Proyecto Final

Ingeniería en Informática

Análisis de imágenes de microscopía

Instituto Tecnológico de Buenos Aires

Autores

Della Sala, Rocío

Rodriguez, Ariel Andrés

Tutores

Dra. Bruno, Luciana

Dra. Gambini, María Juliana

Fecha: 24 de agosto de 2021

Resumen

El análisis automático de imágenes tiene múltiples campos de aplicación, siendo uno de estos el análisis a nivel celular donde se estudian imágenes microscópicas.
En el presente trabajo se implementan métodos para poder extraer tanto información cuantitativa como también cualitativa de una serie de películas de distintos citoesqueletos. El objetivo es construir una herramienta que facilite el estudio de imágenes y nos permita obtener información que no se podría extraer a simple vista.
El trabajo se centra en el estudio de 3 propiedades las cuales son el contorno de la estructura del citoesqueleto, el movimiento de la estructura a lo largo de la película y la textura en distintas zonas de la imagen.
Para cada una de estas propiedades se desarrollan múltiples métodos y se opta por elegir aquellos que presenten los mejores resultados en el menor tiempo posible, teniendo en cuenta que su funcionamiento sea independiente de la forma de los citoesqueletos o de la intensidad de cada imagen.
Si bien este estudio presenta dificultades como el ruido en este tipo de imágenes o la diversidad de las estructuras de los citoesqueletos, para todas las propiedades se logran obtener interesantes resultados.

Índice general

Capítulo 1

Introducción

En el presente trabajo se estudian, a través del análisis automático de imágenes, procesos que involucran el desplazamiento de estructuras intracelulares que van desde pequeñas organelas hasta filamentos del citoesqueleto.

El citoesqueleto es una red de filamentos proteicos presente en células eucariotas. Provee soporte interno en las células, organiza las estructuras internas e interviene en los fenómenos de transporte, tráfico y división celular. Está compuesto por 3 familias de filamentos semi flexibles (polímeros): microtúbulos, filamentos de actina y filamentos intermedios. Podemos ver un ejemplo de un citoesqueleto en la Figura 1.1 donde los microtúbulos se muestran en color verde, los filamentos de actina en rojo y los núcleos en azul.

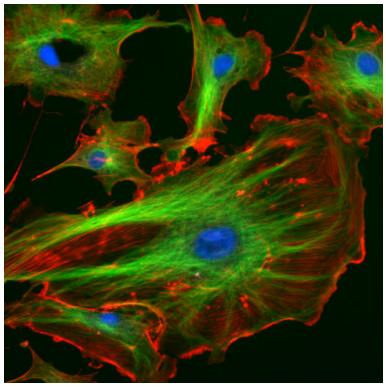


Figura 1.1: Imagen confocal de célula endotelial. Fuente: http://rsb.info.nih.gov/ij/images/.

La Figura 1.2 muestra el esquema de la distribución celular de los tres principales componentes del citoesqueleto de una célula animal.

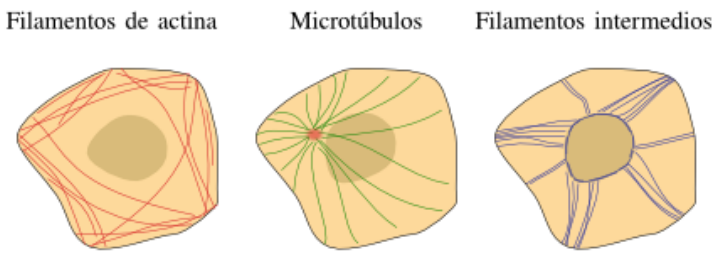


Figura 1.2: Organización típica del citoesqueleto en una célula animal.

Estos filamentos constituyen un entramado con propiedades mecánicas que permite a las células generar y reaccionar ante la acción de fuerzas de compresión manteniendo la forma de la célula. El origen de estas fuerzas es variado; puede ser la contracción generada por células vecinas, la adhesión con el sustrato, entre otras causas [1].

El interés es evaluar la evolución de la dinámica de los filamentos. Una forma de realizarlo es a partir de la recuperación de la forma de los mismos, dado que se puede entender la fuerza que está actuando. Esto nos permite estudiar la configuración general de la red de filamentos. Por ejemplo, cuando una célula realiza una migración celular las redes se modifican y comportan de manera diferente a cuando la célula está en una posición fija. Cuando están perturbadas o enfermas, las propiedades mecánicas como la flexibilidad de los filamentos se modifica.

Estudiar tanto el movimiento como la variación morfológica de los filamentos representa un desafío debido al ruido y bajo contraste que poseen estas imágenes. Esto puede atribuirse a que las imágenes son mediciones tomadas a través de microscopía de fluorescencia.

La microscopía de fluorescencia es un efecto cuántico que involucra la interacción entre los electrones de los átomos de una proteína y una luz láser o led. Se la considera una herramienta disruptiva en la investigación biológica al permitir la observación de procesos que ocurren a nivel celular de manera no invasiva.

La fluorescencia se basa en una captura de fotones que emite la muestra y este número de fotones suele ser bajo, produciendo que la razón señal/ruido sea un factor influyente en el análisis.

1.1. Objetivos

El objetivo general del proyecto final es desarrollar una herramienta computacional que permita realizar rutinas para imágenes de microscopía con el fin de extraer información cuantitativa de las propiedades mecánicas de una célula.

A lo largo del trabajo, se implementan métodos que realizan un análisis automático de las imágenes del cual se extraen parámetros y características morfológicas de la red y de su desplazamiento.

Cada uno de estos métodos poseen objetivos específicos:

- Obtención del área, longitud del contorno y razón de los ejes: Permitirá analizar variaciones de la geometría de la célula a lo largo del tiempo.
- Comparación de la distribución del área, longitud del contorno y razón de los ejes entre las diferentes células.
- Cuantificación de las zonas de mayor movilidad.
- Análisis de textura: Permitirá obtener información acerca de cuan intrincada es la red.

Capítulo 2

Estado del arte

Hasta el momento, existen dos formas de abordar el problema. La primera es a través del seguimiento de filamentos únicos y la segunda es a través del seguimiento de toda la red. En este proyecto trabajamos en el segundo enfoque, es decir se estudia la red en su completitud.

El objetivo de este capítulo es mencionar técnicas utilizadas en trabajos anteriores relacionados a ambos enfoques. De estos trabajos se desprenden las motivaciones que nos llevan a realizar este proyecto y que también son mencionadas en esta sección.

2.1. Seguimiento de filamento único a partir de imágenes de microscopía de fluorescencia

Existen numerosos algoritmos para el seguimiento de partículas individuales, pero recuperar las posiciones de un filamento que se encuentra continuamente cambiando su posición y forma es difícil. A continuación veremos 5 propuestas realizadas por diversos autores para recuperar las coordenadas de filamentos aislados e in vitro [2].

La primera propuesta consiste en localizar un conjunto de puntos en imágenes de un filamento in vitro. Este método requiere seleccionar manualmente los puntos en cada cuadro de una película los cuales luego son interpolados [3]. Sin embargo, se reporta un error de 0,8 % y además el método solo provee formas aproximadas de los filamentos.

La segunda propuesta requiere únicamente la selección de los puntos extremos y de un punto intermedio del filamento, ya que la rutina propuesta es semi-automática. Esta rutina extrae perfiles de intensidad alrededor del punto de intermedio, los ajusta a un modelo cuadrático y selecciona aquel en el cual la pendiente del ajuste es mayor. La orientación de dicho perfil representa la dirección perpendicular al filamento. Los autores reportan un error de 11nm [4].

La tercera propuesta consiste en estudiar los cambios de la forma de filamentos aislados in vitro. Esta rutina requiere de la selección manual del punto x de inicialización. Luego, se extrae un perfil de intensidad vertical alrededor de la posición x analizada. Dicho perfil es convolucionado con un período de la función seno para poder obtener la posición del máximo de los datos convolucionados que representa la posición y del centro del filamento. Esto se ve en la Figura 2.1.

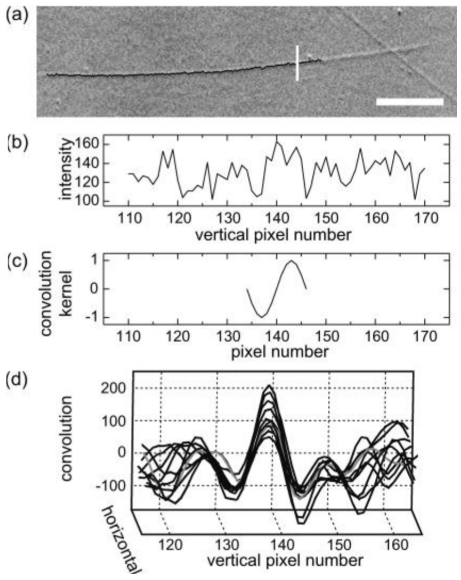


Figura 2.1: (A) Imagen DIC parcialmente analizada (B) Perfil de intensidades correspondiente a la columna de píxeles marcada en A (C) Función seno utilizada como kernel de convolución, para modelar el gradiente de grises de la imagen. (D) Resultados de la convolución. Fuente: [5].

Luego, pasa a la siguiente posición horizontal x donde se analiza el perfil de intensidad alrededor de la posición corregida y, obtenida en la iteración anterior. Este sistema de tracking requiere que los filamentos estén aislados y tengan una forma relativamente recta para su buen funcionamiento, lo cual es posible en condiciones in vitro, pero no es el caso de los filamentos hallados en el entorno celular. [5].

La cuarta propuesta consiste en umbralizar la imagen para separar el filamento que queremos analizar de las fluctuaciones restantes. Luego, se realiza una esqueletización donde los píxeles por encima del umbral cubren el filamento fluorescente en un grupo de varios píxeles de ancho. La Figura 2.2 muestra estos pasos.

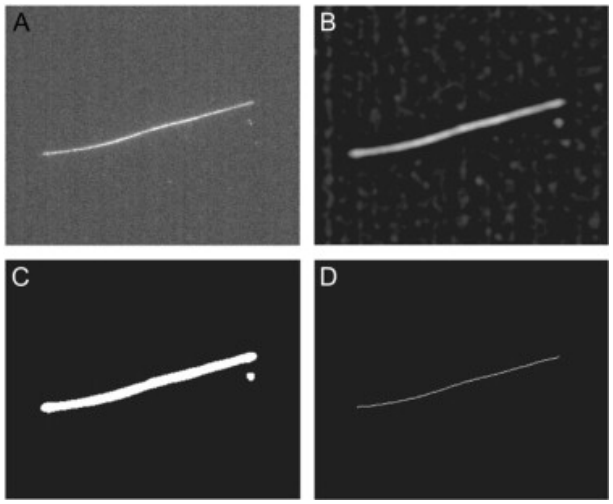


Figura 2.2: (A) Imagen sin procesar (B) Eliminación de ruido (C) Umbralización (D) Esqueletización. Fuente: [6].

Los autores han reportado un error de 20 nm en la recuperación de las posiciones de los filamentos [6].

La quinta propuesta consiste en un algoritmo que combina umbralización, detección de características, segmentación de la imagen, proceso de ajuste e interpolación. El algoritmo logra reconocer elementos puntuales, extremos y el centro del filamento, filamentos cortos e intersección de filamentos [7]. La Figura 2.3 muestra los pasos principales de este proceso.

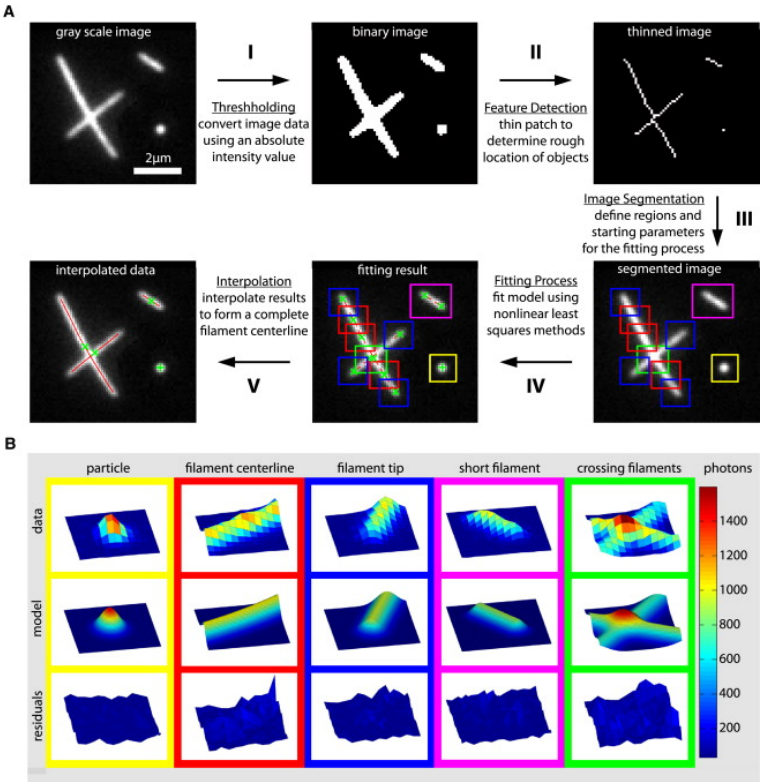


Figura 2.3: (A) Flujo del algoritmo de seguimiento (B) Perfiles de intensidad para diferentes regiones. Fuente: [7].

2.2. Seguimiento de microtúbulos individuales en células vivas

Esta rutina es semi-automática ya que inicialmente necesita de la interacción del usuario para que marque en la imagen el filamento que se desea estudiar.

Está basado en el análisis de perfiles de intensidad perpendiculares a los filamentos. La diferencia con los algoritmos anteriores radica en la forma de analizar los perfiles de intensidad, ya que se utilizan redes neuronales [2].

Para analizar unívocamente los perfiles se define la dirección perpendicular al filamento. Luego se asigna una coordenada (x_k, y_k) a cada posición a lo largo del segmento. Para cada x_k, se extrae el perfil de intensidad perpendicular al filamento definido por una distancia de 300nm alrededor del punto. Luego, cada perfil es interpolado utilizando una red neuronal y se determina la posición corregida $y_k^{'}$ como el máximo de esa interpolación.

Si bien el algoritmo permite detectar y seguir filamentos con precisión, es importante destacar que sólo puede seguir segmentos del mismo y no el filamento de forma completa.

2.3. Seguimiento de la red completa

En esta sección abordaremos un estudio del retículo endoplasmático donde se cuantifican múltiples propiedades del mismo analizándolo de forma completa [8]. Este estudio se asemeja a aquel que realizamos en el presente trabajo.

El retículo endoplasmático es una red de membranas poligonales dentro de la célula compuesta de túbulos y láminas interconectadas llamadas cisternas. Está ubicado dentro del citoplasma de las células eucariotas y sirve por ejemplo para el transporte de proteínas.

Una de las propiedades estudiadas fue la caracterización de la estructura de la cisterna utilizando en base a la textura. Para lo cual se utilizan las matrices de co-ocurrencia (GLCM), las cuales también fueron utilizadas en el presente trabajo y se abordarán en el capítulo 7.4. Se utilizan las métricas de contraste, correlación, energía y homogeneidad para describir la distribución de las intensidades en cada cisterna. La estructura del GLCM se puede utilizar para dar características de textura de diferentes aspectos de la distribución de grises. Por ejemplo una cisterna completamente homogénea tiene valores de cero para el contraste y la energía y uno para la correlación y homogeneidad.

Otra de las propiedades estudiadas fue el movimiento. Para ello se hace uso del flujo óptico, el cual es un campo vectorial 2D donde cada vector es de desplazamiento que muestra el movimiento entre dos fotogramas. En este caso se utiliza el algoritmo Farneback, con el cual se logra cuantificar el movimiento para distintas escalas, mapas de velocidad y mapas de dirección.

También, fue necesario realizar una segmentación de túbulos y cisternas. Algunos métodos de umbralización son problemáticos debido a que si el ancho y la intensidad de los túbulos varía, el umbral puede excluir estructuras o fusionar regiones adyacentes. Por lo cual, se opta por un análisis llamado phase-congruency que es independiente de la intensidad (para muchas escalas y orientaciones). Es un método para detectar esquinas/bordes y es robusto ante cambios de iluminación y contraste. Se logra resaltar elementos rígidos como los túbulos, independientemente de su intensidad.

2.4. Motivaciones

Al analizar los diferentes algoritmos y métodos que se utilizaron previamente podemos ver que, si bien se realizan exhaustivos análisis sobre cómo realizar un seguimiento continuo de filamentos de la célula, no existen demasiados estudios relacionados con la estructura de la red en su completitud. Como ambos enfoques de estudio son complementarios, la motivación principal de este proyecto es estudiar a la red como un todo debido a la falta de investigaciones previas.

Otra motivación es que las rutinas que fueron implementadas en trabajos anteriores se corren en Matlab, dificultando su portabilidad debido a que para poder utilizar Matlab se debe tener una licencia paga. Por esto, proponemos utilizar otro lenguaje de programación para el desarrollo del proyecto que sea open-source como es el caso de Python.

La utilización de Python es clave para el desarrollo del proyecto puesto que en un futuro podrá ser utilizado por cualquier persona sin necesidad de una licencia. Además, este lenguaje es intuitivo y fácil de comprender, lo que hace que el proyecto una vez finalizado pueda ser mantenible.

Como última motivación, se busca que el trabajo realizado sea extensible ya que se pueden implementar en Python las rutinas de seguimiento de filamentos para luego ser incluidas en este proyecto. Esto permitiría tener un estudio de la célula completo bajo una única herramienta centralizada, la cual tendría todas las rutinas para el análisis de la misma.

Capítulo 3

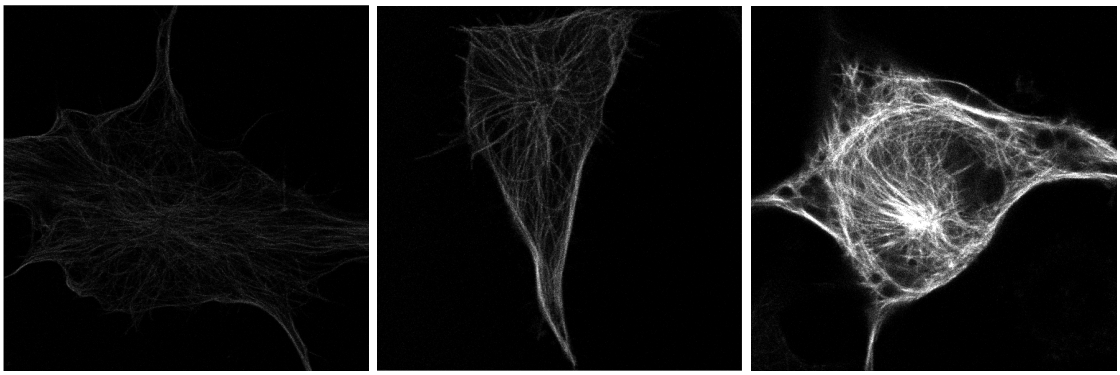
Imágenes a analizar

En el presente capítulo se presentan algunas imágenes representativas de todas las que se analizan. Se introducen sus características técnicas y algunas problemáticas que las mismas poseen. Para realizar el análisis se hace uso del programa ImageJ, el cual permite trazar perfiles de intensidad sobre las imágenes.

Las imágenes a analizar son mediciones tomadas a través de microscopía de fluorescencia a distintas células a lo largo del tiempo. Estas imágenes son obtenidas de células melanofora de Xenopus laevis (una especie de rana) expresando la proteína asociada a microtúbulos XTP, marcadas con una proteína fluorescente verde.

Las imágenes pertenecen a células adherentes, que tienen la particularidad de estar adheridas por puntos de anclaje y tener poca movilidad dándose así únicamente movimientos de expansión y contracción.

La Figura 3.1 muestra ejemplos de imágenes con las que se trabaja en este proyecto y se observa una variación de la escala de grises entre las mismas. Esto representa una problemática que se desarrolla en la sección 3.1.



(a) (b) (c)

Figura 3.1: Imágenes representativas de distintas células ilustrando los distintos niveles de intensidad. (a) Imagen con intensidad baja. (b)Imagen con intensidad media. (c) Imagen con intensidad alta.

En relación a las características técnicas de estas imágenes, son de 16 bits cuyo formato es *.tif*. El espacio que ocupan estas imágenes varía entre 200KB y 800KB, como también sus dimensiones las cuales varían entre 320x320 píxeles y 640x640 píxeles. La Figura 3.2 agrega una barra de escala para representar el tamaño real de la imagen en nanómetros.

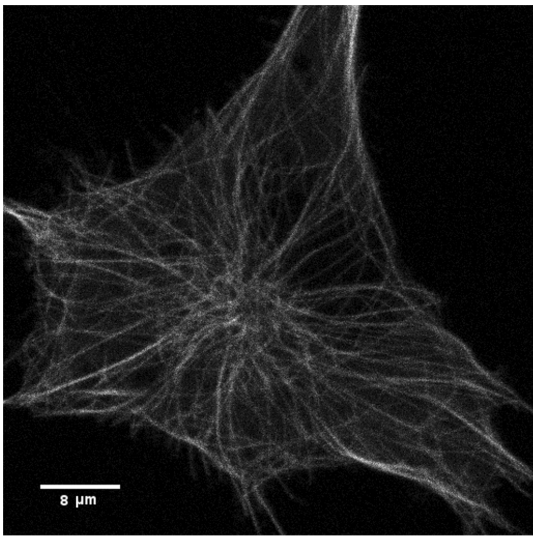
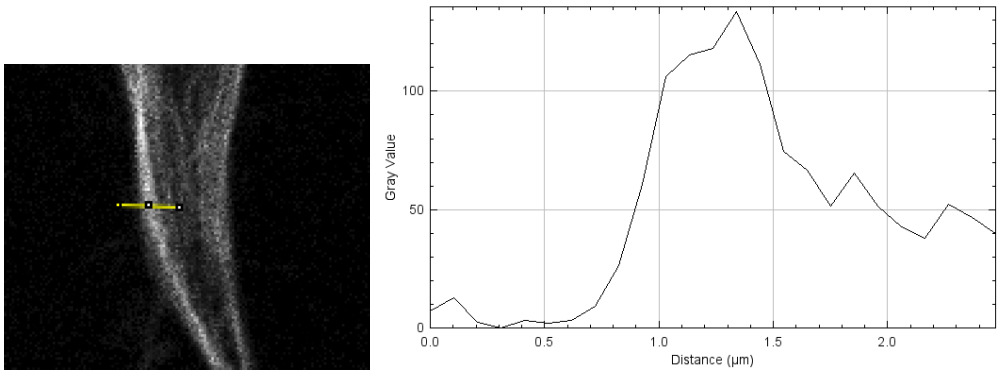


Figura 3.2: Imagen representativa de la célula junto a una barra de escala.

