

Fundamento de Procesamiento Digital de Imágenes

Proyecto Final: “Frotis de Médula Ósea”

Karla Paola Martínez Velázquez

Eduardo Ramírez de la Fuente

David Pedroza Segoviano

Nota: El código se realizó en Python. Y está hecho de tal manera que se puede leer al mismo tiempo que el reporte, debido a que está hecho en un notebook.

INTRODUCCIÓN

En el campo del procesamiento digital de imágenes, la capacidad de manipular y analizar imágenes ha adquirido una relevancia creciente en diversos ámbitos. Este proyecto tiene como objetivo abordar un desafío específico dentro de este campo, centrándose en procesar un “Frotis de Médula Ósea”.

El objetivo principal de este proyecto es contabilizar el número de células que aparecen en una imagen de microscopía, identificando cuantas. A través de técnicas y algoritmos de procesamiento de imágenes, nos proponemos mejorar la calidad visual, detectar y reconocer objetos, o cualquier otro enfoque específico que sea relevante para el problema que se está abordando.

METODOLOGÍA

Librería CV2

La librería **CV2** de Python, también conocida como OpenCV (Open Source Computer Vision Library), es una biblioteca de código abierto ampliamente utilizada en el procesamiento de imágenes y visión por computadora. Proporciona una amplia gama de funciones y algoritmos para el análisis, manipulación y mejora de imágenes, así como para el procesamiento de video.

Histograma de una imagen

La función cv2.calcHist es una función de la librería OpenCV (cv2) en Python que se utiliza para calcular el histograma de una imagen. El histograma es una representación gráfica de la distribución de los valores de intensidad en una imagen, lo cual

proporciona información sobre el contraste, la luminosidad y la distribución de los tonos de la imagen.

La sintaxis básica de cv2.calcHist es la siguiente:

hist = cv2.calcHist(images, channels, mask, histSize, ranges)

Donde:

images: Es la imagen o lista de imágenes de entrada. Puede ser una imagen en escala de grises o una imagen en color.

channels: Especifica los canales de la imagen para los que se calculará el histograma. Puede ser una lista de índices de canales o un solo índice. Por ejemplo, para una imagen en color, [0] representaría el canal azul, [1] el canal verde y [2] el canal rojo.

mask: Opcionalmente, puedes proporcionar una máscara para limitar el cálculo del histograma a una región específica de la imagen.

histSize: Especifica el número de contenedores (bins) para el histograma. Puedes definir un solo valor o una lista de valores para cada canal.

ranges: Especifica el rango de valores de intensidad que se incluirán en el histograma.

Filtro Gaussiano

La función cv2.GaussianBlur es una función de la librería OpenCV (cv2) en Python que se utiliza para aplicar un **suavizado o desenfoque gaussiano** a una imagen. El desenfoque gaussiano es una técnica común en el procesamiento de imágenes para reducir el ruido y suavizar los detalles, creando una apariencia más suave en la imagen.

La sintaxis básica de cv2.GaussianBlur es la siguiente:

blurred = cv2.GaussianBlur(src, ksize, sigmaX[], dst[], sigmaY[], borderType[]])

Donde:

src: Es la imagen de entrada en la que se aplicará el desenfoque gaussiano. Puede ser una imagen en escala de grises o una imagen en color.

ksize: Especifica el tamaño de la ventana del kernel del filtro gaussiano. Debe ser un número impar y positivo. Un valor mayor para ksize producirá un mayor desenfoque en la imagen.

sigmaX: Especifica la desviación estándar en dirección horizontal. Controla la intensidad del desenfoque gaussiano.

dst (opcional): Es la imagen de salida donde se almacenará el resultado del desenfoque. Si no se proporciona, se crea una nueva imagen.

sigmaY (opcional): Especifica la desviación estándar en dirección vertical. Si no se proporciona, se asume que es igual a sigmaX.

borderType (opcional): Especifica el tipo de borde que se aplica a la imagen durante el proceso de convolución. Por defecto, se utiliza el borde replicado.

La función cv2.GaussianBlur devuelve la imagen suavizada resultante.

Método de OTSU

El algoritmo de Otsu, también conocido como el método de Otsu, es un algoritmo utilizado en el procesamiento de imágenes y visión por computadora para realizar una segmentación automática y óptima de una imagen en diferentes regiones o clases basándose en el histograma de niveles de gris.

El objetivo del algoritmo de Otsu es encontrar un umbral de binarización que maximice la varianza entre las clases de píxeles en la imagen. En otras palabras, busca un valor de umbral que minimice la varianza intraclasses y maximice la varianza interclasses.

