Gradu Amaierako Lana/Trabajo Fin de Grado

Ingeniaritza Elektronikoko Gradua/Grado en Ingeniería Electrónica

---

Estudio del movimiento de bacterias magnetotácticas en presencia de campos magnéticos

---

Asier Galicia Martínez

Directora:
Prof. María Luisa Fernández-Gubieda Ruiz

Departamento de Electricidad y Electrónica
Facultad de Ciencia y Tecnología
Universidad del País Vasco UPV/EHU

Leioa, junio de 2019

Índice general

Índice general **3**

1. Introducción **5**
1.1. Bacteria magnetotáctica 6
1.2. Aplicaciones de la bacteria 6

2. Equipo experimental y algoritmos utilizados **9**
2.1. Equipo experimental 9
2.1.1. Microscopio 9
2.1.2. Canales micro-fluidicos 10
2.1.3. Configuración del campo magnético 11
2.2. Algoritmo de detección de bacterias 13
2.2.1. Descripción del problema 13
2.2.2. Algoritmo de análisis LoG 13
2.2.3. Aplicación de LoG 16
2.2.4. Algoritmo de análisis DoG 16
2.3. Seguimiento de las bacterias 18
2.3.1. Linear Assingment Problem (LAP) 18
2.3.2. Linear motion LAP 19

3. Resultados experimentales y análisis de datos **21**
3.1. Preprocesado de imagen 21
3.1.1. Homogeneización de la iluminación 21
3.1.2. Eliminación de las bacterias inmóviles 23
3.2. Detección de las bacterias 24
3.3. Seguimiento de las bacterias. 25
3.3.1. Errores cometidos en el seguimiento 27
3.4. Análisis de los datos 29
3.4.1. Velocidad de las bacterias 30
3.4.2. Direccionalidad del movimiento 31
3.4.3. Fracción de bacterias activas 34
3.4.4. Tamaño de las bacterias 34

4. Conclusiones **37**

Bibliografía **39**

A. Programas creados **41**
A.1. Programa para analizar los algoritmos de detección 41
A.2. Programa para tratar los datos obtenidos de TrackMate . . . 50
A.3. Programa para analizar la robustez del seguimiento 57
A.4. Programa para visualizar datos de la robustez del seguimiento 60

Capítulo 1

Introducción

Uno de los mayores males que nos acecha, y que se cobra una ingente cantidad de vidas al año, es el cáncer. Durante años, se ha buscado una solución perfecta que consiga librarnos de este problema. Sin embargo, los métodos hasta ahora utilizados, además de no ser completamente efectivos, conllevan grandes efectos secundarios.

En las últimas décadas, ha habido un esfuerzo considerable en desarrollar tecnologías capaz de curar este conjunto de enfermedades. Entre ellas, nos gustaría destacar los micro-robots. Estos se conciben como agentes de tamaño microscópico sobre los cuales se dispone de un control externo. Gracias a su tamaño, estos agentes son capaces de actuar en zonas que no serían accesibles de otro modo.

Desgraciadamente la tecnología para la construcción y control preciso de estos está todavía fuera de nuestro alcance. Sin embargo, las bacterias magnetotácticas podrían suponer una alternativa viable con grandes ventajas frente a los micro-robots artificiales. Estas bacterias tienen la capacidad de orientarse en campos magnéticos externos de manera pasiva. Por lo tanto, explotando esta respuesta pasiva, se puede conseguir un control de su movimiento con el fin de dirigirlas a un objetivo marcado. Una vez alcanzado su objetivo, se puede aprovechar la cadena magnética de su interior para el tratamiento del cáncer por medio de la hipertermia. Por último, es posible utilizar a estas bacterias para el transporte de fármacos, de forma que estos actúen de manera localizada.

Por todo esto, es crucial tener una descripción precisa del movimiento de estas bacterias. Es en esta línea de trabajo, en la caracterización de su movimiento, en la que este TFG contribuye. Para ello, se va a analizar la velocidad de movimiento de la bacteria magnetotáctica *Magnetospirillum gryphiswaldense*. Se ha comenzado utilizando un microscopio óptico para la obtención de imágenes de su movimiento. Posteriormente, se han utilizado algoritmos de detección y seguimiento para para poder analizar su movimiento. Además, se ha analizado la viabilidad del equipo experimental

utilizado para su posterior uso en el control preciso de estas bacterias.

Durante el primer capítulo se realizará una introducción de la bacteria utilizada, además de una descripción de las aplicaciones potenciales y el alcance actual de las mismas. El segundo capítulo estará centrado en mostrar el equipo experimental y algunos conceptos teóricos relacionados con los algoritmos de detección y seguimiento utilizados. En el tercer capítulo, se muestra el proceso seguido para el análisis de los datos de manera detalla, así como los resultados experimentales obtenidos. Por último, se concluirá el trabajo con unas conclusiones generales.

1.1. Bacteria magnetotáctica

Las bacterias magnetotácticas (MTB) tienen en su interior un conjunto de nanopartículas magnéticas llamadas magnetosomas. Estas partículas se encuentran alineadas a lo largo de un citoesqueleto formando una cadena. Esta cadena de magnetosomas permite a la bacteria orientarse en las lineas de campo magnético, propiedad que se conoce como magnetotaxis. Su existencia permite a la MTB reducir su movimiento tridimensional a un movimiento unidimensional, facilitando así la búsqueda de zonas con mejores condiciones de habitabilidad.

La bacteria magnetotáctica con la que se ha estado trabajando es la especie *Magnetospirillum gryphiswaldense*. Esta bacteria tiene forma espiral y dispone de dos flagelos, uno en cada extremo. En la figura 1.1 se muestra una imagen de la bacteria *Magnetospirillum gryphiswaldense* tomada utilizando un microscopio de transmisión de electrones. En esta imagen se observa, tanto la cadena de nanopartículas magnéticas, como la forma espiral y los flagelos.

Los magnetosomas de esta bacteria, son cristales de magnetita (Fe_3O_4) con propiedades magnéticas excelentes. Estos cristales tienen forma de cuboctaedro truncado con un tamaño aproximado de 45 nm. Además, están rodeados por una membrana lipídica que dota a estas nanopartículas de una buena compatibilidad biológica.

Otra propiedad que define a estas bacteria es la capacidad de sentir gradientes de oxígeno. Esto les permite moverse hacia la zona que se ajusten a sus condiciones de habitabilidad. Esta zona es de baja concentración de oxígeno, llamada zona de microaerobiosis. En la figura 1.2 se muestra esquemáticamente el comportamiento de la bacteria, mostrando, tanto el efecto del campo, como su preferencia por la zona de microaerobiosis.

1.2. Aplicaciones de la bacteria

En esta sección voy a realizar una descripción de algunas de las aplicaciones potenciales de esta bacteria [1]. Entre ellas se destacan el uso de

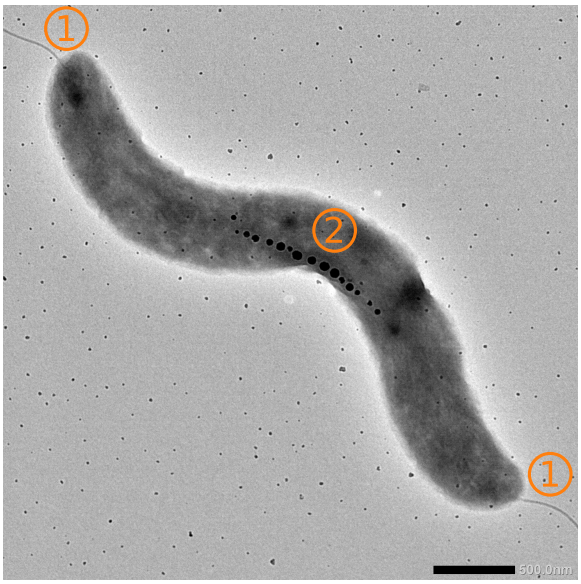


Figura 1.1: Imagen de la bacteria *M. Gryphiswaldense* obtenida con un microscopio de transmisión electrónica (TEM). En (1) se muestran los flagelos de la bacteria. En (2) se muestra la cadena de magnetosomas.

las bacterias para el tratamiento del cáncer por medio de la hipertermia magnética[2] y el transporte de medicamentos[3].

Aplicaciones médicas

La quimioterapia es una terapia para el tratamiento del cáncer que tiene fuertes efectos adversos, dando lugar a un daño tanto en células sanas del cuerpo, como en las del cabello o la sangre. Esto es debido a que se trata de un sistema de acción no localizada.

Un sistema de acción localizada permitiría reducir considerablemente los efectos adversos del tratamiento, centrando así la acción en la zona tumorosa. Es en esta linea de acción donde se estudia el uso de las nanopartículas magnéticas con dos finalidades: hipertermia y transporte de medicamentos.

La hipertermia consiste en calentar una zona localizada de forma que las células de esa zona mueran. Para este tipo de tratamiento podría usarse a la MTB, guiándola a la zona de tumorosa y usando campos magnéticos alternos para calentar la MTB y por ende, la zona tumorosa. Con este tratamiento es posible conseguir aumentar la temperatura hasta los 42-45°C.

En esta misma línea de acción se desarrolla también el uso de la bacteria para el transporte de fármacos. La estrategia consistiría en acoplar a la membrana de la bacteria los fármacos que se desean utilizar. Luego, tras haber guiado satisfactoriamente la bacteria a la zona tumorosa, se utilizarían campos magnéticos alternos, al igual que en la hipertermia magnética, para

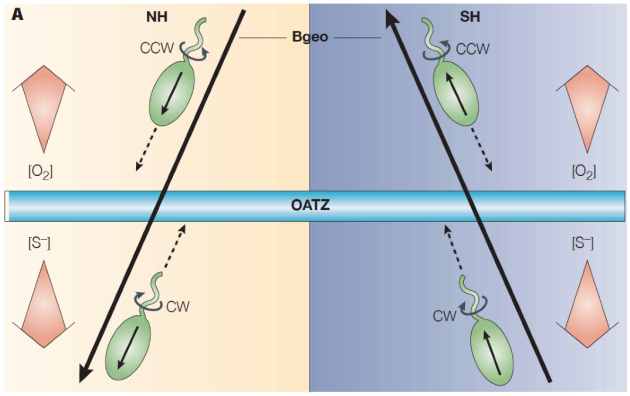


Figura 1.2: Esquema del movimiento de bacterias hacia la zona OATZ (oxic anioxic trasnition zone), que hace referencia a la zona de microaerobiosis. Se utiliza la magnetotaxis para orientarse en el campo magnético terrestre (B_{geo}). El sentido de giro del flagelo cambia en función de la dirección del gradiente de oxígeno y el campo magnético.

aumentar la temperatura de la bacteria. De esta forma, se conseguiría que la MTB liberara los fármacos sobre la zona tumorosa de manera localizada.

Ventajas de las bacterias frente a los micro-robots artificiales.

Como se ha comentado en la introducción, los micro-robots artificiales[4] podrían ser utilizados para este tipo de fines médicos. Sin embargo la construcción de estos micro-robots supone un gran desafío para la tecnología actual debido a su tamaño requerido y las capacidades de propulsión y sensoriales necesarias para sus fines.

Las MTB, por otro lado, superan fácilmente estas limitaciones teniendo un sistema de propulsión propio, capacidad de reacción al entorno propia y siendo fácil su cultivo. Sin embargo, el uso de bacterias como micro-robots tiene la desventaja de no ser posible parar su movimiento, siendo necesarias formas más creativas para conseguir que las bacterias se detenga en una zona particular.

Bacterias magnetotácticas similares han sido utilizadas ya de manera satisfactoria como transporte de medicamentos[3], donde se muestra el uso de campos magnéticos para guiar a las bacterias a zonas hipóxicas en ratones, es decir, zonas de baja concentración de oxigeno. Es aquí donde la capacidad de sentir gradientes de oxigeno entra en juego facilitando la aplicación de las bacterias para fines médicos. Debido al descontrolado crecimiento de células en zonas tumorosas cancerígenas, la concentración de oxígeno es menor que la de su entorno. Estas bajas concentraciones de oxígeno son condiciones de habitabilidad más favorables para las bacterias, lo que facilita que se mantengan en la zona tumorosa.

Capítulo 2

Equipo experimental y algoritmos utilizados

Este capítulo está enfocado a introducir el equipo experimental utilizado durante el proyecto, así como un descripción de los algoritmos utilizados para la detección y seguimiento de la bacteria. Para los algoritmos de seguimiento, se ha utilizado el programa TrackMate[5]. TrackMate se un plugin diseñado para ImageJ[6] que cuenta con una variedad de algoritmos optimizados para el seguimiento de células o bacterias.

Por lo que concierne al algoritmo de detección, se mostrará mediante un ejemplo su viabilidad cuando se intentan detectar objetos elípticos, similares a las bacterias, cuya orientación puede variar. Con respecto al algoritmo de seguimiento, se compararán dos alternativas disponibles, explicando el funcionamiento de cada una de ellas.

2.1. Equipo experimental

En esta sección se va a mostrar el equipo experimental utilizado para la toma de imágenes del movimiento de las bacterias. Más concretamente, se mostrará el microscopio utilizado, así como el canal micro-fluídico usado y la configuración de campos magnéticos usados. En cuanto a esta configuración, se ha caracterizado el campo magnético creado por dos imanes permanentes para comprobar así su viabilidad como plataforma para el análisis del movimiento de las bacterias.

2.1.1. Microscopio

El microscopio utilizado se muestra en el a figura 2.1, donde se observa también la cámara digital usada para la captura de imágenes.

Las características del microscopio se muestran en la tabla 2.1. Entre

Objetivo	20x
Numero de apertura N.A	0.8
Resolución	0.53 μm
Distancia de trabajo	610 μm
Inmersión	Aire

Cuadro 2.1: Tabla de datos con las características del microscopio y su objetivo. El valor de la resolución se ha obtenido de la ecuación 2.1.

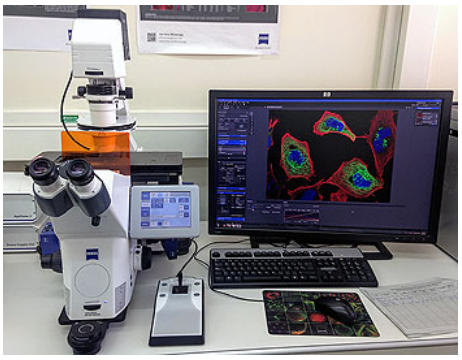


Figura 2.1: Microscopio utilizado para la obtención de imágenes en el laboratorio.

Resolución	2752 x 2208 píxeles
Tamaño real del píxel	4.54 x 4.54 μm
Tamaño de píxel correspondiente en la imagen	0.227 x 0.227 μm
Área total cubierta	624 x 510 μm
Profundidad de bits	16-bits
Frames/s	7.5
Color	Escala de grises

Cuadro 2.2: Características de las imágenes obtenidas.

estos datos es importante destacar la resolución, cuyo valor se obtiene de

$$r \approx \frac{0.61\lambda}{\text{N.A}}, \tag{2.1}$$

donde N.A es la apertura numérica del microscopio, y λ es la longitud de onda de la luz visible ($700nm$ como máximo). La resolución fija tamaño mínimo de los objetos que se van a resolver. Dado que el ancho de la bacteria utilizada es de aproximadamente $0.5\mu m$, esta será una resolución suficiente.

Junto al microscopio, se ha utilizado una cámara digital para la toma de imágenes. Las características de las imágenes obtenidas con esta cámara se muestran en la tabla 2.2. Cabe destacar en este caso que las imágenes que se han tomado a color, aunque luego se modificarán por medios digitales para obtener una versión en blanco y negro.