3.1. Escala de grises

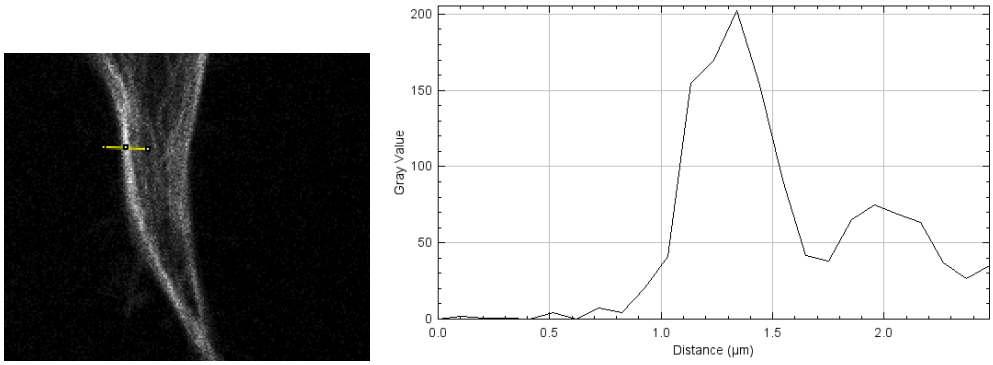
Una de las problemáticas que existen es la diversidad de la escala de grises que estas imágenes poseen como vimos en la Figura 3.1. Nos encontramos con imágenes donde la estructura está representada por píxeles cuya intensidad posee valores cercanos a cero y otras con valores más elevados. Esta variación dificulta algunos métodos que necesiten como parámetro de entrada valores relacionados con la intensidad.

Esta variación de la escala de grises se puede apreciar en las Figuras 3.3 y 3.4 donde trazamos dos perfiles de intensidad sobre el mismo filamento, ambos en ubicaciones adyacentes.



(a) (b)

Figura 3.3: (a) Imagen representativa de una célula con un trazo marcado en color amarillo. (b) Perfil de intensidad de el trazo.



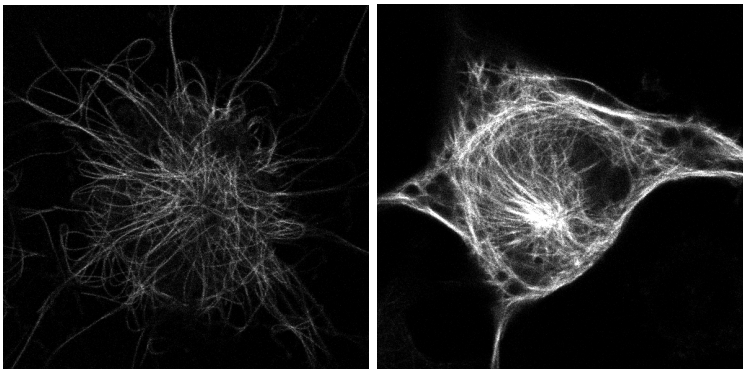
(a) (b)

Figura 3.4: (a) Imagen representativa de una célula con un trazo marcado en color amarillo. (b) Perfil de intensidad de el trazo.

A simple vista, la variación es imperceptible pero los gráficos de ambas figuras nos muestran que a lo largo del perfil se alcanzan dos máximos de intensidad distintos. El valor en la escala de grises de la Figura 3.3 es aproximadamente 130 y el de la Figura 3.4 es aproximadamente 205.

3.2. Estructura

Otra problemática que existe es que estás células poseen estructuras diversas entre sí. Algunas se pueden representar como contornos cerrados mientras que otras poseen una estructura indefinida como muestra la Figura 3.5.



(a) (b)

Figura 3.5: Imágenes representativas de distintas células ilustrando las diferentes estructuras externas. (a) Estructura con contorno abierto. (b) Estructura con contorno cerrado.

3.3. Ruido

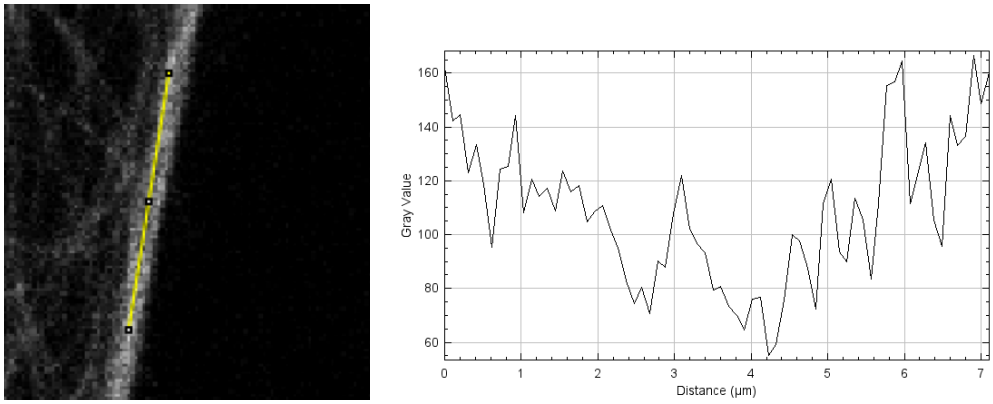
Definimos al ruido como información no deseada que contamina la imagen, dificultando luego su procesamiento. El ruido puede presentarse en la imagen como consecuencia de la captura, digitalización y/o transmisión de la misma.

El ruido tiene un impacto importante en el procesamiento de cualquier imagen, porque puede provocar que los métodos que apliquemos fallen debido a su presencia. Por ende para su correcto procesamiento, es necesario eliminar toda aquella información no deseada que se presente en la imagen.

El ruido se puede modelar de dos formas: aditivo o multiplicativo. Es aditivo es cuando el método de captura produce un ruido que es mínimo, como por ejemplo al utilizar el escáner. Es decir, este

ruido es agregado debido a que no es propio de la imagen. En cambio cuando el ruido es inherente a su método de captura se modela como multiplicativo.

Podemos apreciar el ruido de una imagen representativa en las Figura 3.6 donde trazamos un perfil de intensidad a lo largo de un filamento.

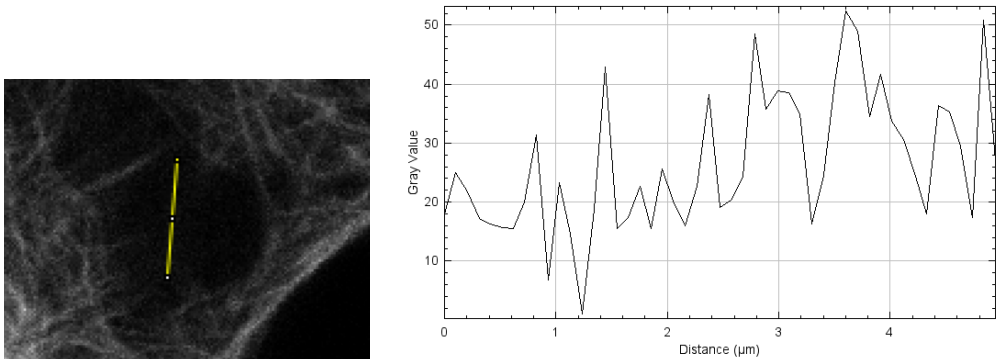


(a) (b)

Figura 3.6: (a) Imagen representativa de una célula con un trazo marcado en color amarillo a lo largo de un filamento (b) Perfil de intensidad del trazo.

El gráfico muestra la variación de intensidad que existe a lo largo de un único perfil. El valor máximo alcanzado es aproximadamente 166 mientras que el valor mínimo es 55.

Por otro lado, en la Figura 3.7 se toma un perfil de intensidad en un sector donde no hay filamentos.



(a) (b)

Figura 3.7: (a) Imagen representativa de una célula con un trazo marcado en color amarillo a dentro de la estructura. (b) Perfil de intensidad de el trazo.

El gráfico muestra que en dicho segmento hay ruido y que el mismo no es uniforme. En este caso no vemos una variación tan pronunciada de los valores de la escala de grises como en la Figura 3.6. Sin embargo, una diferencia de aproximadamente 50 es un valor considerable a tener en cuenta.

Capítulo 4

Pre-procesamiento

En el presente capítulo se presentan algunos métodos de preprocesamiento de imágenes utilizados a lo largo del proyecto. Se incluyen tanto los métodos que resultaron exitosos como los que fueron descartados.

Para comprobar la utilidad de los algoritmos de preprocesamiento, realizamos pruebas utilizando el *modelo de intercambio de píxeles* presentado en el capítulo 5 sección 5.1.

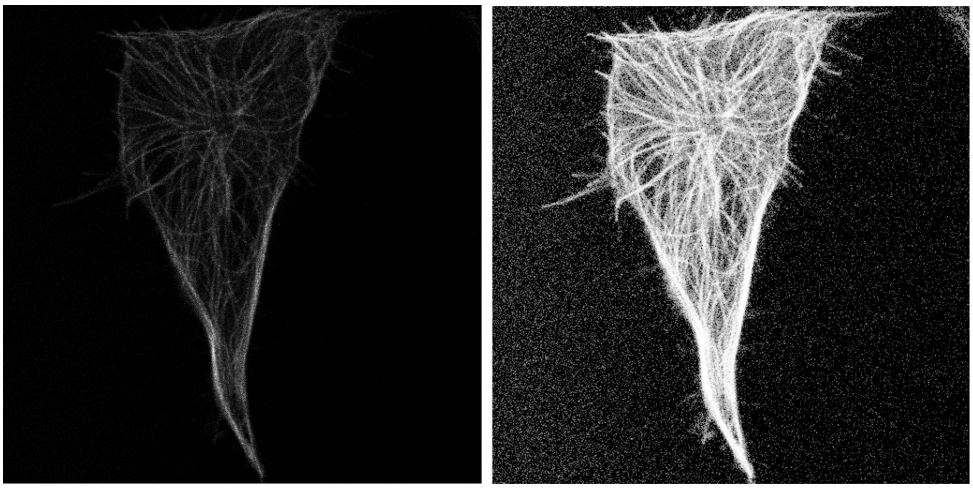
4.1. Ecualización del histograma

Como se mencionó en el capítulo 3 una de las problemáticas que existen es la diversidad de la escala de grises que estas imágenes poseen. Por ende, aplicamos la *ecualización del histograma* a nuestras imágenes como un método previo a la eliminación del ruido.

La *ecualización del histograma* de una imagen es una transformación que pretende obtener para una imagen un histograma con una distribución uniforme de los niveles de gris. Es decir, que exista el mismo número de píxeles para cada nivel de gris.

A continuación se muestra la aplicación del método de ecualización del histograma. El algoritmo utilizado es extraído de la librería *OpenCV* [9]. Este filtro no requiere parámetros para su aplicación.

La Figura 4.1 muestra a la izquierda la imagen original y a la derecha la modificada por la *ecualización del histograma*.



(a) (b)

Figura 4.1: Aplicación de la ecualización del histograma. (a) Imagen original. (b) Imagen ecualizada.

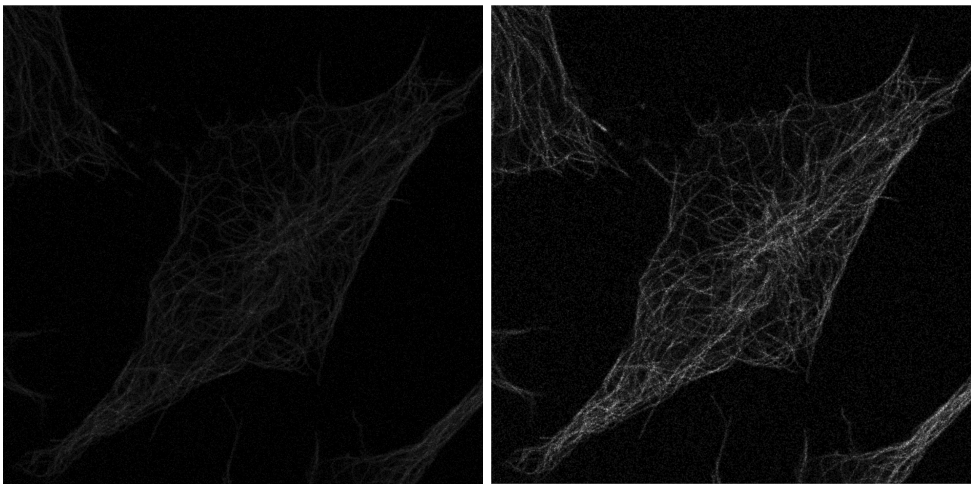
Si bien el método realza de manera eficaz la estructura de la célula, resalta el ruido de la imagen preprocesada. Esto provoca que métodos posteriores de detección de bordes como el *modelo de intercambio de píxeles* o *MGAC* fallen, no siendo capaces de detectar el borde.

Si bien combinamos este método con los filtros de eliminación de ruido como el bilateral y/o anisotrópico que se explican más adelante, los resultados obtenidos con los métodos posteriores no

se adecuan al contorno del objeto de interés. Por lo mencionado anteriormente, decidimos descartar el método.

Para solucionar esto se normaliza la imagen, es decir se mapea el rango de intensidad de sus píxeles al intervalo [*alpha*, *beta*]. Por ejemplo, si una imagen toma valores en el intervalo [5, 50] se transforman sus límites; su valor mínimo se transforma en *alpha* y su valor máximo en *beta*. El resto de los valores se escalan utilizando una constante multiplicativa dependiente de *alpha* y *beta*. En este proyecto, el intervalo utilizado es [0, 255].

El resultado de normalizar la imagen se muestra en la Figura 4.2 donde a la izquierda se muestra la imagen original sin normalizar y a la derecha la modificada.



(a) (b)

Figura 4.2: Aplicación de la normalización. (a) Imagen original. (b) Imagen normalizada.

4.2. Filtros de eliminación de ruido

Un filtro es un método de preprocesamiento de imágenes utilizado principalmente para la eliminación del ruido, suavizado de la imagen o realzar bordes. Su objetivo es obtener a partir de una imagen origen una imagen final cuyo resultado sea más adecuado para la post-aplicación de un método específico.

Si bien existen múltiples métodos de filtrado, todos realizan la misma técnica que consiste en convolucionar la imagen con una máscara tamaño NxN. Los valores de la máscara, llamados pesos, determinan el resultado del método. La idea es reemplazar el valor original del píxel de la imagen correspondiente al centro de la máscara con la suma de los valores originales de los píxeles multiplicados por los pesos de la máscara. Por ende, el tipo de filtrado queda establecido por el contenido de la máscara que se aplica. La Figura 4.3 ilustra lo explicado.

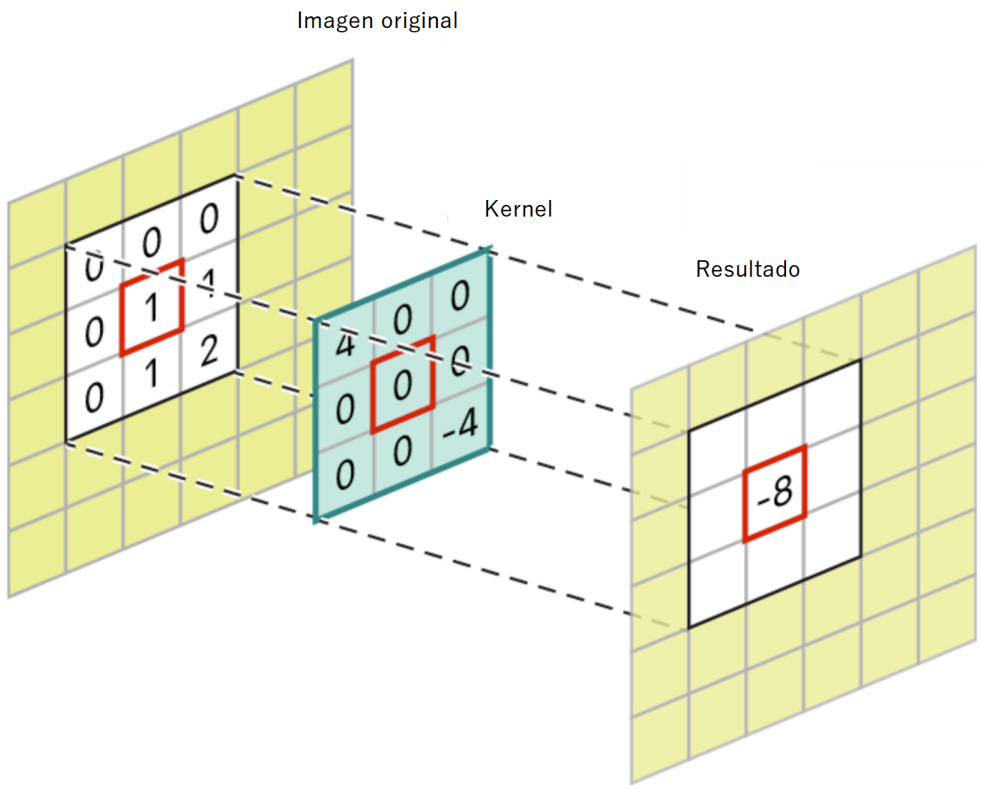


Figura 4.3: Ejemplo de la aplicación de convolución a un píxel.

Decidimos hacer las pruebas utilizando distintos tipos de filtros. Para comprobar la utilidad de los mismos, lo que realizamos es un preprocesamiento de la imagen junto con la aplicación de un método para obtener el contorno de un objeto. Este método, llamado *método de intercambio de píxeles*, se explica en el capítulo 5 en la sección 5.1. La idea es buscar aquel filtro, que al combinarlo con este método, se adecue al contorno a la célula sin perder los detalles de la misma.

4.2.1. Filtro gaussiano

En primera instancia probamos la aplicación del *filtro gaussiano*. Este filtro es el resultado de aplicar la convolución entre la imagen y una máscara construida a través de la función gaussiana de dos dimensiones, presente en la Ecuación (1).

$$G(x, y) = \frac{1}{2\pi\sigma^2} e^{-((x^2+y^2)/\sigma^2)} \tag{1}$$

El valor de σ representa la desviación estándar de la distribución, que a su vez representa el grado de suavizado que se aplica a la imagen. El tamaño de la máscara para realizar convolución se calcula como $2\sigma + 1$.

Al aplicar convolución se devuelve como resultado el promedio pesado de cada píxel de la vecindad, donde el píxel central es aquel con mayor peso. Los valores de los vecinos tienen menor influencia conforme se alejan del píxel central. Esto sucede debido a la distribución gaussiana, la cual se muestra en la Figura 4.4. La misma tiene su pico en el centro y la curva se aplana a medida que avanza hacia los bordes.

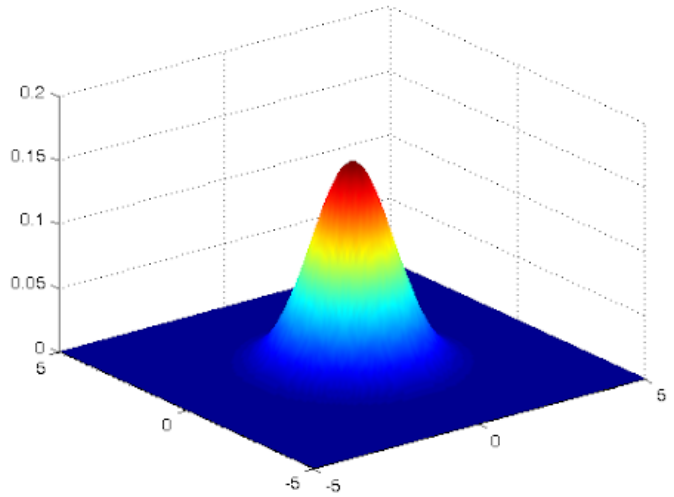
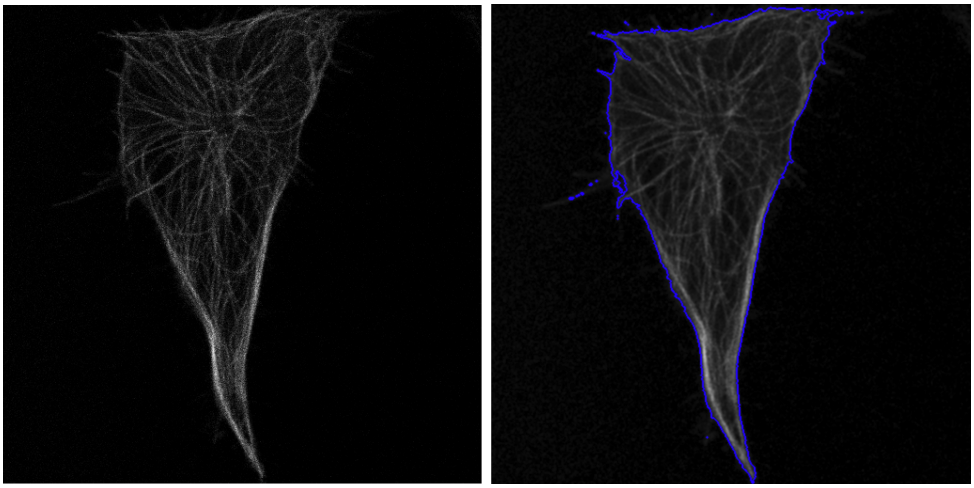


Figura 4.4: Distribución gaussiana.

A continuación se muestra el resultado de aplicar el *filtro gaussiano* junto con el *modelo de intercambio de píxeles*. El algoritmo utilizado es extraído de la librería *OpenCV* [10]. El parámetro que utilizamos correspondiente al filtro es el siguiente:

- $\sigma = 1$

La Figura 4.5 muestra a la izquierda la imagen preprocesada con *filtro gaussiano* y a la derecha la modificada por el *modelo de intercambio de píxeles*.



(a) (b)

Figura 4.5: Aplicación del filtro gaussiano e intercambio de píxeles. (a) Imagen original. (b) Imagen procesada.

Si bien este filtro elimina el ruido, tiene dos problemas. En primer lugar produce un borroneo en la imagen, no preservando tan bien los bordes como lo hacen otros filtros que veremos luego. En segundo lugar, este filtro es útil especialmente con imágenes que contengan ruido gaussiano. Por estos dos motivos, lo descartamos para futuras aplicaciones.

4.2.2. Filtro bilateral

El *filtro bilateral* es un filtro que se caracteriza por preservar bordes a la vez que reduce el ruido. Su implementación se asemeja al *filtro gaussiano* debido a que como en este último, el valor de intensidad de cada píxel de la imagen se reemplaza por un promedio ponderado de los valores de intensidad de los píxeles cercanos.