Como se muestra en el siguiente código:

```
cv2.threshold(img_gray, 0, 255, cv2.THRESH_BINARY_INV + cv2.THRESH_OTSU)
```

Operación apertura

La operación de apertura consiste en dos pasos consecutivos: erosión seguida de dilatación. Se utiliza para eliminar pequeños detalles, ruido o discontinuidades en el interior de los objetos de la imagen, mientras se mantiene la forma general de los objetos.

Funciones:

```
cv2.morphologyEx y cv2.MORPH_OPEN
```

Operación cerradura

La operación de cerradura es similar a la operación de apertura, pero en orden inverso. Consiste en dos pasos consecutivos: dilatación seguida de erosión. Se utiliza para cerrar huecos en el interior de los objetos y suavizar los bordes de los objetos.

Función:

```
cv2.MORPH_CLOSE
```

Operación dilatación

La operación de dilatación es una operación morfológica utilizada en el procesamiento de imágenes para expandir o resaltar regiones de píxeles en una imagen binaria o en escala de grises. Esta operación se basa en el concepto de expansión de conjuntos y se realiza mediante el uso de un elemento estructurante (también conocido como kernel).

Función:

```
cv2.dilate
```

Propiedades de región

La función `regionprops` es la herramienta principal del toolbox para la computación de descripción de regiones:

En esta función, `L` es la matriz de etiquetas (función `bwlabel`), `D` es una estructura de longitud igual al número de etiquetas más grande encontrado en `L`. Los campos de la estructura se refieren a distintas propiedades o mediciones de cada región particular, los cuales son especificados en las propiedades del parámetro por medio de un string para cada propiedad deseada o en su defecto, el string “`all`” para obtener todas las propiedades existentes.

Segmentación

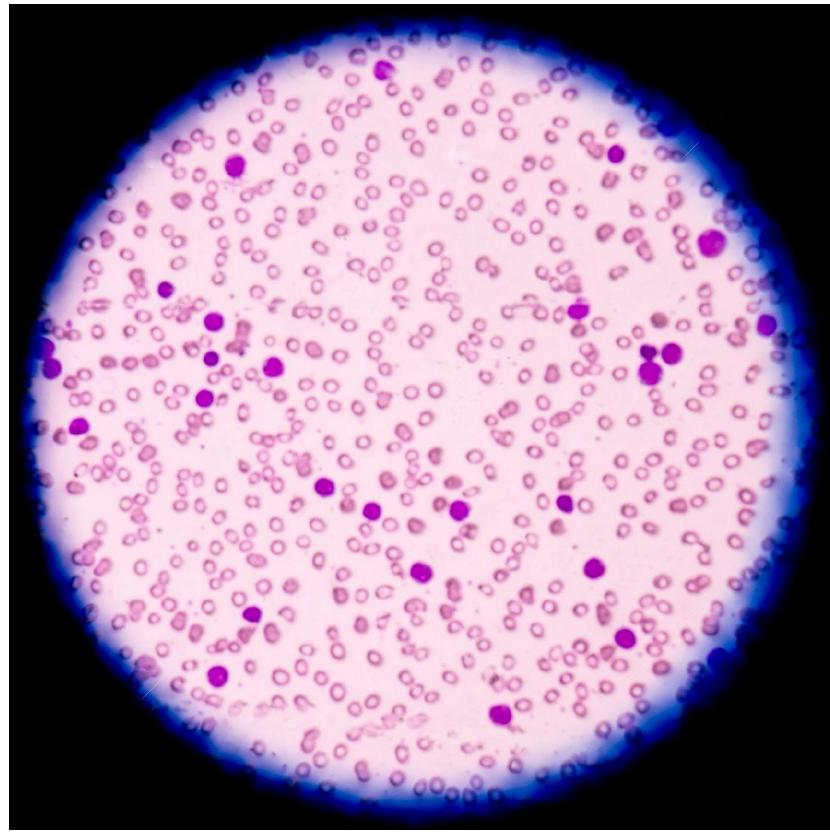
La segmentación de puntos, líneas y bordes es una de las técnicas con más importancia en el procesamiento de imágenes, cada uno de estos elementos puede considerarse como una discontinuidad de intensidad en imágenes digitales. La manera más común de encontrar dichas discontinuidades es con filtros que operan de la siguiente manera teniendo en consideración una máscara de 3x3:

$$R = \sum_{i=1}^9 w_i z_i [1].$$

Cada discontinuidad puede ser detectada según una máscara específica. En el caso de los filtros para la detección de líneas, es posible modificarlos de forma manual para que éstos detecten líneas horizontales, verticales o bien, diagonales, en específico.