2.1.2. Canales micro-fluidicos

Los canales micro-fluidicos utilizados para la observación de las bacterias son los μ-Slide VI$^{0.1}$, de la empresa ibidi, que pueden observarse en la figura 2.2a. Las características de estos canales se pueden observar en la tabla 2.3.

Número de canales	6
Longitud del canal	17 mm
Altura del canal	0.1 mm
Anchura del canal	1 mm
Volumen canal	1.7 μl
Volumen reservorios	60 μl

Cuadro 2.3: Datos con las características del canal micro-fluidico.

En este caso cabe destacar la altura del canal. El microscopio es capaz de enfocar solo parte de esta altura debido a que la profundidad de campo con la que se trabaja es menor que la altura del canal.

A la hora de utilizar los canales, se han llenado los reservorios con $40\mu l$ de bacterias, tras lo cual, se ha esperado un tiempo moderado para que el flujo en el canal alcance un estado estacionario.

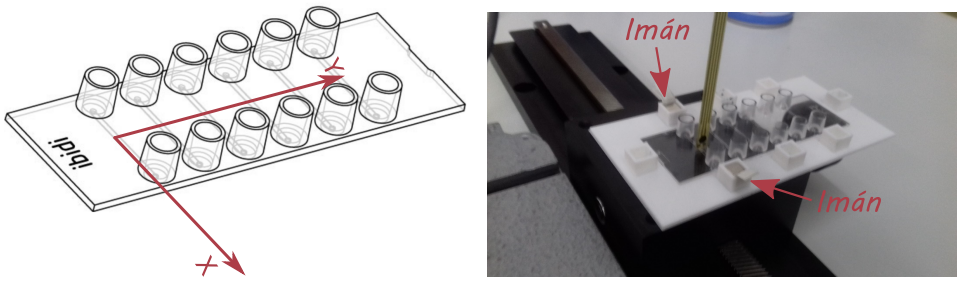
2.1.3. Configuración del campo magnético

Para establecer un campo magnético se ha utilizado una plataforma realizada por impresión 3D, creada por David Gandia, que permite colocar imanes permanentes en paralelo y perpendicular a los canales micro-fluídicos. Esta plataforma puede observarse en la figura 2.2 (b), donde se han marcado las posiciones de los imanes.

Para conocer el campo magnético que se tiene en esta configuración, se ha medido, utilizando un sensor Hall, el campo en la dirección X (a lo largo del canal) y en la dirección Y (perpendicular al canal). En las figuras 2.2 se muestra más concretamente como se han tomado estas medidas. Con estas medidas, se ha calculado el módulo, $|B| = \sqrt{B_x^2 + B_y^2}$, y ángulo, $\theta = \arctan\left(\frac{B_x}{B_y}\right)$, del campo magnético medido. De esta forma, se han obtenido los datos representados en las figuras 2.3.

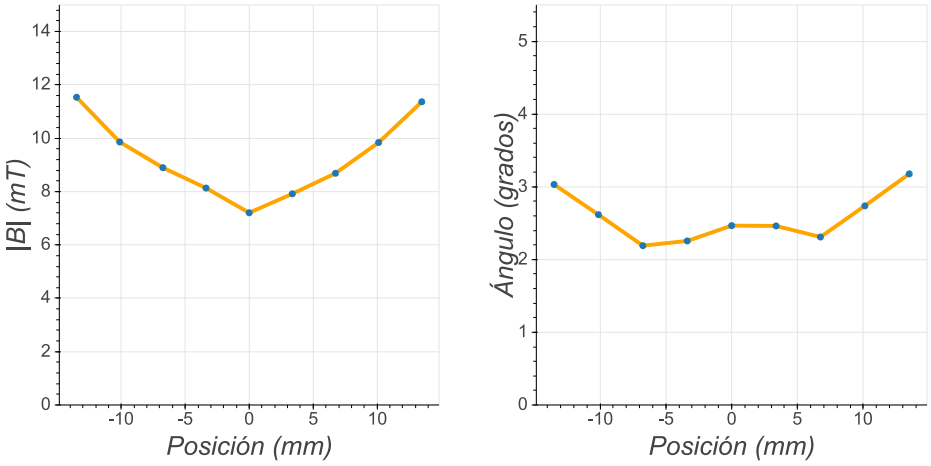
Todos los campos magnéticos medidos tienen una amplitud superior al terrestre (0.04 mT), por lo que las bacterias pueden orientarse en ellos. En cuanto a los ángulos, es cierto que no son completamente constantes. Sin embargo, como se va a enfocar en una pequeña parte del canal, estas variaciones en los ángulos no suponen ningún problema.

Con todo esto se concluye que la configuración de campos magnéticos usada será la adecuada para observar la alineación de la MTB con el campo magnético y así poder medir la velocidad de desplazamiento lineal. Sin embargo, es necesario utilizar otra plataforma si se requiriera tener un control más preciso del movimiento de la MTB. Un alternativa consistiría en el uso de tres pares de bobinas Helmholtz que permitan controlar de manera precisa el campo magnético en todas las direcciones.



(a) (b)

Figura 2.2: En (a) se muestra un esquema del canal microfluídico utilizado. Se ha dibujado el sistema de referencia usado para la representación del campo magnético. La figura (b) muestra un ejemplo de medida del campo magnético creado por los imanes junto con su posición.



(a) (b)

Figura 2.3: Módulo del campo magnético (a) y ángulo (b) creado por los imanes permanentes cuando están en configuración paralela al canal. Las posiciones están representados con respecto al centro del canal. Los puntos azules corresponden a los datos medidos.

2.2. Algoritmo de detección de bacterias

Este apéndice tiene como objetivo explicar el funcionamiento del algoritmo usado para la detección de las bacterias. Aunque a la hora de realizar la parte experimental se ha utilizado la implementación ya existente del algoritmo en el plugin TrackMate del programa ImageJ, se ha creído conveniente añadir esta sección para entender su funcionamiento y sus limitaciones.

En concreto, los algoritmos que se van a presentar son el algoritmo basado en el Laplacian of Gaussian (LoG) y en el Difference of Gaussian (DoG), este último siendo una modificación del primero. Con respecto al algoritmo LoG, se mostrará una explicación de su funcionamiento para continuar con un ejemplo de aplicación a la hora de detectar elipses de diferentes orientaciones. Estas estarán sometidas a un ruido intenso, similar a las condiciones en las que se trabaja con las bacterias.

2.2.1. Descripción del problema

Se va a comenzar simplificando ciertos aspectos de la descripción del algoritmo para poder centrar así la explicación en los aspectos clave del mismo.

En primer lugar, utilizaremos a modo de imagen una función real $I(x, y)$ que representa la intensidad en cada punto de un plano del espacio. Esta función tomará valores reales mayores para puntos más brillantes.

Por otro lado, es necesario introducir la operación de correlación que realizaremos sobre la imagen. Esta operación viene definida por la siguiente ecuación

$$I'(x, y) = \int I(u, u) f(u + x, v + y) du dv, \tag{2.2}$$

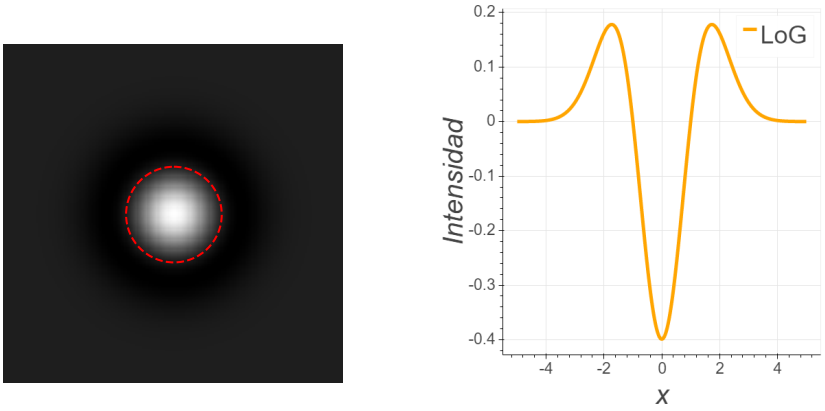
que tiene como finalidad cambiar el valor de intensidad en un punto por un valor ponderado de las intensidades de los puntos de alrededor. A la función $f(x, y)$ de ponderación mostrada en la ecuación 2.2 la llamaremos kernel.

2.2.2. Algoritmo de análisis LoG

El algoritmo LoG consisten en aplicar la operación de correlación ya descrita utilizando como kernel la función $LoG(x, y, \sigma)$, definida como,

$$LoG(x, y, \sigma) = \nabla^2 \left(\frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \right) = \frac{1}{2\pi\sigma^2} \left(\frac{x^2 + y^2}{\sigma^4} - \frac{2}{\sigma^2} \right) e^{-\frac{x^2+y^2}{2\sigma^2}}. \tag{2.3}$$

En la figura 2.4(a) se muestra la representación de esta función como imagen, donde se ha representado con una linea a trazos roja los puntos en los que la función se hace cero. Estos puntos están descritos por una circunferencia de radio $r = \sqrt{2}\sigma$, que separa las regiones donde el kernel toma valores



(a) (b)

Figura 2.4: (1) Imagen bidimensional de la función LoG. En rojo se ha marcado el lugar donde la función se hace cero. Los puntos más blancos representan una menor intensidad. Figura (b) representa el corte en el plano y=0 de la función LoG con $\sigma = 1$.

positivos de las regiones negativas. También se muestra esta misma función cortada por el plano $y = 0$ en la figura 2.4(b).

Para entender el funcionamiento de esta operación, se va a mostrar su comportamiento cuando se realiza la correlación sobre la imagen de prueba mostrada en 2.5 (a). Esta figura muestra un circulo cuyo interior toma valores de 1 y el exterior toma valores -1.

Nuestro objetivo es detectar este círculo de valores 1 usando el kernel LoG. Para ello, vamos a fijarnos en lo que ocurre con la intensidad del punto central del círculo al realizar la correlación de la imagen con el kernel LoG. Denotando como $x = y = 0$ al punto central, tenemos que, al aplicar la correlación, este toma el nuevo valor definido por la siguiente ecuación,

$$I'(0,0) = \int I(x,y) LoG(x,y,\sigma) dxdy. \tag{2.4}$$

Como se puede observar, la operación que realmente se está realizando es la multiplicación de la imagen original por otra imagen que corresponde a la mostrada en la figura 2.4, tras lo cual, se suma la intensidad de todos los puntos. El signo del resultado de la operación de multiplicación se muestra en las figuras 2.5 (b)-(d) para distintos valores σ del kernel LoG. Se puede observar cómo, dependiendo del valor σ, se obtienen regiones de mayor o menor tamaño con signo positivo.

El valor final que se le asigna al punto central, es la suma de todas las intensidades tras la multiplicación. Como resultado de esta integral, es cuando toda la región tiene signo negativo, es decir los ceros de LoG coinciden con el perímetro del círculo, cuando se esperaría obtener un mínimo de intensidad

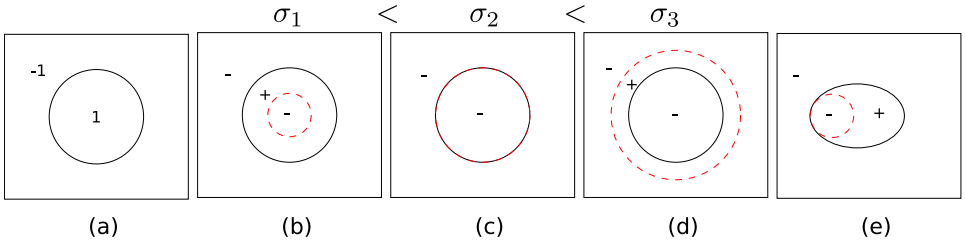


Figura 2.5: (a) imagen original. Se va a suponer que los píxeles toman valores reales entre $[-1, 1]$. En este caso -1 hace las veces de fondo y 1 es el objeto que deseamos detectar. (b) - (d) muestran el resultado de multiplicar el filtro LoG y la imagen (a) para distintos valores σ del filtro. El filtro toma valores negativos solo en el interior del anillo rojo tomando el valor cero justo en el anillo. También se muestra el signo de cada región. En (e) se muestra un caso particular cuando la correlación se hace para obtener el valor de un punto distinto al central.

en el punto central[1]. Al aumentar o disminuir el valor de σ, aparecen regiones de signo positivo que contribuyen aumentando el valor final, es decir, el valor de intensidad del punto central es mayor en estos casos.

Si intentamos obtener el valor de intensidad de un punto tras la correlación distinto al central, nos encontramos en la situación de que no es posible hacer coincidir la circunferencia roja con la circunferencia negra que define nuestro objeto. Es por esto que, tras realizar la operación de correlación, puntos distintos al central tienen valores mayores de intensidad que el central. Buscando, por lo tanto, el mínimo local tras la correlación, se encontraría la posición central de la circunferencia, así como un valor σ que estima el tamaño de la circunferencia. En definitiva, se obtiene un mínimo de intensidad en la posición central tras realizar la operación de correlación con el kernel LoG para $\sigma = \frac{r}{\sqrt{2}}$, siendo r el radio de la circunferencia con valores uno en la figura 2.5 (a).

Esta búsqueda de mínimos locales podría dar un error en una configuración similar a la mostrada en 2.5 (e), donde debido a la forma particular del objeto, es posible obtener un mínimo local para un punto distinto al central. Sin embargo, se mostrará en la siguiente sección que esto no supone un problema.

[1]El filtro LoG no tiene realmente un valor constante, por lo que además del signo del resultado $I(x, y)f(x, y)$, también habría que observar su magnitud. Sin embargo, aún teniendo estas consideraciones en cuenta, los resultados mostrados arriba no cambian. Se ha obviado esta pequeña dificultad por simplicidad.

2.2.3. Aplicación de LoG

Se va a mostrar ahora por medio de un ejemplo la capacidad de detección del algoritmo LoG. Se va a utilizar como imagen un conjunto de elipses de distintos tamaños y orientaciones. También se ha añadido un ruido a la imagen para hacer el ejemplo más realista.

En las figuras 2.6 (b)-(e) se muestra el efecto de realizar la operación de correlación sobre la imagen original 2.6 (a), para distintos valores de σ. En la figura 2.6 (f) se muestra cómo varía la intensidad de varios puntos de la imagen al realizar la operación de correlación. Los puntos en rojo de las figuras 2.6 (b-e) muestra la posición de mínimos locales tras aplicar la correlación de LoG con un σ dado.

Se puede observar en estas imágenes que la intensidad de los puntos centrales de las elipses tienen menor intensidad cuando el valor $\sqrt{2}\sigma$ es similar a su tamaño. Más concretamente, se observa que el valor $\sqrt{2}\sigma$ tiene que ser similar a la longitud del eje largo, en vez de a la del eje corto. Además, como se puede observar con la curva morada, el mínimo de intensidad en puntos distintos al central es significativamente mayor. Esto permite discernir entre este tipo de detecciones y las detecciones correctas. Por último, también se observa que si el valor de $\sqrt{2}\sigma$ usado con el kernel LoG es similar al tamaño de la elipse, entonces el mínimo ocurre en una posición cercana al centro de la elipse, obteniendo así la posición correcta de la elipse.

Este ejemplo muestra la viabilidad del kernel LoG cuando trabaja con imágenes con mucho ruido y se necesita detectar objetos elípticos, similares a las bacterias, cuya orientación puede variar.