Pero, en el *filtro bilateral* los pesos de la máscara no dependen solo de la distancia euclidiana de los píxeles de la ventana al centro de la misma, sino también de las diferencias en la intensidad del color.

El filtro bilateral se define según la Ecuación (2).

$$I^{filtrada}(x) = \frac{1}{W_p} \sum_{x_i \in \Omega} I(x_i) f_r(||I(x_i) - I(x)||) g_s(||x_i - x||) \tag{2}$$

En esta ecuación $I^{filtrada}$ es la imagen filtrada, I es la imagen de entrada original, x son las coordenadas del píxel, ω son los píxeles de la ventana centrada en x, f_r es el núcleo para suavizar diferencias en intensidades y g_s es el núcleo espacial para suavizar las diferencias de coordenadas.

El peso W_p se asigna mediante la cercanía espacial y la diferencia de intensidad. Si consideramos un píxel situado en (i, j) y uno de sus píxeles vecinos en (k, l), entonces el peso asignado viene dado por la Ecuación (3).

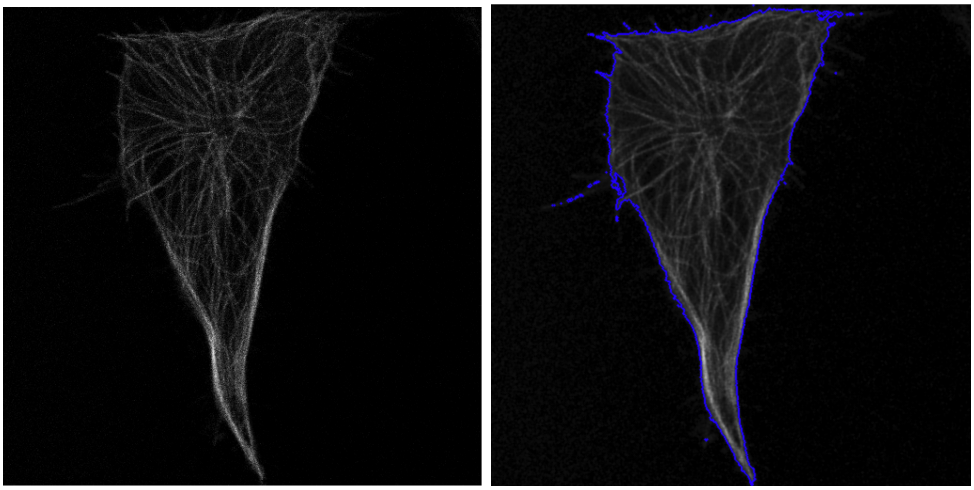
$$w(i, j, k, l) = e^{-\frac{(i-k)^2+(j-l)^2}{2\sigma_s^2} - \frac{||I(i,j)-I(k,l)||^2}{2\sigma_r^2}} \tag{3}$$

En esta ecuación σ_s representa la constante de suavizado en términos espaciales y σ_r representa la constante de suavizado en términos de intensidad de color. A través de estas constantes, se penaliza la distancia entre píxeles y la diferencia de intensidad en la escala de grises. De esta manera, solo los píxeles cercanos y similares en intensidad influyen en el resultado final.

A continuación se muestra el resultado de aplicar el *filtro bilateral* junto con el *modelo de intercambio de píxeles*. El algoritmo utilizado es extraído de la librería *OpenCV* [11]. Los parámetros que utilizamos correspondientes al filtro son los siguientes:

- $\sigma_r = 70$
- $\sigma_s = 70$
- Tamaño de la ventana: 3x3

La Figura 4.6 muestra a la izquierda la imagen preprocesada con *filtro bilateral* y a la derecha la modificada por el *modelo de intercambio de píxeles*.



(a) (b)

Figura 4.6: Aplicación del filtro bilateral e intercambio de píxeles. (a) Imagen original. (b) Imagen procesada.

Este filtro mantiene la fidelidad de la imagen y ayuda a eliminar el ruido. Sin embargo, al tener 3 parámetros de entrada implica que es necesario tener conocimiento del método para poder determinar que valores colocar a fin de obtener buenos resultados.

Por estos motivos, si bien no descartamos su posterior aplicación, decidimos buscar otro método que nos permita poder aplicar el filtro a la imagen sin la necesidad de tener conocimiento acerca de los valores adecuados para los parámetros.

4.2.3. Filtro anisotrópico

El *filtro anisotrópico* es un filtro que reduce el ruido de la imagen sin destruir sus detalles, preservando bordes y suavizando dentro de las regiones limitadas por los mismos pero no a través de ellos.

La idea del método consta de utilizar dos enfoques, por un lado lo que los autores llaman el espacio escala [12] y por otro lado utilizar un descriptor de bordes para no perder los mismos al suavizar.

El espacio escala busca obtener la misma imagen con diferentes niveles de suavizado con el fin de conseguir más información al combinarlas, dado que la primer imagen tiene muchos detalles, bordes y ruido pero la última (la más suavizada) elimina el ruido pero pierde detalles y bordes. Al combinar todas se puede obtener más información que la imagen original. En conjunto, se utiliza un descriptor de bordes, para que solo se suavicen aquellas zonas de la imagen que no se consideren bordes con el objetivo de no perderlos.

A continuación la Figura 4.7 muestra un ejemplo de aplicar el concepto de espacio escala a una imagen de ejemplo. En esta imagen no se utiliza un descriptor de bordes, por lo que se puede ver como los mismos se pierden completamente al suavizar la imagen.

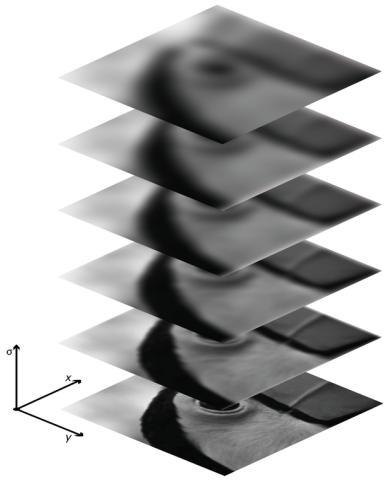


Figura 4.7: Imagen con diferentes niveles de suavizado representando el espacio escala.

Para el algoritmo se utilizó una variante que funciona especialmente para imágenes médicas de la librería [13].

A continuación se muestra el resultado de aplicar el *filtro anisotrópico* junto con el *modelo de intercambio de píxeles*. El algoritmo utilizado es extraído de la librería *Medpy* [12]. Este filtro no requiere parámetros para su aplicación.

La Figura 4.8 muestra a la izquierda la imagen preprocesada con *filtro anisotrópico* y a la derecha la modificada por el *modelo de intercambio de píxeles*.

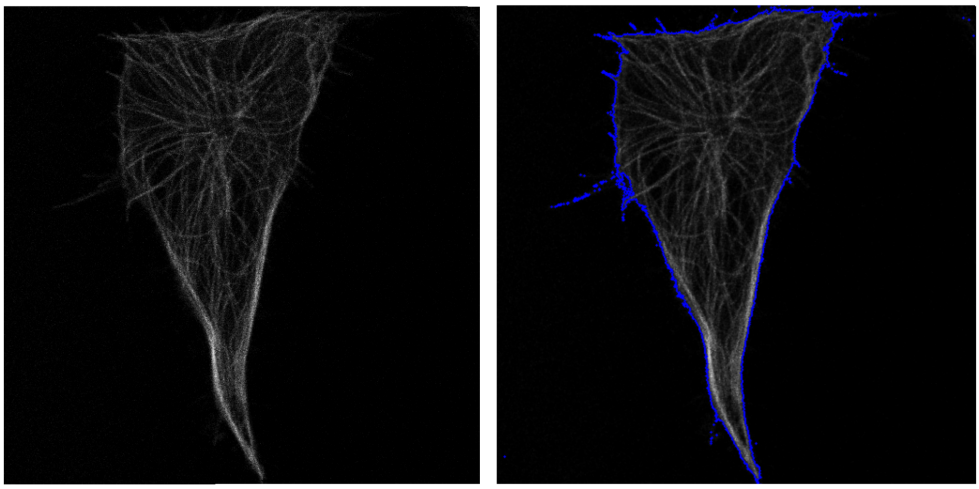


Figura 4.8: Aplicación del filtro anisotrópico y el método de intercambio de píxeles. (a) Imagen original. (b) Imagen procesada.

Este filtro logra mantener la fidelidad de la imagen al eliminar gran parte del ruido preservando sus bordes y además tiene la ventaja de que no es necesario especificar los valores de los parámetros de entrada. Debido a esto, el *filtro anisotrópico* es el método que decidimos utilizar para el preprocesamiento de las imágenes.

4.3. Gradiente Gaussiano Inverso

El gradiente inverso gaussiano es un método de preprocesamiento de imágenes que se utiliza para realzar los contornos en la misma. Como su nombre lo indica, invierte el resultado del método del *gradiente gaussiano* [14]. Este método se rige por la Ecuación (4).

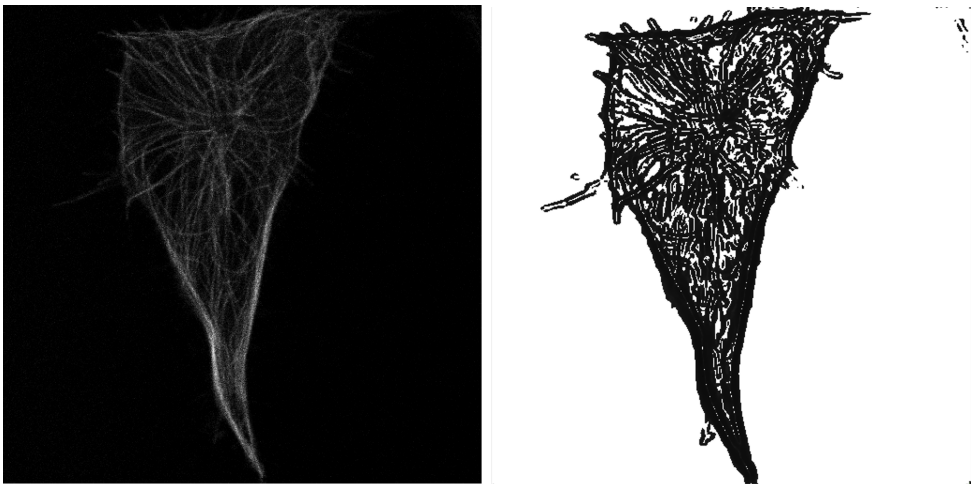
$$g(I) = \frac{1}{\sqrt{1 + \alpha \cdot |\nabla G_\sigma \cdot I|}} \tag{4}$$

El método calcula la magnitud de los gradientes en la imagen haciendo uso de diferencias gaussianas y luego invierte el resultado en el rango [0, 1]. A las áreas planas se les asignan valores cercanos a 1, mientras que a las áreas cercanas a las fronteras se les asignan valores cercanos a 0. De esta manera quedan delimitadas en la imagen las zonas que son frontera de las que no lo son.

Dentro del método se encuentran dos parámetros que son introducidos por el usuario. El parámetro α, controla la inclinación de la inversión. Un mayor valor hará que la transición entre las áreas planas y las áreas fronterizas sea más empinada en la matriz resultante. El parámetro σ tiene como función imponer la desviación estándar del filtro gaussiano aplicado sobre la imagen.

La Figura 4.9 muestra a la izquierda la imagen original y a la derecha la modificada por el método. Los parámetros que utilizamos son los siguientes:

- $\alpha = 200$
- $\sigma = 2$



(a) (b)

Figura 4.9: Aplicación del método del Gradiente Gaussiano Inverso. (a) Imagen original. (b) Imagen procesada.

Este método se utiliza en combinación con otro método que veremos en el capítulo 5, sección 5.2.

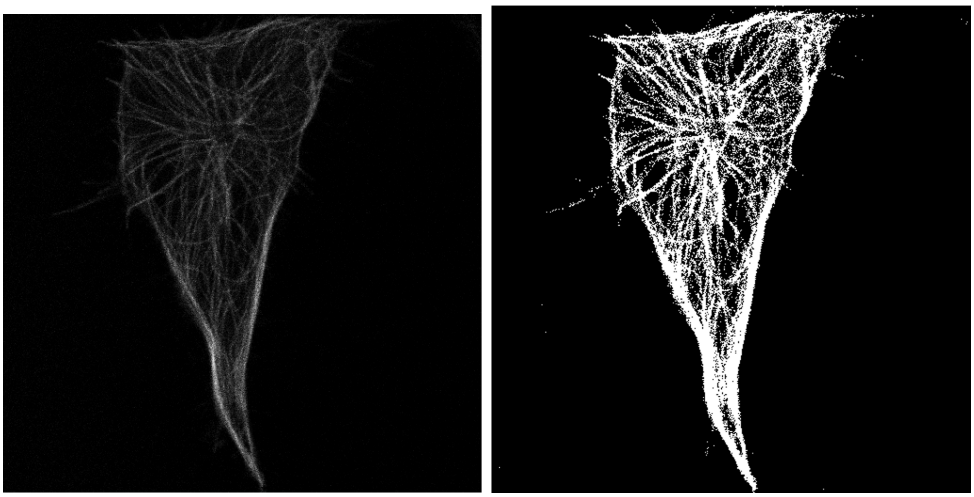
4.4. Método de Otsu

Las técnicas de umbralización buscan obtener un valor de umbral que permita binarizar a la imagen separando adecuadamente el fondo y el objeto de interés. En el caso de *Otsu*, se busca obtener el umbral óptimo [15].

Decidimos probar con técnicas de umbralización debido a la diversidad de la escala de grises de las imágenes, mencionado en el capítulo 3.

A continuación se muestra la aplicación del método de *Otsu*. El algoritmo utilizado es extraído de la librería *OpenCV* [16]. No se utilizan parámetros para su aplicación.

La Figura 4.10 muestra a la izquierda la imagen original y a la derecha la modificada por el método.



(a) (b)

Figura 4.10: Aplicación del método de Otsu. (a) Imagen original. (b) Imagen procesada.

Este método se descarta por varios motivos. Por un lado, la aplicación de un método de umbralización puede mejorar la intensidad de la estructura pero también incrementa la intensidad del ruido. Por otro lado, para métodos como el que veremos en el capítulo 5 sección 5.1 no es conveniente tener una imagen binarizada y la razón se explicará más adelante. Por último, al aplicar la umbralización se pierde una característica fundamental de la imagen que es la textura, la cual estudiaremos luego.

4.5. Método de Canny

El método de *Canny* es un método de detección de bordes más sofisticado que los métodos clásicos porque logra una buena detección minimizando falsos positivos como falsos negativos, una buena localización y una respuesta simple[17].

Este método consiste en una serie de etapas. En primer lugar se realiza un suavizado de la imagen, en este caso se realizó con *filtro bilateral*.

Luego se aplica el método de Sobel, un método basado en gradientes que busca cambios fuertes en la primera derivada de una imagen. Sobel realiza dos convoluciones de 3×3 para obtener G_x y G_y, es decir el gradiente en la dirección x y en la dirección y. Estos resultados se combinan para obtener la magnitud del gradiente mediante la fórmula (5) y el ángulo mediante la fórmula (6)

$$|G| = \sqrt{(G_x)^2 + (G_y)^2} \tag{5}$$

$$\theta = tan(G_y/G_x) \tag{6}$$

El ángulo obtenido se redondea a uno de los cuatro ángulos que representan la vertical, la horizontal y las dos diagonales (0°, 45°, 90° y 135°).

Se realiza la supresión de no máximos, con el fin de eliminar múltiples respuestas.

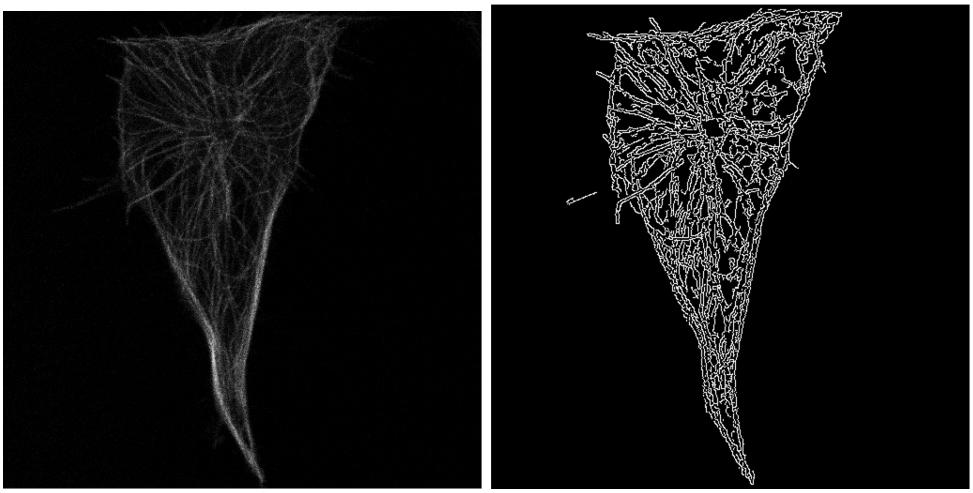
Por último, se realiza una umbralización para eliminar bordes falsos. Para ello se eligen dos umbrales, t_1 y t_2, siendo $t_1 < t_2$. Se marcan como bordes correctos aquellos píxeles cuya magnitud es mayor a t_2 o si su magnitud es mayor a t_1 y menor a t_2, y están conectados con un borde. Los parámetros σ_s y σ_r sirven para la aplicación del *filtro bileteral*, explicados en la sección 4.2.2.

A continuación se muestra la aplicación del método de *Canny*. Los parámetros que utilizamos son los siguientes:

- $\sigma_r = 5$
- $\sigma_s = 5$
- Tamaño de la ventana: 5

- $t_1 = 75$
- $t_2 = 125$

La Figura 4.11 muestra a la izquierda la imagen original y a la derecha la modificada por el método.



(a) (b)

Figura 4.11: Aplicación del método de Canny. (a) Imagen original. (b) Imagen procesada.

Este algoritmo queda descartado como descriptor de bordes dado que se pierde la fidelidad de la imagen. Sin embargo, en la sección 6.5 es utilizado para obtener el perímetro de la máscara generada a través del método $MGAC$ explicado en 5.2.

Capítulo 5

Estudio del contorno

En el presente capítulo se presentan los métodos utilizados para realizar el estudio del contorno de las células junto con los resultados obtenidos.

El estudio del contorno permitirá estudiar la geometría de la red y analizar el movimiento global de la estructura.

5.1. Modelo de intercambio de píxeles

Con el fin de obtener el contorno de las células y luego poder seguir el movimiento de la misma, optamos por utilizar el *modelo de intercambio de píxeles*[18].

El método se basa en la expansión y contracción de una región inicial, basada en dos listas de píxeles vecinos L_{in} y L_{out} . El objeto inicial es marcado a través de un rectángulo y se define a partir de esta región ambas listas de la siguiente forma:

$$L_{in} = \{x \mid \phi(x) \ < \ 0 \ \& \ \exists \ y \ \in N_4 \ (x) \ / \ \phi(y) \ > \ 0 \ \}$$

$$L_{out} = \{x \mid \phi(x) \ > \ 0 \ \& \ \exists \ y \ \in N_4 \ (x) \ / \ \phi(y) \ < \ 0 \ \}$$

donde $N_4(x)$ es el conjunto de píxeles 4-vecinos de x.

Por otro lado, se define ϕ como:

$$\phi(x) = \begin{cases} 3 & \textit{si x es un pixel en el exterior} \\ 1 & \textit{si x es un pixel en el borde exterior} \\ -1 & \textit{si x es un pixel en el borde interior} \\ 3 & \textit{si x es un pixel en el interior} \end{cases}$$

Entonces L_{in} es el borde interno de la región y L_{out} es el borde externo. En la Figura 5.1 se muestra un ejemplo, donde los píxeles amarillos representan el objeto, los píxeles rojos el borde interno, los azules el borde externo y los blancos el fondo.

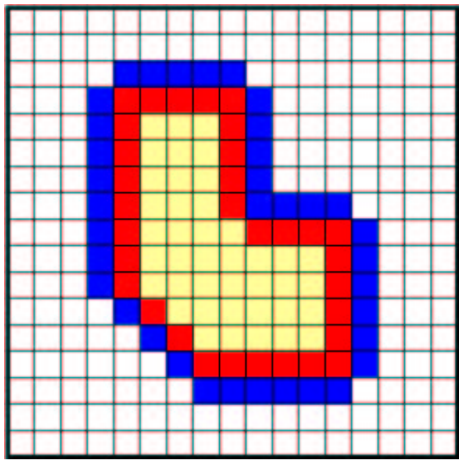


Figura 5.1: Ejemplo de la representación del objeto.

El método realiza la expansión y contracción de una región inicial a través del intercambio de píxeles entre ambas listas. Cuando el contorno se expande se quita un píxel de la lista L_{out} y se lo agrega a L_{in}. En cambio, cuando el contorno se contrae se quita un píxel de la lista L_{in} y se lo agrega a L_{out}

Es importante tener en cuenta que al pasar un píxel de una lista hacia la otra, hay que mover también sus píxeles vecinos. Por ejemplo, si un píxel x estaba en L_{out} y paso a L_{in} será necesario para todo píxel exterior k que sea vecino de x, cambiar su valor de ϕ y pasarlo a L_{out}.

La decisión de contraer o expandir se toma comparando la intensidad del píxel en cuestión con la media de la intensidad de la región actual que representa a nuestro objeto. Para ello se define una constante ϵ que sirve para comparar el módulo de la diferenciad de ambas intensidades. Si la diferencia es menor que ese épsilon, entonces el píxel en cuestión se lo considera parte del objeto de interés.

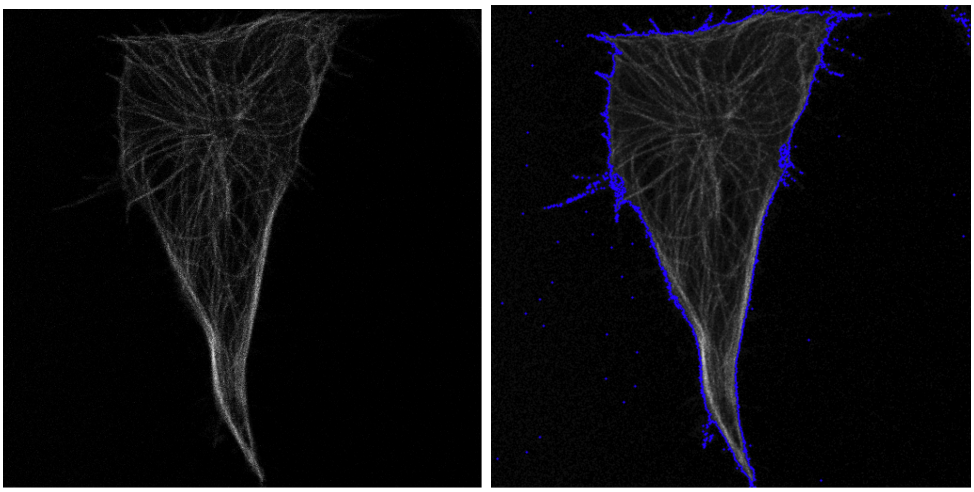
Como mencionamos anteriormente, no es conveniente utilizar este método con una imagen binarizada. En estas imágenes, el contorno del objeto no está claramente definido. Existen regiones donde tenemos saltos en la intensidad de los píxeles que forman el contorno. Por ende al aplicar este método, la región inicial puede adentrarse en el interior de la célula con facilidad al encontrar alguna discontinuidad en la intensidad del contorno.