RESULTADOS

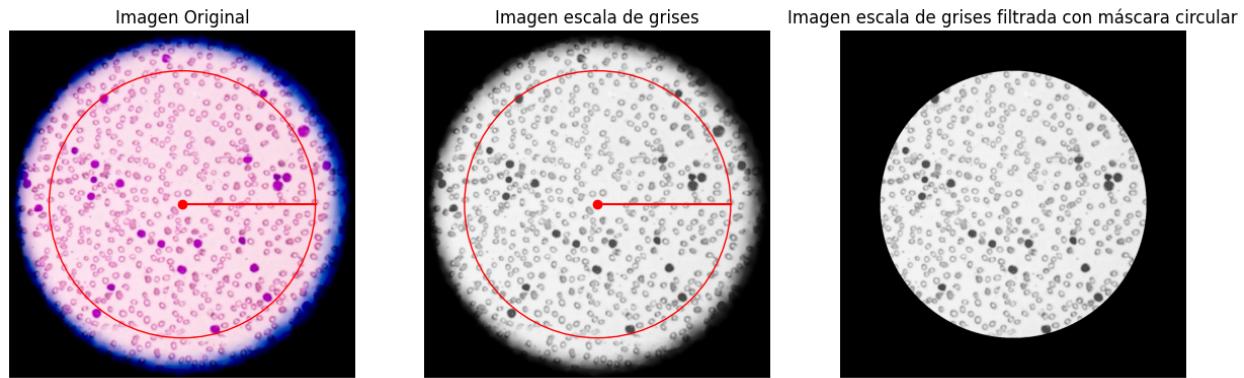
Antes de comenzar, se hace mención de que se utilizó un software para remover la marca de agua de la imagen original: <https://www.watermarkremover.io/>, la imagen que se utilizó para los procesos, que también se encuentra en la sección de resultados, es la siguiente:



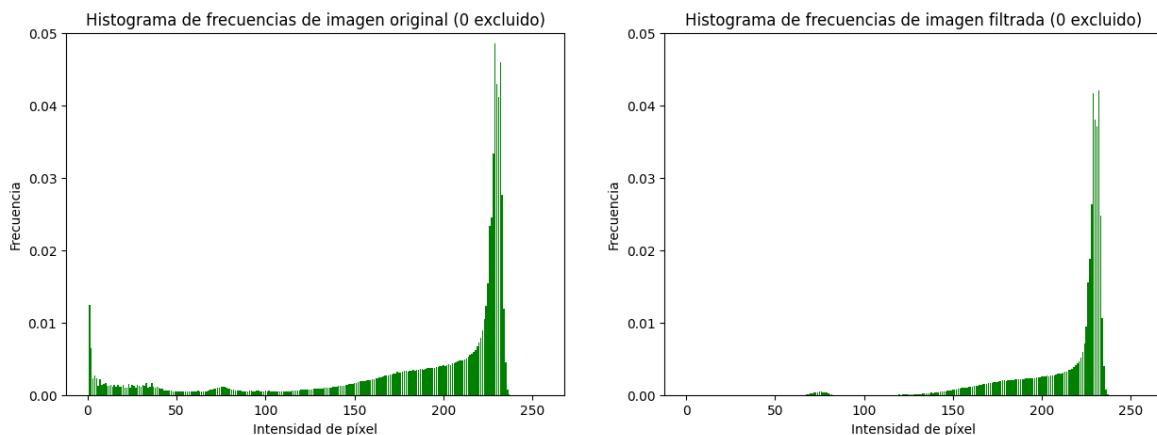
1. (1.0 puntos) Eliminar la región negra y con tinte azulado de la imagen, no se debe usar una herramienta donde manualmente ustedes seleccionen la región de interés. Sugerencia, observe que la región de interés tiene forma circular y está razonablemente bien centrada, el radio del círculo estímenlo como una proporción del tamaño de la imagen. Extraigan la sección circular con iluminación homogénea, a esta imagen la denominaremos FrotisCentral.png (grábela en ese formato que es uno de los archivos a entregar).

Lo primero que se hizo para obtener información que resultara útil, fue estimar la región de la imagen para la cual se tomara el máximo de células sin que se viera el aro azul o que interrumpiera lo mínimo posible a los procesos que se iban a utilizar después, para lo cual se implementó la siguiente práctica.