2.2.4. Algoritmo de análisis DoG

El kernel DoG consiste en aproximar el kernel LoG por la diferencia de dos funciones gaussianas, con objetivo de acelerar el tiempo de ejecución necesario para detectar los objetos. Más concreta mente, la aproximación que se realiza se muestra en la siguiente ecuación

$$LoG(x, y, \sigma) = \nabla^2 f(x, y, \sigma^2) = \frac{\partial}{\partial \sigma^2} f(x, y, \sigma^2) \approx \frac{f(x, y, \sigma^2 + h) - f(x, y, \sigma^2)}{h}, \tag{2.5}$$

donde $f(x, y, \sigma^2) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2++y^2}{2\sigma^2}}$ es una función gaussiana. En la figura 2.7 (b) se muestra las diferencias entre ambos, observando así su similitud. El uso principal de este kernel frente al LoG es aumentar la velocidad de cómputo. Sin embargo, al ser una aproximación, produce en general resultados algo peores que el kernel LoG.

El uso de uno u otro depende de la aplicación concreta, pero en nuestro caso se ha decidido utilizar el algoritmo DoG por ser más rápido y por proporcionar resultados similares al kernel LoG.

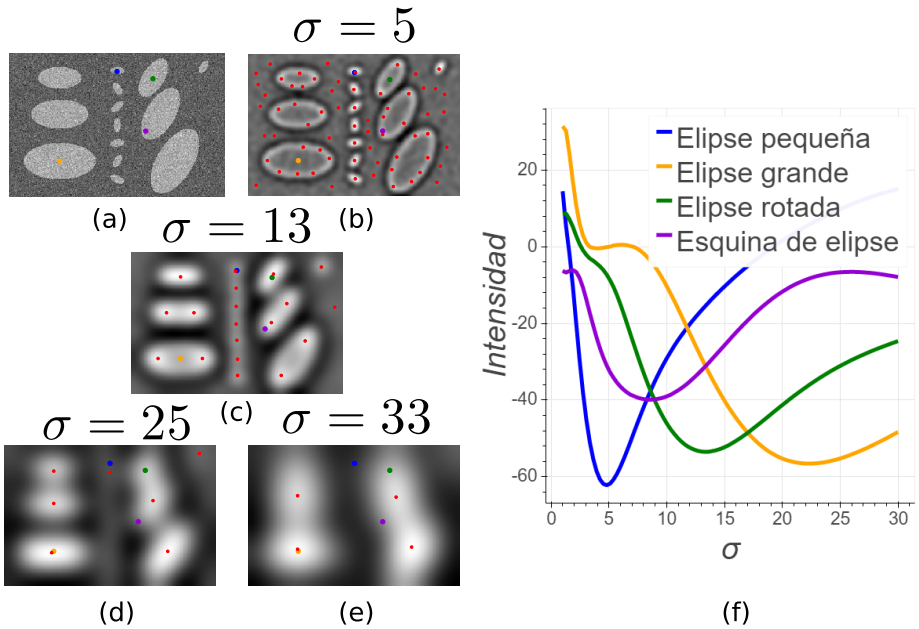
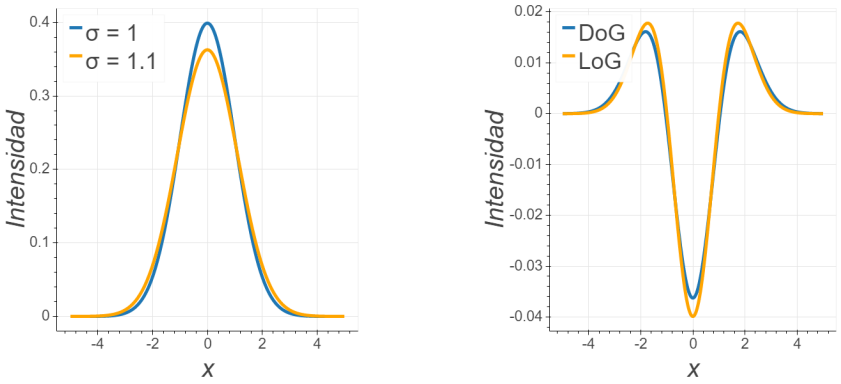


Figura 2.6: La figura (a) es la imagen original que contiene elipses de distintos tamaños y orientaciones. Los tamaños son de 5, 15, 20 y 25 píxeles para el eje corto y 10, 30, 40 y 50 píxeles para el eje largo de las elipses. Las figuras (b) - (e) son los resultados de aplicar el filtro LoG con valores de $\sigma = 5$, 13, 23, 33. Se han invertido las imágenes, de forma que los puntos de menor intensidad son los mas blancos. La figura (f) consiste en la representación de la variación de la intensidad frente al parámetro σ del filtro LoG. La intensidad de los puntos que se muestran, son los marcados en las imágenes (a)-(e).



(a) (b)

Figura 2.7: La figura (a) consiste en dos funciones gaussianas con σ similar. La figura (b) muestra el corte en el plano y=0 de la función LoG, con $\sigma = 1$, y DoG conformado por la diferencia de las funciones mostradas en la figura (a).

2.3. Seguimiento de las bacterias

En esta sección se van a mostrar y comparar los algoritmos para el seguimiento de bacterias disponibles en TrackMate. Para ellos, será necesario realizar una pequeña descripción de ambos algoritmos. Los dos algoritmos disponibles son: Linear Assingment Problem (LAP) y Linear motion LAP. Estos algoritmos tienen como función el enlazar las bacterias ya detectadas en distintos fotogramas. Con los enlaces obtenidos, el algoritmo forma los seguimientos de las bacterias.

2.3.1. Linear Assingment Problem (LAP)

El algoritmo de LAP asigna un coste al enlace de dos bacterias en distintos fotogramas. Este coste depende principalmente del cuadrado de la distancia entre las dos bacterias que se desean unir. Del mismo modo, se puede modificar ligeramente la función de coste para que tenga en cuenta similitudes en las características de las bacterias a enlazar, como por ejemplo, su intensidad media. Se terminan por realizar, por lo tanto, enlaces entre bacterias de distintos fotogramas que minimizan la función de coste total.

El algoritmo tiene la capacidad de enlazar bacterias incluso cuando estas no aparecen durante varios fotogramas. Esta característica es muy útil para poder paliar los fallos cometidos en la detección de bacterias. Por todo esto, el algoritmo funciona mejor cuando tiene que seguir bajas densidades de bacterias cuyo movimiento es browniano.

2.3.2. Linear motion LAP

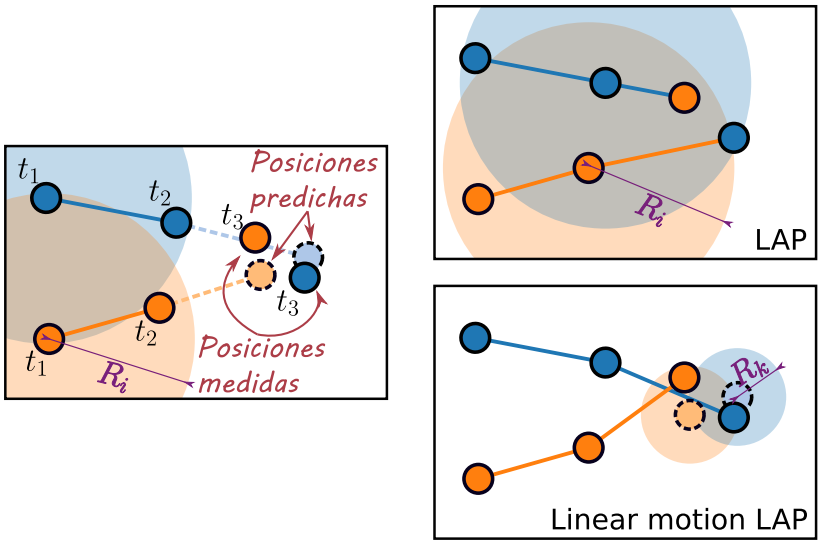


Figura 2.8: Comparación del comportamiento del algoritmo LAP y Linear Morion LAP. En la primera figura se muestra la posición y seguimiento de dos objetos en distintos fotogramas. También se muestra con lineas discontinuas la posición predicha de los objetos. R_i sería tanto el radio de búsqueda inicial, como el radio de búsqueda siguiente en caso de usar el algoritmo LAP. El valor R_k es el radio de búsqueda utilizado por el algoritmo Linear motion LAP tras realizar la predicción de la posición siguiente de la bacteria.

El Linear motion LAP es una modificación del algoritmo LAP que mejora a este en la tarea de seguimiento de bacterias con movimiento lineal. Para ello, se realiza una predicción de la bacteria en seguimiento, suponiendo que se mueve a velocidad constante. A continuación se calculan los costes de enlace, pero esta vez entre la predicción y la posición de las bacterias en el nuevo fotograma, en vez de entre las posiciones medidas de las bacterias entre los dos fotogramas sucesivos.

En la figura 2.8 se muestra un esquema que compara estos dos algoritmos en un caso en el que el algoritmo Linear motion LAP realiza correctamente el seguimiento, a diferencia del algoritmo LAP. Los valores R_i y R_k son los parámetros de entrada del algoritmo Linear motion LAP. Estos parámetros son el radio de búsqueda inicial, R_i, utilizado cuando no se han podido realizar predicciones, y el radio de búsqueda R_k tras la predicción. Solo las bacterias en el interior del radio de búsqueda serán consideradas para el enlace.

La ventaja de utilizar el algoritmo Linear motion LAP frente al algoritmo LAP es, por lo tanto, el poder utilizar un radio de búsqueda R_k mucho menor. Esto conlleva a que el seguimiento sea más preciso, ya que permite

reducir drásticamente la posibilidad de confusión por parte del algoritmo.

Dado que el movimiento de las bacterias bajo campo magnético externo aplicado es de velocidad constante, se ha decidido utilizar el algoritmo Linear motion LAP por su clara superioridad en el seguimiento de objetos con velocidad constante.

Capítulo 3

Resultados experimentales y análisis de datos

En este capítulo se especificará el proceso seguido desde la obtención de la imagen a través del microscopio, hasta la obtención de los seguimientos realizados por TrackMate. Este proceso esta esquematizado en la figura 3.1, donde se muestran las cuatro etapas seguidas: pre-procesado de la imagen, donde se realizarán operaciones que mejoran la calidad para la posterior detección; la detección de las bacterias por medio del algoritmo DoG; el seguimiento por medio del algoritmo Linera motion LAP y por último, el filtrado y análisis de los resultados. También se realizará un análisis de la robustez de los algoritmos de detección y seguimiento, analizando en detalle cuando fallan y porqué.

Cabe destacar que para el análisis de datos, una vez obtenidos los seguimientos de TrackMate, se ha utilizado el lenguaje Python 3.7 junto con las librerías incorporadas en el paquete Scipy[7]. También se ha utilizado la librería Bokeh[8] para la obtención de las figuras.

3.1. Preprocesado de imagen

Para mejorar la calidad de las imágenes obtenidas y poder así usar los algoritmos de TrackMate, se han realizado las siguientes operaciones sobre la imagen: homogeneización de la iluminación, inversión de la imagen y eliminación de las bacterias inmóviles.

3.1.1. Homogeneización de la iluminación

Uno de los mayores problemas a la hora de trabajar con las imágenes, es el no tener una iluminación homogénea. Esto se debe a las condiciones de trabajo (fuente de luz utilizada, microscopio y sensibilidad de los píxeles

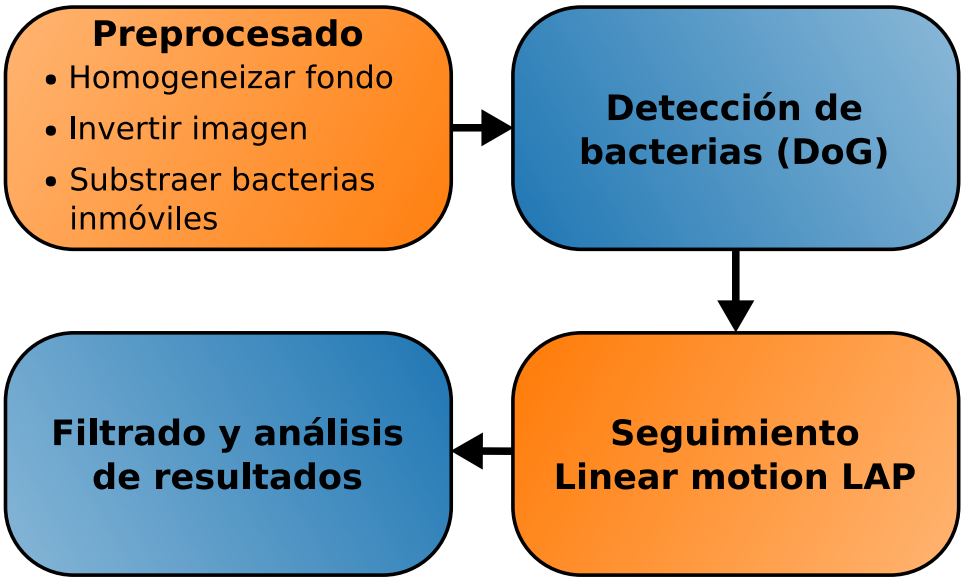


Figura 3.1: Esquema del flujo de trabajo seguido para analizar las imágenes obtenidas con el microscopio.

en la CCD). Para corregir este efecto no deseado, se realiza la siguiente transformación sobre cada píxel de la imagen original:

$$I_{final} = (I_{original} - D)\frac{m}{F - D}. \tag{3.1}$$

Las variables D y F hace referencia a Dark Field (campo oscuro) y Flat Field (campo plano) de la Imagen. La variable m se define en la siguiente ecuación

$$m = \langle F - D \rangle_{pixeles}, \tag{3.2}$$

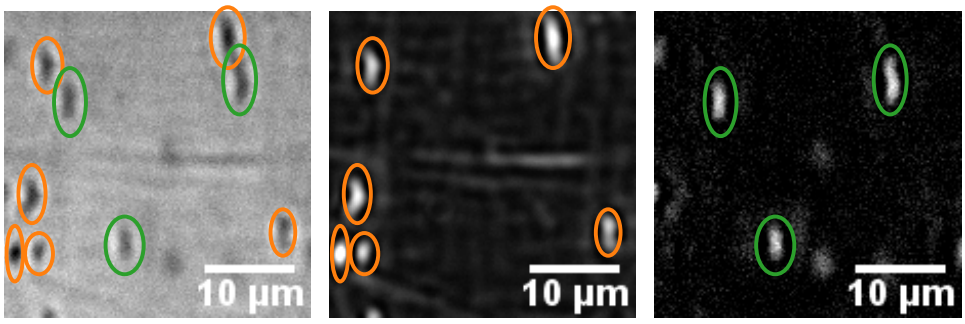
como el valor medio del campo plano menos el campo oscuro a lo largo de todos los píxeles.

El campo plano es la imagen obtenida con una iluminación homogénea sobre la cámara. El campo oscuro, por otro lado, es la imagen obtenida cuando no se hace incidir ninguna iluminación sobre la CCD.

Cabe destacar que nuestro objetivo es corregir la iluminación no homogénea que se utiliza junto al microscopio, por eso no se realizará la corrección correspondiente al campo oscuro (no es necesaria). El procedimiento para obtener D sigue así:

1. Obtener una cantidad suficiente de imágenes (un vídeo) con el microscopio desenfocado antes de analizar la muestra. Esto se hace en las condiciones en las que se trabajará posteriormente.

2. Realizar una media de las imágenes obtenidas. La imagen resultante de la media será F.



(a) (b) (c)

Figura 3.2: La figura (a) es la imagen original obtenida por el microscopio. Se ha señalado con naranja las bacterias quietas y con verde el resto. La figura (b) muestra la imagen invertida de la media sobre todas las imágenes del vídeo. Se pueden observar altos valores de intensidad para las bacterias quietas. La figura (c) muestra la imagen obtenida al restar las figuras (b)-(a), habiendo invertido previamente la imagen (a).