El algoritmo termina al llegar a una determinada cantidad de iteraciones o cuando las listas no sufren cambios luego de dos iteraciones consecutivas.

A continuación se muestra la aplicación del *modelo de intercambio de píxeles*. Los parámetros que utilizamos son los siguientes:

- Número de iteraciones: 1000
- $\epsilon = 20$

La Figura 5.2 muestra a la izquierda se muestra la imagen original y a la derecha la modificada por el método.



(a) (b)

Figura 5.2: Aplicación del método de intercambio de píxeles. (a) Imagen original. (b) Imagen procesada.

Si bien el algoritmo fue capaz de contornear ciertas secciones de la célula nos encontramos con un problema. Este algoritmo solo permite seleccionar una única región inicial y existen imágenes donde es necesario tomar múltiples regiones a fin de obtener todo el contorno de la célula y no sólo una parte.

5.1.1. Mejora utilizando la media del entorno

Para introducir una mejora en el *modelo de intercambio de píxeles*, se agregó una leve modificación a la hora de comparar la intensidad del píxel en cuestión contra la intensidad de la región que se está marcando.

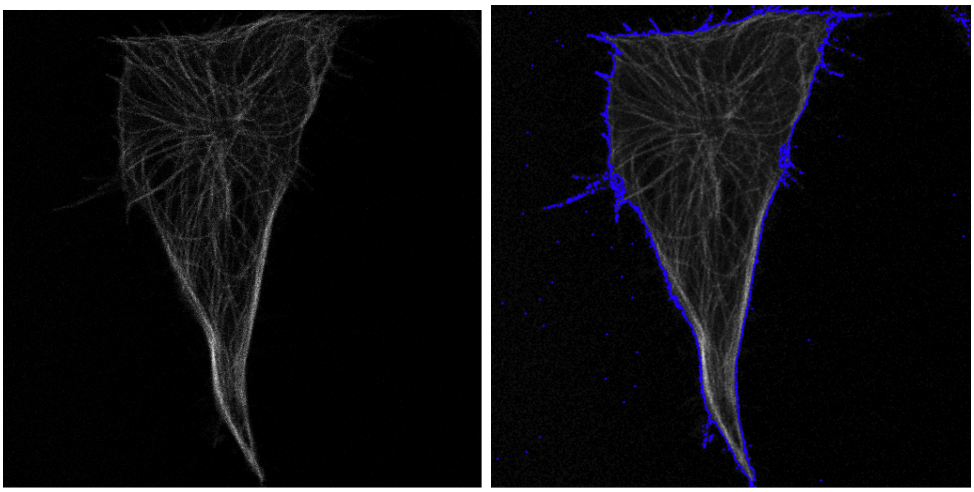
Esta optimización está basada en comparar la intensidad de la media de un entorno del píxel en lugar de comparar únicamente con el valor del píxel.

Para ello se implementaron 3 entornos distintos; de 3x3, 5x5 y 7x7.

A continuación se muestra la aplicación del *modelo de intercambio de píxeles* utilizando el entorno de los mismos. Los parámetros que utilizamos son los siguientes:

- Número de iteraciones: 1000
- Entorno: 7x7
- $\epsilon = 20$

La Figura 5.3 muestra a la izquierda la imagen original y a derecha la modificada por el método.



(a) (b)

Figura 5.3: Aplicación del método de intercambio de píxeles utilizando la media de un entorno de 7x7. (a) Imagen original. (b) Imagen procesada.

5.1.2. Mejora utilizando la varianza del entorno

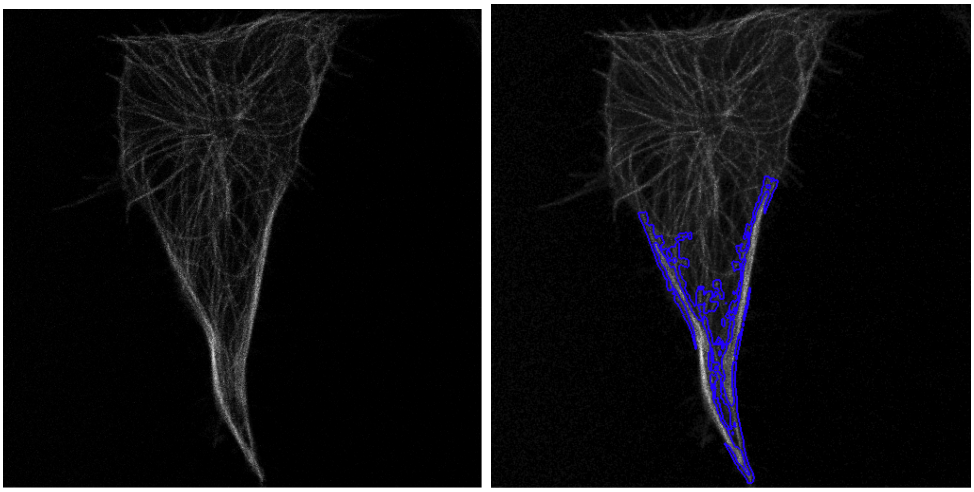
Por otro lado, también se prueba otra modificación similar a la anterior. La misma consiste en comparar con la varianza del entorno, además de con un entorno del píxel. Es decir se agrega una condición extra para establecer si el píxel puede ser incluido en la región al comparar la diferencia de sus varianzas.

Esta modificación surge ante la problemática de poder encontrar algún método que marque la estructura interna de la célula.

A continuación se muestra la aplicación del *modelo de intercambio de píxeles* utilizando el entorno y la varianza de los mismos para marcar la estructura interna. Los parámetros que utilizamos son los siguientes:

- Número de iteraciones: 1000
- Entorno: 7x7
- $\epsilon = 20$

La Figura 5.4 muestra a la izquierda la imagen original y a la derecha la modificada por el método.



(a) (b)

Figura 5.4: Aplicación del método de intercambio de píxeles utilizando la varianza de un entorno de 7x7. (a) Imagen original. (b) Imagen procesada.

Los resultados obtenidos con esta aplicación son muy variados. El método depende de múltiples factores; que la intensidad del contorno a marcar no sea muy variada, que el valor de ϵ elegido sea el correcto respecto a la escala de grises de dicha imagen y que la región inicial marcada sea correcta respecto a ϵ.

Por otro lado, esta implementación podría usarse para marcar la estructura externa como se viene mostrando hasta ahora. Pero decidimos no utilizarla debido a que agregar una condición más de comparación implica un mayor costo del algoritmo en términos temporales.

5.2. Morphological Geodesic Active Contour (MGAC)

Como vimos anteriormente, el método de *intercambio de píxeles* presenta múltiples limitaciones en este tipo de imágenes. Por esa razón, investigamos el método *Morphological Geodesic Active Contour* [19].

Este algoritmo puede usar una región inicial poligonal a diferencia del otro método que requiere una región cuadrada. Sin embargo, a fines de automatizar el método, se utiliza una región inicial cuadrada que abarca toda la imagen.

El método *Morphological Geodesic Active Contours*, por sus siglas *MGAC*, necesita 4 condiciones para que funcione correctamente:

- La imagen debe estar preprocesada para que sus bordes estén resaltados. Esto se logra utilizando el método del *Gradiente Gaussiano Inverso*, explicado en el capítulo 4 sección 4.3.
- Definir una región inicial que esté próxima a los bordes del objeto, ya sea dentro o fuera del objeto. Esto puede ser automatizado si la imagen contiene únicamente el objeto a detectar, o sino se puede usar la intervención del usuario.
- Definir si la región se expande o reduce. Esto lo define el usuario a través de los parámetros.
- Definir un umbral, para el cual se considera si un píxel es borde o no. Este umbral se define por parámetro.

Con la imagen preprocesada, se tiene una matriz de valores entre [0, 1], siendo los valores próximos a 0 los bordes. La región, iteración a iteración, se expande o reduce hasta que todo punto de la misma se encuentre con un borde o hasta que se cumpla el número máximo de iteraciones

Con respecto a los parámetros del método, las iteraciones representan el número de veces máxima que la región se expande o contrae.

El parámetro balloons es aquel que indica si la región debe expandirse o contraerse. Esta expansión o contracción de la región se ve representada por la Ecuación (4).

$$\frac{\partial u}{\partial t} = g(I) \cdot v \cdot |\nabla u| \tag{4}$$

Dentro de la ecuación, la variable v es la que el usuario ingresa en el método. Si dicha variable es negativa, la región se contrae. Por otro lado, si la variable es positiva, la región se expande.

El suavizado (smoothing) representa que tan exacta va a ser la curva de la región con respecto al objeto. A mayor suavizado se obtiene una curva más redondeada, pero su exactitud es menor. La fuerza que describe el suavizado de la curva se observa en la Ecuación (5).

$$\frac{\partial u}{\partial t} = g(I) \cdot |\nabla u| \cdot div(\frac{\nabla u}{|\nabla u|}) \tag{5}$$

Por último, el umbral representa el valor límite en el que un píxel es considerado frontera. Durante la evolución de la curva, los puntos que la conforman detienen su expansión o contracción al encontrar este valor. En las pruebas analizadas se usaron valores entre 0,35 y 0,45.

A continuación se muestra el resultado de preprocesar la imagen con el filtro anisotrópico y el gradiente gaussiano inverso. Los parámetros que utilizamos son los siguientes:

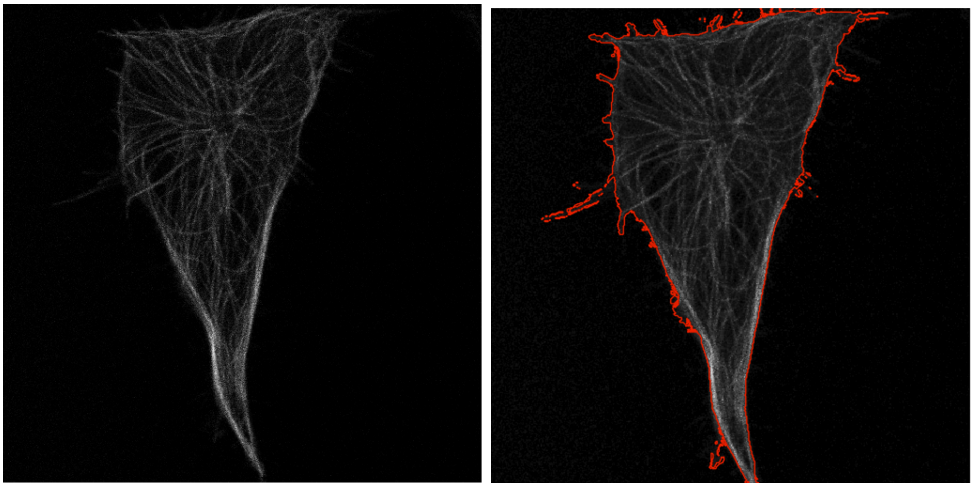
- Número de iteraciones: 250
- Ballons: -1
- Suavizado: 0
- Umbral: 0,35

Y para un procesamiento previo al $MGAC$ con el gradiente inverso gaussiano:

- $\alpha = 200$
- $\sigma = 2$

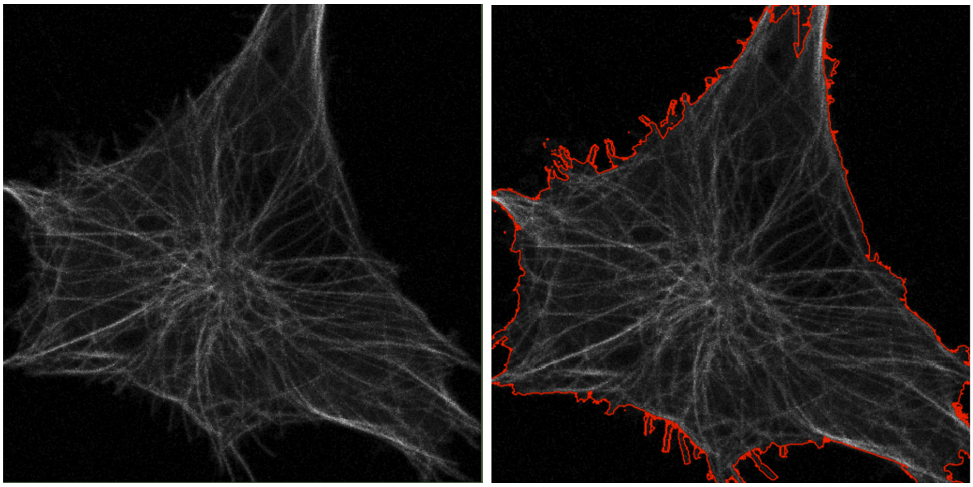
El algoritmo utilizado para el *MGAC* es extraído de la librería *Morphsnakes* [19], pero además se ha modificado parte del código fuente para nuestra aplicación.

Las Figuras 5.5, 5.6 y 5.7 muestran a la izquierda la imagen preprocesada y a derecha la modificada por el algoritmo de *MGAC*:



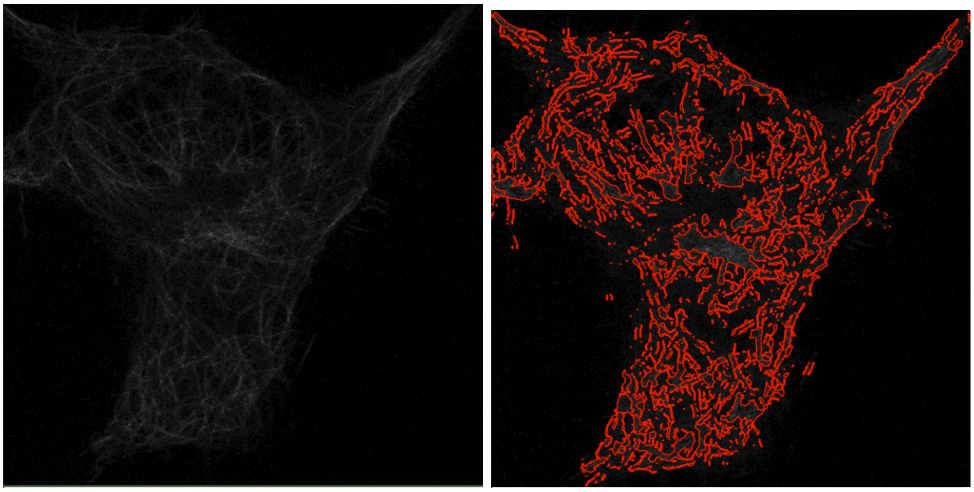
(a) (b)

Figura 5.5: Aplicación del método *MGAC*. (a) Imagen original. (b) Imagen procesada.



(a) (b)

Figura 5.6: Aplicación del método *MGAC*. (a) Imagen original. (b) Imagen procesada.



(a) (b)

Figura 5.7: Aplicación del método *MGAC*. (a) Imagen original. (b) Imagen procesada.

Como se pueden ver en los resultados, no todas las imágenes responden correctamente en la obtención del contorno. Este efecto se puede ver en aquellas imágenes cuya luminosidad es baja. Esto nos lleva a buscar algún preprocesamiento que resalte aún más los contornos a fin de poder usar el *MGAC* para obtener el contorno deseado.

5.3. Umbralización adaptativa gaussiana

El problema indicado en la sección anterior se resuelve utilizando un método de umbralización llamado *adaptive threshold* [20]. El mismo resulta efectivo para imágenes con poco contraste.

Este método requiere de dos variables de entrada. El tamaño del bloque representa el tamaño de la máscara que se utiliza para calcular el valor del umbral. La constante épsilon ϵ se resta del umbral calculado anteriormente.

La máscara se pasa píxel a píxel, centrando la máscara en el píxel para el cual queremos calcular el umbral. Sólo admite máscaras de tamaño impar para poder centrar correctamente la máscara en el píxel.

Existen dos formas de calcular el valor del umbral:

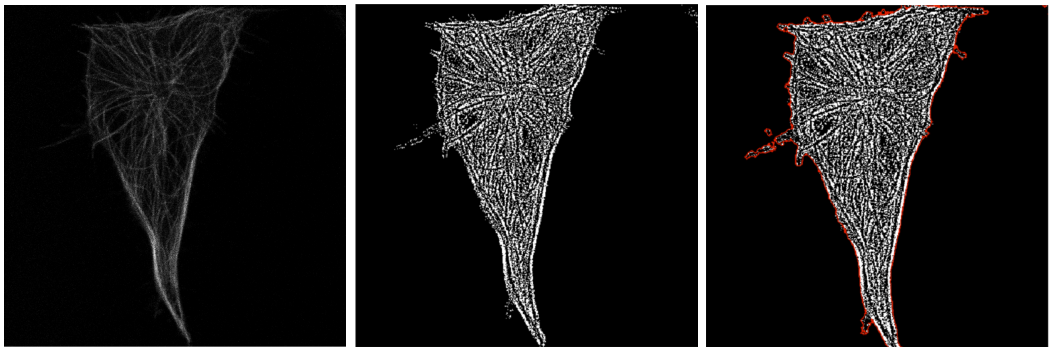
- Método de la media: El valor del umbral es la suma del valor de cada píxel de la máscara sobre el total de píxeles de la máscara.

- Método Gaussiano: El valor del umbral se calcula como la suma de los valores resultantes de multiplicar la máscara Gaussiana con los valores de los píxeles de la imagen.

Una vez calculado el umbral se le resta la constante ϵ. Si el valor del píxel está por encima del nuevo umbral calculado, entonces adquiere el valor 255 y sino adquiere el valor 0.

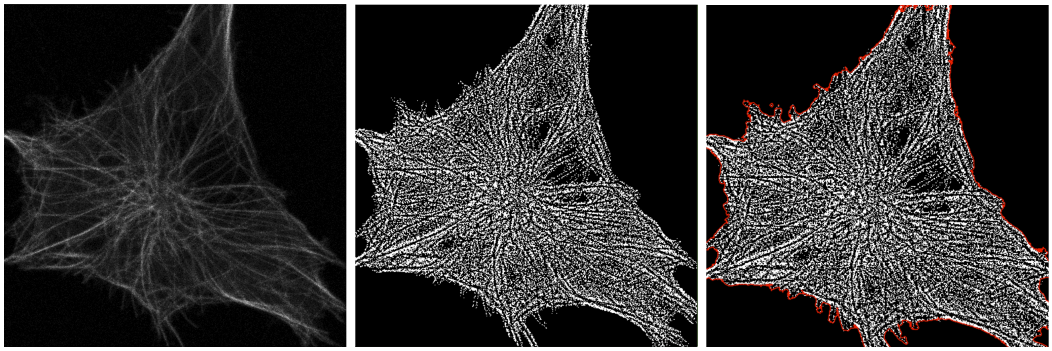
Las Figuras 5.8, 5.9 y 5.10 muestran a la izquierda la imagen original, en el medio la imagen preprocesada con *filtro anisotrópico* y el *método adaptive gaussian threshold* y a la derecha la modificada por el algoritmo de *MGAC*. En cuanto a los parámetros de entrada, usamos los mismos que las pruebas anteriormente mostradas en la sección 5.2. Los parámetros utilizados en las pruebas son:

- Tamaño del bloque: 11
- $\epsilon = 5$



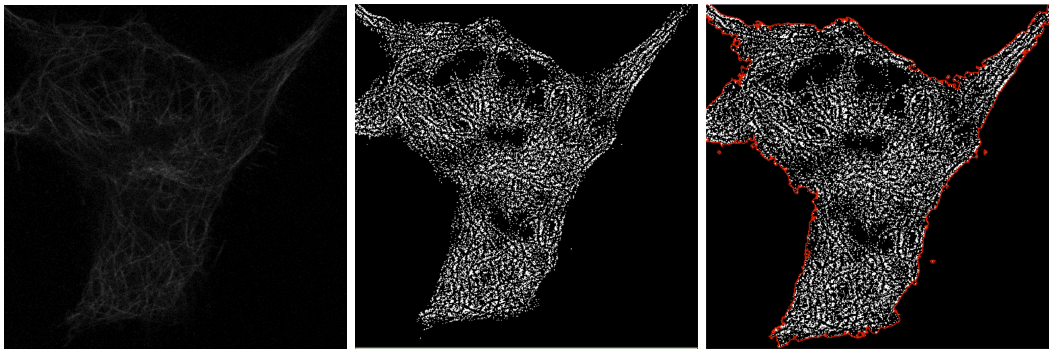
(a) (b) (c)

Figura 5.8: (a) Imagen original. (b) Imagen procesada (c) Imagen con *MGAC* aplicado.



(a) (b) (c)

Figura 5.9: (a) Imagen original. (b) Imagen procesada (c) Imagen con *MGAC* aplicado.



(a) (b) (c)

Figura 5.10: (a) Imagen original. (b) Imagen procesada (c) Imagen con *MGAC* aplicado.

Con este nuevo método de preprocesamiento se ve en los resultados que se sacrifica fidelidad de la imagen pero se puede obtener con un solo movimiento todo el contorno de la célula haciendo uso del *MGAC*.

Capítulo 6

Estudio del movimiento

En el presente capítulo se presentan los métodos utilizados para realizar el estudio del movimiento de la célula dentro de la película junto con los resultados obtenidos.

