la aplicación.
- Presentar una descripción de los parámetros de configuración para simplificar el uso de los mismos y sugerir los valores por defecto.
- Previsualización de los resultados y configuración manual de los parámetros del tracking para adaptarse a los distintos tipos y calidades de imágenes.

- Capacidad de visualizar y descargar las imágenes resultantes de las imágenes originales con los filamentos recuperados en un color contrastante, pudiendo ver así el seguimiento realizado cuadro por cuadro.

#### 4.2. Requisitos No Funcionales

Además de los aspectos puntuales mencionados previamente, también se persiguieron otros objetivos relacionados con el funcionamiento y la usabilidad de la aplicación. La misma debe ser intuitiva pero robusta, pudiendo desplegarse en cualquier navegador y presentar tiempos de respuesta sensatos. Los principales requisitos tenidos en cuenta fueron:

- Precisión de la herramienta. Se debe minimizar la distancia entre las posiciones reales de los filamentos, y las obtenidas por el algoritmo desarrollado.
- Tiempos de respuesta bajos que permitan que el usuario trabaje de manera dinámica con la herramienta.
- Obtención de resultados consistentes al utilizar configuraciones y predicciones similares.
- Presentación de las funcionalidades de manera simple y compacta, evitando que el usuario deba moverse entre pantallas, dejando todas las funcionalidades a su alcance.
- Priorizar la claridad a la hora de presentar los resultados tanto parciales como finales al usuario.
- La aplicación debe funcionar en los navegadores Chrome, Safari y Firefox sin importar el sistema operativo.
- La aplicación debe ser de público acceso, sin requerir de licencias para acceder o hostear.
- La aplicación debe minimizar la cantidad y complejidad de tareas necesarias para mantener el correcto funcionamiento.

#### 4.3. Relevamiento de la aplicación existente y mejoras implementadas

Si bien en el proyecto anterior [16] se desarrolló un algoritmo de seguimiento basado en *raycasting*, analizando los resultados obtenidos se llegó a la conclusión que el mismo no permite obtener la consistencia y los tiempos de respuesta deseados. La principal desventaja es en el análisis del primer cuadro. En el informe de dicho proyecto ya se hace mención a algunos problemas de consistencia. Al explorar la herramienta encontramos tanto el error mencionado en el trabajo como algunos otros. Consideramos que estos problemas están estrechamente relacionados con la manera en la que se obtienen los puntos iniciales en el algoritmo y el pre-procesamiento que se realiza a las imágenes.

El primer problema encontrado es al umbralizar las imágenes a analizar. Debido al ruido que estas presentan y la manera en que se umbralizan, se da lugar a que secciones del fondo sean detectadas como filamentos.

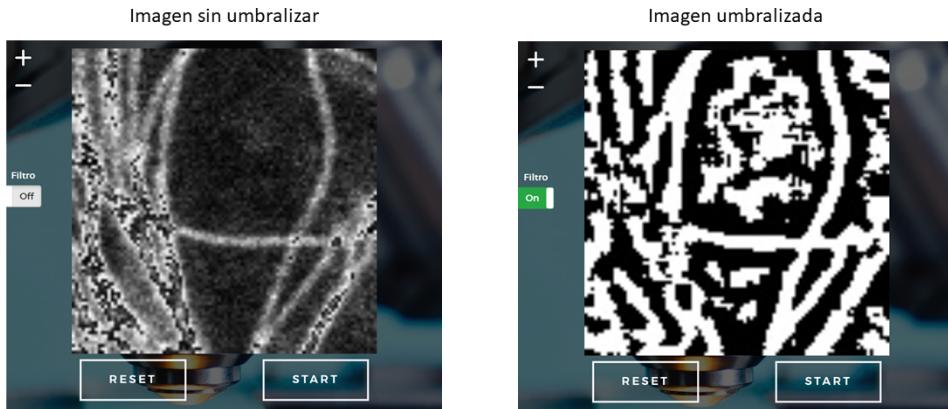


FIGURA 4.1: Exploración de la aplicación presentada [16]. Proceso de umbralización sobre la imagen en la cual se seleccionarán los puntos sobre el filamento que se desea analizar. Obsérvese la región central superior a la que el algoritmo asigna valores de intensidad compatibles con los filamentos.

Como puede comprobarse en 4.1, al aplicar la umbralización, puntos del fondo terminan quedando en blanco. Luego, la técnica de *raycasting* podrá confundir estos píxeles blancos como pertenecientes a un filamento a analizar (ver figura 4.2).

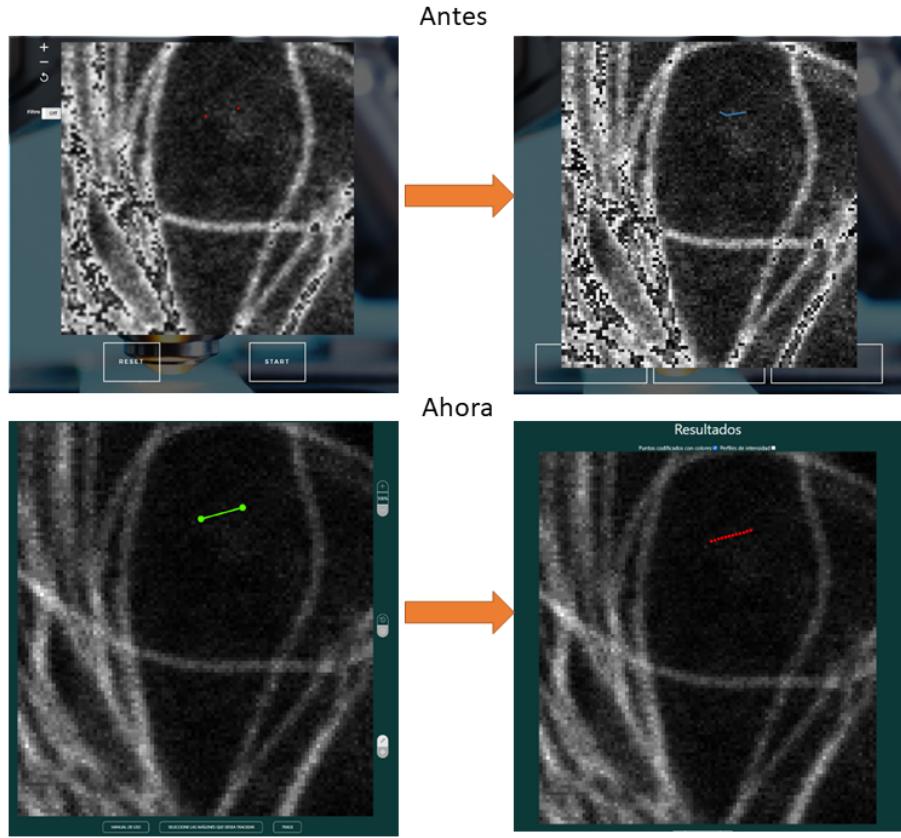


FIGURA 4.2: Selección de dos puntos sobre el fondo. Paneles superiores: el funcionamiento del proyecto previo donde pueden verse los puntos seleccionados (en rojo, izquierda) y el “filamento” recuperado (en celeste, derecha). Paneles inferiores: el mismo caso utilizando la nueva aplicación donde puede verse que la rutina categoriza esos puntos como incorrectos (en rojo).

Como puede apreciarse en 4.2 previamente era posible seleccionar puntos en el fondo y recibir con esos valores de entrada un resultado válido. Actualmente, si bien pueden seleccionarse como valores iniciales dos puntos que representan el fondo (puntos verdes), se mostrará claramente que los valores obtenidos son incorrectos y no representan ningún filamento (puntos rojos). A diferencia de la técnica de *raycasting*, el nuevo algoritmo no detecta los píxeles correspondientes a ruido como un filamento válido. Esto ocurre principalmente por el análisis de intensidad realizado. Si bien los píxeles del fondo presentan, en la mayoría de casos, valores de intensidades representados por una distribución aleatoria no nula, esto no será suficiente para que el nuevo algoritmo los detecte como puntos válidos. En la sección 4.4 se explicará el mismo en detalle.

Otro inconveniente que se encontró en el pre-procesamiento de las imágenes que realizaba el trabajo previo fue que en la mayoría de los casos las imágenes se mostraban con valores de intensidad saturados, aunque las originales no lo estuvieran, causando errores durante el procesamiento del tracking y dificultando la selección de puntos por parte del usuario. Esto se debe principalmente a que, como se explico previamente, las imágenes subidas se truncaban a 8 bits, aún cuando la representación original era de más bits (por ejemplo, imágenes TIFF de 16 bits, o imágenes a color). Esto se ilustra en la figura 4.3: la imagen de la izquierda muestra los colores

totalmente saturados, incluso llegando a mostrar píxeles negros en zonas claras de la imagen. Esto se solucionó utilizando una normalización lineal para pasar del rango de intensidades originales a  $\{0, 255\}$ , es decir, 8 bits. La fórmula de dicha normalización es:  $f(x) = (x - \min) \frac{255}{\max - \min}$ , donde  $\max$  es el valor máximo de intensidad de la imagen original, y  $\min$  el mínimo. Para el caso de imágenes a color, primero se la transforma al formato RGB de 3 canales (ignorando alpha), y luego se la reduce a una imagen de un solo canal tomando el promedio de los 3. Por supuesto, luego de este pre-procesado la imagen no es igual a la original, pero sí se garantiza que los valores relativos de los píxeles se conserven, y por lo tanto, las propiedades que se buscan recuperar.

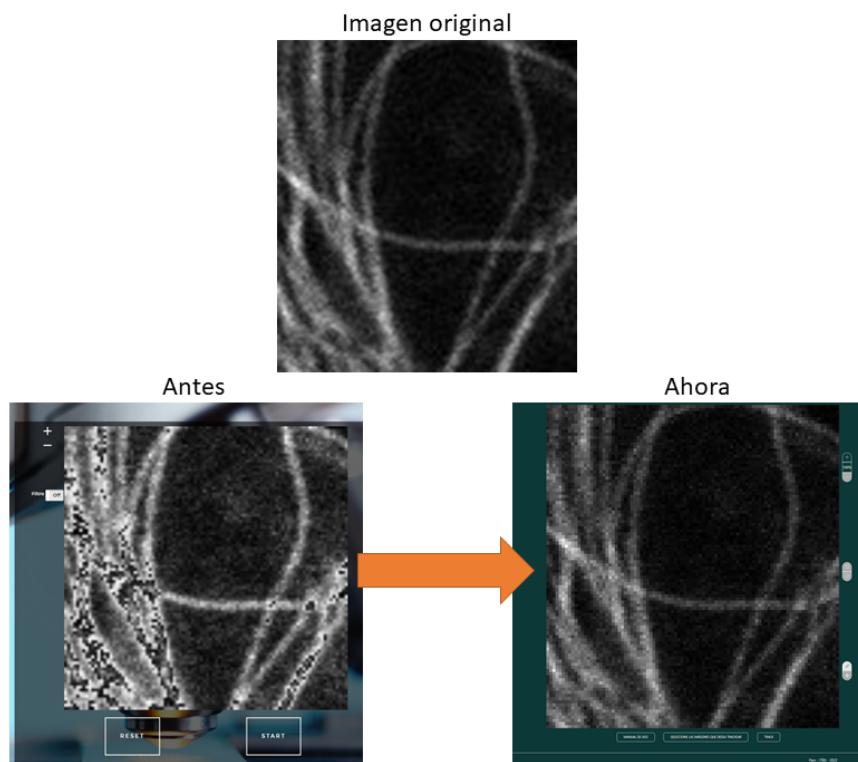


FIGURA 4.3: Imagen original, junto con capturas de la misma al ser seleccionada en el proyecto previo y en el actual.

Por estas razones se decidió desarrollar un nuevo algoritmo, tomando un enfoque más cercano al que se utiliza en el algoritmo AFTER, basado en el análisis de perfiles de intensidad. Teniendo en cuenta los tres enfoques considerados en la sección 2, se considera que esta alternativa es la que mejor se adapta al problema a tratar. En primer lugar, trabajar con los puntos dentro del filamento en base a su entorno permite un análisis más preciso de las coordenadas del mismo. En segundo lugar, como se mencionó previamente, la difracción gaussiana producida por el sistema de captura de las imágenes genera perfiles de intensidad transversales al filamento con estas características.

A su vez, se tuvieron que realizar cambios acerca de la forma en la que se seleccionaban los puntos sobre la primera imagen de la secuencia en relación al trabajo anterior. Esto se debe a que en dicho trabajo se asumía que las imágenes a analizar siempre serían cuadradas. En caso se imágenes rectangulares se presentaba al usuario la imagen distorsionada de forma tal que el

alto y el ancho coincidieran. Como consecuencia de esta distorsión, las coordenadas de los puntos seleccionados por el usuario sobre esta primera imagen modificada ingresaban al algoritmo en posiciones incorrectas, dando lugar a artificios en el tracking. Además, esta distorsión resulta en una visualización de los puntos sustancialmente diferente a la que el algoritmo utiliza para trabajar. Este podría ser uno de los factores por los cuales la salida de la aplicación resultaba, en ocasiones, inconsistente o incorrecta. Un ejemplo de este artefacto se muestra en la figura 4.4: los puntos seleccionados sobre la imagen finalmente no serán los que utiliza el algoritmo. Además, en el último cuadro, puede verse que la imagen resultado no se grafica con la distorsión presente en la selección de puntos y, como consecuencia, se sale de los límites de la aplicación, tapando parte de la interfaz.

Este problema fue resuelto al re-diseñar la interfaz, utilizando la API de Canvas [18] para re-escalar la imagen de manera que alcance los 1080 píxeles de ancho, manteniendo la relación de aspecto en cuanto a la altura. Así, la imagen se puede graficar en cualquier tamaño deseado (que va a depender del tamaño de pantalla utilizado) a una alta resolución y sin distorsionar o suavizar la imagen. Por supuesto, este re-escalado es tomado en cuenta durante la selección de puntos para garantizar que la misma sea correcta. Dado que la selección de los puntos iniciales es una parte vital del algoritmo, se priorizó que esta pueda realizarse de la forma más precisa posible.

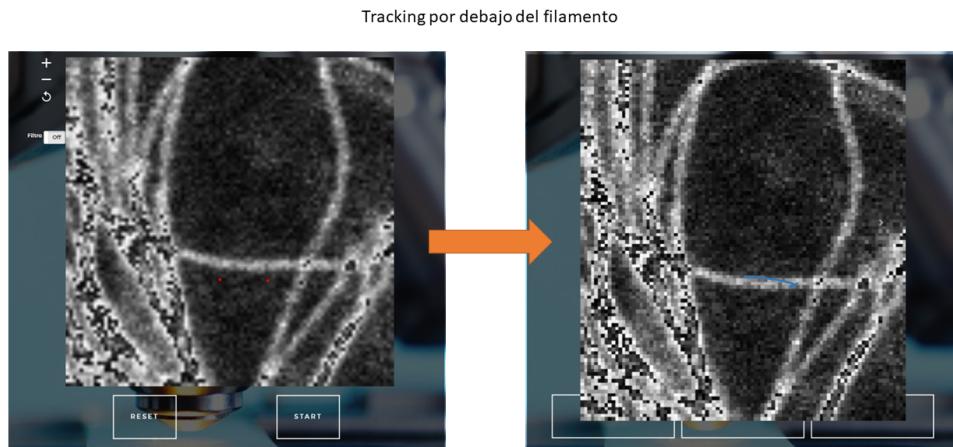


FIGURA 4.4: Selección de dos puntos por debajo del filamento y su resultado en el trabajo anterior. A la izquierda, los puntos seleccionados (en color rojo) por debajo del filamento. A la derecha el resultado mostrado sobre el filamento. Esto muestra como los puntos seleccionados difieren de los puntos analizados por el algoritmo.

Otro problema detectado es la falta de soporte para secuencias de imágenes que se encuentran invertidas, es decir, donde el filamento a trackear es negro y el fondo es blanco. Se decidió que la manera más simple y efectiva de resolver el problema es agregar un parámetro a la aplicación para que el usuario indique los casos donde la secuencia a analizar se encuentre invertida. En caso de estarlo, las imágenes serán invertidas antes de ser procesadas por el algoritmo.

#### 4.4. Backend: Algoritmo de tracking

El primer paso en la construcción de la aplicación fue desarrollar el algoritmo para realizar el seguimiento del filamento seleccionado por el usuario a través de la secuencia de imágenes recibida. Para la definición de este algoritmo, como se mencionó previamente, se parte de la hipótesis de que los movimientos de los filamentos a seguir no son bruscos, es decir, que de una imagen a otra su posición no se ve alterada en gran medida, y que las formas de los filamentos están bien representadas por curvas suaves y continuas.

El método propuesto consiste en calcular la posición del centro del filamento a lo largo de una serie de fotogramas. Para esto, usando como valor inicial la posición recuperada del filamento en el cuadro anterior, se realiza un análisis sobre el entorno y se obtiene la nueva posición. En el caso de la primera iteración esta aproximación inicial sera provista por el usuario. En los demás casos, se utiliza el valor obtenido en la iteración previa.

En cada iteración del algoritmo se comienza tomando los puntos iniciales, que en la primera iteración del algoritmo se corresponden con los puntos obtenidos a partir de una interpolación lineal sobre los puntos seleccionados por el usuario, y que en todas las iteraciones posteriores son los puntos obtenidos en la iteración previa. Luego, para cada uno de estos puntos, se determina la recta en la dirección normal a la tangente del filamento en dicho punto. De esta recta normal se toma un segmento cuyo largo podrá ser configurado por el usuario. Los valores de intensidad de los píxeles ubicados a lo largo de este segmento definirán un **perfil de intensidad**. A continuación, considerando que la difracción de la luz generada en el proceso de captura tiene una distribución similar gaussiana, se realiza el ajuste de dicho perfil de intensidad a una función de este tipo, se obtiene la posición del máximo de la gaussiana ajustada y se la asocia a la coordenada central del filamento. Por último, una vez realizado este procedimiento para todos los puntos a lo largo del filamento, se podrá optar por aplicar una función de suavizado basada en curvas de Bézier.