Tras las aplicar las correcciones arriba mencionadas sobre la imagen, se procede a invertirla para que las bacterias sean los objetos más brillantes de la imagen y no los mas oscuros. Esto se hace por que los algoritmos implementados en TrackMate (LoG y DoG) solamente detectan objetos brillantes.

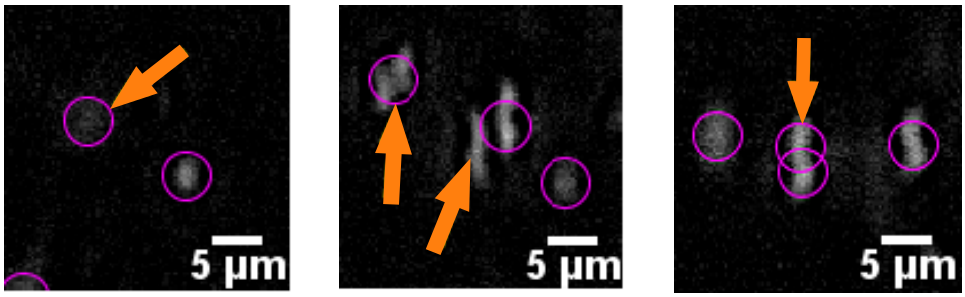
3.1.2. Eliminación de las bacterias inmóviles

Se ha observado en la toma de imágenes cierta cantidad de bacterias que se encuentran inmóviles. Estas bacterias corresponden a bacterias muertas que se han depositado en el fondo del canal. Como este conjunto de bacterias afectan de manera negativa al seguimiento, se ha decidido realizar el siguiente procedimiento:

1. Realizar una media sobre todas las imágenes obtenidas. Esta imagen tendrá picos de intensidad donde las bacterias se encuentren inmóviles.

2. Haciendo la resta de esta imagen sobre la imagen original se obtiene una nueva imagen con la eliminación parcial de las bacterias inmóviles.

Un ejemplo de los resultados del preprocesado se muestran en el grupo de figuras 3.2. Se puede observar como las bacterias señaladas en naranja, que corresponden al grupo de bacterias quietas, son completamente eliminadas en la imagen final (c).



(a) (b) (c)

Figura 3.3: Errores cometido por TrackMate en la detección de bacterias. La figura (a) muestra un error de falso positivo, donde se detecta una bacteria en un lugar donde no es clara la existencia de la bacteria. La figura (b) muestra un error de superposición, detectándose una bacteria donde en realidad hay dos. La figura (c) muestra un error de detección doble, donde se detecta una bacteria dos veces debido a su larga longitud.

3.2. Detección de las bacterias

Para la detección de las bacterias TrackMate tiene implementados varios algoritmos, entre los que se encuentran el DoG y el LoG. Como se ha comentado en el anterior capítulo, se ha decidido utilizar el algoritmo DoG para la detección de bacterias debido a su superior velocidad de trabajo.

El filtro DoG requiere como parámetro de entrada el radio aproximado de los objetos a detectar, donde se ha decidido usar $r = 3.9\ \mu m$ debido a que es el valor que mejores resultados a proporcionado. Esto último se debe a que la longitud de las bacterias es de aproximadamente $4\ \mu m$, aunque, en realidad, la longitud de las bacterias varía mucho de una a otra, habiendo algunas que superen por mucho esta estimación.

La forma alargada de las bacterias y las condiciones de la imagen provocan los errores en la detección que se muestran en las figuras 3.3. En (a) se muestra un error de falso positivo, donde se ha detectado incorrectamente la bacteria señalada. En (b) se muestra un error de superposición donde solamente se detecta una bacteria en el lugar donde debería haber dos. Por último, en (c) se muestra un error de detección doble donde, debido a la larga longitud de la bacteria, se detecta dos veces la misma bacteria.

En un principio ignoraremos los errores del tipo falso positivo por que no necesariamente afectaran al seguimiento de las bacterias que nos interesan. Esto es debido a los filtros que se van a introducir posteriormente.

En cuanto los otros dos errores, estos son debidos al radio utilizado como parámetro para TrackMate. A mayor radio elegido, mayor número de errores de superposición se comenten. Al reducir el radio, aumenta el número de errores de detección doble producidos. El principal problema es

que el algoritmo DoG esta diseñado para detectar objetos con forma circular y, sin embargo, nuestros objetos a detectar tienen una forma más alargada. Es por eso que finalmente se ha decidido utilizar un valor de $3.9\mu m$ como parámetro de entrada para el algoritmo, ya que se ha observado que era el valor que minimizaba mejor estos dos tipos de errores.

Tras la detección de las bacterias se realiza un filtrado en una de las características asociadas a las detecciones y denominada calidad por TrackMate. Esta característica se calcula teniendo en cuenta varias características del objeto detectado. Hace referencia a como de fiable es la detección, siendo más fiable cuanto mayor es su valor. Exigiendo valores mayores, se pueden filtrar los casos de falso positivo. Por lo tanto, se ha decidido filtrar el número de objetos en función de su calidad para reducir así el número de falsos positivos.

3.3. Seguimiento de las bacterias.

Como se ha comentado en el anterior capítulo, TrackMate dispone de los siguientes dos algoritmos implementados: LAP y Linear motion LAP. Dado que las bacterias tienen movimiento lineal bajo la aplicación de campos magnéticos externos, se ha decidido utilizar el algoritmo Linear motion LAP.

Los parámetros de entrada para este algoritmo son el radio de búsqueda inicial R_i y el radio de búsqueda tras la predicción R_k, que se muestran en la figura 2.8.

Se va a mostrar el razonamiento que se ha seguido para fijar los valores de ambos parámetros de entrada del algoritmo. Para ello, se ha comenzado realizando el seguimiento con valores de $R_i = 15\mu m$ y $R_k = 10\mu m$. Con estos valores se ha observado que la velocidad de movimiento de las bacterias es inferior a $v_{max} = 60\frac{\mu m}{s}$. Además, la diferencia máxima entre el valor máximo de la velocidad de la bacteria y el valor medio de su velocidad era de $\Delta v_{max} = 20\frac{\mu m}{s}$. También es necesario obtener el tiempo entre fotogramas τ,

$$\tau = \frac{1}{\text{frames/s}} = 0.133s, \tag{3.3}$$

donde frames/s = 7.5 es el número de fotogramas por segundo de la cámara, mostrado en la tabla 2.2.

Tras este análisis preliminar, se pueden establecer valores más precisos de R_i y R_k. Dado que R_i es el radio de búsqueda inicial, tiene que ser menor que el desplazamiento máximo esperado para las bacterias. Como la velocidad máxima es de $60\frac{\mu m}{s}$ y el tiempo que tiene la bacteria para desplazarse es de τ, el desplazamiento máximo será el producto de estos dos valores. Por lo tanto, se obtiene el siguiente valor para R_i,

$$R_i = v_{max}\tau = 60\frac{\mu m}{s} \cdot 0.133s \approx 8\mu m. \tag{3.4}$$

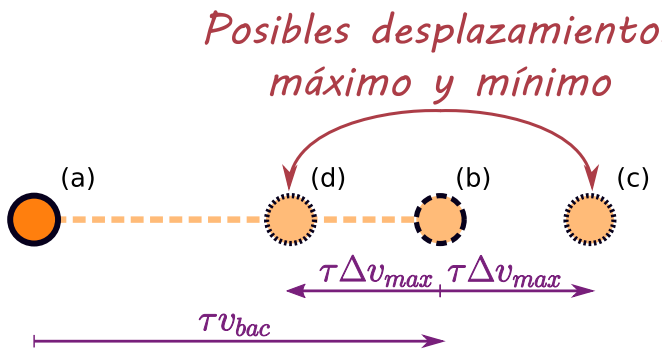


Figura 3.4: Figura explicativa sobre el desplazamiento esperado de la bacteria. En (a)se representa la bacteria inicial, conectada a la posición predicha (b) en base a su velocidad. También se muestra el desplazamiento máximo (c) y mínimo (d) teniendo en cuenta el cambio máximo y mínimo esperado en su velocidad.

En cuanto al valor de R_k, dado que el cambio máximo en la velocidad de una bacteria es menor de $20\frac{\mu m}{s}$, la distancia máxima con respecto a la posición predicha vendrá dada por el cambio máximo en la velocidad. En la figura 3.4 se muestra un esquema de este razonamiento para establecer el valor de R_k, teniendo en cuenta los desplazamientos máximo y mínimos esperados. En esta figura se muestra la posición inicial (a), la posición predicha (b) y las posiciones de desplazamiento máximo (c) y mínimo (d) esperadas. Por todo esto, el radio de búsqueda R_k es,

$$R_k = \tau\Delta v_{max} = 0.133s \cdot 20\frac{\mu m}{s} \approx 3\mu m. \tag{3.5}$$

Es importante destacar en este punto el papel fundamental que tiene el parámetro τ a la hora de realizar el seguimiento. Valores altos de τ dan lugar a seguimientos poco precisos. Esto se debe a que los radios de búsqueda posibles serán mayores, dando lugar a un número mayor de errores cometidos por el algoritmo. Este parámetro está condicionado por los frames/s de la cámara utilizada. Por lo tanto, si se quiere mejorar el seguimiento realizado, es necesario utilizar un equipo que disponga de un mayor frames/s.

Se procederá ahora a analizar los seguimiento que se obtiene de TrackMate para comprobar como de precisos son. Cabe destacar que para los posteriores resultados, se ha realizado un filtrado en los seguimientos. De esta forma, se descartan todos los que no tengan una duración superior a 7 fotogramas. También se filtra estos seguimientos en función de la desviación estándar en el factor de calidad de la bacteria seguida, de forma que se descartan los seguimientos cuyo valor es superior a 10. Este último filtrado se realiza por que se identifica un valor alto en la desviación estándar del factor calidad con un gran número de equivocaciones a la hora de enlazar las bacterias para formar un seguimiento.

3.3.1. Errores cometidos en el seguimiento

Para estimar los errores realizados en el seguimiento, se han analizado tanto los seguimientos realizados por TrackMate, como el número de puntos por seguimiento. Esto último proporciona una idea aproximada de la probabilidad de que TrackMate termine un seguimiento. Este valor tiene interés si lo que se desea hacer es controlar remotamente una bacteria concreta, ya que especifica el tiempo aproximado durante el que va a ser posible controlarla.

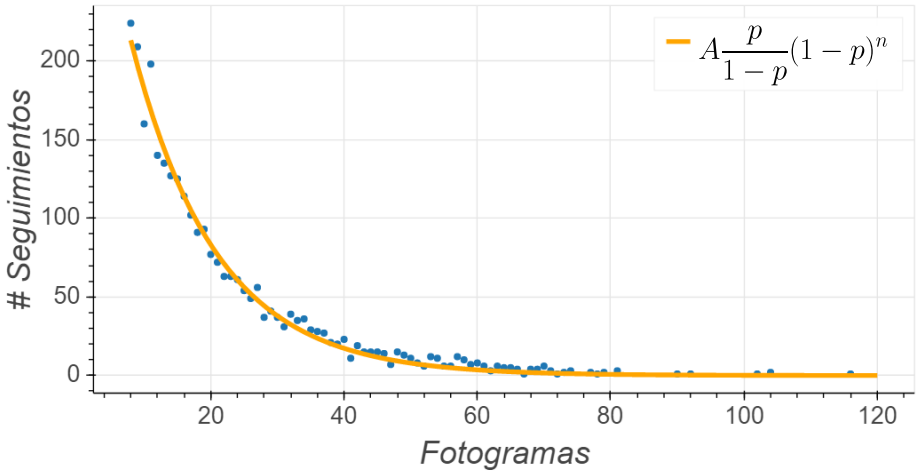


Figura 3.5: Número de seguimientos mantenidos durante un número de fotogramas dado. Se ha realizado un ajuste de estos datos (linea naranja) a una función $A\frac{p}{1-p}(1-p)^n$, siendo n el número de fotogramas. Se ha obteniendo $p = 0.075 \pm 0.001$ y $A = 4890 \pm 60$.

En la figura 3.5 se muestra el número de seguimientos totales con un número de fotogramas dado. Se puede observar un comportamiento exponencial. Este comportamiento se puede asociar a que TrackMate termina un seguimiento entre un fotograma y otro con una probabilidad p que viene condicionada por diversos factores. Siendo esto así, la probabilidad de que un seguimiento termine tras n fotogramas viene determinado por,

$$P(n) = \frac{p}{1-p}(1-p)^n. \tag{3.6}$$

En la figura 3.5 se muestra el ajuste de los datos a la ecuación 3.6.

Es de destacar el número medio de fotogramas en los que se sigue a una bacteria. En este caso, se ha obtenido que el valor medio es de 21 fotogramas, lo cual es un valor extremadamente bajo si se quiere controlar individualmente a una bacteria. Más concretamente, dado que el tiempo entre fotogramas es $\tau = 0.133s$, el tiempo medio de seguimiento de una bacteria sería tan solo de $2.8s$.

Si se quiere aumentar el tiempo medio de seguimiento, convendría mejorar dos factores principalmente. En primer lugar, el número de fotogramas por segundo. Como ya se ha discutido anteriormente, a mayor número de fotogramas por segundo, menor es el desplazamiento entre fotogramas de las bacterias, siendo más fácil realizar el seguimiento de estas. Por otro lado, una menor densidad de bacterias afectaría de manera muy favorable al seguimiento, permitiendo así continuarlo durante más tiempo.

Sin embargo, este tiempo de seguimiento es suficiente si lo que se quiere es analizar las características del movimiento de las bacterias, que es lo que se ha hecho en este trabajo.

Para obtener más información sobre la robustez del seguimiento, se ha decidido realizar una clasificación de los distintos errores que daban lugar a la finalización del seguimiento. Como se podrá observar en la sección 3.4, se ha observado que existen dos grupos de bacterias con velocidades medias de movimiento distintas: un grupo con velocidades superiores a $12\frac{\mu m}{s}$ (bacterias activas) y otro con velocidades inferiores a $12\frac{\mu m}{s}$ (bacterias inactivas). Con este dato en mente, se ha decidido comprobar manualmente varias muestras de seguimientos seleccionadas aleatoriamente, para así obtener la robustez con la que se realiza el seguimiento.

Se ha procedido de la siguiente forma. Tras seleccionar los seguimientos a analizar, solo bacterias activas, se ha comprobado como de exacto ha sido el seguimiento realizado por TrackMate. Para poder realizar un análisis cuantitativo, se han clasificado los seguimientos analizados en cuatro grupos:

Grupo 1: En este grupo los seguimientos se han realizado correctamente.

Grupo 2: En este grupo los seguimientos cuentan con errores en el tracking debidos a una detección incorrecta. Este error ocurre al final del seguimiento realizado por TrackMate, concluyendo en la finalización prematura del seguimiento. Algunas causas más concretas pueden ser:

- No se ha detectado la bacteria que se estaba siguiendo.
- Se han superpuesto dos bacterias impidiendo una detección correcta que ha afectado al posterior seguimiento.

Grupo 3: En este grupo los seguimientos finalizan prematuramente debido a un cambio de sentido en el movimiento de la bacteria. Este cambio de sentido se ha realizado sin observar ningún giro en la bacteria, es decir, la bacteria se sigue moviendo en la dirección de las lineas de campo al realizar el cambio de sentido.