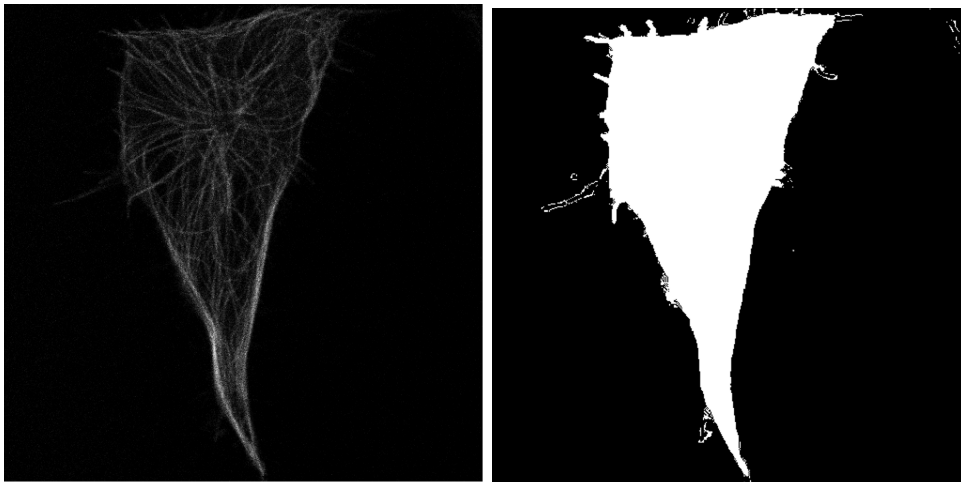
Se muestran los métodos que utilizamos para la detección del movimiento tanto interno como externo de las células. Para ello, se realiza una combinación de los métodos discutidos en capítulos previos, para luego detectar el movimiento que éstas realicen imagen a imagen.

En cada uno de los métodos el resultado final obtenido es un mapa de calor, donde los colores más claros muestran zonas de mayor movimiento y por el contrario los colores más oscuros muestran zonas de menor movimiento.

6.1. Detección de movimiento con MGAC

La utilización del método *MGAC* nos permite obtener el contorno de la célula. Mediante la utilización de este contorno es posible generar una máscara que distinga la célula del fondo.

La Figura 6.1 muestra a la izquierda la imagen de la célula pre-procesada y a derecha la máscara que se puede generar a partir del método *MGAC*:



(a) (b)

Figura 6.1: Máscara de célula a partir del método *MGAC*. (a) Imagen original. (b) Máscara.

La idea del algoritmo es generar una máscara en cada cuadro de la película para luego comparar la imagen actual con la anterior y detectar si se produjeron cambios.

Se inicia con una matriz de ceros donde cada celda corresponde a un píxel. Por cada movimiento detectado, se le suma 1 a la celda. Este proceso se realiza cuadro a cuadro.

La Figura 6.2 muestra el movimiento de la célula:

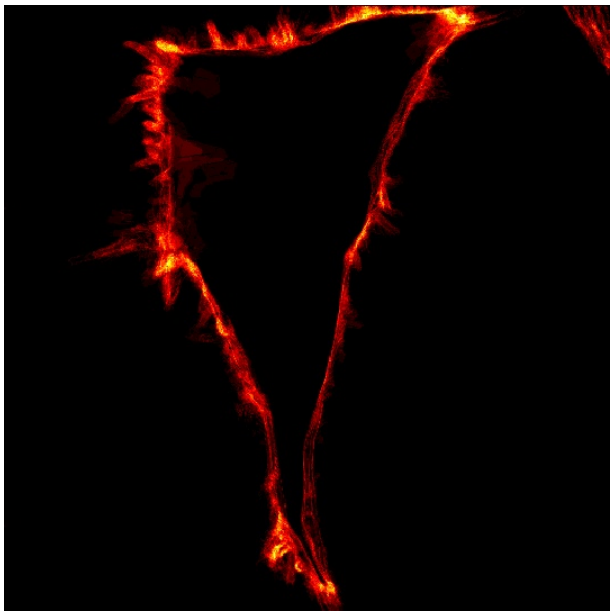


Figura 6.2: Mapa de movimiento perimetral con *MGAC*.

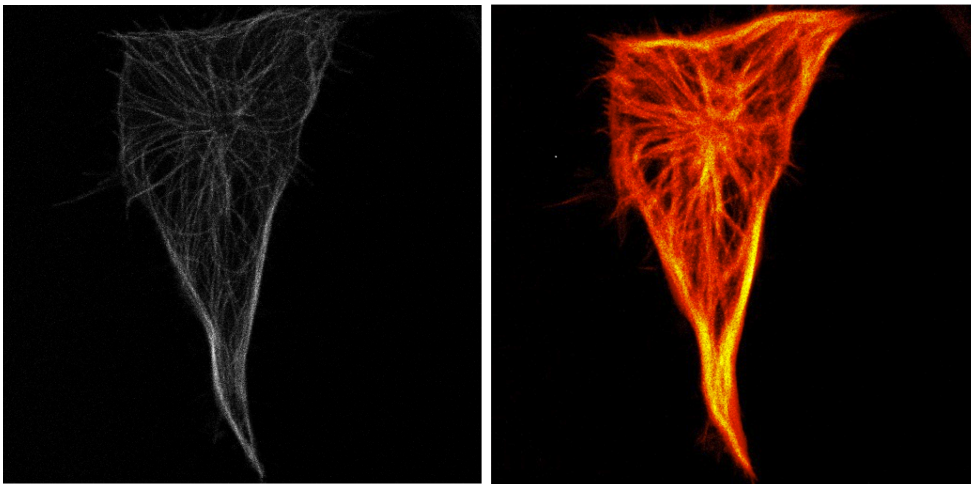
Dado que el *MGAC* solo puede distinguir la célula del fondo, el algoritmo no puede detectar el movimiento que ocurre en el interior de la célula.

6.2. Detección de movimiento por diferencia de píxel

En este algoritmo, se detecta movimiento cuando la diferencia en valor absoluto entre el píxel actual y el píxel en la imagen anterior superan un umbral. Este umbral es el único parámetro del método.

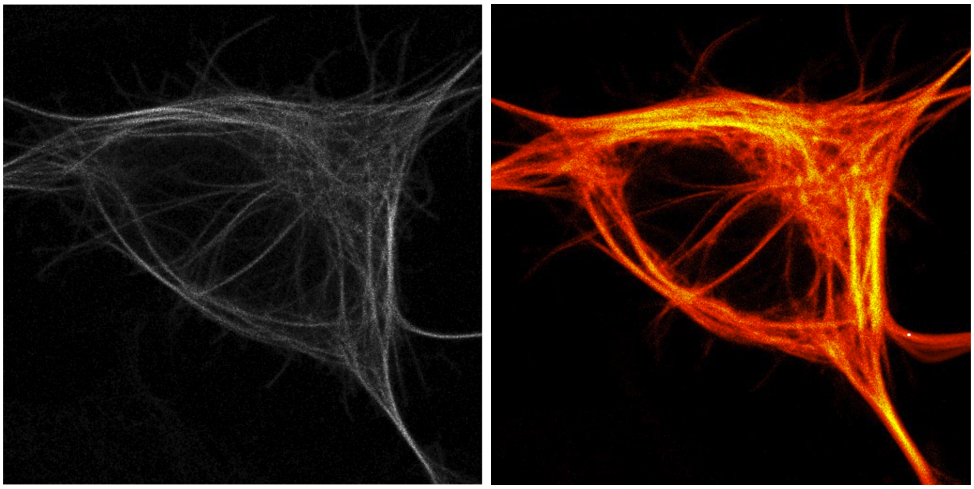
En las Figuras 6.3, 6.4 y 6.5 se muestra a la izquierda la imagen de la célula preprocesada y a derecha el movimiento que realiza en la película. El parámetro de entrada utilizado es:

- $\epsilon = 15$



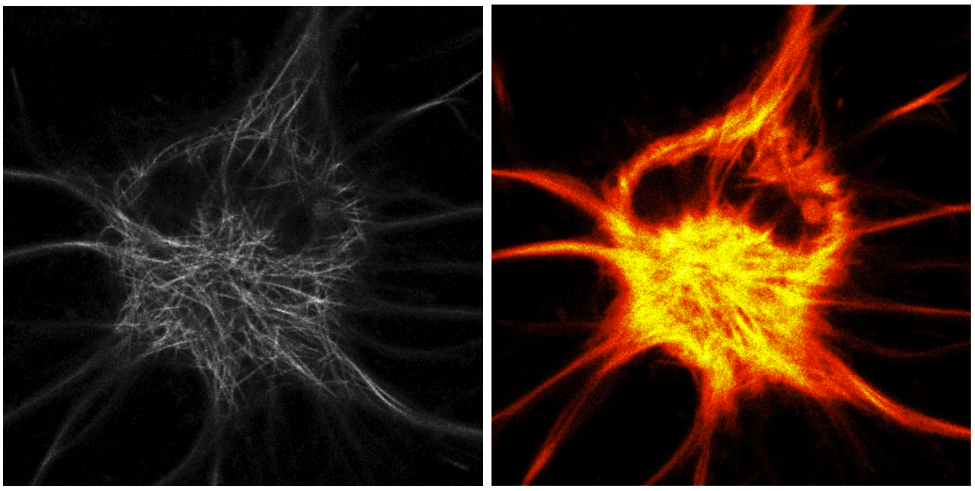
(a) (b)

Figura 6.3: (a) Imagen original. (b) Mapa de movimiento.



(a) (b)

Figura 6.4: (a) Imagen original. (b) Mapa de movimiento.



(a) (b)

Figura 6.5: (a) Imagen original. (b) Mapa de movimiento.

6.3. Detección de movimiento por umbralización

En este algoritmo, la idea es primero umbralizar las imágenes a fin de poder diferenciar la célula del fondo. Con las imágenes umbralizadas, la matriz de píxeles solo puede tener valores binarios que representan donde está la célula y donde no lo está. Todo valor que esté por debajo del parámetro de umbralización será considerado fondo. El umbral que se utiliza en el algoritmo es un parámetro del método.

Una vez realizada la umbralización, se detecta el movimiento cuando la diferencia de píxeles en valor absoluto es distinta de cero. En la Figura 6.6 se muestra el funcionamiento del algoritmo, suponiendo una película de 4 frames. El máximo valor es el píxel que mas fluctúa en la película.

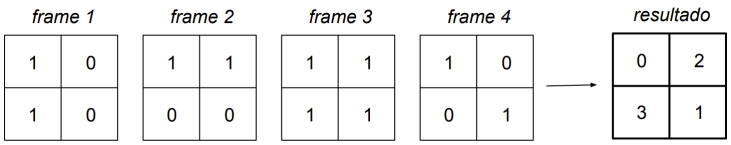


Figura 6.6: Diagrama explicativo del algoritmo.

Para el mapa de calor se utiliza una barra de colores como el ejemplo mostrado en la Figura 6.7. Los valores representan la cantidad de movimiento que hubo en toda la película para un píxel determinado. Los mismos incrementan hacia colores más claros. Por lo tanto habrá mas movimiento en las zonas donde el mapa de calor sea mas claro.

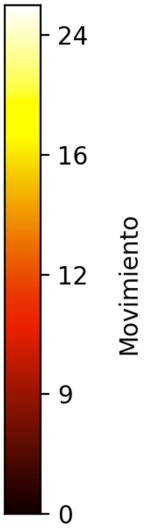
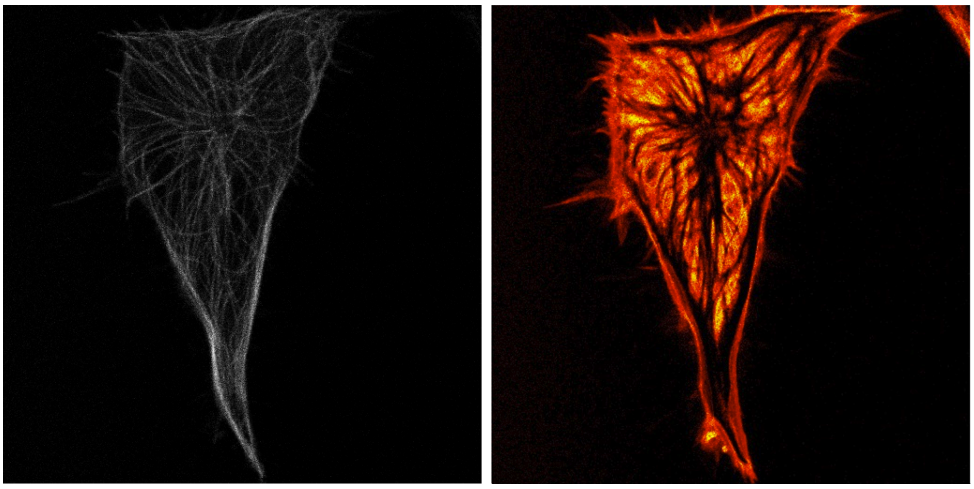


Figura 6.7: Barra de ejemplo utilizada en los mapas de calor.

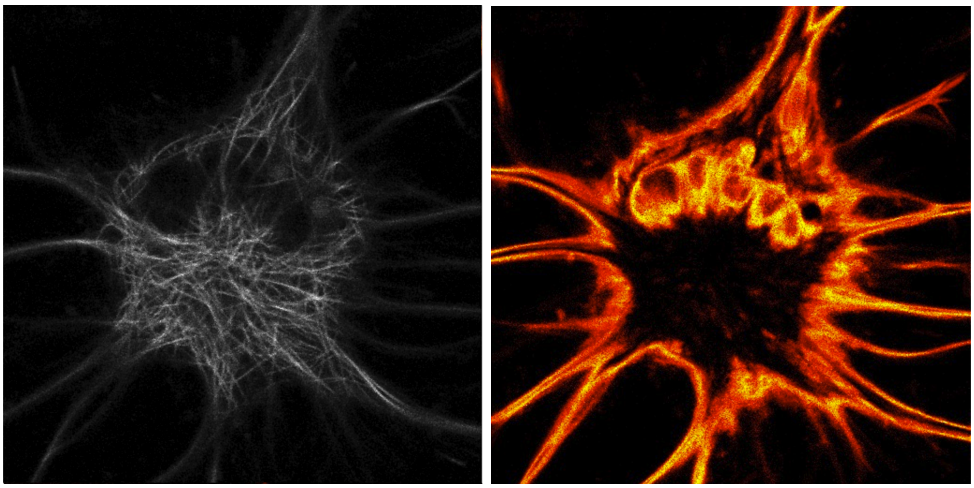
Las Figuras 6.8, 6.9 y 6.10 muestran a la izquierda la imagen de la célula preprocesada y a derecha el movimiento que realiza en la película. El parámetro de umbralización utilizado es:

- Umbral = 10



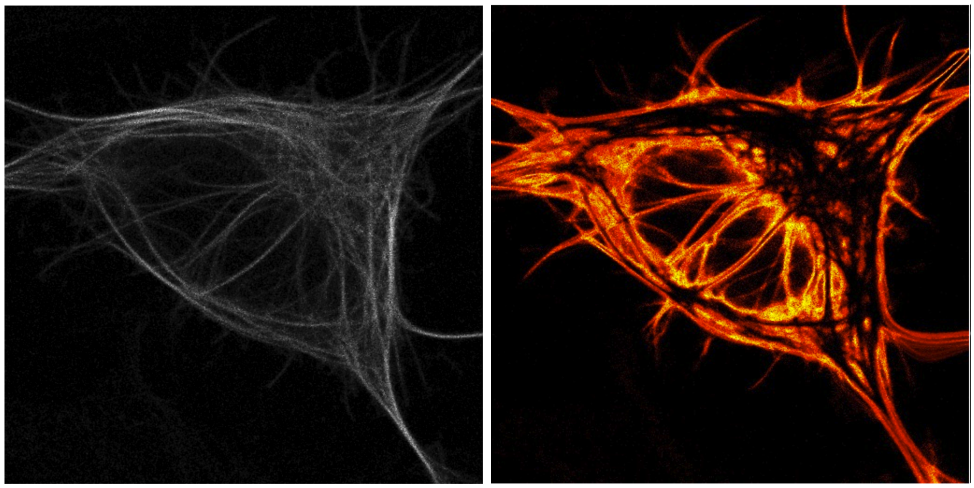
(a) (b)

Figura 6.8: (a) Imagen original. (b) Mapa de movimiento con umbralización.



(a) (b)

Figura 6.9: (a) Imagen original. (b) Mapa de movimiento con umbralización.



(a) (b)

Figura 6.10: (a) Imagen original. (b) Mapa de movimiento con umbralización.

6.4. Detección de movimiento a través del método de intercambio de píxeles

Para poder explicar cómo se detecta el movimiento a través del método de intercambio de píxeles, es necesario retomar la definición de las listas explicadas en el capítulo 5 sección 5.1.

Para cada imagen de la película, el método realiza varias iteraciones donde el contorno se acomoda al objeto de interés. A través de la utilización de una matriz para representar el movimiento, se incrementa la celda correspondiente a un píxel cada vez que en esa posición se produce un movimiento. Esto se hace comparando la lista L_{in} de una iteración con la de la iteración anterior.

Para todos nuestros píxeles en la lista L_{in}, si un píxel no pertenecía a la lista en la iteración anterior, significa que en dicho píxel se realizó un movimiento. Por lo tanto, se incrementa en la matriz su valor.

Esto se repite no solo en las iteraciones dentro de cada imagen sino también en las iteraciones subsiguientes para toda la película. Por último, se realiza una normalización de los valores de la matriz de movimiento y se genera un mapa de calor.

La Figura 6.11 muestra el movimiento de la célula:

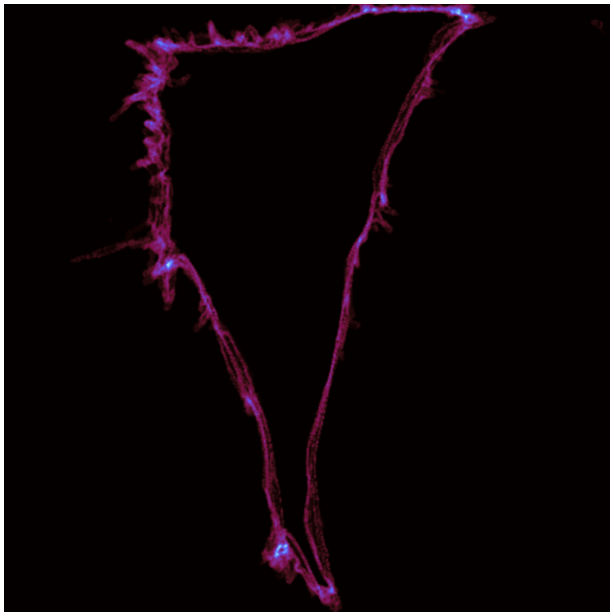


Figura 6.11: Mapa de movimiento utilizando intercambio de píxeles.

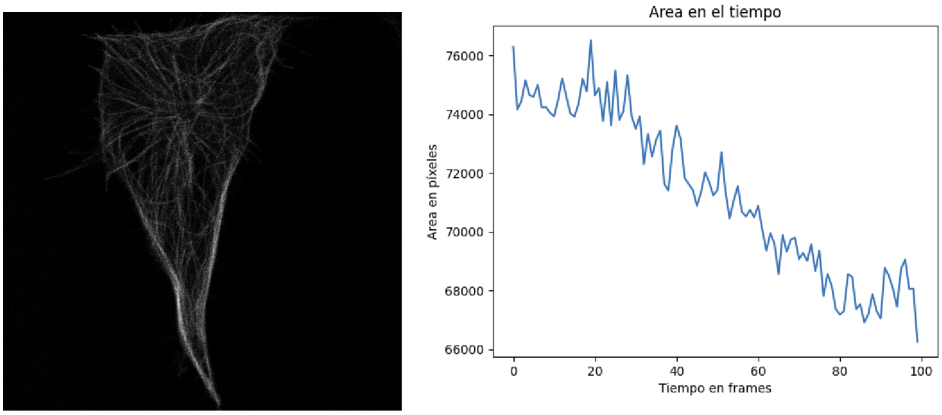
Si bien los colores del mapa de calor no son los mismos que los expuestos en la Figura 6.7, en este caso también se cumple que los colores más claros indican mayor movimiento.

6.5. Analisis del área, perímetro y razón de ejes

Como se observa en la sección 6.1, a través del método *MGAC*, se puede obtener una máscara la cual determina qué es célula y qué es fondo. Esto resulta útil para analizar el área, perímetro y razón de ejes de las células.

Dado que se tiene una película de cada célula, se calculan las tres propiedades en cada frame para luego graficarlas y observar su comportamiento a través del tiempo. Las medidas que se obtienen en los resultados mostrados a continuación están en píxeles.

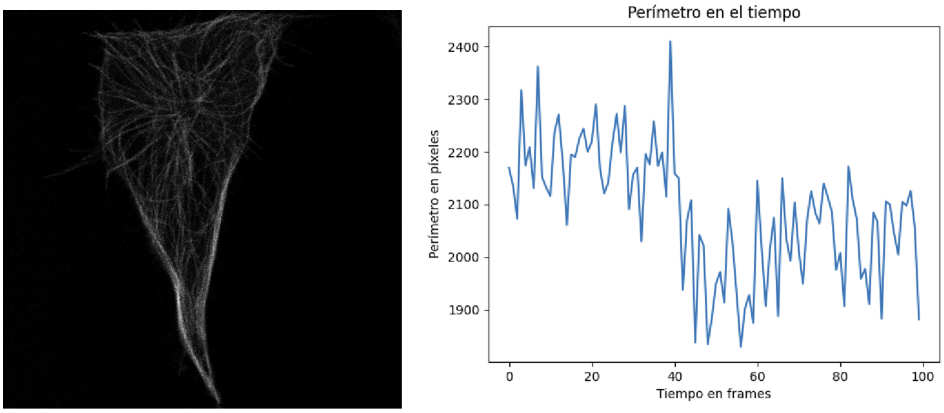
Para calcular el área en cada frame, se cuenta la cantidad de píxeles pertenecientes a la máscara. Los resultados pueden apreciarse en la Figura 6.12.



(a) (b)

Figura 6.12: Cálculo del área en el tiempo. (a) Imagen original. (b) Gráfico.

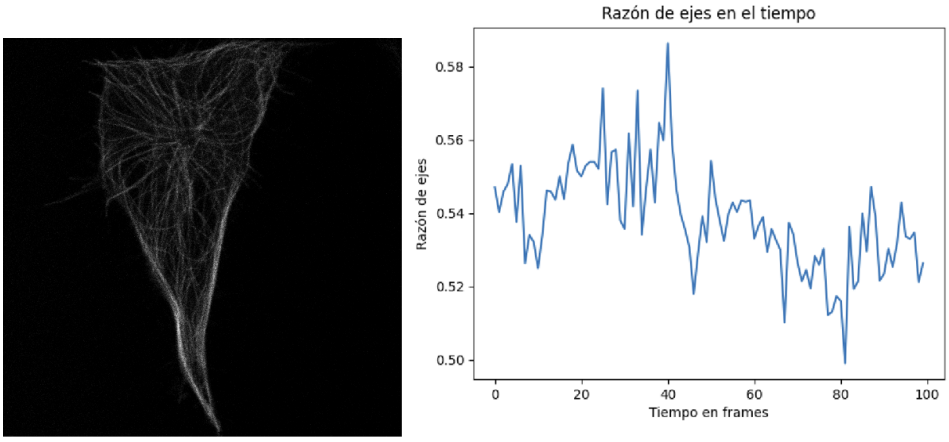
Para calcular el perímetro en cada frame, se utiliza el algoritmo de *Canny*. Una vez que la máscara pasa por dicho algoritmo, quedan únicamente sus bordes. Por lo tanto para obtener el perímetro del frame basta con sumar todos los píxeles de la imagen. Los resultados pueden apreciarse en la Figura 6.13.