A diferencia del proyecto anterior, en el cual se umbralizan los *frames*, y del algoritmo AFTER, en el que se rotan las imágenes; este algoritmo no realiza transformaciones sobre las imágenes.

La figura 4.5 representa esquemáticamente los principales pasos que realiza el algoritmo para la recuperación de las coordenadas centrales del filamento y que se describen en más detalle en las secciones siguientes.

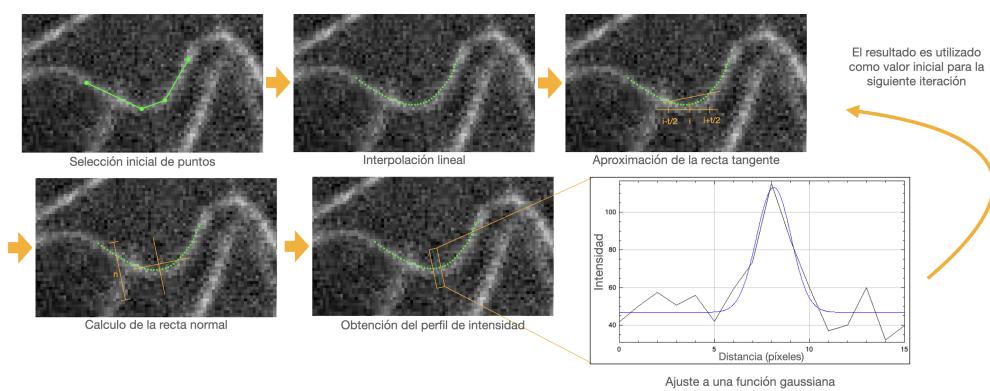


FIGURA 4.5: Figura esquemática con los principales pasos del backend

#### 4.4.1. Primera iteración: determinación aproximada de la forma del filamento

El algoritmo consiste en un proceso iterativo donde se ajustan los resultados del fotograma anterior a las modificaciones en el entorno del fotograma siguiente. Para ello es necesario una aproximación inicial de los puntos que conforman el filamento. Teniendo en cuenta que el filamento que se desea analizar no es conocido y es provisto por el usuario, es necesario que éste pueda seleccionarlo con el menor esfuerzo posible, sin necesidad de seleccionar todos los píxeles que lo conforman. Para ello se separa este proceso en 2 etapas. En la primera se le solicita al usuario que seleccione aquellos puntos necesarios para describir la forma aproximada del filamento. Luego, estos puntos son interpolados linealmente para generar la totalidad de puntos que representan el filamento. Cabe destacar que la resolución de los resultados obtenidos está sujeta a la cantidad de puntos que se generen al interpolar la aproximación inicial provista por el usuario, pues la cantidad de puntos analizados resulta invariante iteración a iteración. El caso ideal es cuando se tiene un punto por cada píxel que conforma el filamento.

Teniendo en cuenta que se está trabajando con una grilla discreta de píxeles, esta interpolación es realizada utilizando el Algoritmo de Bresenham [19]. Este algoritmo permite transformar rectas ideales en una línea discreta sin anti-aliasing y utilizando aritmética de números enteros exclusivamente. Esto lo caracteriza como un método rápido de generación de líneas sin comprometer el resultado. En particular, el algoritmo consiste en recorrer todos los puntos que conforman el segmento de la dimensión que mas varía. Luego, para cada uno de estos valores se calcula un único valor de la dimensión contraria de forma tal que el centro del píxel seleccionado sea el más cercano al punto ideal en la recta.

En la figura 4.6 se pueden observar distintos escenarios de selección. La cantidad de puntos necesarios dependerá de la curvatura del filamento.

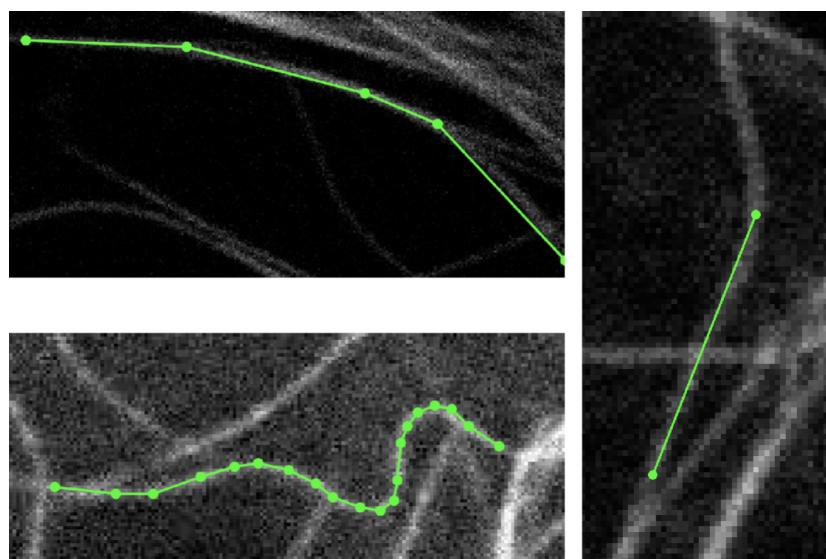


FIGURA 4.6: Distintos escenarios de selección de filamentos durante el primer paso del algoritmo. Los puntos verdes representan las selecciones realizadas por el usuario y los segmentos que los unen muestran la interpolación lineal utilizada para definir las posiciones de los puntos intermedios.

Si bien la resolución del análisis se ve favorecida por una mayor cantidad de puntos, esto tiene un impacto directo en el costo computacional del análisis. Es por esto que puede ser de interés tener un balance entre la precisión y el tiempo, sobre todo al trabajar con imágenes de alta resolución. Para ello, se define la densidad de puntos a interpolar  $d$ , la cual representa la distancia en píxeles que hay entre cada punto. El mismo es un número natural cuyo mínimo es 1, que es el caso donde se coloca un punto por cada píxel. El valor  $d$  es un parámetro del algoritmo y que podrá ser modificado por el usuario, tal como se verá más adelante.

Una vez realizada la primera interpolación, comienza el proceso iterativo que es primero aplicado sobre los puntos interpolados y repetido posteriormente en los fotogramas restantes, utilizando en cada caso la aproximación final del fotograma anterior.

#### 4.4.2. Ajuste de los perfiles de intensidad en la dirección normal al filamento

El primer paso del proceso iterativo consiste en obtener los perfiles de intensidad normales al filamento en cada uno de los puntos pertenecientes al mismo. Para ello, primero se calcula la recta tangencial al filamento en cada uno de los puntos. Luego, a partir de esta recta se podrá determinar la recta normal al punto. Finalmente, se utilizará la recta normal para obtener el perfil de intensidad en cuestión. Los pasos indicados a continuación son realizados de forma individual por cada uno de los puntos que conforman el filamento en cada una de las iteraciones.

Para el cálculo de la dirección tangente al filamento se hace un análisis del entorno del punto. En particular, se toman los extremos de un entorno de longitud  $t$  centrado en el punto y se calcula la pendiente entre los mismos. Si  $p^i$  es el  $i$ -ésimo punto del filamento y  $p^{i-\frac{t}{2}}, p^{i+\frac{t}{2}}$  son los extremos del segmento de longitud  $t$ , el ángulo que forma la tangente respecto al eje  $X$  se calcula como  $\theta = \text{atan2}(p_y^{i+\frac{t}{2}} - p_y^{i-\frac{t}{2}}, p_x^{i+\frac{t}{2}} - p_x^{i-\frac{t}{2}})$ . Por último, el ángulo de la normal (y por ende su dirección) podrá ser obtenido mediante la fórmula  $\theta + \frac{\pi}{2}$ . La longitud  $t$  expresada en cantidad de puntos es un parámetro del algoritmo. Este proceso se puede ver ilustrado en la figura 4.7.

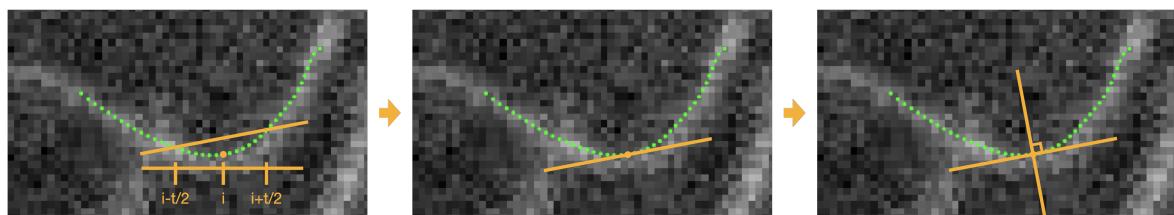


FIGURA 4.7: Cálculo de la recta tangente y normal a un punto del filamento. En la segunda imagen se visualiza el punto que se está analizando ( $i$ -ésimo) y la recta que definen los extremos del intervalo de longitud  $t$  centrado en el  $i$ -ésimo punto. En las últimas dos imágenes se observa, primero la aproximación final de la recta tangente junto a su posterior estimación de la recta normal.

En algunos casos no es posible tomar un punto a una distancia  $\frac{t}{2}$  ya que el filamento es muy corto o el punto está muy cerca de los extremos. En caso de que la longitud del filamento sea menor a  $t$ , se redefinirá el valor de  $t$  como la longitud del filamento. Por otro lado, en el caso de todos aquellos puntos que estén a una distancia menor a  $\frac{t}{2}$  puntos de alguno de los extremos, se utilizará la pendiente del punto ubicado a  $\frac{t}{2}$ , es decir, la pendiente del primer punto con información

suficiente para calcular su pendiente.

Una vez calculada la recta normal al punto en cuestión es necesario obtener el perfil de intensidad. Este representa la intensidad de los píxeles que se encuentran en el segmento perteneciente a la recta normal de longitud  $n$  y centrado en el punto, donde  $n$  es un parámetro del algoritmo expresado en píxeles. Las intensidades de estos píxeles serán utilizados para realizar el ajuste de la función gaussiana.

Para definir los puntos pertenecientes al segmento normal basta con calcular los puntos asociados a los extremos y realizar una interpolación lineal de los mismos, nuevamente utilizando el algoritmo de Bresenham [19]. El cálculo de los extremos es realizado mediante una proyección a partir del punto del filamento en la dirección del ángulo de la recta normal a una distancia de  $\frac{n}{2}$ .

Una vez obtenidos los puntos que conforman el segmento normal se usan las coordenadas de los mismos para obtener la intensidad de los píxeles que representan, obteniendo así el perfil de intensidad buscado.

#### 4.4.3. Obtención de la coordenadas del centro del filamento: ajuste gaussiano del perfil de intensidad

Tal como se menciona en la introducción 1, el fenómeno de difracción genera patrones tipo gaussiano asociados a cada fuente puntual de luz. Si consideramos que un filamento fluorescente del citoesqueleto tiene asociado una gran cantidad de fluorósforos muy pequeños y que cada molécula produce un patrón de difracción tipo gaussiano, es de esperar que los perfiles de intensidad transversales al filamento se caractericen por la presencia de una intensidad máxima en el centro del filamento y un decaimiento de la misma hacia los laterales. Teniendo esto en cuenta, se ajusta el perfil de intensidad obtenido a una función gaussiana (4.1) compuesta por 4 parámetros:

$$g(x) = y_0 + b e^{-\frac{(x-\mu)^2}{2\sigma^2}} \quad (4.1)$$

donde  $\mu$  y  $\sigma$  representan la media y el desvió estándar. Los parámetros  $y_0$  y  $b$  brindan mayor flexibilidad a la función y a los casos que pueden ser ajustados. Por un lado,  $y_0$  corresponde a la intensidad basal debida al ruido. Finalmente,  $b$  es el factor de escalado de la función. Permite que la altura de la función se ajuste a las diferencias de intensidades. Ambos parámetros resultan cruciales para la toma de resultados, pero específicamente en entornos muy ruidosos. De estos 4 parámetros, es de particular interés el valor de  $\mu$ , ya que el mismo representa el valor medio de la función gaussiana y, en este caso, la posición central del filamento. Mientras,  $\sigma$  representa un valor proporcional al ancho del filamento.

Para el ajuste de los perfiles de intensidad se proponen los siguientes valores iniciales:  $\mu = x_{max}$ ,  $\sigma = 1$ ,  $b = b_{max}$ ,  $y_0 = b_{max}/4$ . Se define  $b_{max}$  como la intensidad máxima medida en el perfil que se esta analizando y  $x_{max}$  como el valor que cumple  $g(x_{max}) = b_{max}$ .

Es posible que algunos perfiles no se puedan ajustar a la función propuesta o que el ajuste logrado no sea bueno. Se requiere que el error del ajuste esté por debajo de un valor máximo  $E_g$  tolerado, que es parámetro del algoritmo. En caso de no encontrarse por debajo de esta cota, el punto es marcado como inválido e interpolado a partir de los puntos de su entorno. El error del

ajuste es calculado como el desvió estándar de la estimación de  $\mu$ .

#### 4.4.4. Reposición de coordenadas longitudinales por interpolación

Si bien el objetivo es determinar el centro del filamento utilizando el ajuste gaussiano, este puede no ser suficientemente bueno y contar con un error mayor al límite establecido. Si se descartasen estos puntos se perdería información lentamente y se generaría discontinuidades en el filamento afectando las aproximaciones siguientes.

Para solucionar este problema se completarán los lugares vacíos. Este proceso se realiza una vez ajustados todos los perfiles de intensidad ya que es una operación global. Consiste en llenar las discontinuidades utilizando una interpolación de los extremos. Para este procedimiento se toma como supuesto que la discontinuidad se producirá en una recta. Esta solución no contempla cómo suplir las fallas en situaciones más complejas, pero funciona de manera aceptable en los casos estudiados, tal como se mostrará en la sección de Resultados 5.

Para completar los lugares vacíos, primero se identifican los puntos, o segmentos de puntos contiguos, que no fueron aproximados exitosamente. Como consecuencia, éstos estarán delimitados por uno o más puntos que fueron ajustados de forma exitosa. Se define una cantidad de puntos  $n_i$ , parámetro del algoritmo. Luego, para completar cada uno de estos segmentos, se tomarán los primeros  $n_i$  puntos aproximados exitosamente de cada lado, con un mínimo de 1 punto por cada extremo. Luego, se contará con un conjunto de puntos que fueron ajustados correctamente y representan la posición del filamento. Estos serán ajustados con una regresión lineal para aproximar la pendiente del filamento en esa región. Finalmente, se generan tantos puntos como se hayan descartado y se los distribuye de forma equiespaciada a lo largo del segmento.

Es posible que los puntos que no se hayan podido ajustar se encuentren en uno de los extremos del filamento. De ser así, no sería posible encontrar puntos aproximados de forma correcta en uno de sus lados y, como consecuencia, no se podría realizar la regresión lineal. En estos casos, se conservan los puntos calculados en la iteración anterior. Si bien no se realiza un ajuste de estos puntos al último cuadro, se entiende que la diferencia de posición del filamento entre cuadro y cuadro es baja. Esta estrategia es aceptable, pero no particularmente buena, por lo que queda pendiente para una futura mejora encontrar una mejor opción.

#### 4.4.5. Suavizado de la forma recuperada

Los resultados obtenidos hasta aquí corresponden a una primera aproximación a las coordenadas del filamento analizado. Sin embargo, en algunos casos la forma obtenida presenta una cierta rugosidad. Esto se debe a que la posición de cada punto fue obtenida mediante el ajuste previo de su perfil de intensidad transversal, que no tiene en cuenta la posición de los puntos vecinos sobre el filamento. Esto produce pequeñas variaciones de la posición en la dirección perpendicular al filamento, tal como puede observarse en la imagen A de la figura 4.8.

Una alternativa deseable consiste en suavizar los resultados obtenidos. El objetivo del suavizado es que los puntos describan una curva suave. Una forma de lograr este comportamiento es interpolar los mismos a una función de estas características. Las funciones, por definición, no

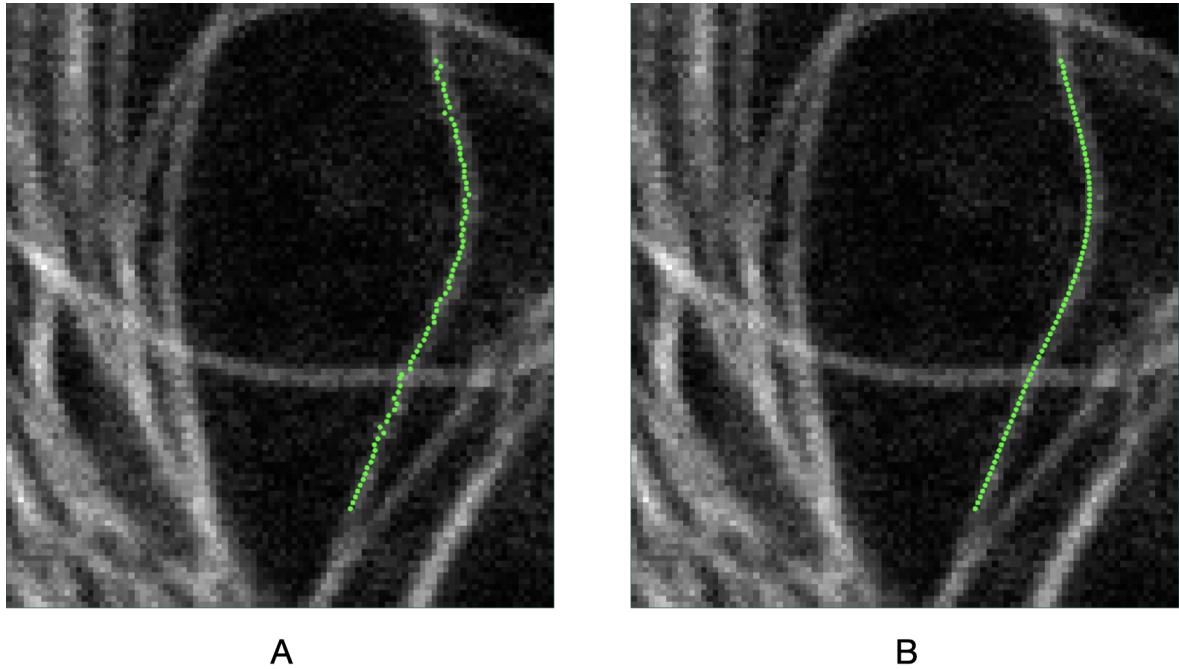


FIGURA 4.8: Imagen A: Resultado crudo del algoritmo al final de una iteración. Imagen B: Resultado del mismo tracking incorporando un suavizado basado en curvas de Bézier a los resultados.

permiten que exista más de un resultado asociado a un valor de entrada. Esto limita la posibilidad de utilizar las componentes de una imagen como ejes de nuestra función objetivo. Una única función impide describir figuras circulares o curvas que vuelvan sobre sí mismas. Se puede ver en la figura 4.9 un caso donde el filamento analizado no es una función del tipo  $f : \text{ancho} \rightarrow \text{alto}$  o del tipo  $f : \text{alto} \rightarrow \text{ancho}$ . La otra estrategia posible consiste en aplicar funciones dependientes del entorno a los valores. Un ejemplo sería el promedio móvil. Esta operación permite atenuar el impacto de los puntos más alejados ajustando su valor en base al valor de sus vecinos. El problema de utilizar esta estrategia es determinar a qué valores aplicar la función. En el caso de la figura 4.8 sería de interés suavizar los valores en el eje vertical, pero no en el horizontal. En cambio, si se hubiese seleccionado el filamento transversal, se tendrían que usar los ejes opuestos para el suavizado. Si bien se podría hacer un análisis sobre los datos provistos para determinar la dirección del filamento y así definir en qué componentes aplicar el suavizado, esto sería más difícil o casi imposible en escenarios como la figura 4.9, es decir, en escenarios donde el filamento cambia su orientación a lo largo del seguimiento.