Se realizó una función en donde se veían los siguientes resultados:



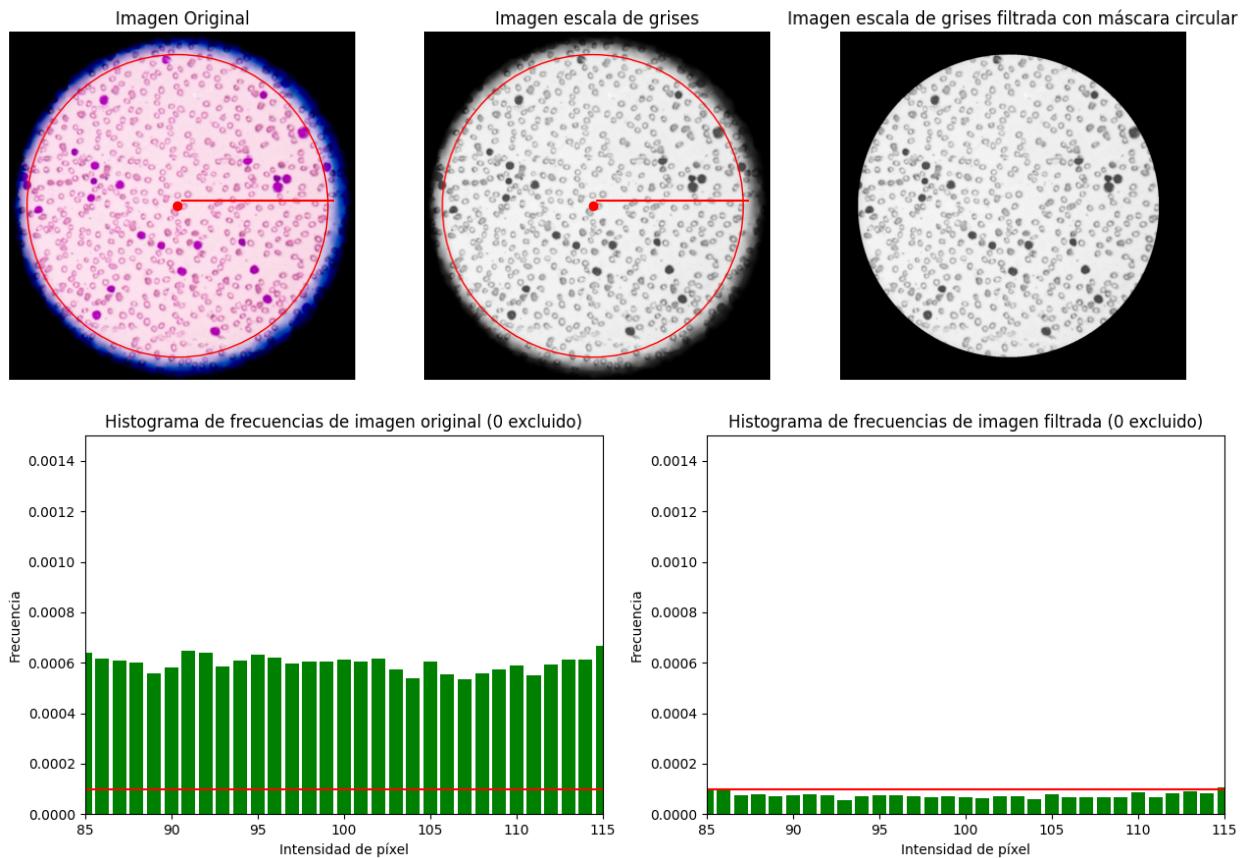
La primera imagen es la imagen original, la segunda es la imagen a escala de grises (esto para calcular su histograma de frecuencias y detectar con qué porción de la imagen nos deberíamos de quedar para quedarnos con la mayoría de la imagen con iluminación homogénea) que se obtuvo a partir de la librería **CV2** de Python, note que estas dos imágenes tienen un círculo, este círculo indica la región sobre la cual vamos a “recortar” la imagen de tal manera que cumpla con las especificaciones que indicamos anteriormente, el radio y centro de este círculo son parámetros que le pasamos a la función, así se puede variar, la tercera imagen es la imagen recortada sobre este círculo, este recorte se hizo creando una máscara donde hay 0's en toda la imagen excepto dentro del círculo creado, esta función también nos devuelve las siguiente información:



Estos son los histogramas de frecuencias normalizados de la imagen en escala de grises y la imagen recortada respectivamente, se ignoró el valor para la intensidad 0 puesto que el fondo que no corresponde a la región que se analizará, es negra, por tanto tiene valores 0. Se hizo un análisis para distintos radios, con este análisis pudimos concluir que los píxeles correspondientes a la región azulada, corresponden a los valores entre 1 a 60 y entre 80 a 120 (aproximadamente claro). Por tanto, si buscamos el radio más grande que no agregue valores en las intensidades 0-60 y 80-120, Se podrá analizar la mayor cantidad de información, sin tener interrupciones de esta parte azul de manera significativa.