Grupo 4: En este grupo los seguimientos se caracterizan por tener algún error durante el seguimiento a diferencia de los anteriores, que solo tenían errores al final de su seguimiento. Este tipo de errores suelen estar dados principalmente por una superposición en las bacterias.

El resultado de este análisis se muestra en la figura 3.6, donde se observa que el grupo que perjudica a los posteriores análisis, el grupo 4, es una minoría.

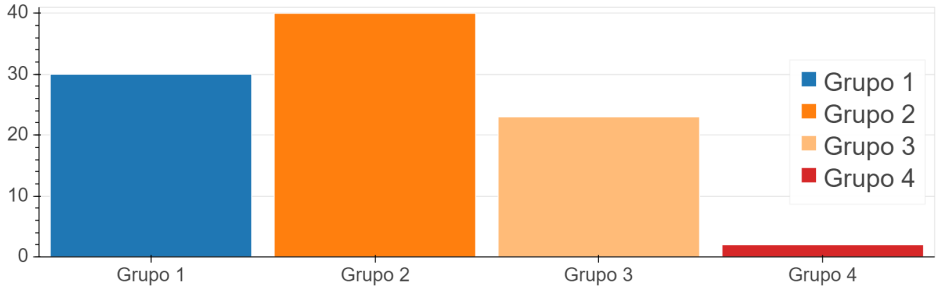


Figura 3.6: Análisis de los errores realizados por TrackMate en el seguimiento de las bacterias. El número total de seguimientos analizados es 95, todos pertenecientas al grupo de bacterias activas.

También es importante destacar que los errores cometidos por el grupo 3 son debidos principalmente al algoritmo de seguimiento utilizado. Este algoritmo funciona mejor cuanto más uniforme sea el movimiento de la bacteria y es por eso que los cambios en el sentido del movimiento afectan muy negativamente al seguimiento, terminando por concluirlo prematuramente. Cabe destacar que se ha observado que todos estos seguimientos son de corta duración. En la sección 3.4.2 se volverá a estos resultados para dar una posible explicación a este comportamiento.

En cuanto al grupo 2, este tipo de errores son debidos principalmente a la alta densidad de bacterias. En cualquier caso, como estos errores dan lugar a la finalización del seguimiento, se concluye que no afectaran a los datos obtenidos posteriormente.

Debido a estas consideraciones, como el único grupo del que se obtendrán datos incorrectos es el 4, se ha concluido que los resultados posteriores serán correctos y que están limitados en precisión por otros factores.

3.4. Análisis de los datos

Durante esta sección se van a analizar los resultados del seguimiento. De esta forma, se obtiene la velocidad del movimiento de las bacterias, su dirección y sentido de movimiento, la fracción de bacterias activas y el tamaño de las bacterias. También se compararán algunos de estos resultados con trabajos realizados por otros grupos, observando un claro acuerdo entre los resultados.

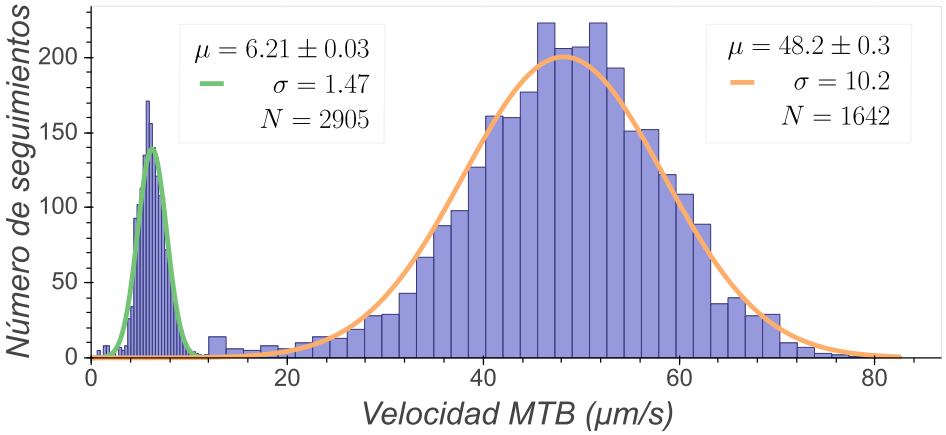


Figura 3.7: Distribución de velocidades medias de las trayectorias seguidas por las bacterias con campo magnético paralelo al canal. Los valores obtenidos para las distribuciones expresados como $\mu \pm \sigma$ son $6.2 \pm 1.5 \frac{\mu m}{s}$ para las bacterias inactivas y $48.2 \pm 10.2 \frac{\mu m}{s}$ para las bacterias inactivas. El número total de trayectorias analizadas es 4547.

3.4.1. Velocidad de las bacterias

Se comienza representado el histograma de las velocidades de movimiento de la bacteria, el cual se muestra en la figura 3.7. Esta figura deja clara la existencia de dos grupos de bacterias bien diferenciados. Al grupo de bacterias con $v > 12 \frac{\mu m}{s}$ se le hará referencia como el grupo de bacterias activas y al grupo con $v < 12 \frac{\mu m}{s}$ se le hará referencia como bacterias inactivas.

Tras observar con más detalle las bacterias inactivas, se ha llegado a la conclusión de que su baja velocidad se debe a que están muertas, o a que no disponen de flagelos para moverse. Se atribuye su velocidad no nula a un flujo del medio en el que se encuentran.

Este movimiento, debido al flujo del medio en el que se encuentran las bacterias, ya ha sido observado anteriormente en [9] y utilizado para estimar la velocidad del flujo del medio en el que se encuentran las bacterias. De esta forma, en [9] utilizan el movimiento de las bacterias inactivas para estimar la velocidad del fluido. Estos datos son utilizados posteriormente en [9] para analizar el movimiento de la bacteria *Magnetospirilium magneticum* ante flujos en el medio.

La velocidad de nuestras bacterias activas se ha obtenido con flujo en el mismo sentido y dirección de movimiento de las bacterias. En principio se podría pensar que la velocidad de la bacteria en condiciones en las que no hay flujo sería la resta de la velocidad obtenida menos la velocidad del flujo. Sin embargo, en [9] se observa que la velocidad de la especie MTB observada no varía prácticamente para flujos pequeños. Esto nos lleva a concluir que la

velocidad de movimiento de la bacteria *Magnetospirillum gryphiswaldense*, cuando se encuentra activa, es de media $48 \pm 10\frac{\mu m}{s}$. También concluimos en este análisis que la velocidad del flujo en el que se encuentran estas bacterias es de $6 \pm 2\frac{\mu m}{s}$.

Las velocidades obtenidas en nuestro caso, son cercanas a las obtenidas en otra publicación[10]. En este trabajo se obtiene, para la misma especie de bacteria utilizada por nosotros, una velocidad de movimiento sin flujo de $49.5 \pm 8.6\frac{\mu m}{s}$, expresado como $\mu \pm \sigma$.

Con respecto a estos resultados, cabe destacar que solo se ha analizado la velocidad de movimiento de la bacteria para un cultivo. Cuando se ha querido repetir el experimento, no se ha podido conseguir un número significativo de bacterias activas que poder analizar.

3.4.2. Direccionalidad del movimiento

Se ha procedido a analizar la dirección de movimiento de las bacterias, tanto para el grupo de las bacterias activas, como para el grupo de las bacterias inactivas. Para ello, se ha medido la desviación de la bacteria con tres métodos diferentes, seleccionando aquel que a proporcionado los resultados más precisos. La tres formas están esquematizadas en la figura 3.8.

El primero de los métodos, figura 3.8 (a), consiste en calcular el valor medio de todos los ángulos entre los puntos sucesivos de un mismo seguimiento. El segundo de ellos, figura 3.8 (b) consiste en ajustar una recta uniendo los extremos de la trayectoria. El tercero de los métodos, figura 3.8 (c), consiste en ajustar una recta por mínimos cuadrados. Se ha elegido este último método para la posterior representación, debido a que el histograma obtenido de esta forma tiene una menor desviación estándar. Sin embargo, cabe destacar que este método no proporciona el sentido de movimiento, solo la dirección.

En la figura 3.9 se muestra la distribución de ángulos de las trayectorias seguidas por las bacterias. En ella se observa un movimiento menos direccional en el caso de las bacterias inactivas. Esto se debe a que su movimiento se asemeja más a uno browninano.

En cuanto a las bacterias activas, se observa que su movimiento tiene una dirección media de 4.0 ± 0.1 grados con una desviación estándar de $\sigma = 4.3$. El alto valor en la desviación estándar puede ser debido a que el eje de la bacteria no está alineado con el eje principal de la cadena de magnetosomas. Esto es algo que ya se ha estudiado en el artículo [11] para la especie *Magnetospirillum magneticum*, donde se concluye que la desalineación entre la cadena y el eje principal de la bacteria es de 6° de media, con máximos de hasta 20°. Esta desalineación puede variar para cada bacteria, dando lugar a la alta desviación estándar obtenida. Cabe destacar que el campo magnético tiene un ángulo como máximo de 3.2°, como puede observarse en la figu-

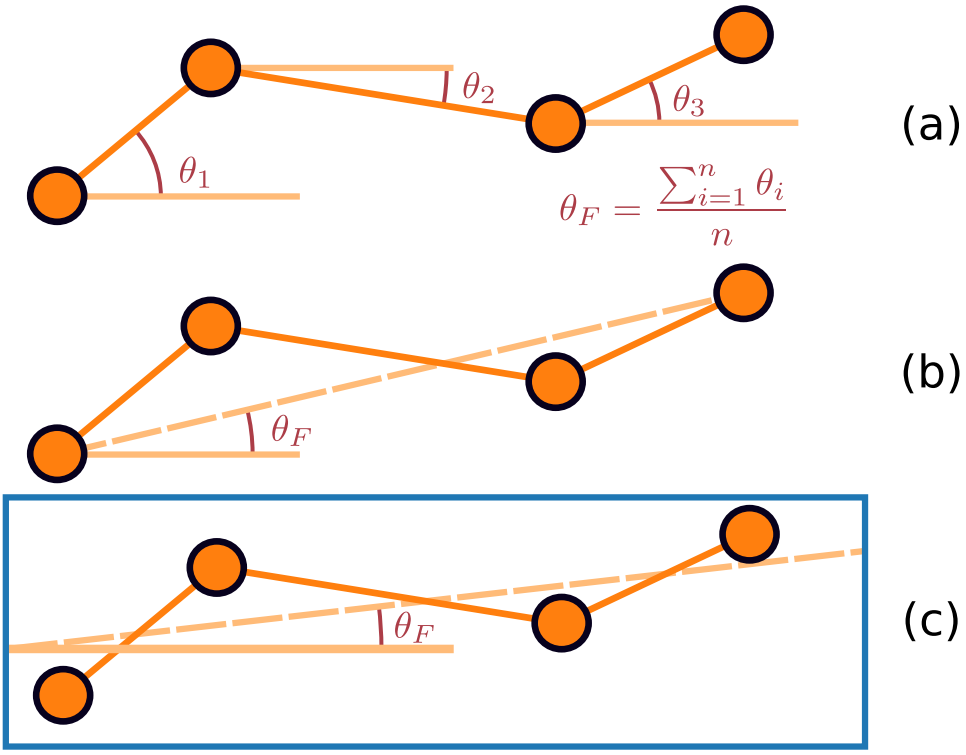


Figura 3.8: Esquema de los tres métodos distintos utilizado para obtener la dirección de movimiento, θ_F, de las bacterias. El método finalmente utilizado, (c), se ha marcado con un recuadro azul.

ra 2.3b. Este valor está dentro de la desviación estándar obtenida para la dirección de movimiento, por lo que no se puede concluir que la diferencia entre el valor de la dirección del campo y la dirección de movimiento de las bacterias sea debida exclusivamente a la desalineación entre la cadena y el eje principal de la bacteria.

Cabe destacar que se han observado bacterias activas con sentido de movimiento opuesto al campo magnético. Esto se ha obtenido calculando el número total de bacterias activas que se mueven con ángulos mayores a 170° en valor absoluto. Para ello, se ha usado el método (b) mostrado en la figura 3.8, ya que el método (c) no proporciona información acerca del sentido de movimiento. De esta forma, se ha obteniendo un total de 31 seguimientos con sentido opuesto al resto (el número total de seguimientos es 1642).

Es importante destacar que el tiempo medio de estos seguimientos es de $1.159s$, siendo el tiempo mínimo impuesto por los filtros comentados anteriormente $1.064s$. Esto pone de relieve la corta duración de estas trayectorias. En cualquier caso, dado que el tiempo de desplazamiento medio para el resto de los seguimientos es de $2.8s$, queda claro que el principal desplazamiento es en el sentido del campo.

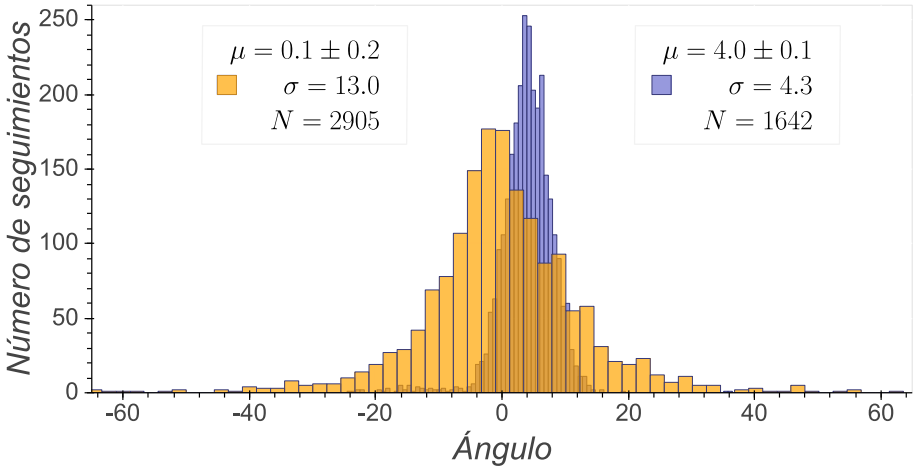


Figura 3.9: Distribución de la dirección de movimiento de las trayectorias seguidas para las bacterias con $v < \frac{12\mu m}{s}$ naranja y $v > \frac{12\mu m}{s}$ azul. El número total de trayectorias es 2905 naranja y 1642 azul. El ajuste a una gaussiana expresado como $\mu \pm \sigma$ es: 0 ± 13 (a) y 4 ± 4 (b).

En este punto hay que recordar los resultado obtenidos en la sección 3.3.1, donde se ha mostrado en la figura 3.6 que el 24.2 % de los seguimientos analizados termina por un cambio abrupto en el sentido de movimiento de la bacteria. Este resultado puede resultar contradictorio ya que en esta sección se han detectado tan solo 31 (de 1642) seguimientos en el sentido opuesto al flujo. La razón de esta diferencia se debe a que los seguimientos en el sentido opuesto son de muy corta duración. Esto da lugar a que el filtro impuesto sobre el número mínimo de fotogramas en un seguimiento los descarte.

Las razones por las que se da este cambio brusco en el sentido de la bacteria pueden ser varias. En primer lugar, no es extraño que puedan realizar este tipo de cambios en su sentido, gracias a los dos flagelos que disponen. En el artículo[12] muestran como la especie *Magnetospirillum magneticum* utiliza los dos flagelos de los que dispone para moverse en un sentido u otro y pueden mostrar cambios de sentido repentinos al chocar con un objeto. Esta especie no es con la que nosotros hemos estado trabajando, pero se espera que tenga un comportamiento similar.