Para solucionar este problema y teniendo en cuenta las limitaciones de las opciones mencionadas, se optó por una estrategia basada en curvas paramétricas, definidas por la expresión 4.2, en particular de Bézier [20].

$$f(s) = (f_x(s), f_y(s)) \quad (4.2)$$

Estas estrategias permiten describir curvas en un espacio 2D utilizando un único parámetro  $s$ , sin estar limitadas a los ejes coordenados. En el caso de las curvas de Bézier, estas curvas son

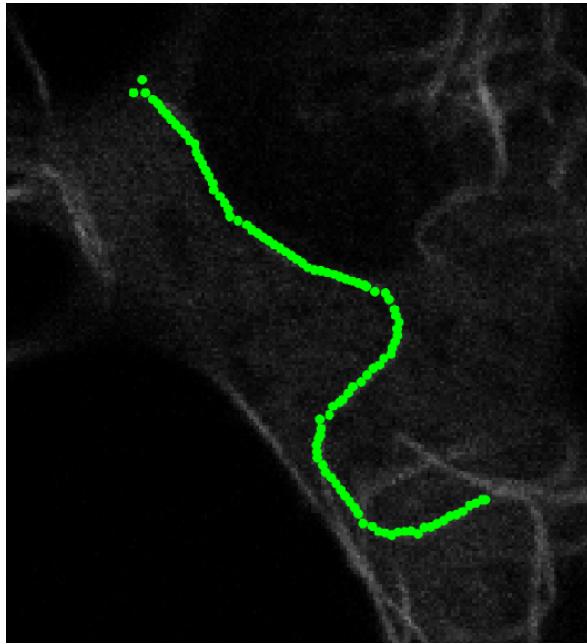


FIGURA 4.9: Resultados del tracking de un filamento sin realizar el suavizado.

definidas utilizando puntos de control. Estos representan puntos por los cuales la curva intentará pasar. Estos pueden estar distribuidos de cualquier forma en el espacio 2D. Además, el método de Bézier es de tipo global. Esto significa que todos los puntos que conforman la curva tienen efecto en la totalidad de la misma. Estas propiedades brindan mayores libertades a la hora de definir curvas arbitrarias en el espacio 2D. En la aplicación se incorporó una rutina de suavizado basada en curvas de Bézier (ver imagen B en figura 4.8). Esta es utilizada al final de cada iteración y puede ser activada de forma opcional.

La estrategia propuesta consiste en utilizar las coordenadas recuperadas hasta ahora como los puntos de control, y luego obtener los puntos de la curva suavizada aplicando directamente la definición por Polinomio de Bernstein [21]. Como cada punto de control tiene un efecto en todos los puntos de la curva, esto permite suavizar el resultado final de una manera significativa, aprovechando la información obtenida previamente. Una vez definida la curva, se muestran las coordenadas finales en intervalos equiespaciados para cada cuadro de la película. Esto se realiza para evitar que se formen zonas que posean distintas densidades de puntos a través del filamento.

Una consecuencia negativa de este procedimiento es que no permitiría recuperar cambios abruptos en la curvatura del filamento. Si un filamento se curvase mucho de un cuadro a otro, es posible que el suavizado evite que el resultado lo acompañe. Para contemplar estos casos, el algoritmo no se limita a tomar una sola curva de Bézier para todo el filamento. Sino, se parte el resultado en secciones de una misma longitud  $l_s$  (expresada en puntos) parámetro del algoritmo, las cuales son ajustadas individualmente a una curva de Bézier, y posteriormente unidas por sus extremos para formar la curva final. Esto es posible gracias a que los puntos extremos de cada intervalo no se ven modificados por el ajuste. Otro beneficio de segmentar la curva para aplicar el suavizado es que preserva el funcionamiento del procedimiento para números altos de puntos.

La principal desventaja de esta estrategia es que se pueden generar artefactos indeseados en los puntos de unión de los segmentos. Si bien se tendrá una curva continua, la unión puede presentar un cambio abrupto de dirección. Es posible evitar estos artefactos eligiendo una longitud de segmento de curva  $l_s$  mayor a la cantidad total de puntos del filamento seguido, en cuyo caso solo será una curva. Queda pendiente para una futura mejora el suavizado o eliminación de estos artefactos, algo perfectamente factible de conseguir mediante una segunda pasada de post-procesado.

#### 4.4.6. Complejidad temporal

Es interesante analizar, al menos superficialmente, la complejidad temporal del algoritmo propuesto. Si bien el mismo posee una gran cantidad de parámetros distintos, se considera que los que mayor impacto tienen en la complejidad temporal son: la cantidad de cuadros que componen la secuencia (*frames*); la cantidad de puntos del filamento analizados durante el tracking, (*points*) y la longitud de los perfiles de intensidad en píxeles (previamente definido como  $n$ ). Asumiendo que el método utilizado para el ajuste gaussiano de los perfiles de intensidad posee una complejidad temporal  $O(f(n))$ , y teniendo en cuenta que el algoritmo esta diseñado para trackear solamente una vez cada una de las imágenes, analizar una sola vez cada uno de los puntos del filamento de dicha imagen y ajustar una única vez el perfil de cada uno de dichos puntos, se concluye que la complejidad temporal del algoritmo es de  $O(\text{frames} * \text{points} * f(n))$ .

### 4.5. Backend: Implementación

#### 4.5.1. Implementación del algoritmo

Para la implementación del algoritmo se decidió mantener el lenguaje de programación Python. El mismo cuenta con muy buen soporte para el análisis de imágenes mediante paquetes externos. Esto lo vuelve simple y flexible sin sacrificar el rendimiento, ya que muchas rutinas de estos paquetes están implementadas en lenguajes conocidos por su alto rendimiento, como C++. Las versiones de todas las dependencias externas se detallan en el archivo `/requirements.txt` del proyecto.

El principal paquete externo utilizado es Numpy [22], que provee eficientes implementaciones de arreglos y matrices, junto con rutinas para manipularlos. El mismo fue utilizado a lo largo de toda la implementación. Por ejemplo, la representación interna de las imágenes a procesar son matrices de 8 bits de tipo `ndarray`, provisto por Numpy.

Otro paquete externo utilizado con frecuencia fue SciPy [23]. El mismo provee múltiples implementaciones de algoritmos fundamentales para el cómputo científico. Específicamente, se utilizan:

- `scipy.optimize.curve_fit` [24]: Rutina utilizada para calcular el ajuste gaussiano de los perfiles de intensidad para aproximar las posiciones de los puntos. La misma permite ajustar cualquier función real de 2 dimensiones mediante el método de cuadrados mínimos no lineales (también ofrecido por SciPy bajo el nombre `scipy.optimize.least_squares` [25]). El algoritmo de minimización elegido para que la rutina utilice fue el de Levenberg-Marquardt, que se identifica en SciPy mediante el nombre `lm`, y es la configuración por

defecto cuando no se imponen cotas, como es el caso.

- `scipy.stats.linregress` [26]: Esta rutina calcula la regresión lineal de cuadrados mínimos de un conjunto de puntos. Fue utilizada durante la rutina de reposición de puntos, para calcular la recta por la cual se deben interpolar los puntos faltantes.
- `scipy.special.comb` [27]: Implementación de la función de combinatoria estándar. Utilizada durante la rutina de suavizado para el cálculo de curvas de Bézier, específicamente los coeficientes del polinomio de Bernstein.

Se utilizó la rutina `skimage.draw.line` [28] provista por `scikit-image` [29] como implementación del Algoritmo de Bresenham [19].

Por último, se utilizó `Pillow` [30] para la lectura de los diferentes formatos de imágenes, y posterior transformación a matrices de tipo `ndarray`.

#### 4.5.2. Arquitectura de la aplicación web

A grandes rasgos, se decidió preservar la arquitectura general de la aplicación propuesta por el trabajo anterior [16]. Es decir, Python como lenguaje de programación, `Flask` [31] como *framework* web, `Gunicorn` [32] como servidor web y `Docker` [33] como mecanismo de *deployment* estandarizado. El principal agregado es un *proxy* reverso `Nginx` [34], el cual permite manejar la carga de manera eficiente y es ampliamente configurable.

Por otro lado, se decidió exponer el algoritmo de 3 maneras distintas, para mayor flexibilidad a la hora de utilizarlo. Las mismas son: como aplicación web completa, como *endpoint* HTTP-REST público y como librería de Python. Hay que tener en cuenta que la única opción que ofrece un entorno amigable para la selección de puntos es la aplicación web completa.

Dentro del paquete `tracking` se encuentra toda la implementación del algoritmo de tracking, disponible para ser embebido en otra aplicación Python como una dependencia. Este paquete se encuentra totalmente aislado de cualquier lógica web. Posee un único punto de entrada, la función `track_filament` dentro del módulo `tracking.main`, que recibe la secuencia de imágenes, los puntos provistos por el usuario y la configuración de parámetros con la cual correr el algoritmo. En el módulo `tracking.models` se declara la estructura de configuración que el algoritmo recibe, y las estructuras en las que se retorna la respuesta, es decir, las coordenadas de los filamentos junto con *metadata* adicional. En el módulo `tracking.image_utils` se exponen rutinas que ayudan en el pre-procesado de las imágenes, antes de que sean analizadas por el algoritmo principal. Se consideró importante facilitar el acceso directo a la implementación del algoritmo, sin la necesidad de interactuar con un servidor web, para casos donde se desee evitar la latencia introducida por internet.

Por otro lado, en el módulo `main` ubicado en el *root* de la aplicación, se encuentra la lógica que aprovecha `Flask` para configurar el servidor web y declarar los *endpoints* del mismo. En particular, el *endpoint* `/track` es el que expone la lógica del algoritmo, recibiendo los puntos seleccionados, la secuencia de imágenes y la configuración del algoritmo en formato `multipart/form-data`, y retornando las posiciones del filamento (junto con *metadata* adicional) en formato `application/json`.

El resto de los *endpoints* (/ y /manual) retornan las distintas páginas de la aplicación web, diseñadas para ser consumidas mediante un navegador. Es importante aclarar que la página web es simplemente un consumidor más del *endpoint* principal que ofrece acceso al algoritmo, por lo que es perfectamente factible para terceros implementar interfaces alternativas a la página web presentada, por ejemplo, una herramienta de línea de comandos.

Por último, mediante Docker se declara cómo la aplicación debe ser desplegada. El objetivo es que esta configuración sea genérica para cualquier servidor Linux, y no sea necesario modificarla dependiendo del *deployment*. Bajo este mecanismo se declaran las configuraciones del servidor Gunicorn en */Dockerfile*, de la generación de certificados HTTPS utilizando el script */generate\_certificate.sh* y del proxy reverso Nginx en la carpeta */nginx/*, que luego son compuestos en */docker\_compose.yml*.

Uno de los requisitos no funcionales de la aplicación es la minimización del mantenimiento que la misma requiere. Para ello se decidió que la aplicación no aumente su huella de almacenamiento a lo largo de su vida útil. Es por esto que las imágenes utilizadas durante la ejecución de la rutina son enviadas en cada pedido, y únicamente almacenadas de forma temporal por Flask durante el ciclo de vida del pedido HTTP. Esta decisión junto a la no utilización de logs permiten a la aplicación operar con un capacidad de disco fija y sin necesidad de realizar limpiezas programadas.

Al momento de la presentación del trabajo, existen dos *deployments* de la aplicación corriendo: uno en servidores de la UBA, accesible en <https://fernet.exp.dc.uba.ar>, y y otro en servidores del ITBA, accesible en <https://pf-pipo.it.itba.edu.ar>. Ambos fueron desplegados de forma equivalente, mediante Docker.

#### 4.6. Frontend: Interfaz web

El segundo aspecto más importante del trabajo fue el rediseño de la interfaz visual. Este proceso consistió en modificar la distribución de la aplicación, corregir errores de la implementación anterior e incorporar nuevas funcionalidades que mejoren la experiencia de usuario.

El principal factor que se tuvo en cuenta a la hora de rediseñar la aplicación fue que la misma iba a ser utilizada por un grupo reducido de personas, por tiempo prolongado. Por ello, se buscó un diseño que fuera simple de usar, pero altamente funcional.

El proceso de trabajo con la aplicación está compuesto por 2 etapas claramente marcadas: la selección de imágenes y la configuración del seguimiento junto con la visualización de los resultados. Se buscó que el nuevo diseño presentara todas sus funcionalidades en una misma página, para poder simplificar al máximo el uso de la aplicación.

Dentro de la primer etapa es necesario que el usuario pueda realizar las siguientes operaciones:

- Seleccionar las imágenes con las que se desea trabajar.
- Visualizar el primer fotograma de la secuencia de imágenes y seleccionar sobre el mismo los puntos del segmento de filamento que se desea analizar.
- Definir los valores de los parámetros del algoritmo.

- Deshacer las selecciones incorrectas.
- Hacer zoom en cualquier sección de la imagen.

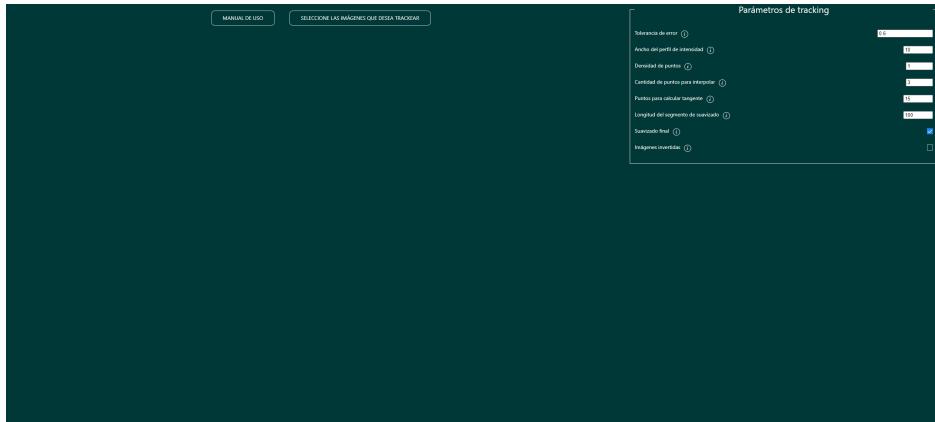


FIGURA 4.10: Captura de pantalla de la interfaz en su estado inicial. Se observan el botón que permiten acceder al manual de usuario y el botón para la selección de las imágenes. A la derecha se observa el formulario con los parámetros a ser utilizados en el seguimiento.

El primer paso al utilizar la aplicación es seleccionar las imágenes. Se puede comprobar en la figura 4.10 su posición central en la interfaz. El foco esta puesto en que esta sea la primer opción con la que se encuentra el usuario. Los formatos de imagen soportados son: tiff, jpeg y png. Los archivos tiff pueden ser tanto una secuencia de archivos como un único archivo con toda la secuencia, también conocido como multi tiff.

Una vez seleccionadas las imágenes se mostrará el primer fotograma de las mismas. Sobre esta imagen se seleccionarán los puntos que aproximan el filamento. En particular, solo los puntos que permitan definir la curva que lo representa, el resto de puntos será generado por interpolación tal como fue explicado en la sección 4.4.1. Para que el posterior proceso de interpolación sea comprensible se generan rectas que unen los puntos conforme son seleccionados para facilitar al usuario la visualización de la forma del filamento que será analizada por el algoritmo. Estos segmentos pueden observarse en la figura 4.11.