(a) (b)

Figura 6.13: Cálculo del perímetro en el tiempo. (a) Imagen original. (b) Gráfico.

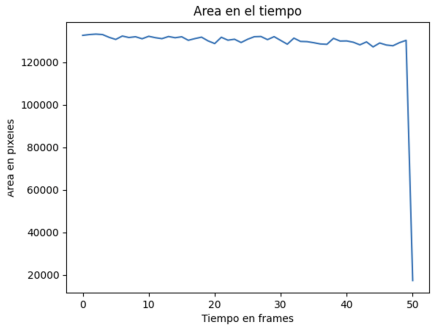
Para calcular la razón de ejes, se debe primero buscar dentro de la máscara aquella fila y columna que tengan la mayor cantidad de píxeles. Luego la razón se calcula como la división entre aquella fila y columna. Los resultados pueden apreciarse en la figura 6.14.



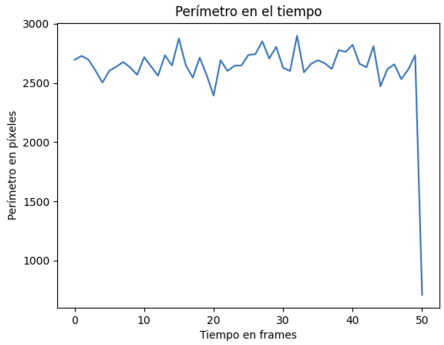
(a) (b)

Figura 6.14: Cálculo de la razón de ejes en el tiempo. (a) Imagen original. (b) Gráfico

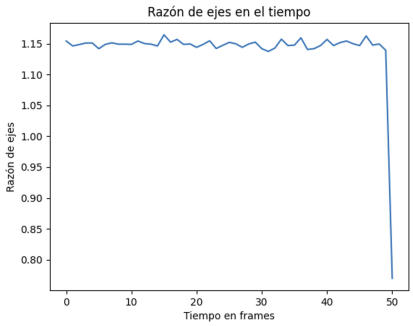
Para finalizar, es importante aclarar que para el cálculo de dichas propiedades en el tiempo se omite el último frame de la película. Esto se debe a una falla en el detector que genera, en ciertos casos, que no detecte la célula completamente. Podemos ver a continuación en la Figura 6.15 el error en el cálculo de las tres propiedades en el último frame.



(a)



(b)



(c)

Figura 6.15: Cálculo de las tres propiedades en el tiempo. Se observa en el último frame una caída abrupta debido al error en la detección. (a) Gráfico del área. (b) Gráfico del perímetro. (c) Gráfico de la razón de ejes.

6.6. Resultados

Para finalizar el capítulo, se presentan a continuación los resultados finales. Estos resultados sirven para poder analizar la distribución de áreas, perímetros y razones de ejes que tienen las células del conjunto.

Se puede ver en la Figura 6.12 la distribución de áreas de las 32 células del conjunto.

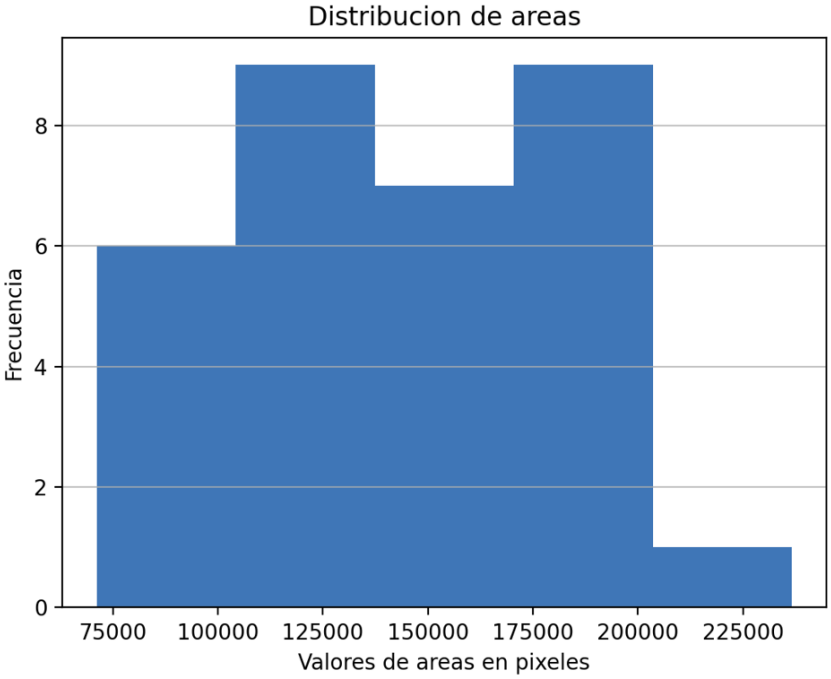


Figura 6.16: Distribución de áreas encontradas en el conjunto.

Se puede ver en la Figura 6.16 que la mayoría de las células tienen un área que oscila entre 100.000 y 180.000 píxeles cuadrados.

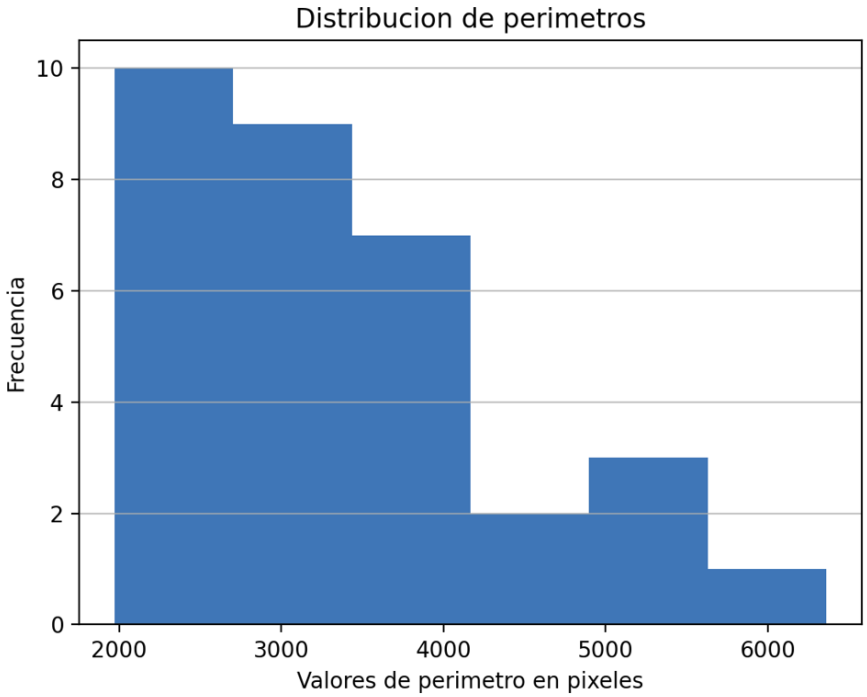


Figura 6.17: Distribución de perímetros encontrados en el conjunto.

Se puede ver en la Figura 6.17 que la mayoría de las células tienen un perímetro menor a 3750 píxeles.

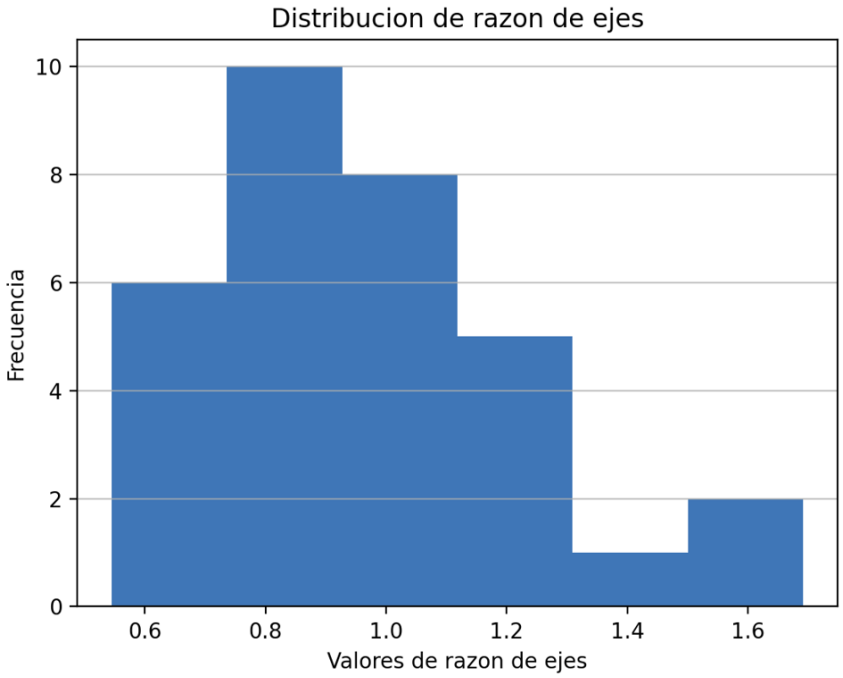


Figura 6.18: Distribución de razón de ejes encontrados en el conjunto.

Se puede ver en la Figura 6.18 que más de la mitad de las células tienen más altura que anchura. Por otro lado, muy pocas células gozan de tener la propiedad de ser redondas, dado que sólo 3 se acercan al valor 1.0.

Capítulo 7

Estudio de textura

En el presente capítulo se presentan los métodos utilizados para realizar el estudio de la textura junto con los resultados obtenidos.

La textura es una de las características principales de cualquier imagen, que nos permite conocer la relación entre los niveles de grises. Nos interesa conocer la complejidad de la red de filamentos que forman el interior de la célula.

7.1. Entropías

La entropía es una medida de incertidumbre. Para una imagen, puede detectar variaciones en los niveles de grises. Toma valores entre 0 y 1:

1. Los valores próximos a 0 representan muestras más homogéneas.
2. Los valores próximos a 1 representan muestras más heterogéneas.

La *entropía de Shannon* está definida según la Ecuación 6.

$$S = -\sum_{0}^{255}(p_k * log(p_k)) \tag{6}$$

donde p_k es la probabilidad (obtenida de la normalización del histograma de la imagen) asociada a un nivel de gris k.

Esta fórmula computa la entropía global de una imagen, lo que nos da como resultado un número. Pero a su vez, se puede definir la entropía local, es decir la entropía relacionada con la complejidad contenida en un vecindario. El método utilizado es extraído de la librería *scikit-image* [21]. Este método nos permite obtener un mapa de calor de la entropía de toda la imagen.

A continuación la Figura 7.1 muestra un ejemplo del cálculo de la entropía de una imagen:

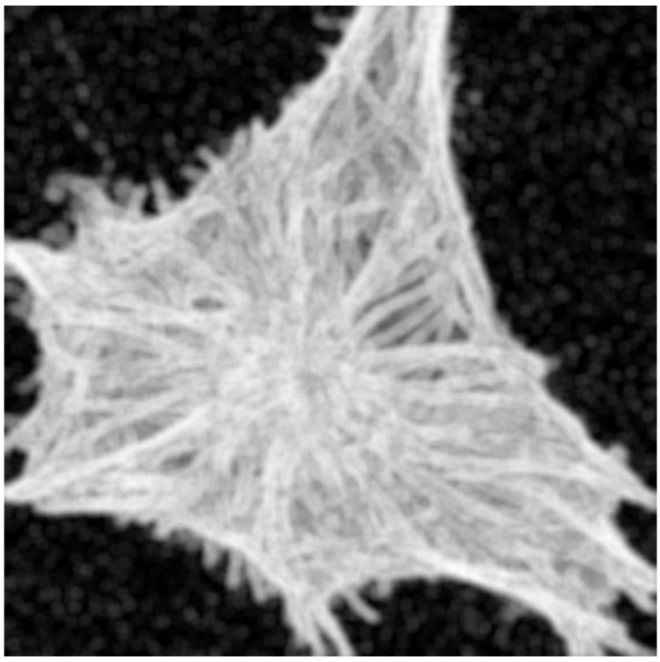


Figura 7.1: Cálculo de la entropía en una célula. Píxeles más blancos representan zonas heterogéneas. Píxeles más negros representan zonas más homogéneas.

7.2. Dimensión fractal

Un fractal es un objeto geométrico cuya estructura irregular, se repite a diferentes escalas.

La dimensión fractal es una proporción que indica como el patrón/fractal cambia con la escala que se mide. También se la caracteriza como una medida de la capacidad de llenado del espacio de un patrón/fractal. La dimensión fractal está definida solamente para un conjunto autosimilar pero puede estimarse para una imagen.

Existen múltiples formas de estimar la dimensión fractal de una imagen. En nuestro caso utilizaremos la dimensión *box-counting* (Minkowski-Bouligand).

En un conjunto acotado X considerado espacio euclidiano, se dice que el espacio es autosimilar cuando X es la unión de N_r copias no superpuestas de si mismo. Cada copia es similar a X, pero reducida por un ratio r que representa la escala. La dimensión fractal se puede describir a través de la Ecuación 7.

$$D = \frac{log(N_r)}{log(1/r)} \tag{7}$$

Para calcular la dimensión de la *box-counting*, la figura tiene que estar dividida en una cuadrícula con una longitud r y cubrir la figura con dichas çajas". Todas las cajas que cubren la figura serán contadas como N.

Como muestra la Figura 7.2 el número de cajas cubriendo la figura cambia cuando se toma una caja de diferente longitud.

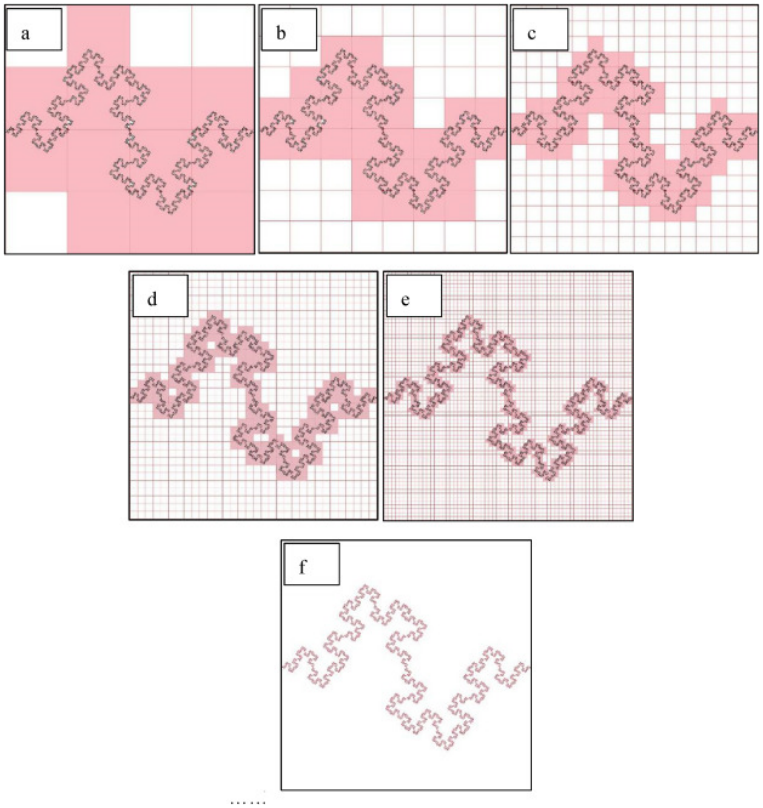


Figura 7.2: El número N de cajas cambia con la longitud de r. Fuente: https://www.sciencedirect.com/science/article/pii/S2590123020300128.

El algoritmo para realizar la *box-counting* consiste en la siguiente serie de pasos:

1. Binarizar la imagen mediante un umbral.
2. Determinar las escalas r que se van a utilizar.
3. Para cada escala se calcula el box-count. Se cuenta la mínima cantidad de cajas que fueron tocadas por la figura. Para esto se cuenta la cantidad de cajas donde no todos los píxeles sean ceros y donde no todos los píxeles sean uno. Si la suma de los píxeles da como resultado un valor entre $(0, r^2)$, entonces significa que tenemos dentro píxeles que son cero y píxeles que son uno.
4. Aproximamos linealmente el conjunto de escalas con el conjunto de los N calculados.
5. La dimensión *box-counting* es la pendiente de la recta aproximante.

7.3. Perfil de textura

Tomando como idea los perfiles de intensidad del programa ImageJ utilizados en el capítulo 3, se grafica la textura a través de un perfil.

Este perfil se traza sobre la imagen original y se calcula para cada píxel de dicho perfil su valor de dimensión fractal y su valor de entropía. Luego, estos valores se normalizan y posteriormente se grafica respecto a la coordenada de cada píxel.

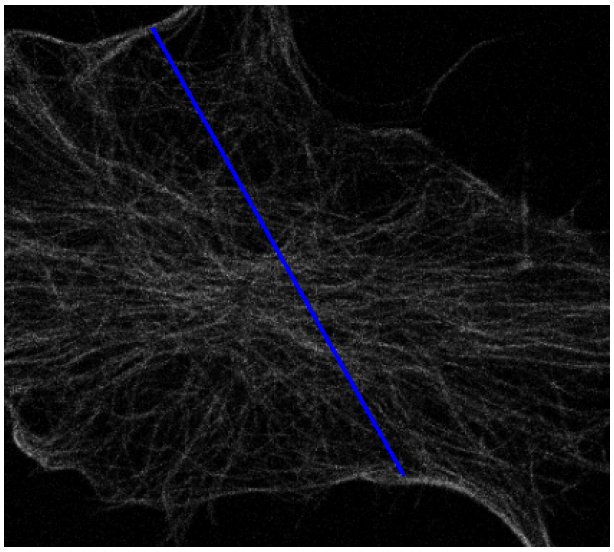


Figura 7.3: El perfil sobre la imagen para la cual se calcula la entropía y la dimensión fractal sobre cada píxel del mismo.

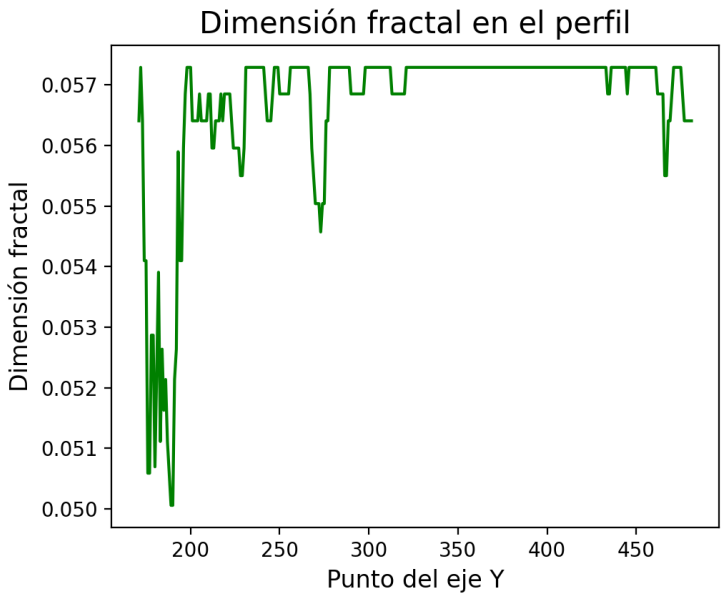


Figura 7.4: Gráfico de la dimensión fractal en relación a cada punto del perfil trazado.

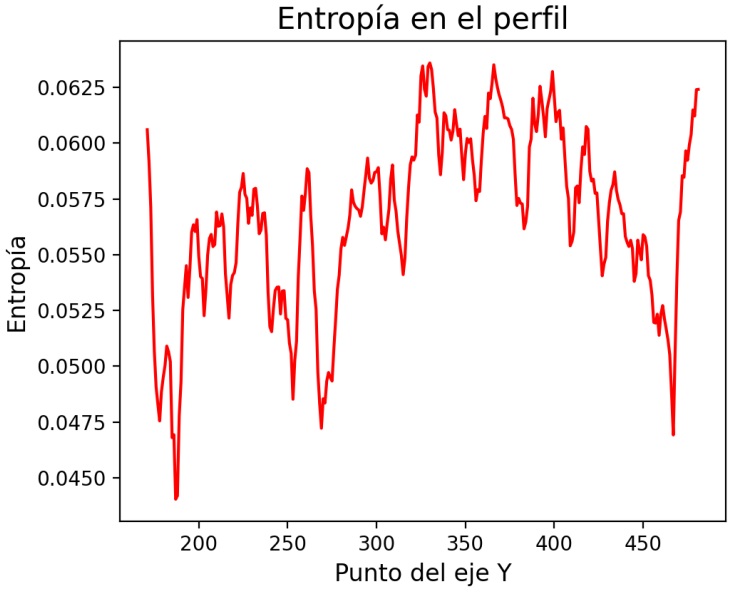


Figura 7.5: Gráfico de la entropía en relación a cada punto del perfil trazado.

También, se genera un gráfico combinando ambas métricas de la textura. La combinación que se realiza consiste en tomar para cada píxel ambos valores, multiplicarlos por 0,5 y finalmente sumarlos.

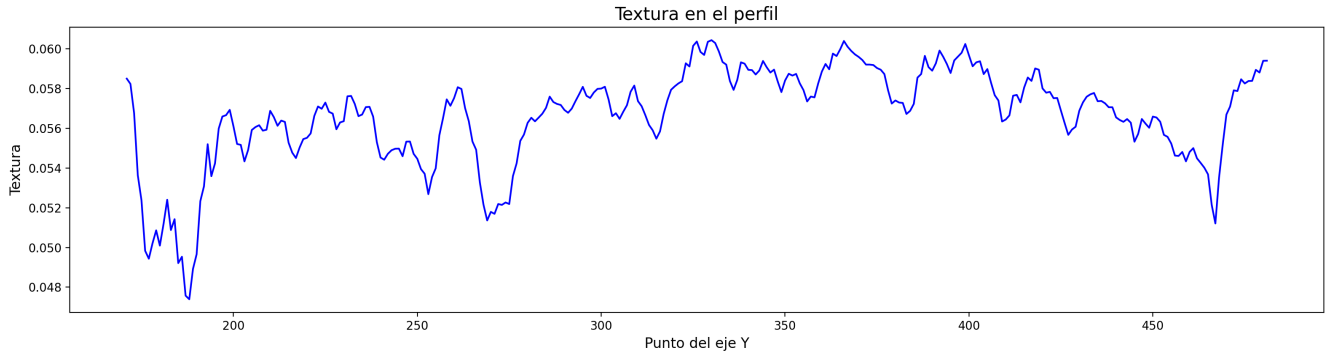


Figura 7.6: Gráfico de la combinación en relación a cada punto del perfil trazado.