En cuanto a las razones sobre este comportamiento, las imágenes del movimiento han sido tomadas enfocando cerca del fondo del canal. Es posible que este fondo afecte al movimiento de las bacterias dando lugar al cambio de sentido observado. Por otro lado, en el artículo [13] se menciona un comportamiento similar para la bacteria *E.Coli*. Más concretamente, se muestra un seguimiento de esta bacteria donde su dirección de movimiento cambia rápidamente cada cierto tiempo. La explicación que se muestra en ese artículo, es que la bacteria cambia su movimiento para explorar las condiciones de

habitabilidad en otra dirección. También se menciona en este artículo, que la bacteria seguida tiende a moverse durante una mayor distancia hacia los lugares con condiciones más favorables. Sin embargo, cabe destacar que la bacteria *E.Coli* no es siquiera magnetotáctica, por lo que el comportamiento y los motivos del mismo no tienen que ser similares. En nuestro caso, se ha observado que la mayor parte de las bacterias se mueven en un mismo sentido, posiblemente debido a la existencia de un gradiente de oxígeno. En cualquier caso, se desconoce la razón final para este comportamiento.

3.4.3. Fracción de bacterias activas

En esta sección se va a obtener una estimación de las bacterias activas. Una primera opción para obtener esta estimación podría consistir en contar el número total de seguimientos de bacterias activas y bacterias inactivas y obtener la fracción a partir de estos valores. Sin embargo, dado que el seguimiento de una bacteria puede dividirse (debido a los errores cometidos en el seguimiento) en dos seguimientos distintos, este valor medido para la fracción de bacterias activas daría resultados incorrectos. Además, como la velocidad de las bacterias activas es muy superior a la de las inactivas, es de esperar que el número de bacterias activas totales que atraviesan la imagen sea muy superior al número de bacterias inactivas, sobrestimando así la fracción de bacterias activas.

Para obtener una representación más fiable de esta fracción de bacterias activas, conviene calcular el número de seguimientos de bacterias activas e inactivas en cada fotograma. De esta forma, se obtiene el histograma de la fracción de bacterias activas mostrado en la figura 3.10. El resultado obtenido expresado como $\mu \pm \sigma$ es una fracción de 0.553 ± 0.028 bacterias activas.

Con respecto a este resultado, se ha observado en experimentos posteriores una fracción de bacterias in-usualmente inactivas que ha impedido en varias ocasiones analizar más resultados. Se desconoce la causa de este hecho, y por lo tanto, convendría en un futuro analizar cuales son las causas que afectan severamente a la actividad de la MTB. Este análisis beneficia no solo al análisis de su movimiento, sino también a su posible futuro uso como micro-robots.

3.4.4. Tamaño de las bacterias

TrackMate dispone de la capacidad de calcular la morfología de los objetos detectados. En concreto, es posible obtener información acerca del tamaño de las bacterias. Esto se hace por medio del ajuste de una función gaussiana de varias variables. Con este resultado, se puede obtener la longitud del semieje mayor mostrada en el histograma de la figura 3.11. Se observa que el semieje mayor tiene una longitud, expresada como $\mu \pm \sigma$ de

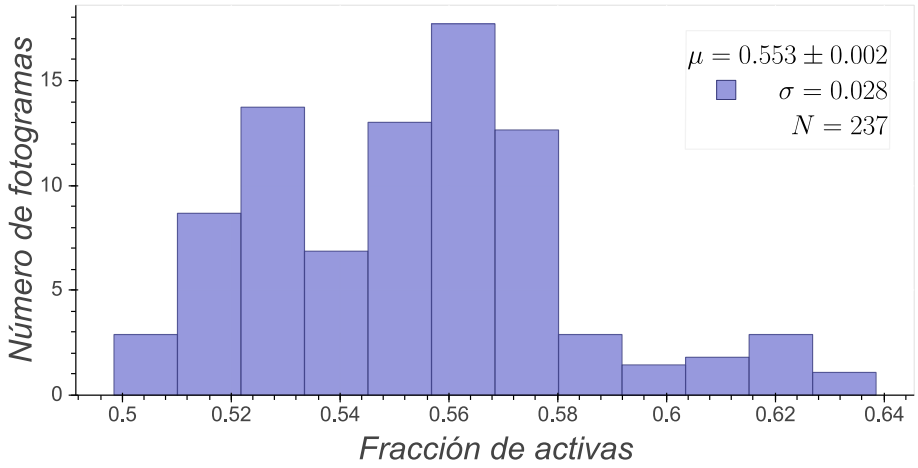


Figura 3.10: Histograma de la fracción total de bacterias activas. El valor medio de la fracción de bacterias activas expresado como $\mu \pm \sigma$ es $f = 0.55 \pm 0.03$.

$4.2 \pm 0.5 \mu m$, resultado similar al obtenido mediante el uso de un microscopio TEM. Sin embargo, el error obtenido para esta longitud esta condicionado fuertemente por el tamaño de $0.227 \mu m$ del píxel. De esta misma forma, ya que la anchura de la bacteria es de aproximadamente $0.5 \mu m$, no se obtienen resultados significativos de su medida mediante este método.

También cabe destacar el efecto negativo que tiene la alta densidad de bacterias en el medio, dando lugar a resultados poco fiables sobre la orientación de las bacterias y su tamaño. El efecto que tiene la densidad alta de bacterias se muestra en la figura 3.12, donde se observa que la proximidad de las bacterias afecta negativamente a la estimación de las características ya comentadas.

En definitiva, el método seguido para la medida de la longitud de las bacterias no es adecuado. Aunque el objetivo era obtener información sobre posibles correlaciones de parámetros con la longitud, por ejemplo la velocidad, la baja resolución de las imágenes y la alta densidad de bacterias no permite obtener información relevante.

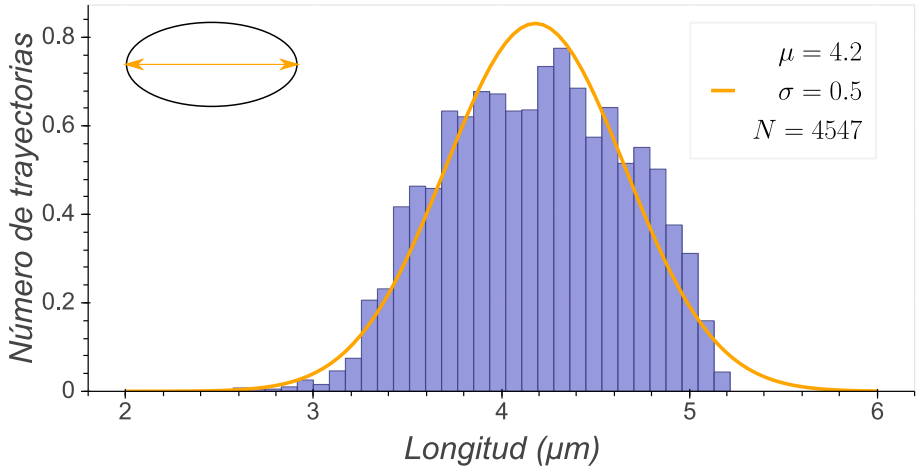
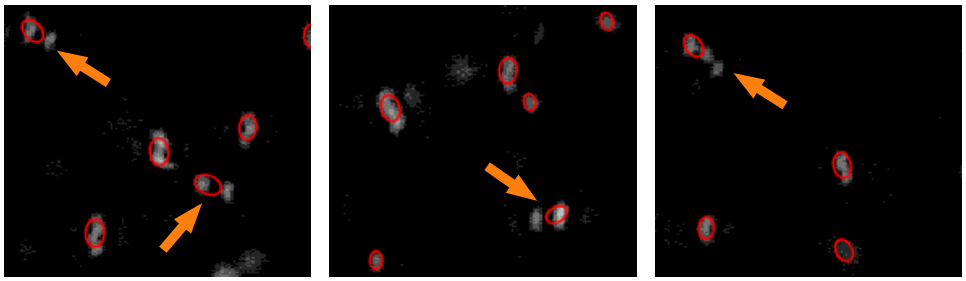


Figura 3.11: Longitud de las bacterias para el eje mayor marcado en naranja. Los valores del ajuste a una gaussiana expresado como $\mu \pm \sigma$ son $4.2 \pm 0.5 \mu m$. En este caso no se ha calculado el error de la media debido a que hay muchos factores que afectan negativamente a la media de de este valor.



(a) (b) (c)

Figura 3.12: Imagen donde se muestran algunos resultados obtenidos del análisis de la morfología obtenidos por TrackMate. Con una flecha naranja se señalan los ajustes a una elipse claramente erróneos.

Capítulo 4

Conclusiones

El objetivo del trabajo ha sido analizar el movimiento de las bacterias para contribuir en el desarrollo de tecnologías con posible uso en el ámbito médico.

Para ello, se ha realizado un estudio exhaustivo de la configuración de campos magnéticos utilizados, así como de los algoritmos disponibles en TrackMate para el seguimiento y detección de las bacterias. En cuanto a este estudio, se ha concluido que el equipo experimental es adecuado para el análisis del movimiento lineal de las bacterias, ya que los campos magnéticos son suficientemente intensos como para orientarlas. Sin embargo, si se desea en un futuro trabajo realizar un control preciso sobre la bacteria, será necesario utilizar una densidad menor de bacterias y una cámara con mayor fotogramas por segundo para poder así facilitar el seguimiento realizado.

En consideración a los resultados del capitulo 3, se ha comenzado discutiendo los parámetros utilizados para los distintos algoritmos, así como los errores cometidos por estos. Finalmente, se ha concluido que los resultados no están limitados por estos errores.

De esta forma, se ha podido obtener la velocidad media de las bacterias con una alta precisión y se ha comparado satisfactoriamente este valor con el proporcionado por un artículo publicado por otro grupo. Se destaca en este punto la existencia de dos grupos de bacterias, activas e inactivas, y que el grupo de bacterias inactivas se puede usar para medir la velocidad del flujo del medio en el que navegan las bacterias. En este caso se ha obtenido un flujo con velocidades de $6 \pm 2\frac{\mu m}{s}$, expresado como $\mu \pm \sigma$. En cuanto a las bacterias activas analizadas se refiere, se mueven con velocidades medias de $48 \pm 10\frac{\mu m}{s}$, expresado como $\mu \pm \sigma$.

También se ha analizado la dirección de movimiento, estando esta de acuerdo con la caracterización realizada para el campo magnético. En cuanto a la dirección de movimiento, se ha obtenido una notable diferencia con respecto a la anchura de las distribuciones mostradas en la figura 3.9 para bacterias activas e inactivas, debido a que la naturaleza de cada uno de estos

movimientos es diferente. También es notable destacar que se han observado cambios de sentido repentinos en el movimiento de las bacterias, los cuáles duran un corto periodo de tiempo.

Por otro lado, se ha obtenido que la fracción de bacterias activas con las que se ha trabajado es de 0.553 ± 0.028. Sin embargo, no se ha podido repetir el análisis del movimiento de las bacterias con diferentes cultivos debido a la extremadamente baja actividad encontrada. En esta línea de trabajo se puede analizar cuales son los factores decisivos que mejoran la actividad de las bacterias.

Bibliografía

[1] A. S. Mathuriya, "Magnetotactic bacteria for cancer therapy," *Biotechnology letters*, vol. 37, no. 3, pp. 491–498, 2015.

[2] E. Alphandéry, S. Faure, L. Raison, E. Duguet, P. Howse, and D. Bazylinski, "Heat production by bacterial magnetosomes exposed to an oscillating magnetic field," *The Journal of Physical Chemistry C*, vol. 115, no. 1, pp. 18–22, 2010.

[3] O. Felfoul, M. Mohammadi, S. Taherkhani, D. De Lanauze, Y. Z. Xu, D. Loghin, S. Essa, S. Jancik, D. Houle, M. Lafleur, *et al.*, "Magneto-aerotactic bacteria deliver drug-containing nanoliposomes to tumour hypoxic regions," *Nature nanotechnology*, vol. 11, no. 11, p. 941, 2016.

[4] K. Bente, A. Codutti, F. Bachmann, and D. Faivre, "Biohybrid and bioinspired magnetic microswimmers," *Small*, vol. 14, no. 29, p. 1704374, 2018.

[5] J.-Y. Tinevez, N. Perry, J. Schindelin, G. M. Hoopes, G. D. Reynolds, E. Laplantine, S. Y. Bednarek, S. L. Shorte, and K. W. Eliceiri, "Trackmate: An open and extensible platform for single-particle tracking," *Methods*, vol. 115, pp. 80–90, 2017.

[6] J. Schindelin, I. Arganda-Carreras, E. Frise, V. Kaynig, M. Longair, T. Pietzsch, S. Preibisch, C. Rueden, S. Saalfeld, B. Schmid, *et al.*, "Fiji: an open-source platform for biological-image analysis," *Nature methods*, vol. 9, no. 7, p. 676, 2012.

[7] E. Jones, T. Oliphant, P. Peterson, *et al.*, "SciPy: Open source scientific tools for Python," 2001.

[8] Bokeh Development Team, *Bokeh: Python library for interactive visualization*, 2014.

[9] S. Rismani Yazdi, R. Nosrati, C. A. Stevens, D. Vogel, P. L. Davies, and C. Escobedo, "Magnetotaxis enables magnetotactic bacteria to navigate in flow," *Small*, vol. 14, no. 5, p. 1702982, 2018.

[10] M. Pichel, T. Hageman, I. Khalil, A. Manz, and L. Abelmann, "Magnetic response of magnetospirillum gryphiswaldense," *arXiv preprint arXiv:1710.00405*, 2017.

[11] L. L. Nagard, L. Yu, M. Rajkotwala, S. Barkley, D. Bazylinski, A. P. Hitchcock, and C. Fradin, "Misalignment between the magnetic dipole moment and the cell axis in the magnetotactic bacterium *Magnetospirillum magneticum* AMB-1," *Physical Biology*, 2019.

[12] S. Rismani Yazdi, R. Nosrati, C. A. Stevens, D. Vogel, and C. Escobedo, "Migration of magnetotactic bacteria in porous media," *Biomicrofluidics*, vol. 12, no. 1, p. 011101, 2018.

[13] E. M. Purcell, "Life at low reynolds number," *American journal of physics*, vol. 45, no. 1, pp. 3–11, 1977.

Apéndice A

Programas creados

En este apéndice he introducido algunos de los programas desarrollados en Python 3.7 durante el TFG. En concreto, el apéndice A.1 muestra el programa realizado para entender el funcionamiento de los algoritmos de detección. El apéndice A.2 muestra el proceso seguido para leer y modificar los datos del seguimiento realizado por TrackMate. Los apéndices A.3 y A.4 muestran, el programa desarrollado para guardar varias selecciones de seguimiento en un archivo XML que pueda ser leído por TrackMate y el programa que muestra las figuras de los datos del análisis de estas muestras de seguimiento. Cabe destacar que las imágenes obtenidas en algunas de las salidas de los programas mostrados en A.4 y A.1 no han sido posible incluirlas.

Otros programas desarrollados y que no han sido incluidos son los utilizados para visualizar los datos del seguimiento y los utilizados para automatizar la llamada al programa de TrackMate mediante Python. Tampoco se han incluido algunos scripts que permiten automatizar el procesado de imágenes en TrackMate.

A.1. Programa para analizar los algoritmos de detección

Este notebook tiene como objetivo analizar entender el funcionamiento de los algoritmos empleados por TrackMate. Para ello, se muestra mediante un ejemplo como funcionan.