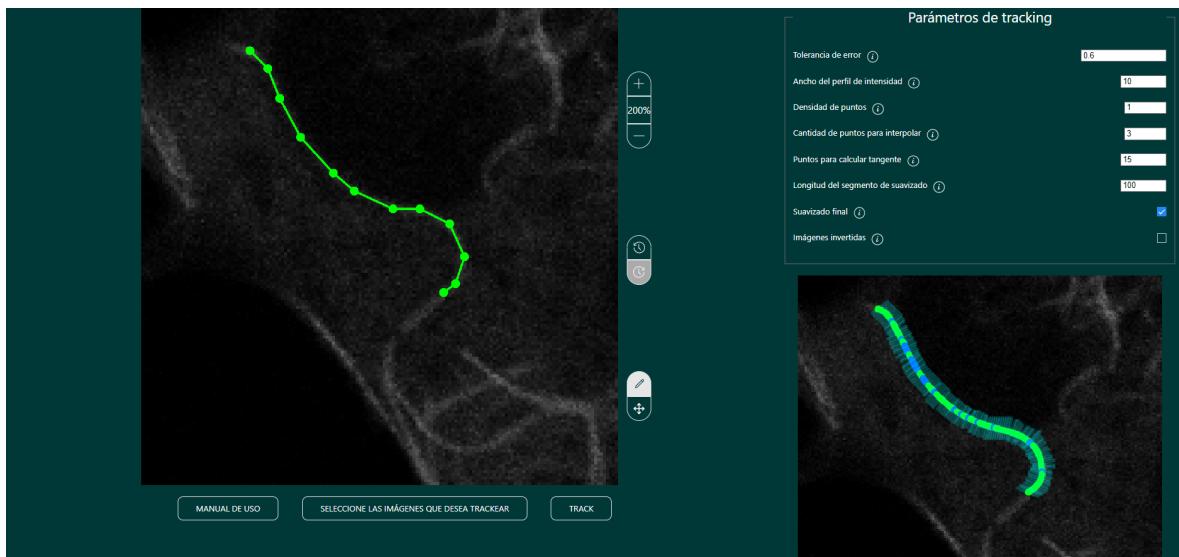


FIGURA 4.11: Captura de pantalla de la interfaz luego de haber seleccionado una imagen y un conjunto de puntos. La imagen principal representa el lienzo sobre el cual el usuario marcará los puntos necesarios para ejecutar el algoritmo. A la derecha de esta imagen se encuentran 3 unidades con funcionalidades para facilitar el trabajo con la imagen. La primera unidad es utilizada para el manejo de zoom. Permite acercar y alejar la imagen. La segunda unidad permite deshacer y redo los puntos ya dibujados para no tener que reiniciar la operación. La última permite alternar entre dibujar y mover la imagen (para las imágenes ampliadas). A la derecha se observa el tablero de parámetros y, debajo de este, el resultado de aplicar el algoritmo en forma de previsualización.

Si bien con las herramientas de selección ya mencionadas es posible marcar los puntos necesarios, se cuenta con 3 funcionalidades que buscarán simplificar este proceso. En primer lugar, se brinda la posibilidad de magnificar una región de la imagen mediante el botón para hacer zoom, ya sea porque se está trabajando con imágenes de alta resolución o porque el usuario desea seleccionar los puntos con mayor claridad. En segundo lugar, se cuenta con la posibilidad de deshacer y redondear selecciones mediante los botones indicados con un reloj. Esto permite corregir los errores de selección sin tener que volver a empezar. Por último, se cuenta con una previsualización de la selección donde se muestra el resultado de ejecutar la rutina de seguimiento sobre el primer fotograma. En caso de que se esté haciendo zoom, esta ampliación de la imagen se verá reflejada.

Además de seleccionar las imágenes, otro aspecto importante es seleccionar los parámetros del algoritmo presentados durante la sección 4.4. La interfaz cuenta con un formulario que permite modificar cada uno de forma individual. El mismo se puede observar en la figura 4.11 arriba a la derecha. Los valores que se pueden modificar son:

- La tolerancia del error  $E_g$ .
- El ancho del perfil de intensidad  $n$ .
- La densidad de puntos  $d$ .
- La cantidad de puntos para interpolar  $n_i$ .
- Cantidad de puntos tomados para calcular la recta tangente  $t$ .

- Longitud del segmento de suavizado  $l_s$ .
- Utilización de suavizado de Bézier.
- Inversión de la secuencia de imágenes previo al seguimiento.

La modificación de los parámetros puede tener un impacto importante sobre el resultado del seguimiento. Cada conjunto de imágenes puede requerir distintos valores para estos parámetros. La previsualización resulta altamente valiosa dado que para ver los resultados obtenidos con la configuración elegida se hubiese tenido que esperar a que se ejecute el algoritmo sobre toda la secuencia, lo que podría llegar a tomar una cantidad considerable de tiempo. De esta forma, se podrá comprobar si la configuración seleccionada brinda resultados similares a los esperados, evitando analizar la secuencia completa y permitiendo modificar los parámetros elegidos, recibiendo los nuevos resultados en un menor tiempo.

Una vez definida una configuración aceptable se ejecutará el algoritmo. Esto enviará las imágenes y los puntos seleccionados al servidor para ser procesados. Una vez obtenidos los resultados, estos serán mostrados al usuario utilizando un visualizador.

El visualizador permitirá ver cuadro por cuadro o en forma animada la secuencia de resultados, pudiendo controlar la velocidad de la animación, detenerla y volver a empezar. Esta visualización puede ser configurada haciendo uso de dos *checkbox*. Estos permiten activar y desactivar visualizaciones que funcionan a modo de información complementaria, la cual es de gran utilidad cuando se intenta razonar sobre posibles problemas en el seguimiento. Estas opciones incluyen la posibilidad de dibujar los perfiles de intensidad para verificar su correcta orientación y longitud, así como utilizar un esquema de codificación por colores para los puntos. Los colores representan la forma en la que fueron determinados esos puntos. De esta manera, el color verde corresponde a puntos bien determinados mediante ajuste gaussiano, el color azul se refiere a puntos interpolados y el color rojo señala puntos que no pudieron ser interpolados y se reemplazaron por el punto de la iteración anterior. Este código de colores permite al usuario tener una idea cualitativa de la calidad del tracking y facilita la búsqueda de parámetros adecuados. En la previsualización se grafican tanto los perfiles de intensidad como los puntos con este esquema de colores.

Además de visualizar los resultados en la página, se puede descargar un archivo zip que contiene todos los cuadros del resultado en formato png. En caso de no necesitar las imágenes y querer analizar los resultados numéricos del análisis, se pueden descargar archivos tipo json y tsv con la posición de todos los puntos a lo largo de todos los cuadros de la secuencia.

## 5. Resultados

### 5.1. Validación

#### 5.1.1. Metodología

Para el análisis de precisión y rendimiento del algoritmo, se generaron imágenes sintéticas de filamentos que emulan las características de las tomadas mediante microscopía de fluorescencia, para luego comparar el error entre la posición ideal del filamento con la retornada por el algoritmo, a medida que se reduce el Signal to Noise Ratio (SNR).

La generación de las imágenes sintéticas se realizó mediante el siguiente procedimiento:

1. Se comienza con una imagen de 8 bits monocromática toda en 0 (negra) de alto `height` y ancho `width` en píxeles ( $\text{width} > \text{height}$ ).
2. A partir de una función  $f : \{0, \dots, \text{width}\} \rightarrow \{0, \dots, \text{height}\}$  y un `thickness` natural positivo, se asigna a cada píxel  $(x, f(x) + \text{offset})$  una intensidad igual a `max_value`, donde  $\text{offset} \in \{-\text{thickness}, \dots, \text{thickness}\}$  e ignorando los resultados negativos.
3. Se suaviza el filamento mediante una convolución gaussiana de  $\sigma = 10$  y un kernel de 3x3.
4. Se agrega ruido gaussiano aditivo caracterizado por un  $\sigma$  a cada píxel de la imagen con una probabilidad del 85 %.
5. Por último, se normaliza linealmente el resultado al rango [0, 255] para que sea una imagen de 8 bits válida.

A menos que se aclare lo contrario, los parámetros para la generación de las imágenes fueron:

- `height` = 96
- `width` = 166
- `thickness` = 3
- `max_value` = 150

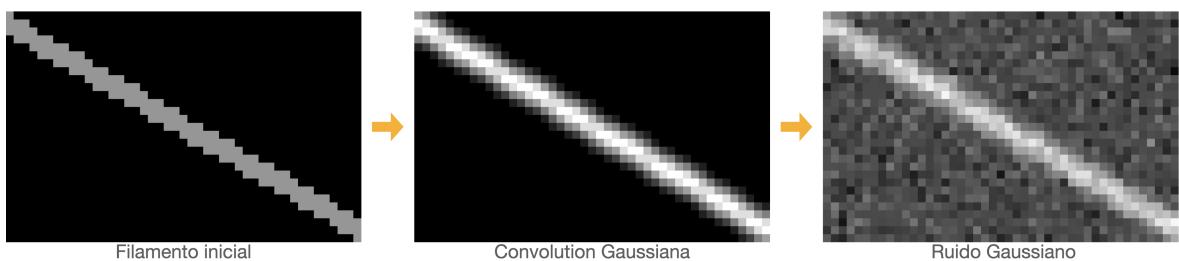


FIGURA 5.1: Pasos intermedios del procedimiento de generación de imágenes sintéticas. Se utiliza una función lineal como ejemplo.

El cálculo del SNR se realizó definiendo un entorno rectangular de fondo en la imagen sintética, luego restando la media de la intensidad del filamento (es decir la señal, cuyas posiciones fueron previamente computadas durante la generación de la imagen) con la media de la intensidad del fondo y, por último, dividiéndolo por la desviación estándar de la intensidad del fondo.

La métrica de error utilizada es el Root Mean Square Error (RMSE), cuya unidad es en píxeles.

Por cada  $\sigma_i \in \{0,0001, 0,0002, \dots, 0,0060\}$ , se obtiene un SNR y un RMSE asociado, es decir 60 puntos, los cuales luego serán graficados. El procedimiento es el siguiente:

1. Se genera la imagen sintética a partir de la función  $f$ .
2. Se ejecuta el algoritmo de tracking y se recuperan las posiciones de los puntos resultantes.  
Los puntos que serían seleccionados por el usuario se obtendrán mediante  $(x, f(x))$ , donde  $x \in \{\text{trim} + 0 \cdot \text{step}, \text{trim} + 1 \cdot \text{step}, \dots, \text{width} - \text{trim}\}$ ,  $\text{trim} = 20$  y  $\text{step} = 15$ . El rango de  $x$  se definió con un paso de 15 píxeles, alejado de los extremos unos 20 píxeles para evitar problemas de borde. Para la configuración general del algoritmo se utilizarán los siguientes valores a menos que se indique lo contrario:
  - `max_fitting_error = 0,6`
  - `normal_line_length = 10`
  - `point_density = 1`
  - `missing_inter_len = 3`
  - `max_tangent_length = 15`
  - `bezier_segment_len = 500`
  - `bezier_smoothing` habilitado
  - `inverted` deshabilitado
3. Se calcula el SNR de la imagen.
4. Se calcula el RMSE entre el resultado del algoritmo y  $f$ .
5. Se repite cada paso anterior 50 veces para obtener la media de los SNR, y la media y error estándar de los RMSE, los cuales serán utilizados para graficar.

### 5.1.2. Análisis de precisión y rendimiento

Se analizó la precisión y rendimiento del algoritmo para 3 casos distintos: un filamento lineal con pendiente, el caso más sencillo de trackear, y por tanto donde se esperan mejores resultados; el mismo filamento lineal, pero ahora en una imagen con el doble de resolución horizontal y vertical, para poder evaluar cómo el tamaño de la imagen modifica los resultados; y, por último, un filamento curvo, cuya complejidad es mayor a la del caso lineal. Tanto las imágenes generadas, como los parámetros y los resultados obtenidos, pueden ser encontrados en: <https://drive.google.com/drive/folders/13zcnyXbnx2uPWhtsS2Ypl-swtDMRoblj?usp=sharing>.

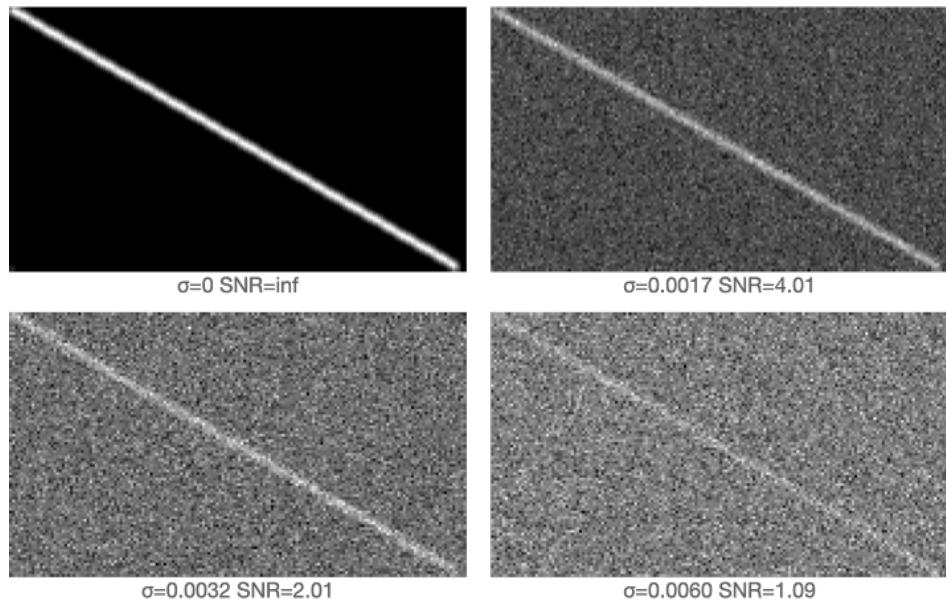


FIGURA 5.2: Ejemplos de imágenes sintéticas generadas de filamentos lineales.

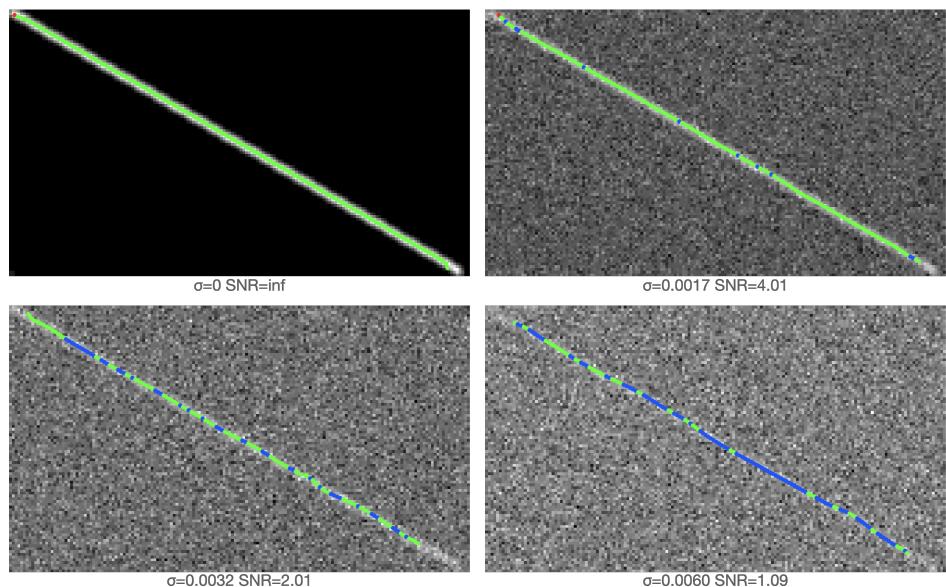


FIGURA 5.3: Resultado de trackear las imágenes de la figura 5.2 mediante la aplicación.

Para generar las imágenes del caso lineal, se definió la función  $f(x) = \lfloor \frac{\text{height}}{\text{width}} x \rfloor$ , es decir, un filamento cuya pendiente es la relación de aspecto de la imagen.

Para el caso lineal grande, los parámetros de generación de imágenes fueron los mismos que en el caso lineal, con la única diferencia de que las imágenes generadas poseen el doble de resolución de alto y ancho. Es decir, ahora  $\text{height} = 192$  y  $\text{width} = 332$ .

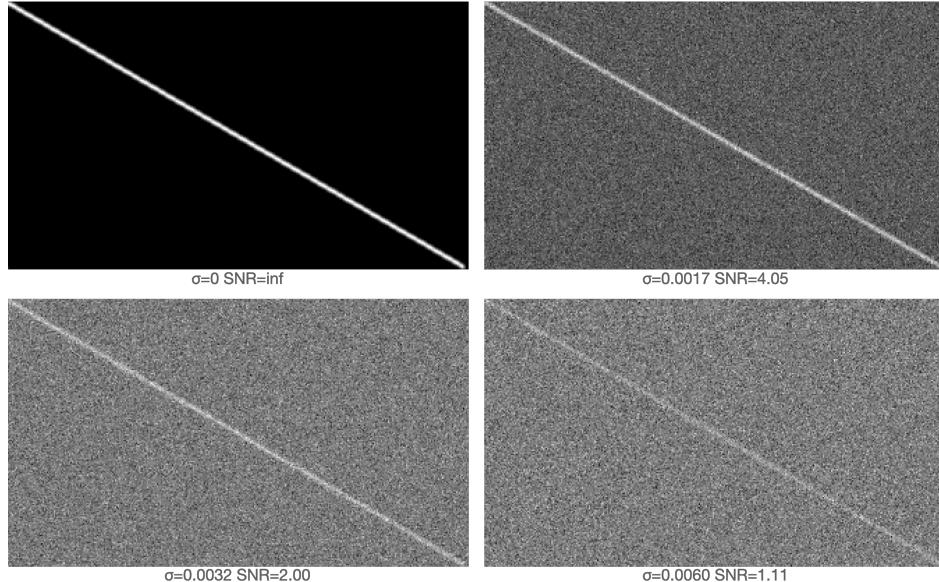


FIGURA 5.4: Ejemplos de imágenes sintéticas generadas de filamentos lineales de 4 veces la resolución.