7.4. GLCM

El método de matrices de niveles de grises de co-ocurrencia ($GLCM$)[22] se utiliza para obtener diferentes descriptores de textura según el enfoque que se esté buscando. Cuando se habla del enfoque, significa la función que el método utiliza para calcular la matriz resultante.

El método consta de dos partes principales. En la primera parte, se calculan las matrices de co-ocurrencia que luego se combinarán en la segunda parte formando una única matriz para estudiar la textura según la función utilizada para combinar dichas matrices.

A continuación se detallan las funciones elegidas que provee el método para calcular la combinación de las matrices de co-ocurrencia:

1. Entropía: Es una medida que permite conocer la complejidad de la imagen. Como se explico anteriormente, muestra las zonas que son más homogéneas y aquellas que son más heterogéneas.

2. Disimilitud: Es una medida que permite conocer la variación lineal local del valor de grises.

3. Homogeneidad: Es una medida que permite conocer la uniformidad de tonos en la imagen.

4. Media: Es una medida que permite conocer el valor medio del píxel según el entorno.

Sin embargo, el método tiene otras funciones para el cálculo de la matriz resultante como por ejemplo máximos, segundo momento angular, contraste, etc. Estas funciones no fueron elegidas dado que no dan como resultado un descriptor que provea de información relevante.

Para el cálculo de las matrices de co-ocurrencia se realizan los siguientes pasos:

1. Los valores de los píxeles de la imagen que queremos analizar se normalizan entre 0 - 9. Se dice que se obtiene la imagen digitalizada.

2. Se crea una matriz de 10 x 10 donde cada celda representa a un par de co-ocurrencia. Los pares de co-ocurrencia son de la forma (X, Y) donde X,Y $\in 0 - 9$. Cada celda de la matriz (es decir, cada par de co-ocurrencia) apunta a una matriz del mismo tamaño que la imagen a analizar inicializando sus valores con ceros.

3. Por cada par de co-ocurrencia se calculan los valores de la matriz a la que apunta. Para ello, sobre la imagen digitalizada se pasa una ventana de 3 x 3 donde se buscan los pares de co-ocurrencia, de forma horizontal, según corresponda. En el siguiente ejemplo, se muestran dos iteraciones de cómo se calculan los valores de la matriz de co-ocurrencia del par (0,0):

Matriz digitalizada

0	0	0	0	0	0	0
0	0	0	1	2	1	0
0	0	0	0	0	0	0
0	0	1	5	0	1	0
0	5	1	2	0	0	0
0	0	0	2	1	1	0
0	0	0	0	0	0	0

Matriz de co-ocurrencia de pares (0,0)

6	5	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0

Matriz digitalizada

0	0	0	0	0	0	0
0	0	0	1	2	1	0
0	0	0	0	0	0	0
0	0	1	5	0	1	0
0	5	1	2	0	0	0
0	0	0	2	1	1	0
0	0	0	0	0	0	0

Matriz de co-ocurrencia de pares (0,0)

6	5	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0

Figura 7.7: Iteraciones del método *GLCM* para obtener los valores de la matriz de co-ocurrencia del par (0,0).

4. Una vez calculadas las matrices, se combinan en una única matriz (del mismo tamaño que la imagen digitalizada) según la función utilizada (entropía, media, homogeneidad, etc.). Como resultado, se obtiene una matriz que funciona como descriptor de textura.

5. Se debe normalizar el resultado entre 0 - 255 para poder graficar la matriz como una imagen.

El método es muy efectivo cuando se busca tener muchos descriptores de una imagen según varios enfoques. Esto cobra mayor importancia cuando se quiere analizar la relación de los descriptores para poder clasificar el nivel de textura en toda la imagen.

Otro aspecto importante del método es que no necesita ningún parámetro por parte del usuario, lo cual es útil para poder automatizar rutinas de clasificación de la textura.

A continuación vemos un ejemplo de los descriptores de textura resultantes según los diferentes enfoques que ofrece el método. La Figura 7.8 muestra los resultados de cada enfoque mediante mapas de calor.

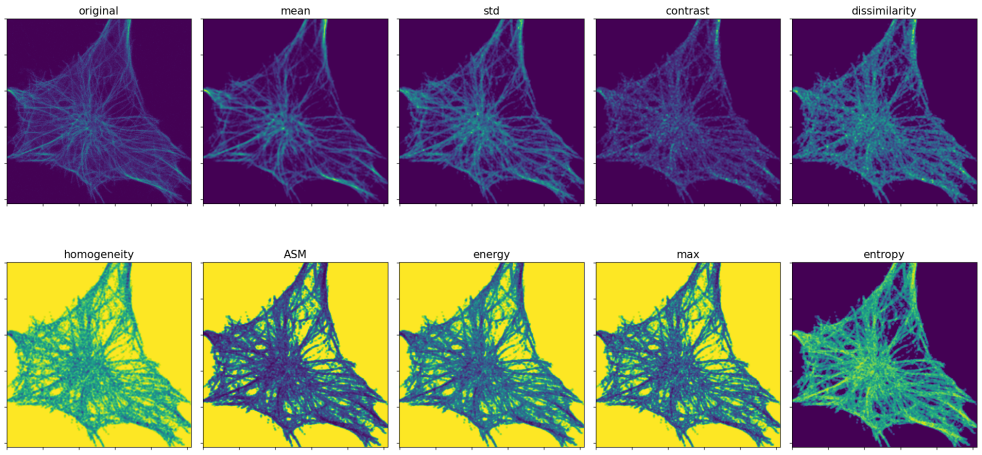


Figura 7.8: Resultados de descriptores utilizando los enfoques de media, desviación estándar, contraste, disimilitud, homogeneidad, segundo momento angular (ASM), energía, máximos y entropía respectivamente.

7.5. Métodos de clasificación de textura no supervisados

En esta sección, se presentan diferentes métodos de clasificación no supervisada, analizando su funcionamiento, sus ventajas y desventajas y se detalla su efectividad en cuanto a la clasificación de la textura para las células.

A la vez, se muestran los resultados de probar dichos métodos con diferentes combinaciones de descriptores de textura, para poder visualizar cuales descriptores funcionan mejor para la clasificación de la textura.

Los descriptores a tener en cuenta son los siguientes:

- Entropía ($GLCM$)
- Homogeneidad ($GLCM$)
- Disimilitud ($GLCM$)
- Media ($GLCM$)
- Entropía Local

El resultado que se obtienen son mapas de calor donde la barra que se muestra en la Figura 7.9 representa la distribución de los clústers en los que se clasifica cada píxel.

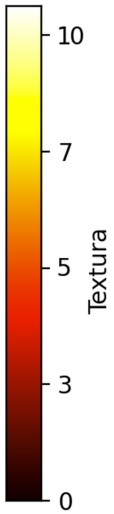


Figura 7.9: Barra de ejemplo utilizada en los mapas de calor mostrados en este capítulo.

Los valores indican el clúster al cual se clasifica a cada píxel según su textura. Los mismos incrementan hacia colores más claros, por lo tanto habrá mayor textura en las zonas donde el mapa de calor sea mas claro.

7.5.1. K-Medias

El algoritmo de *K-Medias*[23] es uno de los métodos de clasificación no supervisados más simples. Sin embargo, debido a su simpleza es rápido a la hora de mostrar los resultados de la clasificación.

El único parámetro que utiliza como entrada es la cantidad de clústers que se quiere tener como resultado en la clasificación. Si bien que un método requiera parámetros es un aspecto negativo para la automatización, en este caso poder elegir la cantidad de clústers en los que clasificar la textura resulta un aspecto positivo.

El algoritmo utiliza la idea de lo que llama elemento representativo para determinar qué observaciones (píxeles, en este caso) corresponden a cada clúster. A este elemento representativo se lo llama centroide. El algoritmo procede de la siguiente forma:

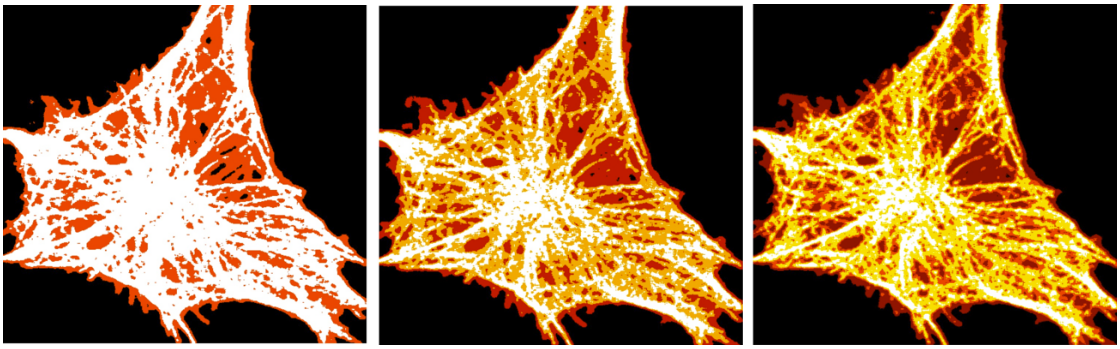
1. Se coloca cada observación en un clúster de manera aleatoria.

2. Se calcula el centroide de cada clúster como la media de todos los elementos del clúster.

3. Se calcula la distancia euclídea entre cada observación y los centroides. Se coloca cada observación en aquel clúster en el que la distancia euclídea entre el centroide y la observación es mínima

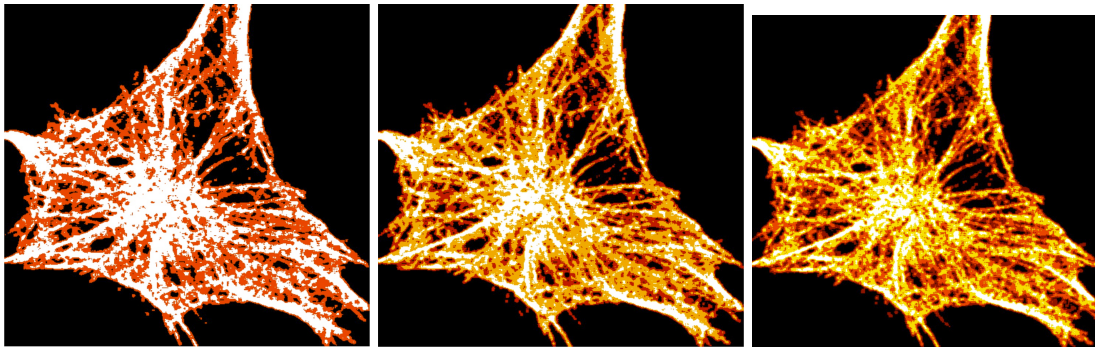
4. Se repiten los pasos 2 y 3 hasta que de una iteración a otra no se produce ningún cambio.

A continuación veremos algunos resultados de la clasificación utilizando K medias:



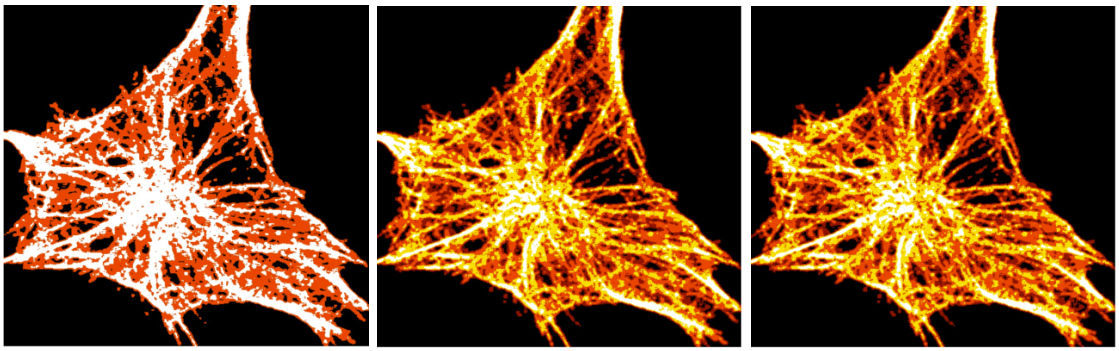
(a) (b) (c)

Figura 7.10: Resultados de la clasificación de la textura usando los descriptores media, disimilitud, homogeneidad, entropía ($GLCM$) y entropía local con (a) 3 clústers. (b) 4 clústers. (c) 5 clústers.



(a) (b) (c)

Figura 7.11: Resultados de la clasificación de la textura usando los descriptores media, disimilitud, homogeneidad y entropía ($GLCM$) con (a) 3 clústers. (b) 4 clústers. (c) 5 clústers.



(a) (b) (c)

Figura 7.12: Resultados de la clasificación de la textura usando los descriptores media, disimilitud y entropía ($GLCM$) con (a) 3 clústers. (b) 4 clústers. (c) 5 clústers.

Como se detalla en la sección, el algoritmo tiene dos ventajas. Por un lado es rápido en producir los resultados y por otro lado permite especificar la cantidad de clústers del resultado.

Además, la clasificación da buenos resultados debido a que, empíricamente, parece agrupar los píxeles correctamente en los clústers según la textura encontrada en los mismos.

Es por ello, que *K-Medias* resulta ser un buen algoritmo de clasificación de la textura en estas células.

Es importante destacar que dentro de los descriptores de textura, la entropía local parece generar que la clasificación sea menos precisa. Sin embargo, el descriptor de la homogeneidad no parece generar cambios en la clasificación, ya sea que esté presente como si no lo está.

7.5.2. Redes de Kohonen

El método de redes de *Kohonen*[24], o mapas auto-organizados se basa en proponer una red neuronal artificial donde no existe ningún maestro o supervisor externo que indique cual es la salida deseada para cada una de las entradas de entrenamiento. Por ello, se dice que es un método de clasificación no supervisado.

El algoritmo procede de la siguiente manera:

1. Se normalizan los atributos de las observaciones entre 0 y 1.

2. Se crea una matriz de K x K neuronas.

3. Se inicializan los pesos de la matriz de manera aleatoria entre 0 y 1.

4. Se muestra una observación aleatoria del conjunto a la red. Se calcula la distancia euclídea entre la observación y cada neurona. Aquella neurona con menor distancia euclídea es la ganadora, por lo cual es la que actualiza sus pesos según los atributos de la observación. Además, según el radio de vecindad, se actualizan los pesos en menor medida de las neuronas vecinas de la neurona ganadora.

5. Se repite el paso 4 hasta un cierto número de iteraciones.

Una de las grandes desventajas del algoritmo, es que necesita de varios parámetros de entrada:

- Learning Rate: Se utiliza para calcular la actualización de los pesos.

- Radio de vecindad: Se utiliza para determinar qué neuronas vecinas a la neurona ganadora deben actualizar sus pesos.
- Número de iteraciones.
- Estructura de la red.

Esto es un problema al intentar automatizar el método, dado que para algunas células ciertos parámetros funcionarán mejor que otros.

La figura 7.13 algunos resultados de la clasificación utilizando el algoritmo de Redes de *Kohonen*:

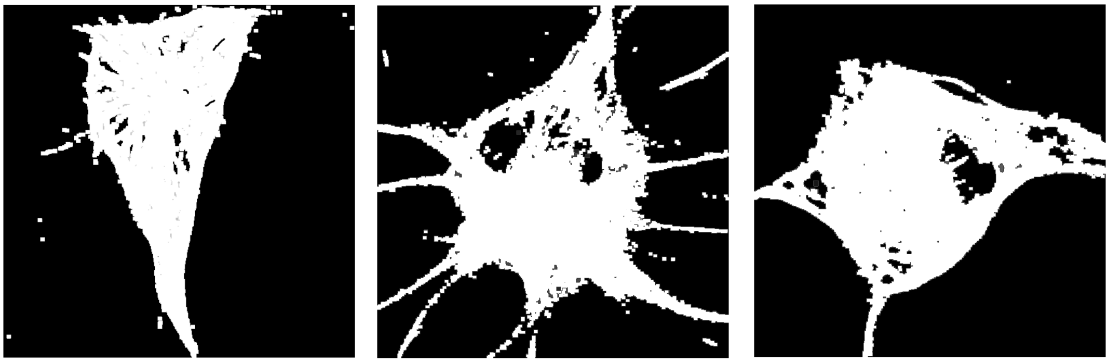


Figura 7.13: Resultados de la clasificación de la textura usando los descriptores media, disimilitud, entropía ($GLCM$) y homogeneidad.

Como puede verse en los ejemplos, la agrupación generada por este método es binaria. Separa únicamente la célula del fondo.

Dentro de los resultados, se utilizan diferentes variaciones de los parámetros:

- Learning Rate: 0.05, 0.1, 0.25 y 0.5.
- Número de iteraciones: 5, 10 y 25
- Estructura de la red: 3x3, 5x5, 9x9 y 15x15

Sin embargo, en todos los casos se obtienen agrupaciones binarias en la clasificación, por lo que puede concluirse que el método de redes de *Kohonen* no es un buen clasificador de textura para estas imágenes.

Por lo dicho en la sección, se opta por descartar el método debido a los múltiples parámetros que necesita y los malos resultados obtenidos.

7.5.3. Mean Shift

El algoritmo de *Mean Shift*[25], al igual que K-Medias, utiliza la idea de elementos representativos para determinar que observaciones corresponden a cada clúster. Sin embargo, a diferencia de K-Medias, utiliza el enfoque de agrupación jerárquica al momento de unir las observaciones en clústers.

Mean Shift procede de la siguiente forma:

1. Inicialmente, cada observación conforma un clúster en sí misma. En este paso, tenemos N observaciones, por lo tanto, N clústers. Cada clúster tiene su centroide, que en este paso corresponde a la única observación que tiene.

2. Se calcula la distancia euclídea entre el centroide de cada clúster y las observaciones. Toda observación que tenga una distancia euclídea menor a un umbral, quedará agrupada dentro del mismo clúster.

3. Se calcula el centroide de cada clúster como la media de todos los elementos del clúster.

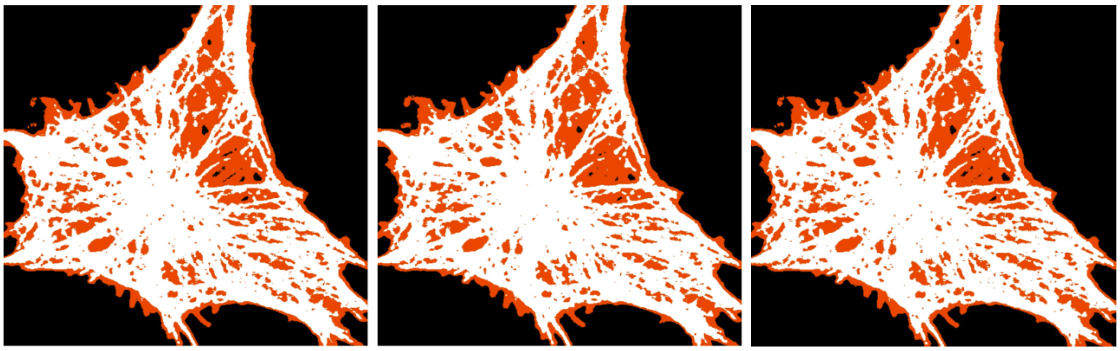
4. Se repiten los pasos 2 y 3 hasta que de una iteración a otra no se produce ningún cambio.

Este algoritmo posee dos grandes desventajas. Por un lado, el enfoque de utilizar agrupación jerárquica genera que el cómputo sea elevado, por lo que a mayor cantidad de observaciones, mayor será el tiempo que le tome calcular los resultados de la agrupación. Dado esta problemática, para clasificar las imágenes es necesario partir el conjunto en testeo y entrenamiento. Con el conjunto de píxeles de entrenamiento se busca entrenar el método para reconocer los diferentes niveles de textura para luego etiquetar los píxeles del conjunto de testeo.

Además, el algoritmo para realizar las agrupaciones requiere de un parámetro, llamado bandwidth, que es el umbral utilizado en el paso 2 para determinar el clúster de cada observación.

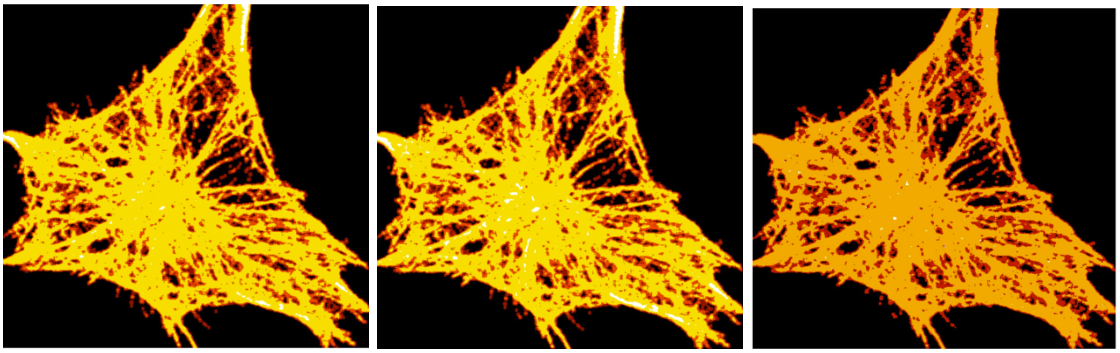
Es importante ver que no se puede elegir la cantidad de clústers resultantes de la clasificación, si no que depende de las observaciones y el parámetro de entrada mencionado anteriormente.

A continuación se muestran algunos resultados de la clasificación utilizando *Mean Shift*:



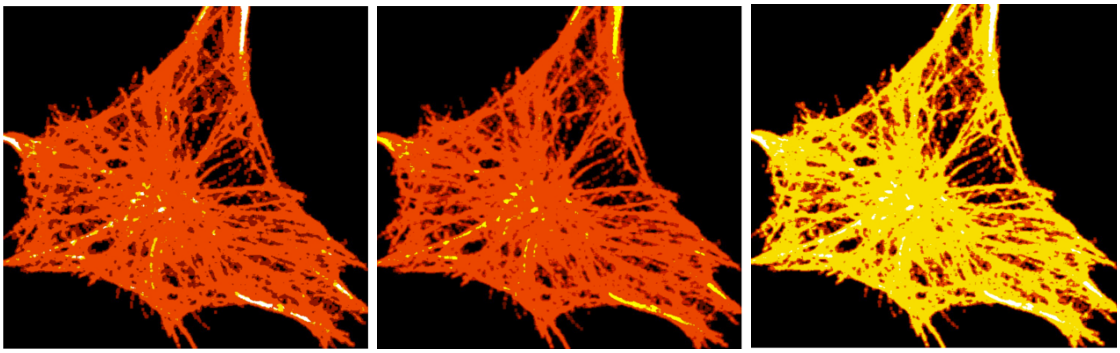
(a) (b) (c)

Figura 7.14: Resultados de la clasificación de la textura usando los descriptores media, disimilitud, homogeneidad, entropía ($GLCM$) y entropía local, con un porcentaje de entrenamiento de (a) 7.5 %. (b) 10 %. (c) 15 %.