In [1]:
```
import numpy as np

import skimage
from skimage.feature import peak_local_max
from skimage import exposure, io, data, img_as_float
import matplotlib.pyplot as plt
import scipy.ndimage as tools


from bokeh.layouts import gridplot
from bokeh.plotting import figure, show, output_notebook
from bokeh.io import export_png
import bokeh
output_notebook()

import matplotlib.pyplot as plt

%matplotlib notebook
folder_to_save = r'C:\Users\agali\Dropbox\TFGBac\TFG_document\images\ImagesTheory'
```

In [2]:
```
#Estas son algunas funciones que se utilizarán para mostrar las imágenes y los histogram

def plotImageBokeh(images,showPlot=True):
    from bokeh.layouts import gridplot
    #Una función simple que imprime la imagen introducida como argumento
    figures = []
    for im in images:
        aspect_ratio=im.shape[0]/im.shape[1]
        p = figure(x_range=(0,im.shape[1]), y_range=(0,im.shape[0]),plot_width=400, plot
        bokeh.palettes.gray(256)
        #p.x_range.range_padding = p.y_range.range_padding = 0
        p.xaxis.visible = False
        p.yaxis.visible = False

        p.image([im], x=0, y=0, dw=im.shape[1], dh=im.shape[0], palette = bokeh.palettes
        figures.append(p)
    if showPlot:
        grid = gridplot(figures,ncols = 2)
        show(grid)
    else:
        grid = figures
    return grid


def plot_hist(data):
```

```
    #Una función simple que imprime el histograma introducido en el argumento data.
    p = figure( background_fill_color="#fafafa",plot_width=400, plot_height=400,x_range

    hist, edges = np.histogram(data,bins=500,range =(data.min(),data.max()))
    p.quad(top=hist, bottom=0, left=edges[:-1], right=edges[1:],
           fill_color='navy', line_color="navy", alpha=0.5)

    #make facy plot
    p.y_range.start = 0
    p.y_range.end = hist.max() + hist.max()*5/100
    p.xaxis.axis_label = 'Intensidad'
    p.yaxis.axis_label = 'Número de píxeles'
    p.grid.grid_line_color = "white"

    show(p)

    return p

def plotImageMatlplotlib(images,ncols=1,figsize = (8,3)):
    n = len(images)
    nrows = n//ncols
    if n%ncols != 0: nrow += 1
    fig, axes = plt.subplots(nrows, ncols, figsize=figsize, sharex=True, sharey=True)
    if not n-1==0:
        ax = axes.ravel()
        for i in range(n):
            ax[i].axis('off')
            ax[i].imshow(images[i], cmap=plt.cm.gray)
    else:
        axes.axis('off')
        axes.imshow(images[0], cmap=plt.cm.gray)
    return fig,axes
```

1 Detección de regiones de interés

El siguiente cuaderno tiene como objetivo mostrar el funcionamiento de algunos algoritmos usados para la detección de regiones de interés.

Se utilizará como ejemplo una imagen con elipses de distintos tamaños y diferentes orientaciones. También se añadirá ruido a la imagen para que este caso ficticio se asemeje al caso de querer detectar bacterias en una imagen.

```
In [3]: from skimage.draw import ellipse as draw_ellipse
        from skimage.util import random_noise as add_noise
        ellipses = np.zeros((200,300))
        #-----------------y------x----
        ellipses[draw_ellipse(r=35, c=70, r_radius=15, c_radius=30)] = 100
        ellipses[draw_ellipse(r=85, c=70, r_radius=20, c_radius=40)] = 100
```

```
ellipses[draw_ellipse(r=150, c=70, r_radius=25, c_radius=50)] = 100
ellipses[draw_ellipse(r=35, c=200, r_radius=15, c_radius=30,rotation=45)] = 100
ellipses[draw_ellipse(r=85, c=210, r_radius=20, c_radius=40,rotation=45)] = 100
ellipses[draw_ellipse(r=150, c=230, r_radius=25, c_radius=50,rotation=45)] = 100

ellipses[draw_ellipse(r=25, c=150, r_radius=5, c_radius=10,rotation=-0)] = 100
ellipses[draw_ellipse(r=75, c=150, r_radius=5, c_radius=10,rotation=-15)] = 100
ellipses[draw_ellipse(r=125, c=150, r_radius=5, c_radius=10,rotation=-25)] = 100
ellipses[draw_ellipse(r=175, c=150, r_radius=5, c_radius=10,rotation=-35)] = 100
ellipses[draw_ellipse(r=50, c=150, r_radius=5, c_radius=10,rotation=-45)] = 100
ellipses[draw_ellipse(r=100, c=150, r_radius=5, c_radius=10,rotation=-55)] = 100
ellipses[draw_ellipse(r=150, c=150, r_radius=5, c_radius=10,rotation=-65)] = 100


ellipses[draw_ellipse(r=20, c=270, r_radius=5, c_radius=10,rotation=-65)] = 100

ellipses = add_noise(ellipses,'gaussian',seed = 1,clip = False,mean=50,var=1000)


plotImageBokeh([ellipses])
```

```
Out[3]: Column(id='1059', ...)
```

A continuación se muestra un filtro gaussiano. Se va a utilizar este filtro como modelo y se va a realizar la correlación de imagen con la imagen de las bacterias. De esta forma, se espera que los detalles más grandes se difuminen dando importancia a objetos de tamaño aproximado igual a la varianza del filtro.

```
In [4]: mask_gaussian_filter = np.zeros((300,300))

        mask_gaussian_filter[mask_gaussian_filter.shape[0]//2,mask_gaussian_filter.shape[1]//2]
        mask_gaussian_filter = tools.gaussian_filter(mask_gaussian_filter,(30,30))
        plotImageBokeh([mask_gaussian_filter])
```

```
Out[4]: Column(id='1182', ...)
```

Se puede observar en las imágenes de abajo el efecto de utilizar el filtro gaussiano. Lo que se ha hecho es realizar la correlación de la imagen con el funciones gaussianas de distinta varianza. Esto da lugar a que, cuanto mayor es la varianza, menor es el detalle de las imágenes. De esta forma, con varianzas grandes solo se detectan regiones de tamaño grande.

```
In [5]: %matplotlib notebook
        for r in [0,10,17,30]:
            ellipse_smothed = tools.gaussian_filter(ellipses,(r,r),mode='reflect')
            coordinates = peak_local_max(ellipse_smothed, min_distance=5,indices = True)

            images_to_show = [ellipses,ellipse_smothed,ellipses]
            fig,ax = plotImageMatlplotlib(images_to_show,ncols=3)
```

```
            ax[2].plot(coordinates[:,1], coordinates[:,0], 'r.')
            ax[1].set_title('sigma = {}'.format(r))
```

```
<IPython.core.display.Javascript object>
```

```
<IPython.core.display.HTML object>
```

```
<IPython.core.display.Javascript object>
```

```
<IPython.core.display.HTML object>
```

```
<IPython.core.display.Javascript object>
```

```
<IPython.core.display.HTML object>
```

```
<IPython.core.display.Javascript object>
```

```
<IPython.core.display.HTML object>
```

1.1 Filtro LoG

En vez de utilizar el filtro gaussiano, es más común utilizar el filtro Laplacian of Gaussian (LoG) que tiene mejores respuestas a regiones de tamaño $\sqrt{2}\sigma$. A continuación se muestra la forma del filtro. Este filtro tiene simetría de revolución y toma valor cero para $x^2 + y^2 = 2\sigma^2$.

```
In [6]: def f(x,var):
            var2 = var**2
            var4 = var**4
            pi = np.pi
            return -1/np.sqrt(2*pi*var2)*(1/var2 - x*x/var4)*np.exp(-x*x/(2*var2))


        x = np.linspace(-5,5,1000)
        y = f(x,1)

        p = figure()
        p.line(x,y,color = 'orange',legend = 'LoG',line_width=5)


        p.xaxis.axis_label_text_font_size = "30pt"
        p.yaxis.axis_label_text_font_size = "30pt"
```

```
        p.xaxis.major_label_text_font_size = '17pt'
        p.yaxis.major_label_text_font_size = '17pt'
        p.legend.label_text_font_size = "27pt"


        p.xaxis.axis_label = "x"
        p.yaxis.axis_label = 'Intensidad'




        show(p)
```

```
In [7]: image = np.zeros((200,200))
        image[100,100] = 1

        #Para mostrar una imagen del filtro se va a correlaciconar el filtro con una imagen dond
        #Esto tiene como consecuencia que la imagen resultante de la correlación se el propio fi
        #A partir de esta imágen se ha cogido el caso y=100, obteniendo así una representación b
        image = tools.gaussian_filter(image,sigma = 20,order = 0)
        image = -tools.laplace(image,mode='constant')


        fig,ax = plotImageMatlplotlib([image])
        #Se va a dibujar en rojo la zona donde la función toma valor cero.
        circle = plt.Circle((100,100),20*np.sqrt(2),edgecolor = 'red',fill = False,linestyle = '
        ax.add_artist(circle)
        plt.show()
        fig.savefig(folder_to_save + r'\LoG_1D.pdf', bbox_inches='tight',pad_inches = 0)
```

```
<IPython.core.display.Javascript object>


<IPython.core.display.HTML object>
```

Se va a aplicar este filtro y buscar los nuevos máximos de intensidad.

Realizando esta operación se observa que para distintos distintos valores de σ se detectan distintos objetos de manera correcta. Para σ pequeños, las elipses pequeñas dan una respuesta muy fuerte. Para σ grande, los las elipses grandes las que dan una fuerte respuesta.

Con objetivo de obtener de manera exacta la posición de las elipses, además de su tamaño, se va a analizar la intensidad de varios puntos de la imagen tras haber aplicado el filtro LoG con distintos valores de σ.

```
In [9]: images_to_plot = [ellipses]
        fig.tight_layout()
        fig,ax = plotImageMatlplotlib(images_to_plot,ncols = 1)
        colors = ["blue", "orange", "green", 'darkviolet']
```

```
        points = np.array([[25,150],[150,70],[35,200],[108,190]])

        for i in range(len(colors)):
            ax.plot(points[i,1], points[i,0], '.',markersize=10,color = colors[i])
        plt.savefig(folder_to_save + r'\LoG_aplicacion\sigma_0.pdf',bbox_inches='tight')
```

```
<IPython.core.display.Javascript object>
```

```
<IPython.core.display.HTML object>
```

```
In [10]: points = np.array([[25,150],[150,70],[35,200],[108,190]])
         colors = ["blue", "orange", "green", 'darkviolet']
         for r in [5,13,23,33]:
             ellipse_smothed = -r**2 * tools.gaussian_laplace(ellipses,sigma = r,mode='reflect')
             coordinates = peak_local_max(ellipse_smothed, min_distance=10,indices = True)

             images_to_plot = [ellipse_smothed]
             fig.tight_layout()
             fig,ax = plotImageMatlplotlib(images_to_plot,ncols = 1)
             ax.set_title('sigma = {}, r = {}'.format(r,np.sqrt(2)*r))
             for i in range(len(colors)):
                 ax.plot(points[i,1], points[i,0], '.',markersize=10,color = colors[i])
             ax.plot(coordinates[:,1], coordinates[:,0], 'r.')
             #plt.savefig(folder_to_save + r'\LoG_aplicacion\sigma_{}.pdf'.format(r),bbox_inches
```

```
<IPython.core.display.Javascript object>
```

```
<IPython.core.display.HTML object>
```

```
<IPython.core.display.Javascript object>
```

```
<IPython.core.display.HTML object>
```

```
<IPython.core.display.Javascript object>
```

```
<IPython.core.display.HTML object>
```

```
<IPython.core.display.Javascript object>
```

```
<IPython.core.display.HTML object>
```

```
In [11]: x = np.linspace(1,30,100)
         y = []
         y2 = []
         y3 = []
         y4 = []

         for r in x:
             ellipse_smothed = -r**2*tools.gaussian_laplace(ellipses,sigma = r, mode='reflect')
             y.append(-ellipse_smothed[ 25, 150])
             y2.append(-ellipse_smothed[150,  70])
             y3.append(-ellipse_smothed[ 35, 200])
             y4.append(-ellipse_smothed[108, 190])

         x_data = [x,x,x,x]
         y_data = [y,y2,y3,y4]


         p = figure()
         x_data = [x,x,x,x]
         y_data = [y,y2,y3,y4]
         legend = ['Elipse pequeña', 'Elipse grande','Elipse rotada','Esquina de elipse']
         color  = ["blue", "orange", "green", 'darkviolet']
         for i in range(len(color)):
             p.line(x_data[i],y_data[i],color=color[i],legend = legend[i],line_width=5)

         p.xaxis.axis_label = "σ"
         p.yaxis.axis_label = 'Intensidad'
         p.xaxis.axis_label_text_font_size = "30pt"
         p.yaxis.axis_label_text_font_size = "30pt"
         p.xaxis.major_label_text_font_size = '17pt'
         p.yaxis.major_label_text_font_size = '17pt'

         p.legend.location = 'top_right'
         p.legend.label_text_font_size = "27pt"

         show(p)
```

Como se puede observar, se obtiene un pico de intensidad en los píxeles centrales de las elipse para filtros LoG donde σ es aproximadamente el tamaño de la elipse. Esto no ocurre si usamos el filtro gausiano, como puede mostrarse a continuación. Por lo tanto, se concluye que el filtro LoG proporciona mejor respuesta que el gaussiano.

Para obtener las distintas regiones de la imagen, se busca máximos locales de (x, y, σ^2). Es decir, se busca que sea máximo tanto para la escala σ usada, como para la posición.

También se puede eliminar la respuesta indeseada de las esquinas (curva morada) teniendo en cuenta que su respuesta es diferente si se usar el determinante del hessiano como filtro, en vez del LoG.

Si se realiza este mismo análisis con el filtro gaussiano sólo, se observa que la respuesta no proporciona ninguna información sobre el tamaño de la región.

In [12]:
```
x = np.linspace(1,30,100)
y = []
y2 = []
y3 = []
y4 = []

for r in x:
    ellipse_smothed = tools.gaussian_filter(ellipses,sigma = r, mode='reflect')
    y.append(ellipse_smothed[ 25, 150])
    y2.append(ellipse_smothed[150,  70])
    y3.append(ellipse_smothed[ 35, 200])
    y4.append(ellipse_smothed[108, 190])

x_data = [x,x,x,x]
y_data = [y,y2,y3,y4]
legend = ['Elipse pequeña', 'Elipse grande','Elipse rotada','Esquina de elipse']
color  = ["blue", "orange", "green", 'darkviolet']
for i in range(len(color)):
    p.line(x_data[i],y_data[i],color=color[i],legend = legend[i])

p.xaxis.axis_label = "$\sigma$"
p.yaxis.axis_label = 'Intensidad'
p.legend.location = 'bottom_right'
show(p)
```

A.2. Programa para tratar los datos obtenidos de TrackMate

Este notebook tiene como objetivo acceder a los datos en forma de xml que proporcion TrackMate. Además, realiza algunas operaciones que son usadas posteriormente para las representaciones.