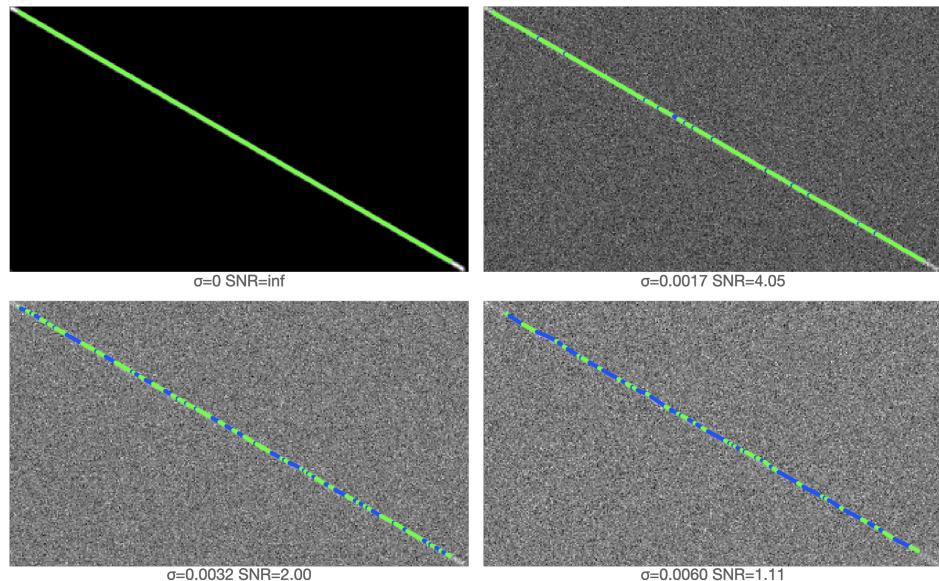


FIGURA 5.5: Resultado de trackear las imágenes de la figura 5.4 mediante la aplicación.

El último caso analizado es el de un filamento curvo. Para esto, se lo modela con una función sinusoidal. Se define la función:  $f(x) = \lfloor \frac{\text{height}}{4} \sin(\frac{8}{\text{width}} x - \frac{\pi}{4}) + \frac{\text{height}}{2} \rfloor$ , para generar una curva sinusoidal con un mínimo y un máximo centrados.

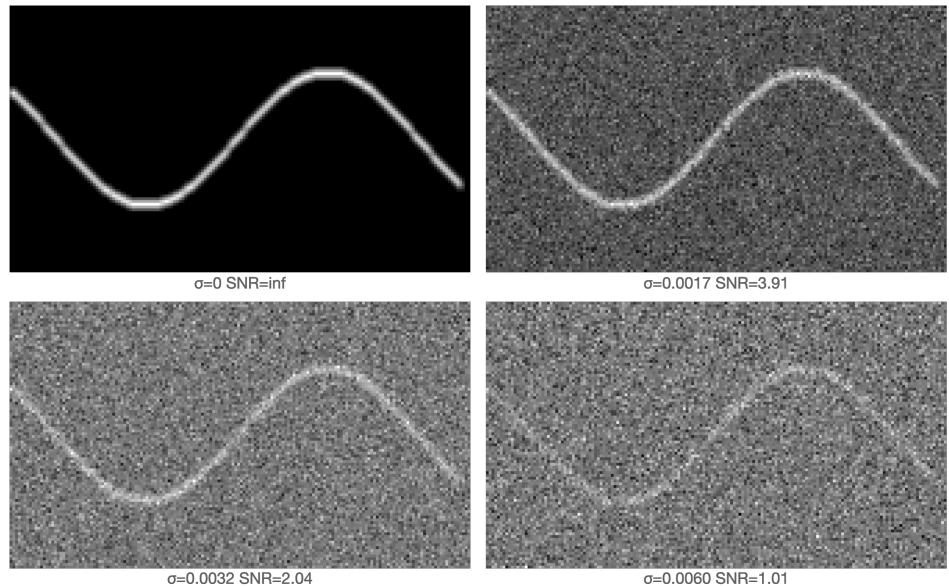


FIGURA 5.6: Ejemplos de imágenes sintéticas generadas de filamentos curvos.

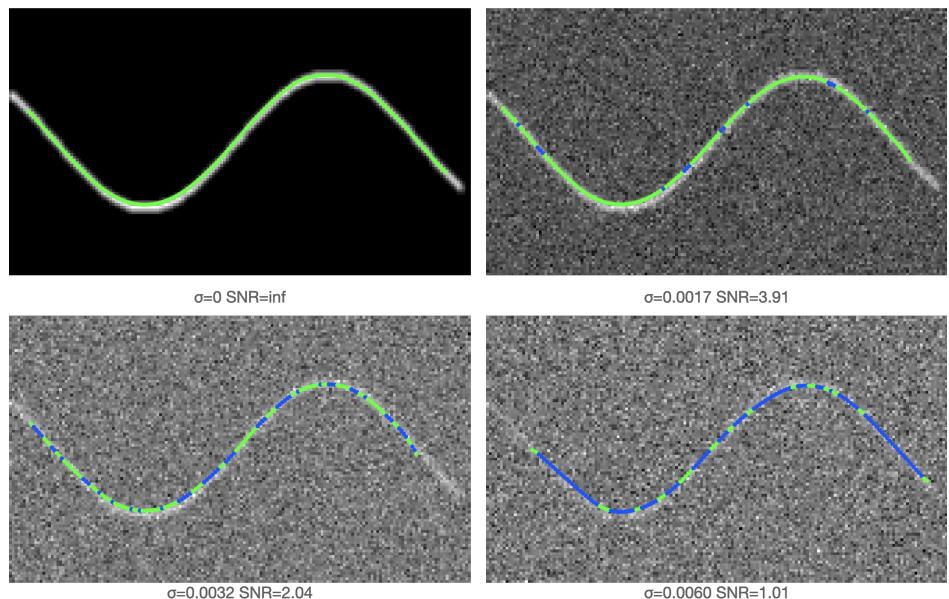


FIGURA 5.7: Resultado de trackear las imágenes de la figura 5.6 mediante la aplicación.

Para relacionar el método de generación de las imágenes sintéticas y la cantidad de ruido presente en cada una, se podrá analizar la figura 5.8 que muestra la relación entre el  $\sigma$  utilizado para la adición de ruido gaussiano y el SNR que este ruido genera. Como es esperable, esta tiene forma decreciente ya que al aumentar el  $\sigma$ , se mantiene el nivel de señal constante y aumenta el ruido presente. Además, podemos apreciar que en los 3 casos esta relación es prácticamente la misma, lo cuál aporta confianza a la metodología utilizada para la generación de ruido y el cálculo del SNR, al menos para los tipos de funciones analizadas.

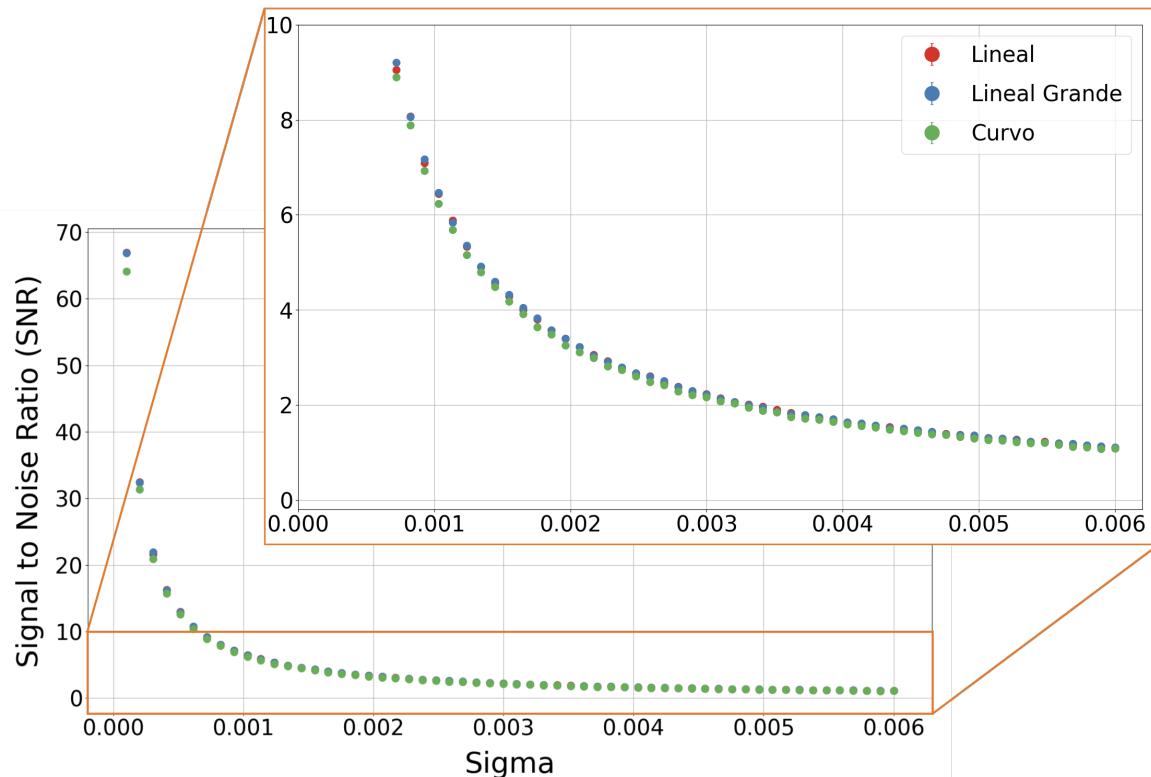


FIGURA 5.8: Relación entre sigma y SNR. El gráfico de la derecha es un acercamiento del gráfico de la izquierda.

En la figura 5.9 se muestra cómo aumenta el error de cada caso a medida que lo hace el ruido en la imagen. Se puede observar con claridad que los casos lineal y lineal grande se comportan de manera similar, por lo que el cambio de resolución no parece afectar significativamente la precisión del algoritmo. De todas maneras, en la zona de mayor ruido sí se puede percibir una leve mejora en el caso de mayor resolución. Esto era esperable debido a que, a mayor resolución, el algoritmo posee más información para trabajar y utiliza una mayor cantidad de puntos para muestrear la función. Por otro lado, se puede ver que el error del algoritmo en ausencia de ruido es cercano a los 0.3 píxeles; y que como máximo llega a los 0.6 píxeles, es decir, cuando hay casi tanto ruido como señal. Por lo tanto, el error nunca supera el pixel. Era esperable que los casos lineales tuvieran excelentes resultados: no solo son el caso de menor complejidad, sino que además es el caso beneficiado por la rutina de reposición de puntos, cuando los mismos superan el error límite, pues se utiliza una interpolación lineal.

En el caso del filamento curvo, el error es significativamente mayor al de los demás, con un

error basal de casi 0.6 píxeles, y llegando a casi 1.6 píxeles. Como se mencionó previamente, era esperable que la precisión del algoritmo en este caso fuese peor, debido a la mayor complejidad de la forma. Sin embargo, en general, el SNR hallado en las imágenes experimentales se encuentra por encima de 2 (ver Sección 5.2), por lo que el error en píxeles sería siempre menor a 0.8 píxeles, lo que representa una precisión totalmente aceptable.

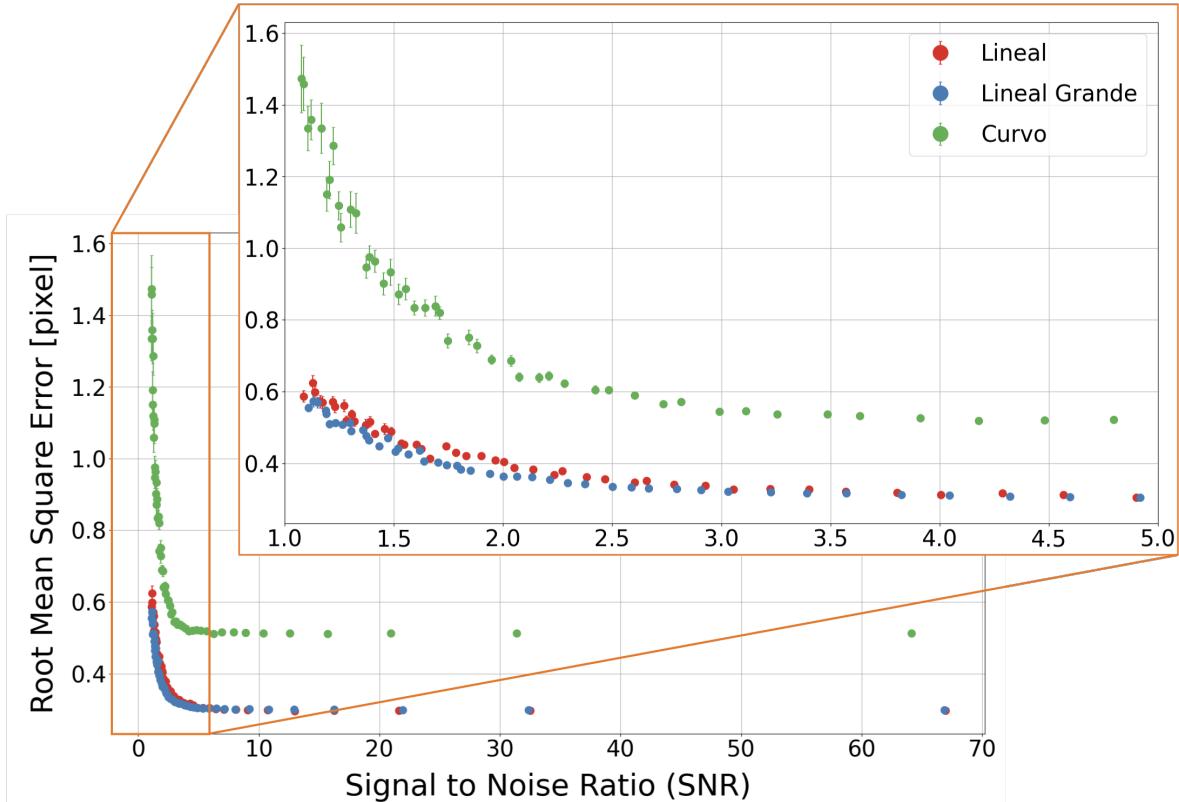


FIGURA 5.9: Dependencia del error de tracking (en píxeles) con la SNR para los casos lineal, lineal grande y curvo. El gráfico de la derecha es un acercamiento del gráfico de la izquierda.

En la figura 5.10 se observa cómo aumenta el tiempo de ejecución de cada caso a medida que lo hace el ruido en la imagen.

En primer lugar, se puede ver claramente que el caso lineal y el caso curvo poseen tiempos de ejecución muy similares, por lo que la forma del filamento no parecería afectar significativamente el tiempo de ejecución. Por supuesto, el caso lineal obtuvo tiempos ligeramente menores, lo cual es esperable debido a la menor complejidad del mismo. Se puede comprobar en ambos casos que el tiempo de ejecución basal estuvo cerca de los 50 ms, y que nunca supera los 150 ms.

Por otro lado, es interesante como el caso lineal grande es aproximadamente el doble de lento que los otros dos casos. La diferencia principal entre estos casos es el tamaño de la imagen. Al duplicar la resolución de cada eje de la imagen, se duplica el largo del filamento a analizar. Al suceder esto, se muestrean el doble de puntos en el filamento, y, como el algoritmo posee un orden lineal en la cantidad de puntos muestreados, el tiempo se duplica.

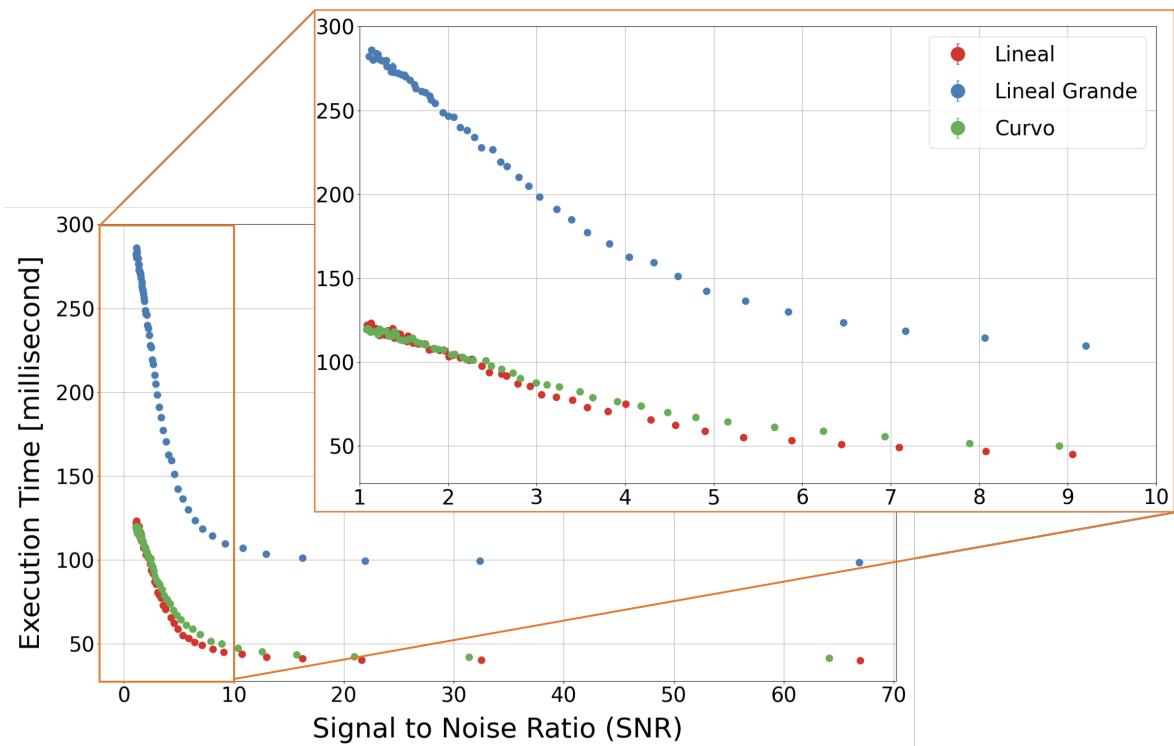


FIGURA 5.10: Tiempo de ejecución del algoritmo vs la SNR de la imagen para los casos lineal, lineal grande y curvo. El gráfico de la derecha es un acercamiento del gráfico de la izquierda.

### 5.1.3. Intersección de filamentos lineales

Uno de los requerimientos del algoritmo era soportar intersecciones entre filamentos sin perder la identidad del que se está trackeando. El caso más simple de diferenciar es cuando dichas intersecciones son perpendiculares, y se va volviendo cada más difícil a medida que disminuye el ángulo de intersección entre los dos filamentos, hasta que el mismo es tan pequeño que no hay suficiente información para diferenciarlos.