(a) (b) (c)

Figura 7.15: Resultados de la clasificación de la textura usando los descriptores media, disimilitud, homogeneidad y entropía ($GLCM$), con un porcentaje de entrenamiento de (a) 7.5 %. (b) 10 %. (c) 15 %.



(a) (b) (c)

Figura 7.16: Resultados de la clasificación de la textura usando los descriptores media, disimilitud y entropía (*GLCM*), con un porcentaje de entrenamiento de (a) 7.5 %. (b) 10 %. (c) 15 %.

Debido al elevado tiempo de cómputo que conlleva obtener los resultados y dado que no mejora lo obtenido en *K-Medias*, decidimos descartar el algoritmo.

7.5.4. HDBScan

El método de *HDBScan*[26] busca extender el algoritmo de *DBScan*[27] utilizando un enfoque orientado a agrupación jerárquica para luego extraer los clústers basados en su estabilidad.

El algoritmo de *HDBScan* cuenta de los siguientes pasos principales:

1. Transformar el espacio según la densidad/escasez: La idea en este paso es poder distinguir entre aquellas observaciones que son consideradas ruido, de las que no lo son. Para dicha distinción, el método utiliza una métrica llamada distancia mutua de alcance que se calcula de la siguiente forma:

   $$d_{mreach-k}(a,b) = max(core_k(a), core_k(b), d(a,b))$$

   $d(a,b)$ es la distancia euclídea entre dos observaciones a y b.

   $core_k(a)$ es la distancia euclídea que existe entre a y el k-ésimo vecino más lejano de a

   Se piensa cada observación como un vértice y que la métrica es la arista que los une.

   Con esta métrica lo que se busca es alejar los puntos considerados ruido manteniendo intactos aquellos que no lo son.

2. Construir el árbol de expansión mínimo del gráfico según la distancia euclídea: Con el grafo ya formado del punto anterior lo que se busca es formar el árbol de expansión mínimo usando el algoritmo de Prim.

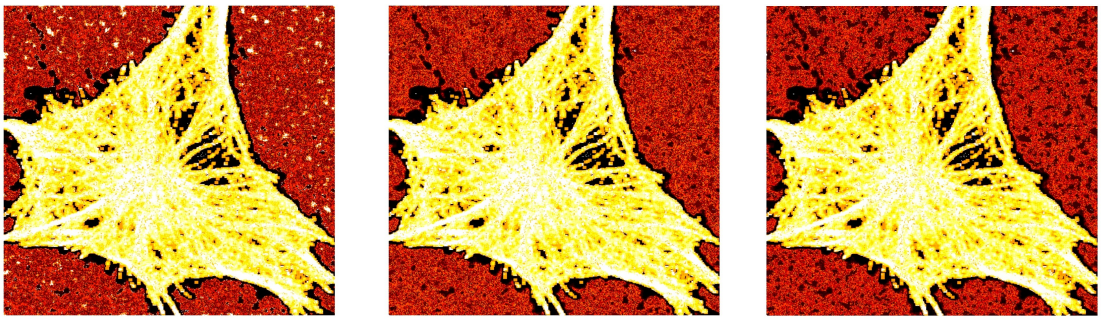
3. Construir la jerarquía de clústers: Una vez construido el árbol, se busca transformarlo en un dendrograma ordenando las aristas del árbol por distancia (en orden creciente) y luego iterar, creando un nuevo clúster combinado para cada arista

4. En el final, lo que buscamos es mergear los clústers hasta que cada uno tenga un mínimo número de observaciones.

HDBScan tiene la ventaja de que, si bien no podemos elegir la cantidad de clústers en el resultado, nos permite elegir la cantidad mínima de observaciones que debe tener un clúster para ser considerado como tal.

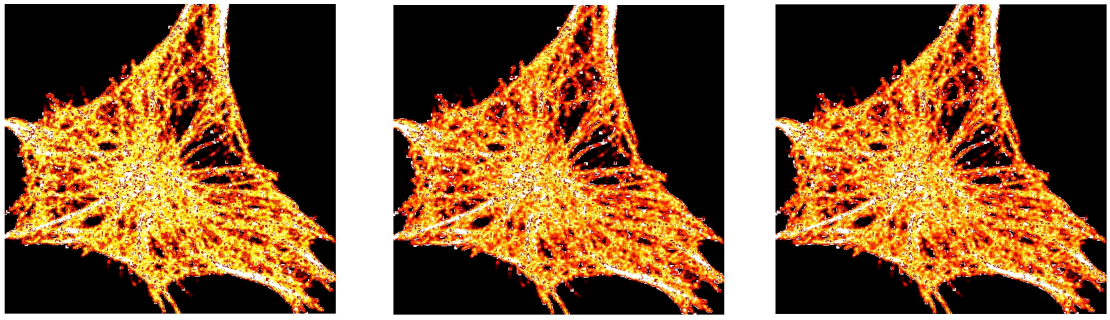
Por otro lado, si bien utiliza el enfoque de agrupación jerárquica que tiene un costo computacional alto, se obtienen resultados de manera veloz, a comparación con *Mean Shift*.

A continuación veremos algunos resultados de la clasificación utilizando *HDBScan*:



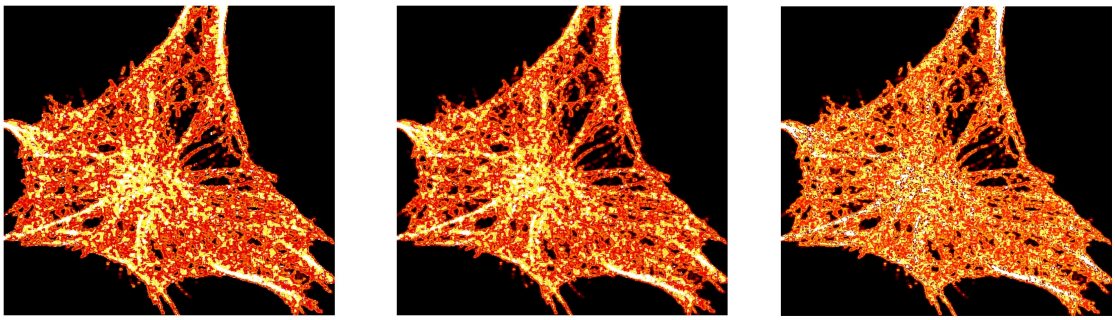
(a) (b) (c)

Figura 7.17: Resultados de la clasificación de la textura usando los descriptores media, disimilitud, homogeneidad, entropía ($GLCM$) y entropía local, con una cantidad de observaciones por clúster igual a (a) 1250. (b) 1750. (c) 2250.



(a) (b) (c)

Figura 7.18: Resultados de la clasificación de la textura usando los descriptores media, disimilitud, homogeneidad y entropía ($GLCM$), con una cantidad de observaciones por clúster igual a (a) 1250. (b) 1750. (c) 2250.



(a) (b) (c)

Figura 7.19: Resultados de la clasificación de la textura usando los descriptores media, disimilitud y entropía ($GLCM$), con una cantidad de observaciones por clúster igual a (a) 1250. (b) 1750. (c) 2250.

Como puede verse, *HDBScan* produce buenos resultados a la hora de clasificar la textura. Aún teniendo un costo computacional más alto que *K-medias*, es un candidato prometedor para la clasificación de la textura.

7.6. Resultados

Para finalizar el capítulo, se expresan algunas conclusiones respecto de los métodos utilizados.

Como se puede ver en la sección 7.5.1, los resultados de la textura con el método de *K-Medias* dan una distribución acorde a lo que uno puede ver a la vista. Otra característica importante del

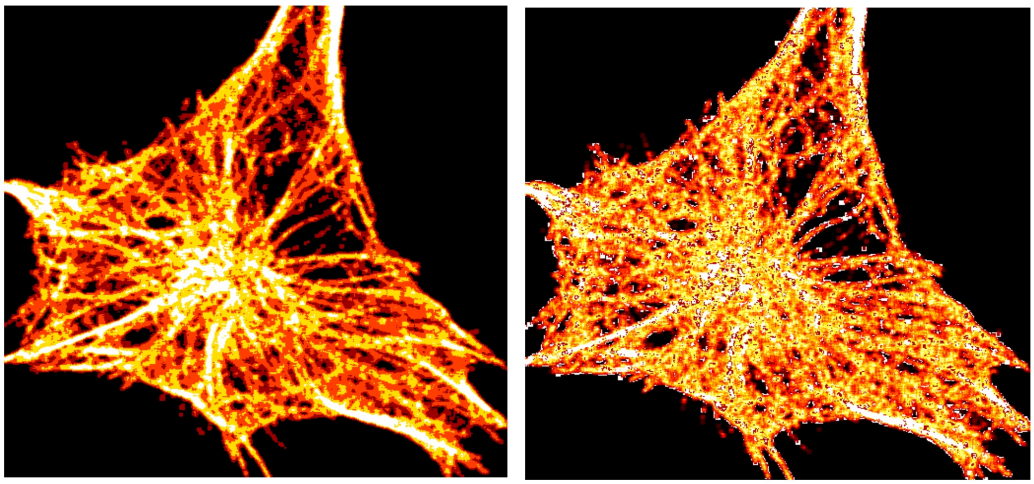
método es que es muy veloz a la hora de traer resultados. Aproximadamente tarda entre 6 - 10 segundos[1].

En la sección 7.5.2 se puede ver la utilización de un algoritmo de redes neuronales. En el caso de las células, los resultados no son favorables ya que las redes neuronales solo pueden identificar la célula por un lado y el fondo por otro. Es decir, da una distribución compuesta por dos clústers, fondo y célula. Si bien el algoritmo tiene una velocidad equiparable a la de *K-Medias*, dado los resultados obtenidos, queda descartado.

En la sección 7.5.3 se presenta el uso de un algoritmo basado en clusterización jerárquica, llamado *Mean Shift*. Este método, dado el enfoque que utiliza, tiene un costo computacional muy alto. Esto no solo afecta a la velocidad de obtención de resultados, si no que también limita la precisión obtenida en la clusterización. En general, tarda entre 40 - 50 minutos en generar resultados. Por lo dicho anteriormente queda descartado.

Finalmente en la sección 7.5.4 se presenta el método *HDBScan* que si bien está basado en clusterización jerárquica, utiliza otros enfoques que agilizan el algoritmo. Cuantitativamente, tarda entre 6 - 10 minutos en mostrar los resultados.

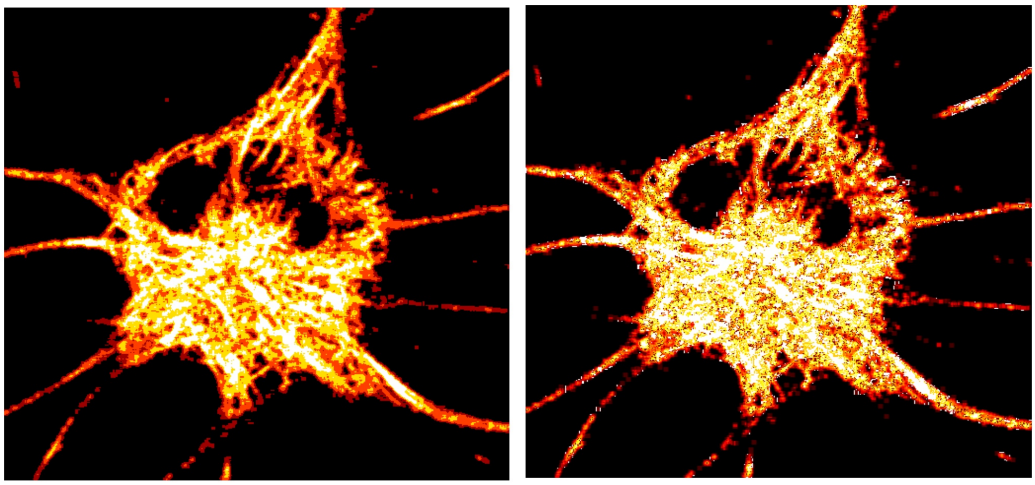
A continuación se muestra algunos resultados para comparar el algoritmo de *K-Medias* frente a *HDBScan*:



(a) (b)

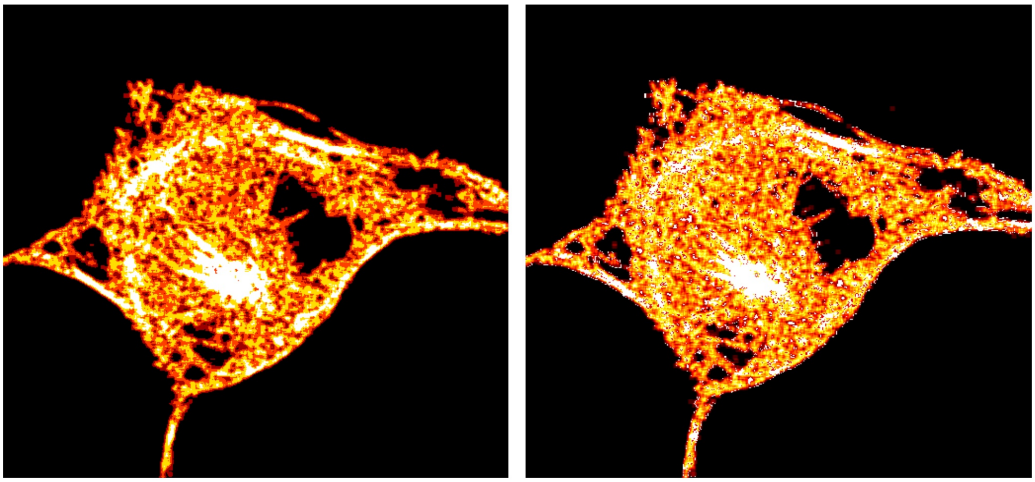
Figura 7.20: Resultados de la clasificación de la textura usando los descriptores media, disimilitud, homogeneidad y entropía (*GLCM*), con (a) *K-Medias* y (b) *HDBScan*.

[1] Los tiempos detallados en esta sección fueron tomados utilizando un procesador Intel i7-9750H de 2.6GHz



(a) (b)

Figura 7.21: Resultados de la clasificación de la textura usando los descriptores media, disimilitud, homogeneidad y entropía (*GLCM*), con (a) *K-Medias* y (b) *HDBScan*.



(a) (b)

Figura 7.22: Resultados de la clasificación de la textura usando los descriptores media, disimilitud, homogeneidad y entropía (*GLCM*), con (a) *K-Medias* y (b) *HDBScan*.

En las pruebas mostradas, se puede ver que los resultados son muy parecidos. Una diferencia notable, es que los resultados a partir de usar el método *HDBScan* dan imágenes que parecen tener ruido, mientras que en *K-Medias* dan resultados fluidos.

Es importante recordar la velocidad de obtención de los resultados. Mientras que *HDBScan* produce sus resultados en aproximadamente 6 - 10 minutos, *K-medias* los obtiene en tan sólo 6 - 10 segundos. Esta diferencia fue clave a la hora de elegir el método a descartar.

Por todo lo dicho anteriormente, de los cuatro métodos probados, se decide dejar a *K-Medias* como el algoritmo que mejor presenta los resultados de textura.

Capítulo 8

Conclusiones y trabajo futuro

En este último capítulo se exponen las conclusiones del trabajo realizado en base a los resultados mostrados en capítulos anteriores. Se recopilan los métodos implementados para cada estudio realizado y se detalla cuál fue el que dio mejores resultados.

8.1. Conclusiones

En primer lugar, en relación al estudio del contorno externo del citoesqueleto, el método de *MGAC* simplifica la obtención del contorno debido a que es más automático que *el método de intercambio de píxeles*. Este último presenta problemas que el *MGAC* no, como su dependencia con la forma del citoesqueleto. Si la estructura del mismo toca los bordes de la imagen, es necesario tomar más de una región inicial, lo cual incrementa el tiempo de procesamiento del método. En este sentido, el método del *MGAC* es independiente del input del usuario. Este nos permite obtener la máscara del citoesqueleto que luego es utilizada para obtener el área, el perímetro y la razón de los ejes en la película.

En segundo lugar, en cuanto al estudio del movimiento, si bien todos los métodos dieron resultados exitosos el método *MGAC* y el *método de intercambio de píxeles* no evalúan el movimiento en toda la célula sino que únicamente en los contornos. El *método de umbralización* da resultados más exactos que *el método por diferencia de píxel*, ya que en este último se obtienen mapas de calor con grandes zonas homogéneas.

Por último, el estudio de textura fue más complejo que el resto. Se prueba la aplicación de una cantidad de métodos mayor y también se combina el análisis clásico de imágenes como lo es el uso

de *GLCM* o la entropía con métodos de aprendizaje automático como lo son *K-Medias*, *Mean Shift*, *Redes de Kohonen* y *HDBScan*. Esto permite obtener más de una herramienta de estudio de esta propiedad, como los mapas de calor o los perfiles de intensidad.

Para las 3 propiedades que se propusieron analizar al inicio del trabajo, fue posible obtener un resultado final óptimo para cada una de ellas, más allá de las dificultades que este tipo de imágenes microscópicas agregan. Utilizando métodos de limpieza de ruido más sofisticados de los que el alcance de este proyecto propuso, los resultados finales obtenidos probablemente sean aún mejores.

8.2. Trabajo Futuro

Existen dos tipos de células que se pueden estudiar con las rutinas. Por un lado las células adherentes estudiadas en el presente trabajo, cuya particularidad es estar adheridas por puntos de anclaje y tener poca movilidad. Por otro lado las células no adherentes (migratorias), es decir que gozan de movimiento libre.

Un posible trabajo futuro es probar el funcionamiento de los métodos desarrollados para células no adherentes y en el caso que los resultados obtenidos no sean óptimos, extender las rutinas para que funcionen con este tipo de células.

Bibliografía

[1] Adrian F Pegoraro, Paul Janmey, and David A Weitz. Mechanical properties of the cytoskeleton and cells. *Cold Spring Harbor Perspectives in Biology*, 9(11):a022038, 2017.

[2] Carla Pallavicini. Mecánica y dinámica del citoesqueleto en células vivas. `http://hdl.handle.net/20.500.12110/tesis_n6316_Pallavicini`.

[3] Frederick Gittes, Brian Mickey, Jilda Nettleton, and Jonathon Howard. Flexural rigidity of microtubules and actin filaments measured from thermal fluctuations in shape. *The Journal of cell biology*, 120(4):923–934, 1993.

[4] J Käs, H Strey, JX Tang, D Finger, R Ezzell, E Sackmann, and PA Janmey. F-actin, a model polymer for semiflexible chains in dilute, semidilute, and liquid crystalline solutions. *Biophysical journal*, 70(2):609–625, 1996.

[5] M. E. Janson and M. Dogterom. A bending mode analysis for growing microtubules: evidence for a velocity-dependent rigidity. *Biophysical journal*, 87:2723–2736, 2004.

[6] Brangwynne C. P. and col. Bending dynamics of fluctuating biopolymers probed by automated high-resolution filament tracking. *Biophysical journal*, 93:346–359, 2007.

[7] Felix Ruhnow, David Zwicker, and Stefan Diez. Tracking single particles and elongated filaments with nanometer precision. *Biophysical journal*, 100:2820–2828, 2011.

[8] Charlotte Pain, Verena Kriechbaumer, Maike Kittelmann, Chris Hawes, and Mark Fricker. Quantitative analysis of plant er architecture and dynamics. *Nature communications*, 10(1):1–15, 2019.

[9] OpenCV. Histograms - 2: Histogram equalization. `https://docs.opencv.org/master/d5/daf/tutorial_py_histogram_equalization.html`.

[10] OpenCV. Gaussian blurring. `https://docs.opencv.org/master/d4/d86/group__imgproc__filter.html#gaabe8c836e97159a9193fb0b11ac52cf1`.

[11] OpenCV. Bilateral filtering. https://docs.opencv.org/master/d4/d86/group__imgproc__filter.html#ga9d7064d478c95d60003cf839430737ed.

[12] Perona and Malik. Scale-space and edge detection using anisotropic diffusion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 629 – 639, 1990.

[13] Medpy. Medpy filter. https://loli.github.io/medpy/generated/medpy.filter.smoothing.anisotropic_diffusion.html.

[14] Scipy. Gaussian gradient magnitude. https://docs.scipy.org/doc/scipy/reference/generated/scipy.ndimage.gaussian_gradient_magnitude.html.

[15] Nobuyuki Otsu. A threshold selection method from gray-level histograms. *IEEE Transactions on Systems, Man, and Cybernetics*, 9(1):62–66, 1979.

[16] OpenCV. Image thresholding. https://docs.opencv.org/master/d7/d4d/tutorial_py_thresholding.html.

[17] John Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-8(6):679–698, 1986.

[18] Y. Shi and W.C. Karl. Real-time tracking using level sets. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 2, pages 34–41 vol. 2, 2005.

[19] Luis Alvarez, Luis Baumela, Pablo Márquez-Neila, and Pedro Henríquez. A Real Time Morphological Snakes Algorithm. *Image Processing On Line*, 2:1–7, 2012. https://doi.org/10.5201/ipol.2012.abmh-rtmsa.

[20] OpenCV. Adaptive thresholding. https://docs.opencv.org/master/d7/d4d/tutorial_py_thresholding.htmll.

[21] OpenCV. Local entropy. https://scikit-image.org/docs/dev/api/skimage.filters.rank.html#skimage.filters.rank.entropy.

[22] Shruti Singh, Divya Srivastava, and Agarwal S. Glcm and its application in pattern recognition. pages 20–25, 08 2017.

[23] Xin Jin and Jiawei Han. *K-Means Clustering*, pages 563–564. Springer US, Boston, MA, 2010.

[24] O. J. Vrieze. *Kohonen network*, pages 83–100. Springer Berlin Heidelberg, Berlin, Heidelberg, 1995.

[25] Yizong Cheng. Mean shift, mode seeking, and clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(8):790–799, 1995.

[26] Leland McInnes, John Healy, and Steve Astels. hdbscan: Hierarchical density based clustering. *The Journal of Open Source Software*, 2(11), mar 2017.

[27] Michael Hahsler, Matthew Piekenbrock, and Derek Doran. dbscan: Fast density-based clustering with R. *Journal of Statistical Software*, 91(1):1–30, 2019.