In [1]:
```
import numpy as np
import pandas as pd

from ScriptsToUseIJ.CallTracking import callTracking as ct
from ScriptsToUseIJ.DataConverter import *

import matplotlib.pyplot as plt
import matplotlib

from scipy.stats import norm
from scipy.stats import linregress as linreg
import scipy.optimize as scop

import xml.etree.ElementTree as ET

import math as math

from bokeh.layouts import gridplot
from bokeh.plotting import figure, show, output_notebook
output_notebook()
```

1 Loading data

1.1 Load data in df

In [2]:
```
#Getting the tracks info
#Start by making a tree structure out of the xml file
modelDir = r"..\Data\Model_1.xml"

tree = ET.parse(modelDir)

root = tree.getroot()

#Iterate trough all the tracks, getting their attributes
tracksList = list()
for track in root.iter('Track'): #get the attributes of each track
    tracksList.append(track.attrib)

dfTracks = pd.DataFrame(tracksList)#convert those attributes into a dataframe

dfTracks = dfTracks.set_index("name") #change index of the dataframe to be the name of t
dfTracks = dfTracks.apply(pd.to_numeric,errors='coerce') #the values in dataframe are st
```

In [3]:
```
#Getting the spots info
spotList = list()
```

```
for spot in root.iter('Spot'):
    spotList.append(spot.attrib)

dfSpots = pd.DataFrame(spotList)
dfSpots = dfSpots.set_index('name')
dfSpots = dfSpots.apply(pd.to_numeric,errors='coerce')
dfSpots.head()
```

Out[3]:

name	ELLIPSOIDFIT_AXISPHI_A	ELLIPSOIDFIT_AXISPHI_B	ELLIPSOIDFIT_AXISPHI_C	ELLIPSOIDFIT_AXISTHETA_A	\
ID7170	0.0	-0.025627	1.545169	0.0	
ID7171	0.0	-0.181041	1.389756	0.0	
ID7172	0.0	0.177903	-1.392893	0.0	
ID7173	0.0	0.252192	-1.318604	0.0	
ID7175	0.0	0.032540	-1.538257	0.0	

...

In [4]:
```
#Create a dict:
#             -index : Track name
#             -valule: a listo of spots that blong to that track. Sorted by the frame th
#TODO Existe serguro una forma mas eficiente de leer del archivo
#TODO la creacion de s sobra. Se ha añadido al loop el introducir el TRACK id de cada sp
#     No se ha quitado el proceso de creacion de debido a que este es utilizado mas alan

def changeToID(spot):
    #TODO vectorize?
    "change an array of spots id in int form to str for adding the prefix ID"
    spot = np.array(spot)
    spot = spot.astype(str)
    for i in range(spot.size):
        spot[i] = 'ID'+str(spot[i])
    return spot

def sortByFrame(spots):
    series = dfSpots.loc[spots]['FRAME'] #Get a series with the spots we want to sort as
    series = series.sort_values()
    spots = series.index.values
    return spots
```

```
n=0
dfSpots['TRACK_ID'] = ""
spotsInTrack = {}
for track in root.iter("Track"):
    n+=1
    spots =[]
    for edge in track:
        sourceID = int(edge.attrib['SPOT_SOURCE_ID'])
        if sourceID not in spots:
            spots.append(sourceID)
        targetID = int(edge.attrib['SPOT_TARGET_ID'])
        if targetID not in spots:
            spots.append(targetID)
    spots = changeToID(spots)
    dfSpots.loc[spots, 'TRACK_ID'] = track.attrib['name']
    spots = sortByFrame(spots)
    spotsInTrack[track.attrib["name"]] = spots

FilteredTrackIndex = dfTracks.index.values
s = pd.Series(spotsInTrack)
s = s.loc[FilteredTrackIndex]

dfTracks["SPOTS"] = s
```

In [5]:
```
#I could have added some features to use in the filter, but I was lazy
dfSpots.head()
```

Out[5]:

name	ELLIPSOIDFIT_AXISPHI_A	ELLIPSOIDFIT_AXISPHI_B	ELLIPSOIDFIT_AXISPHI_C	ELLIPSOIDFIT_AXISTHETA_A
ID7170	0.0	-0.025627	1.545169	0.0
ID7171	0.0	-0.181041	1.389756	0.0
ID7172	0.0	0.177903	-1.392893	0.0
ID7173	0.0	0.252192	-1.318604	0.0
ID7175	0.0	0.032540	-1.538257	0.0

...

1.2 Filter desired Tracks

```
In [6]: print("Number of tracks before filtering them: {}".format(dfTracks.LONGEST_GAP.count()))
        dfTracks = dfTracks[dfTracks["NUMBER_SPOTS"] > 7]  #filtrar el numero de puntos minimo
        print("Number of tracks after filtering them (\" in the number of spots\"): {}".format(d
        dfTracks = dfTracks[dfTracks["TRACK_STD_QUALITY"] < 10] #pedir que el desvio en la calid
        print("Number of tracks after filtering them (\" + in the std quality of the track\"): {


        print("Number of spots before filtering them: " ,dfSpots.ELLIPSOIDFIT_AXISPHI_A.count())
        dfSpots = dfSpots[dfSpots.TRACK_ID.isin(dfTracks.index)]
        print("Number of spots after filtering them: " ,dfSpots.ELLIPSOIDFIT_AXISPHI_A.count())
```

```
Number of tracks before filtering them: 14155
Number of tracks after filtering them (" in the number of spots"): 5366
Number of tracks after filtering them (" + in the std quality of the track"): 4547
Number of spots before filtering them:  176491
Number of spots after filtering them:  112743
```

1.3 Add more features

1.3.1 Start and Stop Frames of a track.

```
In [7]: #Add last and start frame
        s = dfTracks.SPOTS
        def lastSpot(spotArray):
            return spotArray[-1]
        def firstSpot(spotArray):
            return spotArray[0]
        lastSpots = s.apply(lastSpot).values
        firstSpots = s.apply(firstSpot).values

        dfTracks['START_FRAME'] = dfSpots.FRAME.loc[firstSpots].values
        dfTracks['STOP_FRAME'] = dfSpots.FRAME.loc[lastSpots].values
```

1.3.2 Directionality calculated by taking the mean angle of each edge.

```
In [8]: #add the directionality of the track
        def getAngle(spotArray):
            x = dfSpots.POSITION_X.loc[spotArray].values
            y = dfSpots.POSITION_Y.loc[spotArray].values
            dx = x[1:] - x[:-1]
            dy = -(y[1:] - y[:-1])
            ang =  np.arctan2(dx,dy)
            return ang

        ang = s.apply(getAngle)*180/np.pi
```

```
        dfTracks['ANGLE_MEAN_EDGE'] = ang.apply(np.mean)
        dfTracks['ANGLE_STD_EDGE'] = ang.apply(np.std)
```

```
In [9]: getAngle(s['Track_2623'])*180/np.pi
```

```
Out[9]: array([ 176.39936967,  176.88567002,  176.56081846, -177.44847393,
                177.04985888, -176.71412926,  178.77665357, -176.74761719,
               -173.29358252])
```

1.3.3 Directionality computed using only the first and last spots of a given track.

```
In [10]: #Claculate directionality in another way
         dxTotal = dfSpots.POSITION_X.loc[lastSpots].values - dfSpots.POSITION_X.loc[firstSpots]
         dyTotal = -(dfSpots.POSITION_Y.loc[lastSpots].values - dfSpots.POSITION_Y.loc[firstSpot
         ang = np.arctan2(dxTotal,dyTotal)/np.pi*180
         dfTracks['ANGLE_TWO_SPOTS'] = ang
```

1.3.4 Directionality computed fitting a line in de x-y plane.

Directionality computed fitting a line in de x-y plane created with the point that belongs to each track. The directionality is the angle of the fitted line. Take into account that the fitting has been made taken the y-position to be the x axis since the bacteria move primary towards the north. In this way we have been able to compute more accurately the angle of the line.

```
In [11]: def getAngle3(spotArray):
             x = dfSpots.POSITION_X.loc[spotArray].values
             y = dfSpots.POSITION_Y.loc[spotArray].values
             m = linreg(y,x)[0]
             ang =  math.atan(m)
             return ang
         ang = s.apply(getAngle3)/np.pi*180
         dfTracks['ANGLE_MEAN_FIT'] = ang
         #print(dfTracks['SPOTS'].apply(func=calculateDistanceVariance))
```

1.3.5 Calculation of the acceleration.

I aim to find the tracks whose velocity changes abruptly. This could probably be a symptom of a bad tracking.

NOT USEFUL-> I didn't find anything of significance.

```
In [12]: #DO NOT FORGET TO CHANGE T!!!
         def calculate_instantaneous_accel_max(spots,t = 0.133):


             t = 0.133 #Change this. it depends on the image set taken.



             x_position = dfSpots.POSITION_X.loc[spots].values
```

```
        y_position = dfSpots.POSITION_Y.loc[spots].values

        frame = dfSpots.FRAME.loc[spots].values
        change_in_frame = frame[1:]-frame[:-1]
        change_in_time = change_in_frame*t

        vx = (x_position[1:] - x_position[:-1])/change_in_time
        vy = (-y_position[1:] + y_position[:-1])/change_in_time

        ax = (vx[1:] - vx[:-1])/0.133
        ay = (vy[1:] - vy[:-1])/0.133

        #v = np.sqrt(vx*vx+vy*vy)

        a = np.sqrt(ax*ax + ay*ay)
        return a.mean()

    dfTracks["TRACK_MEAN_ACCELERATION"] = s.apply(calculate_instantaneous_accel_max)
```

1.3.6 Mean length of the spot along the track

```
In [13]: SpotsLength = dfSpots.loc[:,['ELLIPSOIDFIT_SEMIAXISLENGTH_B','TRACK_ID']].groupby('TRAC

         #Compute mean and std of the lengths of the spots for each track

         dfTracks['LENGTH_STD_B'] = SpotsLength.std().loc[dfTracks.index]
         dfTracks['LENGTH_MEAN_B'] = SpotsLength.mean().loc[dfTracks.index]

         SpotsLength = dfSpots.loc[:,['ELLIPSOIDFIT_SEMIAXISLENGTH_C','TRACK_ID']].groupby('TRAC
         dfTracks['LENGTH_STD_C'] = SpotsLength.std().loc[dfTracks.index]
         dfTracks['LENGTH_MEAN_C'] = SpotsLength.mean().loc[dfTracks.index]
```

2 Save data

```
In [14]: dfTracks.to_csv("Tracks_data_processed_1")
         dfSpots.to_csv("Spots_data_processed_1")
```

A.3. Programa para analizar la robustez del seguimiento

Este notebook tiene como objetivo elegir un conjunto de tracks para guardarlos en un archivo xml que pueda leer TrackMate. De esta forma, se puede elegir un conjunto reducido de tracks para observarlos en TrackMate, aprovechando así las herramientas de visualización que brinda.

In [10]:
```
import numpy as np
import pandas as pd

import xml.etree.ElementTree as ET

import math as math
```

In [2]:
```
#Import tracks from which I will select the data to see in TM
dfTracks = pd.read_csv("Tracks_data_processed_1",index_col = 0)
dfSpots = pd.read_csv('Spots_data_processed_1',index_col = 0)

dfTracksFast = dfTracks[dfTracks.TRACK_MEAN_SPEED > 12]
dfTracksSlow = dfTracks[dfTracks.TRACK_MEAN_SPEED < 12]
```

In [3]:
```
#Random selection process
totalNumberOfTracks = dfTracksSlow.count().values[0]
randomSelection = np.random.randint(0,totalNumberOfTracks,size=20)
randomSelection.sort()

dfTracks_to_save = dfTracksSlow.iloc[randomSelection]#select the random tracks
Tracks_to_save = dfTracks_to_save.index.values
#print(Tracks_to_save)
```

In [4]:
```
#Change Tracks to save here if you want to save a custom set of tracks
Tracks_to_save = ['Track_67', 'Track_1011', 'Track_1811', 'Track_2550', 'Track_2623',
       'Track_2927', 'Track_3303', 'Track_4741', 'Track_5186', 'Track_5641',
       'Track_5914', 'Track_6200', 'Track_6771', 'Track_7587', 'Track_8597',
       'Track_8994', 'Track_8999', 'Track_9359', 'Track_9989', 'Track_10759',
       'Track_10951', 'Track_11062', 'Track_11628', 'Track_11721',
       'Track_11890', 'Track_12586', 'Track_12821', 'Track_13016',
       'Track_13099', 'Track_13285', 'Track_13748']
#Tracks_to_save
```

In [5]:
```
#Open the model and settings xml to modify
modelDir = r"..\Data\Model_1.xml"
settingsDir = r"..\Data\Settings_1.xml"

tree = ET.parse(modelDir)
settings_tree = ET.parse(settingsDir)

settings_root = settings_tree.getroot()
root = tree.getroot()

root.append(settings_root[0]) #add the setting xml so that TM can load the image along u
```

In [6]:
```
#Eliminate all the tracks from the model. More especifically, from the child AllTracks
n = 0
Tracks = root[0][2]
for track in Tracks.findall('Track'): #it is important to use find all function since it
#Othrewise, you would be removing elements of and object while iterating it -> Catastrop
    if not track.get('name') in Tracks_to_save:
        Tracks.remove(track)
```

In [7]:
```
#Take only the number of the track instead of its name. It is essential for the next ste
for i in range(len(Tracks_to_save)):
    Tracks_to_save[i] = Tracks_to_save[i].split('_')[1]
```

In [8]:
```
#Elimintae all the track from the model. More especifically, from the child filteredTrac
filteredTracks = root[0][3]
for track in filteredTracks.findall('TrackID'):
    if not track.get('TRACK_ID') in Tracks_to_save:
        filteredTracks.remove(track)
#for track in filteredTracks:
    #print(track.get('TRACK_ID'))
```

In [9]:
```
saveAs =  'DataToanalizeTracks_InverseDirection.xml'
tree.write(r'..\data\\' + saveAs ) #Save the results in a new model
```

A.4. Programa para visualizar datos de la robustez del seguimiento

In [3]:
```
import pandas as pd
import numpy as np

from bokeh.layouts import gridplot
from bokeh.plotting import figure, show, output_notebook
output_notebook()
```

1 This notebook aims to analize the different errors that happened during the tracking

In [4]:
```
df = pd.read_csv(r"C:\Users\agali\Desktop\Projects\TrackingBacteriasPython\Data\Data_ana
                 sep = ',')
df.head()
```

Out[4]:

	Track	Error_Type
0	2446	1
1	2503	4
2	2888	2
3	3570	2
4	3664	5

In [5]:
```
dfErrors = df.groupby(' Error_Type').count()
dfErrors = dfErrors.rename(columns={'Track ':'Total_number'})
newIndex = np.empty(5,dtype=object)
for i in range(dfErrors.size):
    newIndex[i] = "Grupo " + str(dfErrors.index.values[i])
dfErrors = dfErrors.set_index(pd.Index(newIndex,name = 'Error_type'))
dfErrors

#I will d the error type 3 to 4.


dfErrors.iloc[3]  += dfErrors.iloc[2]
dfErrors = dfErrors.iloc[[0,1,3,4]]
dfErrors = dfErrors.rename(index={'Grupo 4':'Grupo 3','Grupo 5':'Grupo 4'})
```

In [6]:
```
dfErrors
```

Out[6]:

Error_type	Total_number
Grupo 1	30
Grupo 2	40
Grupo 3	23
Grupo 4	2

In [7]:
```
from bokeh.io import show, output_file
from bokeh.models import ColumnDataSource
from bokeh.palettes import Spectral6
```

```
from bokeh.plotting import figure
from bokeh.transform import factor_cmap


index = dfErrors.index.values
values = dfErrors.Total_number.values


p = figure(x_range=index, y_range = (0,15), plot_height=300,plot_width = 1000)
p.vbar(x='Error_type', top='Total_number', width=0.9, source=dfErrors, legend="Error_typ
       line_color='white', fill_color=factor_cmap('Error_type', palette=['#1f77b4','#ff7

p.xgrid.grid_line_color = None
p.y_range.start = 0
p.y_range.end = dfErrors.Total_number.max() + 1
p.legend.orientation = "vertical"
p.legend.location = "center_right"

p.xaxis.axis_label_text_font_size = "23pt"
p.yaxis.axis_label_text_font_size = "20pt"
p.xaxis.major_label_text_font_size = '15pt'
p.yaxis.major_label_text_font_size = '15pt'
p.legend.label_text_font_size = "20pt"

show(p)
```