El siguiente análisis busca explorar la precisión del algoritmo para distintos ángulos de intersección.

La metodología es muy similar a la anteriormente descripta, exceptuando los siguientes cambios.

En primer lugar, se generan 2 filamentos, por lo tanto se necesitan dos funciones:  $f$  (asociada al filamento trackeado) y  $g$ . Se define  $m = \tan(\frac{\theta_i}{2})$  y  $b = \frac{\text{height}}{2} - m \frac{\text{width}}{2}$  como ayuda. Luego,  $f(x) = \lfloor mx + b \rfloor$  y  $g(x) = \text{height} - 1 - f(x)$ . Así, se garantiza que la intersección ocurra en el centro de la imagen.

Luego, se fija el valor de  $\sigma = 0,0010$  (es decir,  $\text{SNR} = 2,08$ ), y se generan imágenes en las que variamos el ángulo de intersección entre 2 filamentos. Para eso definimos  $\theta_i \in \{10^\circ, 11^\circ, \dots, 60^\circ\}$ , y por cada uno de ellos se obtendrá un valor de RMSE.

Por último, se modificaron los puntos seleccionados por el usuario ingresados al algoritmo de tracking. En este caso  $\text{trim} = 50$  y  $\text{step} = 7$ . De esta manera, sólo se toman en cuenta en el análisis la parte de la imagen donde sucede la intersección, es decir, el centro.

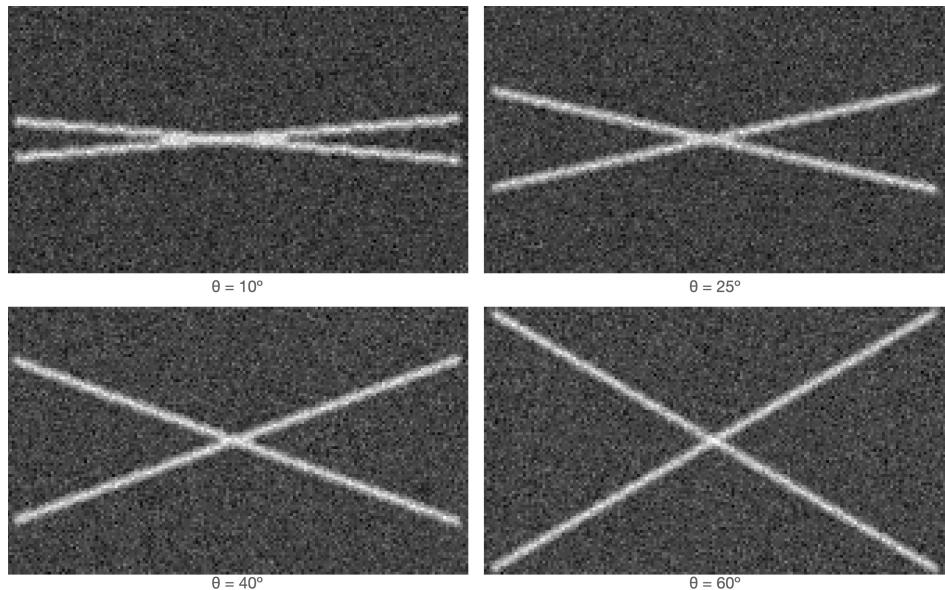


FIGURA 5.11: Ejemplos de imágenes sintéticas generadas de filamentos lineales que se yuxtaponen.

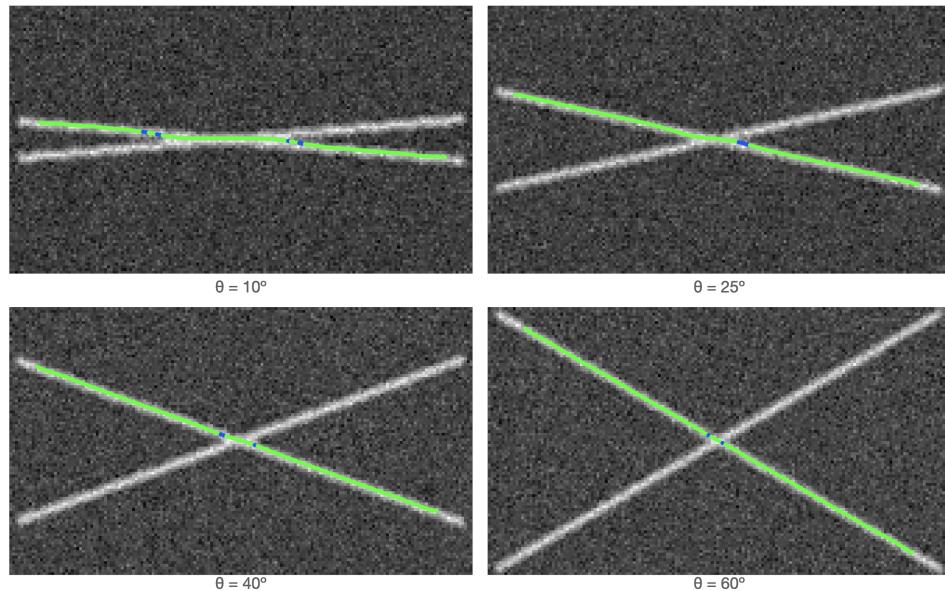


FIGURA 5.12: Resultado de la aplicación al trackear uno de los filamentos (el que cruza de manera decreciente la imagen de izquierda a derecha) que se interceptan en las imágenes de la figura 5.11.

En la figura 5.13 se muestran los resultados del error de tracking como función del ángulo entre los filamentos para un valor fijo de SNR  $\sim 2$ . Puede verse que los puntos parecen estar escalonados. Esto se debe a que sólo se registra un cambio en el centro de la imagen cuando entre un ángulo y otro se cambia de píxel. Por otro lado, cuando el ángulo de intersección es mayor a  $40^\circ$ , el error es muy similar al que se obtiene en ausencia de intersección. Sin embargo, cuando el ángulo es menor a  $20^\circ$ , el error introducido por la intersección es muy notorio. Esto se ve reflejado en las imágenes de la figura 5.12: cuanto menor es el ángulo de intersección, mayor es el área de confusión entre los dos filamentos ubicada en el centro de la imagen. Por supuesto, cuanto mayor es la misma, mayor es el error total.

Si bien esta prueba se realiza con filamentos lineales, se espera un comportamiento similar con curvas más complejas.

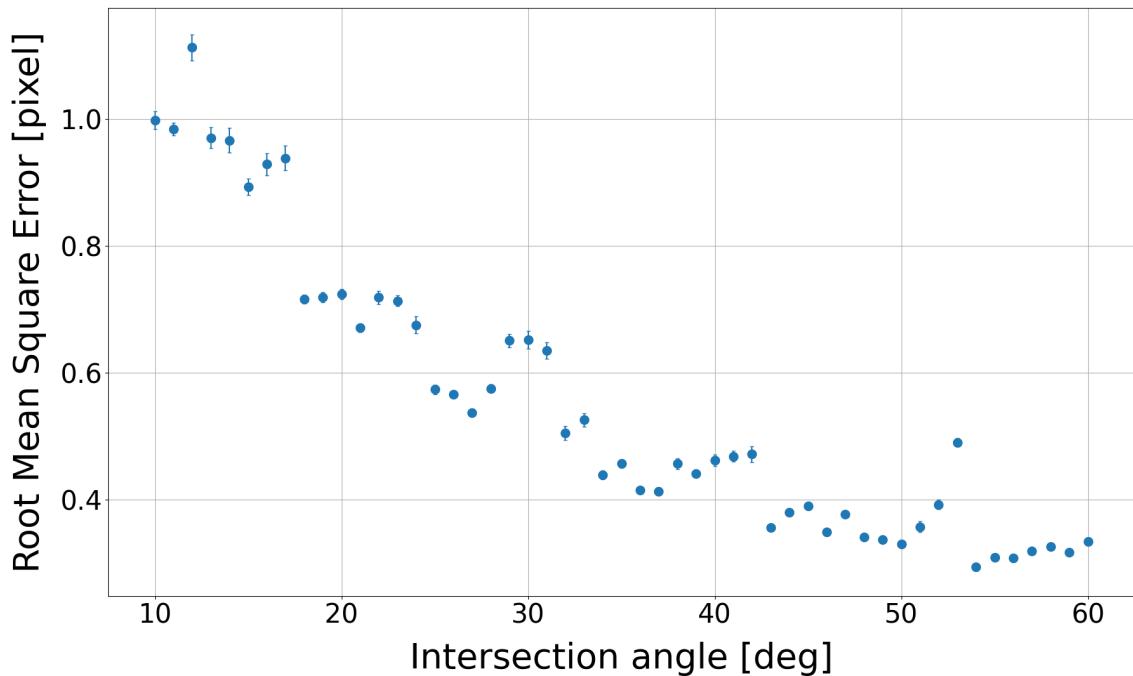


FIGURA 5.13: Error del algoritmo de tracking en píxeles en base al ángulo de intersección de los filamentos de la imagen.

## 5.2. Casos de uso reales

Tanto los resultado de los seguimientos como los parámetros de todos los ejemplos mostrados a continuación podrán encontrarse en la carpeta provista en el Anexo 9.

### 5.2.1. Tracking de filamentos en imágenes de microscopía confocal

Las imágenes obtenidas en células mediante microscopía confocal muestran, obviamente, una complejidad mayor a la que presentaban las imágenes obtenidas *in silico* de la sección anterior. Tal como puede apreciarse en las figuras 5.14 , 5.15 y 5.16 existen múltiples puntos de intersección entre filamentos, con diferentes ángulos, filamentos curvados y un ruido no despreciable. Más

importante aún, los filamentos cambian de posición y de forma en los distintos cuadros, y lo mismo ocurre con la SNR que suele ir disminuyendo a medida que avanza el experimento debido al fenómeno de fotoblanqueo. A pesar de estas complejidades, mostraremos a continuación que la aplicación permite obtener muy buenos resultados.

En el caso mostrado en la figura 5.14 se observa que el algoritmo supera con éxito la intersección con otro filamento conservando la identidad del filamento seleccionado. La SNR de estas imágenes ronda un valor de 7, por lo que se espera un error de tracking por debajo del píxel. El tiempo total de tracking fue de 2.3 segundos, para una secuencia de 22 imágenes. Para este caso, en el siguiente enlace se podrán ver las imágenes resultado obtenidas del algoritmo y la película formada por ellas (en las que se muestran los resultados con el código de colores verde, azul y rojo para ilustrar explícitamente la performance del tracking punto por punto): [https://drive.google.com/drive/folders/1ELrEPMybXXRodhqDUAx\\_-Wa9Dg6lpsBt?usp=share\\_link](https://drive.google.com/drive/folders/1ELrEPMybXXRodhqDUAx_-Wa9Dg6lpsBt?usp=share_link).

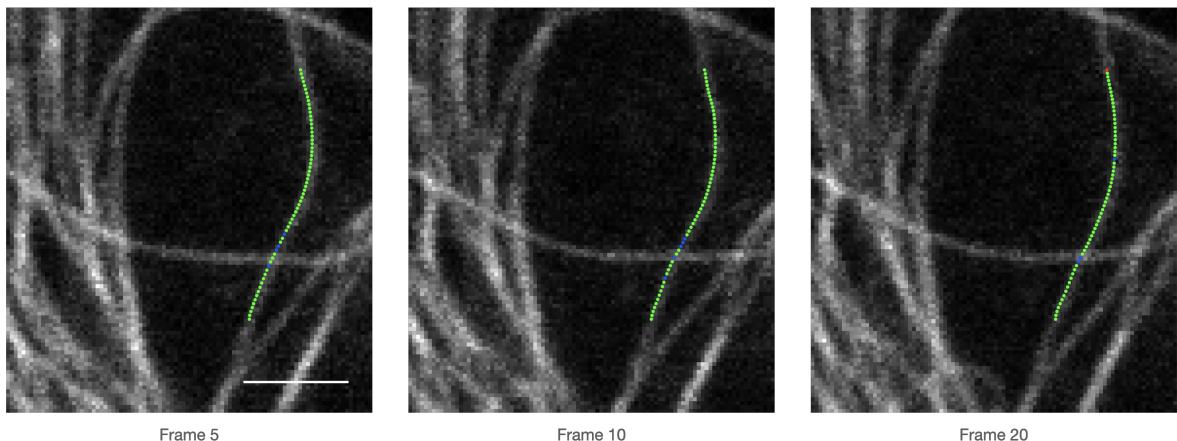


FIGURA 5.14: Tres fotogramas, 5, 10 y 20 de la secuencia analizada. La escala representa 2  $\mu m$ . El SNR presente fue de 7.5, 7.1 y 7.4 respectivamente. El tamaño de la imagen es de 94x104 píxeles.

El caso mostrado en la figura 5.15 es aún más problemático ya que tanto la forma del filamento como la SNR varía de manera considerable ( $SNR \in [3, 9]$ ) y existen múltiples intersecciones. Sin embargo, los resultados obtenidos son muy buenos. El tiempo total de tracking fue de 28.3 segundos, para un total de 260 imágenes. Los tiempos de corrida en relación al caso anterior escalan casi linealmente con la cantidad de imágenes analizadas. Se podrán ver los resultados obtenidos y la película de los mismos en el siguiente enlace: [https://drive.google.com/drive/folders/1QFDdbngcJra0LM\\_-Vx9K3MwMM\\_CR2yEX?usp=share\\_link](https://drive.google.com/drive/folders/1QFDdbngcJra0LM_-Vx9K3MwMM_CR2yEX?usp=share_link)

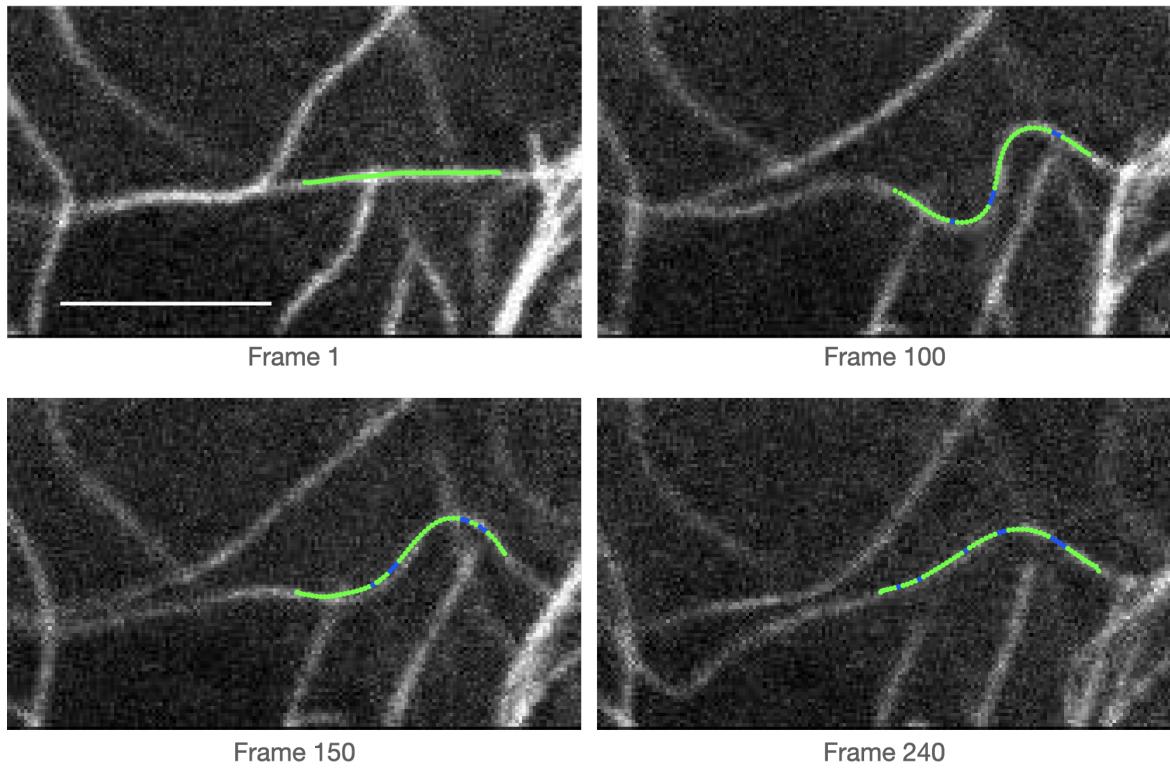


FIGURA 5.15: Cuatro fotogramas 1, 100, 150 y 240 de la secuencia analizada. La escala representa  $5 \mu m$ . El SNR presente fue de 8.1, 5.3, 4.3 y 3.6 respectivamente. El tamaño de la imagen es de 166x96 píxeles.

Por último, en la figura 5.16, se observa una mayor cantidad de píxeles incorrectos en comparación a los casos anteriores. Esto se debe sobre todo a que el filamento analizado no se encuentra aislado. El ángulo de intersección entre los filamentos cercanos es muy bajo. Como se mostró previamente, esto genera problemas a la hora de hacer el seguimiento. Aún así los resultados fueron aceptables. El tiempo total de tracking fue de 37.7 segundos, para un total de 140 imágenes. Puede comprobarse como el tiempo al realizar el seguimiento aumenta considerablemente, siendo mayor que el de la figura 5.15 en la cual se tiene un mayor número de imágenes y, sin embargo, el tiempo del algoritmo fue menor. Esto está directamente relacionado con la cantidad de píxeles de las imágenes. En el siguiente enlace se puede descargar la película con los resultados obtenidos: [https://drive.google.com/drive/folders/1guU10JCyPA02\\_oTIPirQVvenC66MFI5c?usp=share\\_link](https://drive.google.com/drive/folders/1guU10JCyPA02_oTIPirQVvenC66MFI5c?usp=share_link)

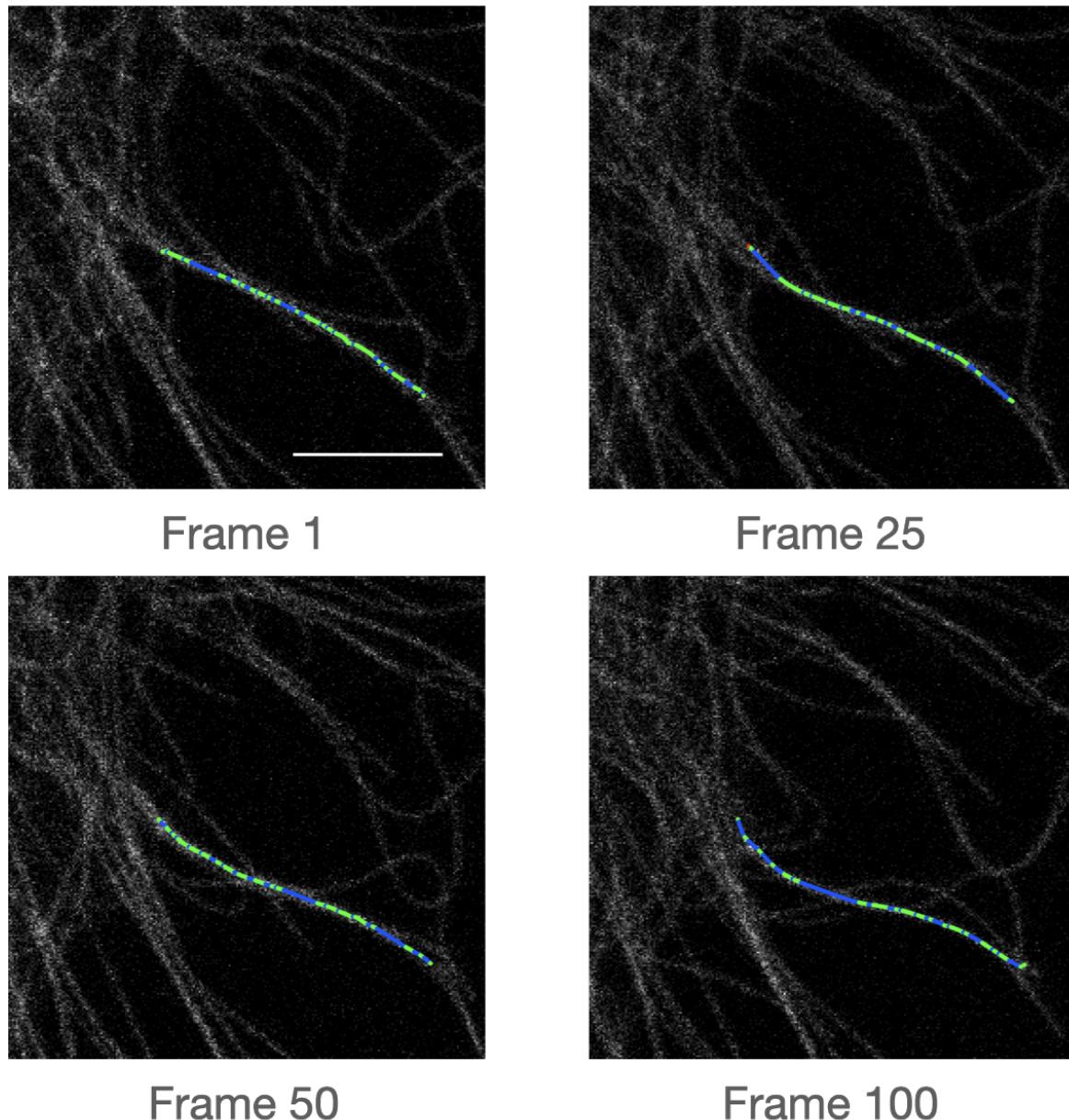


FIGURA 5.16: Cuatro fotogramas, 1, 25, 50 y 100 de la secuencia analizada. La escala representa  $5 \mu m$ . El SNR presente fue de 8.3, 8.9, 9.1 y 6.9 respectivamente. El tamaño de la imagen es de 256x256 píxeles.

### 5.2.2. Tracking de filamentos en imágenes de microscopía Airyscan

A diferencia de las imágenes de microscopía confocal, la microscopía Airyscan genera imágenes de mayor resolución y menor ruido. Como contraposición, las imágenes resultan más pesadas y los tiempos de procesamiento aumentan. En la figura 5.17 se muestran dos imágenes de células melanóforas de *Xenopus laevis* en las que los microtúbulos fueron marcados con una sonda fluorescente. Tal como se puede apreciar, las imágenes obtenidas con la técnica de Airyscan presentan ruido más homogéneo y mayor resolución.

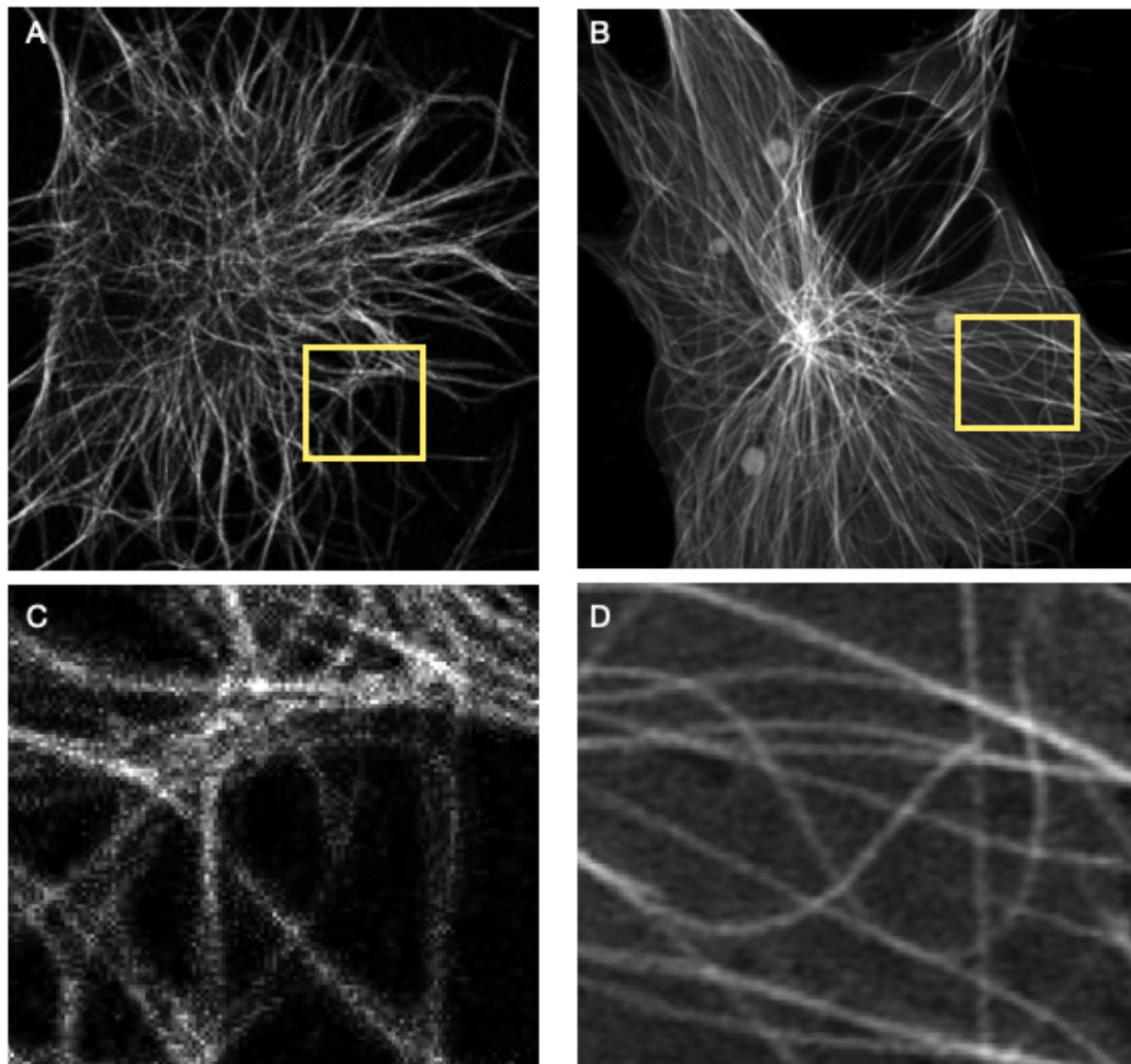


FIGURA 5.17: Comparación imágenes microscopía confocal y Airyscan. (A-B) Imágenes de células con micrótubulos fluorescentes tomadas con un microscopio confocal (A) y Airyscan (B). El tamaño de las células es de aproximadamente  $50 \mu m$ . (C-D) Magnificación de las regiones indicadas en A y B, respectivamente.

A continuación se analizan resultados obtenidos trabajando con imágenes de este tipo.

En primer lugar, en la figura 5.19 se ilustra el análisis realizado sobre una pequeña sección de una imagen obtenida por microscopía Airyscan. La misma pudo ser seguida con precisión aprovechando la funcionalidad de zoom provista por la interfaz, tal como se muestra en la figura 5.18. El tiempo total de tracking fue de 17.5 segundos, para una secuencia de 30 imágenes. Este tiempo aumentó considerablemente en comparación al analizado en 5.14. Esto se debe principalmente a la mayor resolución y tamaño de las imágenes analizadas. En el siguiente enlace podrán obtenerse las imágenes resultado y la película construida a partir de las mismas. [https://drive.google.com/drive/folders/1VMDRNv2PeVqM70jaqv3J\\_RM\\_-UZh0IKS?usp=share\\_link](https://drive.google.com/drive/folders/1VMDRNv2PeVqM70jaqv3J_RM_-UZh0IKS?usp=share_link)

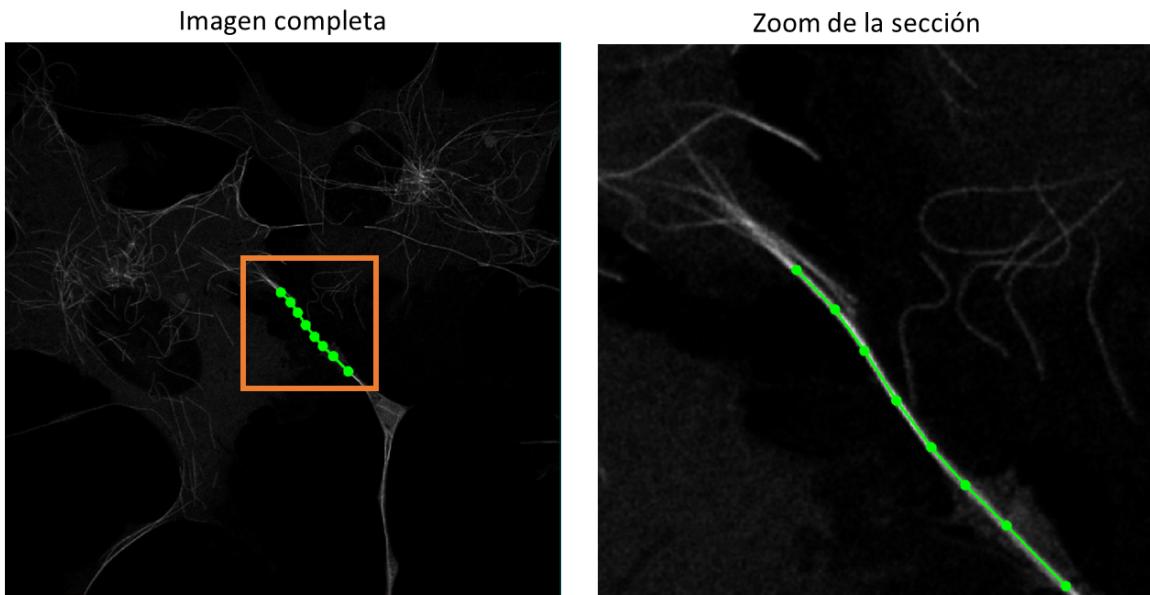


FIGURA 5.18: A la izquierda la imagen completa con los puntos marcados. A la derecha, magnificación de la región indicada en la imagen completa.

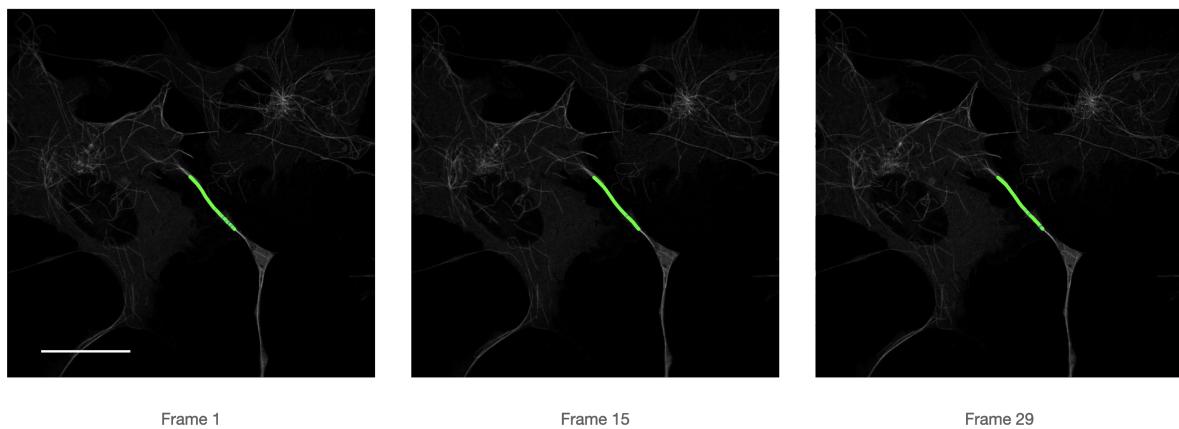
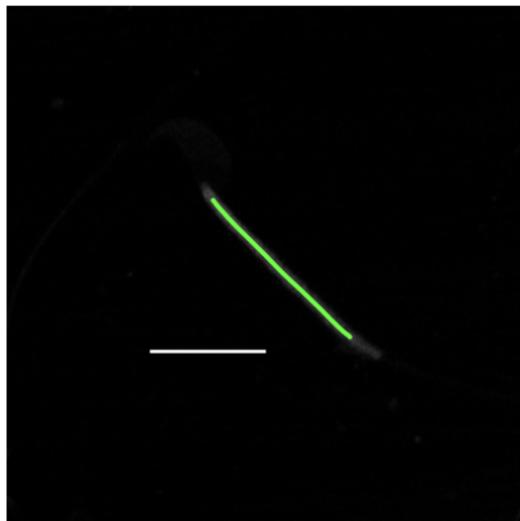


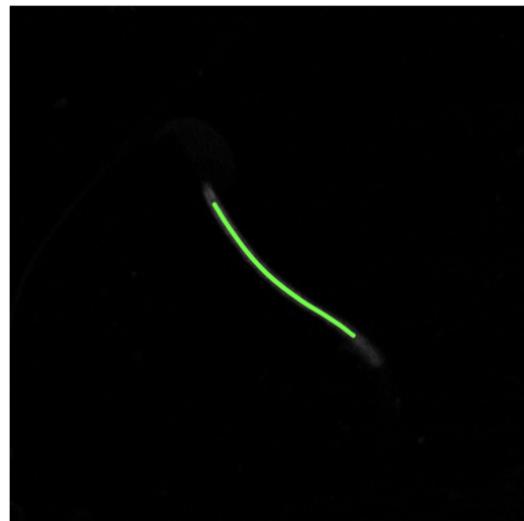
FIGURA 5.19: Tres fotogramas, 1, 15 y 29 de la secuencia analizada. La escala representa  $25 \mu m$ . El SNR presente fue 118, 93 y 116, respectivamente. El tamaño de la imagen es de 2424x2424 píxeles.

En segundo lugar, se puede apreciar en la figura 5.20 un caso muy particular. La secuencia de imágenes muestra el movimiento del flagelo de un espermatozoide de ratón marcado con una proteína fluorescente obtenido mediante microscopía de Airyscan. Las imágenes fueron cedidas gentilmente por la Dra. Vanina Da Ros, del IByME-CONICET. Al tratarse de un sistema diferente, en el que el filamento se encuentra inmerso en una solución acuosa, las imágenes presentan intensidades de fondo prácticamente nulas. Esta observación es compatible con los altos valores de SNR hallados (entre 70-80). La dificultad en este caso consiste en la velocidad del movimiento del flagelo, que requirió utilizar perfiles de intensidad de longitud muy grande en comparación con los utilizados en las imágenes de microtúbulos, ya que la distancia entre localizaciones entre cuadros

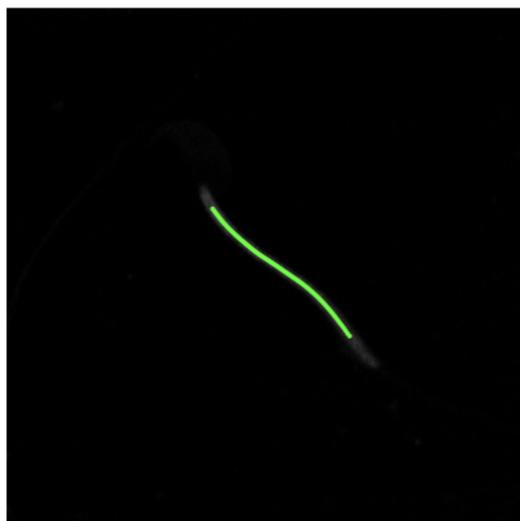
sucesivos era grande debido a la gran amplitud de movimiento. Sin embargo, los resultados del tracking resultaron excelentes y el tiempo total del análisis fue de 28 segundos, para un total de 100 imágenes. Las imágenes resultado y la película obtenida a partir de las mismas puede encontrarse en: [https://drive.google.com/drive/folders/15MEWE\\_fungPmrBUUUJ8e4auwLLmp3ah?usp=share\\_link](https://drive.google.com/drive/folders/15MEWE_fungPmrBUUUJ8e4auwLLmp3ah?usp=share_link).



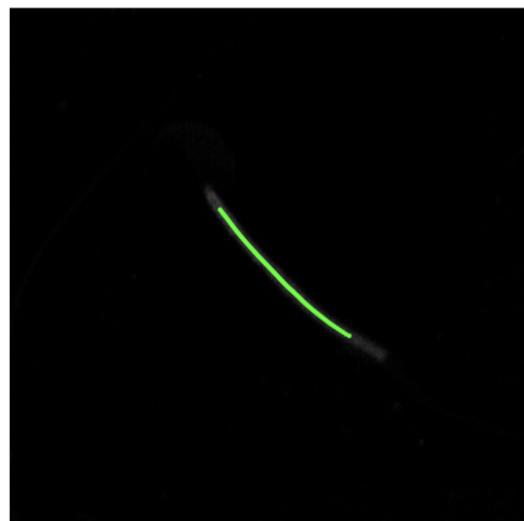
Frame 1



Frame 25



Frame 50



Frame 75

FIGURA 5.20: Cuatro fotogramas, 1, 25, 50 y 75 de la secuencia analizada. La escala representa  $10 \mu\text{m}$ . El SNR presente fue de 74.8, 76.8, 77.5 y 77.8 respectivamente. Para la siguiente configuración, se modificaron los parámetros: ancho del perfil de intensidad = 100 píxeles y densidad de puntos = 3. Imágenes: Dra. Vanina Da Ros (IByME-CONICET). El tamaño de la imagen es de 1044x1044 píxeles.

### 5.2.3. Tracking de filamentos macroscópicos tomados con cámara CCD

En este caso, los filamentos seguidos son macroscópicos. Una de las principales dificultades al analizar estas imágenes es que las mismas, a diferencia del resto de casos analizados hasta aquí, presentan un patrón de colores invertido: los filamentos son de color negro y el fondo es claro. Para poder trabajar con este tipo de imágenes se implementó el parámetro de imágenes invertidas.

Tal como puede verse en el ejemplo mostrado en la figura 5.21, el seguimiento en estas imágenes apenas presenta puntos interpolados, a pesar de la marcada curvatura presente. Esto es principalmente por el alto contraste entre la intensidad del filamento y la del fondo, similar a 5.20. A diferencia de los casos anteriores, el tiempo de respuesta aumenta considerablemente teniendo en cuenta la cantidad de imágenes, lo cual está directamente relacionado con la resolución de las mismas. El tiempo total de tracking fue de 52.1 segundos, para un total de 22 imágenes. Estos resultados muestran que el algoritmo es robusto y extiende el campo de aplicación de la interfaz. Las imágenes resultado obtenidas con el algoritmo y la película: [https://drive.google.com/drive/folders/1BejDJhL4piehNnoMUTXBgr-2Mh5bika2?usp=share\\_link](https://drive.google.com/drive/folders/1BejDJhL4piehNnoMUTXBgr-2Mh5bika2?usp=share_link).

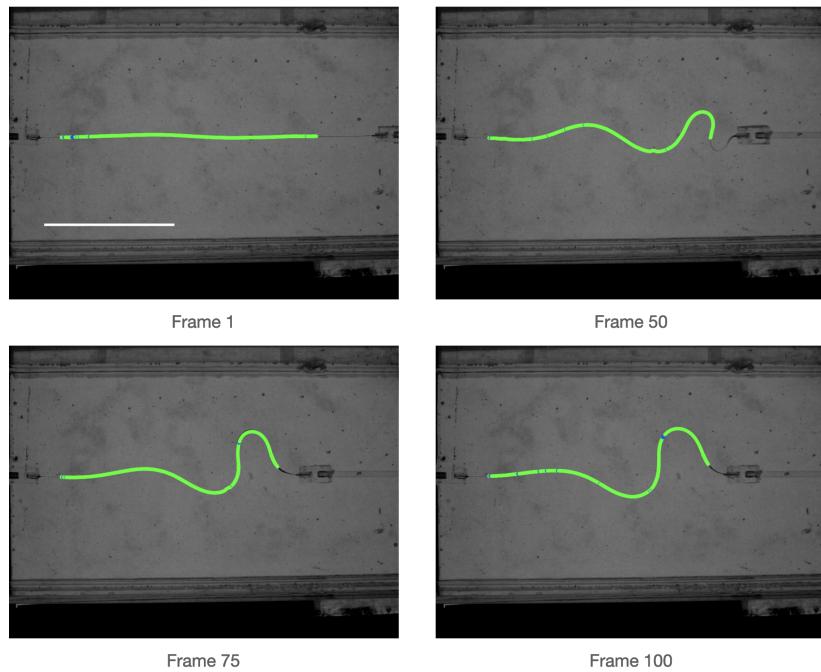


FIGURA 5.21: Cuatro fotogramas, 1, 50, 75 y 100 de la secuencia analizada. La escala representa 5 cm. La SNR presente fue de 6.2, 4.4, 9.4 y 5.6 respectivamente. Para la siguiente configuración, se modificó el ancho del perfil de intensidad a 20 píxeles. El tamaño de la imagen es de 1024x768 píxeles.

## 6. Manual de uso

La aplicación web cuenta con un manual de uso accesible mediante el botón “Manual de Uso” ubicado arriba a la izquierda de la interfaz al ingresar, o accediendo a `/manual`. En el mismo se detalla paso a paso como utilizar la herramienta, y se explica en detalle los parámetros del algoritmo, como se visualizan los resultados y los distintos métodos de exportación.

Un instructivo de como levantar un servidor de la aplicación a partir del código fuente se puede encontrar en el archivo `README.md` en la raíz del proyecto.

## 7. Conclusiones y trabajo futuro

Como conclusión, se considera que, luego de analizar trabajos previos, algoritmos y el proyecto final de Osimani y Favaron [16] se logró construir un algoritmo que permite recuperar las formas de filamentos en secuencias de cientos de imágenes con y sin ruido de manera quasi-automática. El mismo utiliza Python, un lenguaje de código abierto versátil, capaz de construir herramientas de alto rendimiento. El algoritmo desarrollado resulta robusto incluso ante altos niveles de ruido para imágenes obtenidas por microscopía confocal de filamentos fluorescentes, microscopía Ayrscan y microscopía CCD. Las primeras destacan la capacidad del algoritmo de trabajar con imágenes de baja resolución y altos niveles de ruido. Las segundas muestran el alto rendimiento del procedimiento al trabajar con imágenes de altas resoluciones. Por último, al analizar imágenes capturadas mediante cámaras CCD, demuestran la extensión del campo uso de la aplicación más allá de imágenes de microscopía de fluorescencia y con contraste inverso de intensidad.

En todos los casos se puede comprobar la capacidad del algoritmo de soportar movimientos considerables de los filamentos, a pesar de que se toma como hipótesis inicial que los mismos no presentan movimientos bruscos. Cabe destacar también la tolerancia del algoritmo a imágenes con baja señal-ruido. Así mismo, se remarca la resistencia del algoritmo ante cruces del filamento analizado con otros, fallando sólo bajo condiciones altamente complejas. Por último en materia del algoritmo, se logró que el mismo sea capaz de utilizarse en múltiples escenarios gracias a los distintos parámetros configurables ofrecidos.

Por otro lado, se construyó una interfaz web que permite al usuario interactuar con el algoritmo de manera sencilla. Se puso el foco en facilitar el proceso de cargado de imágenes y selección de los puntos sobre las mismas. Se buscó que el usuario pudiera ajustar los parámetros de la rutina de seguimiento de manera ágil y dinámica pudiendo ver el impacto de estos cambios rápidamente gracias a la pre-visualización en tiempo real. A su vez, se reduce la carga cognitiva del usuario al presentar todas las funcionalidades y aspectos configurables en una misma página. Por ultimo, la disponibilidad on-line de la aplicación accesible mediante cualquier navegador estándar ahorra el trabajo de instalar la aplicación localmente, con las dificultades que esto puede conllevar.

Al momento de la presentación del trabajo, cualquier usuario puede acceder a la aplicación a través de las siguientes direcciones:

- Servidor UBA: <https://fernet.exp.dc.uba.ar/>.
- Servidor ITBA: <https://pf-pipo.it.itba.edu.ar/>.

A partir de todo lo mencionado, se considera que se logró cumplir con todos los objetivos y requerimientos planteados.

Desde ya, existen algunos puntos a tener en cuenta para futuras extensiones, tanto en el algoritmo como en la interfaz. Acerca del algoritmo se destaca:

- Mejorar el proceso de reposición de puntos, en particular de puntos de los extremos del filamento. En la configuración actual, si los puntos de los extremos deben ser interpolados, debido a que la interpolación requiere dos extremos para realizarse, simplemente se decide

mantener los puntos de la iteración previa. Se podría desarrollar una heurística más avanzada que permita, utilizando información de la iteración actual, obtener una mejor aproximación al punto deseado.

- La longitud de los segmentos de suavizado es parametrizable. En caso de estar usando una longitud baja, los segmentos no ocuparán todo el filamento. En estos casos se deberán unir los segmentos de Bézier, lo que, como ya fue mencionado, puede generar artefactos indeseados. Los mismos podrían ser eliminados mediante una segunda pasada de suavizado.

Para la interfaz se consideran:

- Presentar una barra de progreso que detalle específicamente cuánto falta para la finalización del tracking.
- Mostrar los resultados parciales a medida que se van generando y agregar un botón que permita interrumpir el análisis en caso de que los resultados dejen de ser satisfactorios. Actualmente, los resultados se despliegan al finalizar el seguimiento.
- Capacidad de aumentar el tamaño de los resultados obtenidos y de hacer zoom sobre ellos sin necesidad de descargarlos.

Finalmente, en cuanto a funciones extras que se podrían agregar a la aplicación se mencionan las siguientes:

- Tracking de filamentos de longitud variable y tracking de extremos de los filamentos. En la aplicación actual la ventana analizada es siempre la misma, ya que la rutina está diseñada para recuperar segmentos de filamentos de longitud constante. Sin embargo, existen casos en los cuales los filamentos pueden cambiar su longitud y en esos casos el algoritmo fallaría.
- Post-procesamiento de los resultados. Una funcionalidad muy valiosa sería por ejemplo realizar el cálculo de la curvatura punto a punto del filamento a partir de las coordenadas recuperadas y graficar las formas del filamento con un mapa de calor según la curvatura local. También se podrían determinar las trayectorias del centro de masa o analizar las fluctuaciones laterales del filamento en posiciones pre-determinadas.

## 8. Bibliografia

- [1] Chirlmin Joo y col. «Advances in single-molecule fluorescence methods for molecular biology». En: *Annu. Rev. Biochem.* 77 (2008), págs. 51-76.
- [2] Jin Zhang y col. «Creating new fluorescent probes for cell biology». En: *Nature reviews Molecular cell biology* 3.12 (2002), págs. 906-918.
- [3] *introduction-to-fluorescence-microscopy*. [www.microscopyu.com/techniques/fluorescence/introduction-to-fluorescence-microscopy](http://www.microscopyu.com/techniques/fluorescence/introduction-to-fluorescence-microscopy). Accessed: 2022-11-21.
- [4] Randy O Wayne. *Light and video microscopy*. Academic Press, 2019.
- [5] James Pawley. *Handbook of biological confocal microscopy*. Vol. 236. Springer Science & Business Media, 2006.
- [6] Frederick Gittes y col. «Flexural rigidity of microtubules and actin filaments measured from thermal fluctuations in shape.» En: *The Journal of cell biology* 120.4 (1993), págs. 923-934. DOI: [10.1083/jcb.120.4.923](https://doi.org/10.1083/jcb.120.4.923).
- [7] Clifford P Brangwynne y col. «Bending dynamics of fluctuating biopolymers probed by automated high-resolution filament tracking». En: *Biophysical journal* 93.1 (2007), págs. 346-359. DOI: [10.1529/biophysj.106.096966](https://doi.org/10.1529/biophysj.106.096966).
- [8] Felix Ruhnow, David Zwicker y Stefan Diez. «Tracking single particles and elongated filaments with nanometer precision». En: *Biophysical journal* 100.11 (2011), págs. 2820-2828. DOI: [10.1016/j.bpj.2011.04.023](https://doi.org/10.1016/j.bpj.2011.04.023).
- [9] Y. Shi y W.C. Karl. «Real-time tracking using level sets». En: *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)* 2 (2012), 34-41 vol. 2. DOI: [10.1109/CVPR.2005.294](https://doi.org/10.1109/CVPR.2005.294).
- [10] David Valdman y col. «Spectral analysis methods for the robust measurement of the flexural rigidity of biopolymers». En: *Biophysical journal* 102.5 (2012), págs. 1144-1153. DOI: [10.1016/j.bpj.2012.01.045](https://doi.org/10.1016/j.bpj.2012.01.045).
- [11] Hongsheng Li y col. «Automated actin filament segmentation, tracking and tip elongation measurements based on open active contour models». En: *2009 IEEE International Symposium on Biomedical Imaging: From Nano to Macro*. IEEE. 2009, págs. 1302-1305.
- [12] Matthew B Smith y col. «Segmentation and tracking of cytoskeletal filaments using open active contours». En: *Cytoskeleton* 67.11 (2010), págs. 693-705.
- [13] J Käs y col. «F-actin, a model polymer for semiflexible chains in dilute, semidilute, and liquid crystalline solutions». En: *Biophysical journal* 70.2 (1996), págs. 609-625. DOI: [10.1016/S0006-3495\(96\)79630-3](https://doi.org/10.1016/S0006-3495(96)79630-3).
- [14] Marcel E Janson y Marileen Dogterom. «A bending mode analysis for growing microtubules: evidence for a velocity-dependent rigidity». En: *Biophysical journal* 87.4 (2004), págs. 2723-2736. DOI: [10.1529/biophysj.103.038877](https://doi.org/10.1529/biophysj.103.038877).

- [15] Carla Pallavicini y col. «Lateral motion and bending of microtubules studied with a new single-filament tracking routine in living cells». En: *Biophysical journal* 106.12 (2014), págs. 2625-2635. DOI: [10.1016/j.bpj.2014.04.046](https://doi.org/10.1016/j.bpj.2014.04.046).
- [16] Agustina Osimani y Sebastián Favaron. «Análisis de imágenes de microscopía de fluorescencia para segmentación y tracking de filamentos celulares». En: (2021).
- [17] *OpenCV: Image Thresholding*. [https://docs.opencv.org/4.x/d7/d4d/tutorial\\_py\\_thresholding.html](https://docs.opencv.org/4.x/d7/d4d/tutorial_py_thresholding.html). Accessed: 2022-11-21.
- [18] *the-canvas-element*. <https://html.spec.whatwg.org/multipage/canvas.html#the-canvas-element>. Accessed: 2022-11-21.
- [19] *Algoritmo de Bresenham*. [https://es.wikipedia.org/wiki/Algoritmo\\_de\\_Bresenham](https://es.wikipedia.org/wiki/Algoritmo_de_Bresenham). Accessed: 2022-11-21.
- [20] *Curva de Bézier*. [https://es.wikipedia.org/wiki/Curva\\_de\\_B%C3%A9zier](https://es.wikipedia.org/wiki/Curva_de_B%C3%A9zier). Accessed: 2022-11-21.
- [21] *Polinomio de Bernstein*. [https://es.wikipedia.org/wiki/Polinomio\\_de\\_Bernstein](https://es.wikipedia.org/wiki/Polinomio_de_Bernstein). Accessed: 2022-11-21.
- [22] *NumPy*. <https://numpy.org/doc/1.23/>. Accessed: 2022-11-21.
- [23] *SciPy*. <https://docs.scipy.org/doc/scipy-1.8.1/index.html>. Accessed: 2022-11-21.
- [24] *scipy.optimize.curve\_fit*. [https://docs.scipy.org/doc/scipy-1.8.1/reference/generated/scipy.optimize.curve\\_fit.html#scipy.optimize.curve\\_fit](https://docs.scipy.org/doc/scipy-1.8.1/reference/generated/scipy.optimize.curve_fit.html#scipy.optimize.curve_fit). Accessed: 2022-11-21.
- [25] *scipy.optimize.least\_squares*. [https://docs.scipy.org/doc/scipy-1.8.1/reference/generated/scipy.optimize.least\\_squares.html#scipy.optimize.least\\_squares](https://docs.scipy.org/doc/scipy-1.8.1/reference/generated/scipy.optimize.least_squares.html#scipy.optimize.least_squares). Accessed: 2022-11-21.
- [26] *scipy.stats.linregress*. <https://docs.scipy.org/doc/scipy-1.8.1/reference/generated/scipy.stats.linregress.html#scipy.stats.linregress>. Accessed: 2022-11-21.
- [27] *scipy.special.comb*. <https://docs.scipy.org/doc/scipy-1.8.1/reference/generate/scipy.special.comb.html#scipy.special.comb>. Accessed: 2022-11-21.
- [28] *skimage.draw.line*. <https://scikit-image.org/docs/0.19.x/api/skimage.draw.html?highlight=draw%20line#skimage.draw.line>. Accessed: 2022-11-21.
- [29] *scikit-image*. <https://scikit-image.org/docs/0.19.x/>. Accessed: 2022-11-21.
- [30] *Pillow*. <https://pillow.readthedocs.io/en/stable/>. Accessed: 2022-11-21.
- [31] *Flask*. <https://flask.palletsprojects.com/en/2.1.x/>. Accessed: 2022-11-21.
- [32] *Gunicorn*. <https://docs.gunicorn.org/en/20.1.0/>. Accessed: 2022-11-21.
- [33] *Docker*. <https://www.docker.com/>. Accessed: 2022-11-21.

[34] Nginx. <https://www.nginx.com/>. Accessed: 2022-11-21.

## 9. Anexo

Todos los resultados mostrados en la sección 5, junto con los parámetros para replicarlos, se encuentran en: [https://drive.google.com/drive/folders/1OaJf6WDuhMytULURFSWoWjjCbYAnTFX2?usp=share\\_link](https://drive.google.com/drive/folders/1OaJf6WDuhMytULURFSWoWjjCbYAnTFX2?usp=share_link).