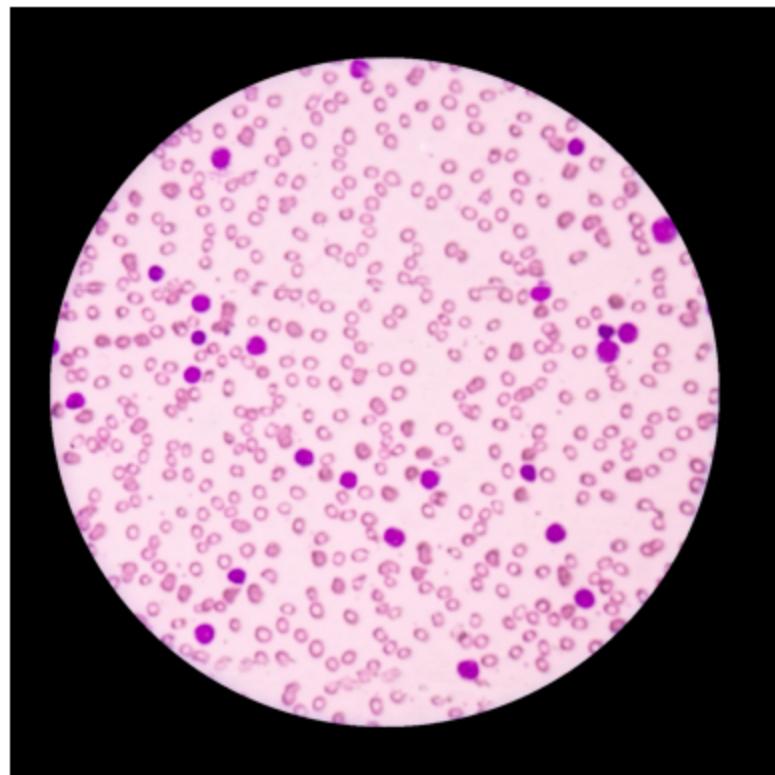
Para esto, se agregó a la función un parámetro de "exploración de región" para observar cambios en regiones pequeñas del histograma, en este caso en las regiones 83-115 píxeles, se experimentó modificando parámetros de centro y radio de la máscara de recorte hasta que fue

lo más grande posible con esta frecuencia de píxeles lo más baja posible. El resultado fue lo siguiente



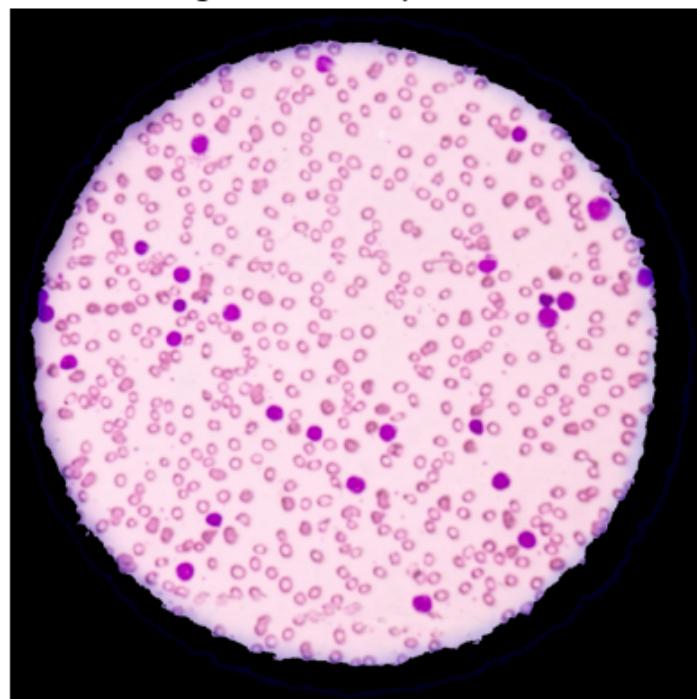
Donde la línea roja corresponde al máximo de píxeles en esa región que fueron aceptados (puesto que estos corresponden al aro azul y tal vez a algunas células muy específicas, y por ello se agregó un margen de error). Estos resultados corresponden a un círculo con radio **565** píxeles y un centro en $x = \text{width} / 2 - 18$, $y = \text{height} / 2$. Dónde width es la cantidad de píxeles en la altura y width en el ancho (aunque la imagen mide 1300x1300 así que son los mismos valores en este caso). Por tanto, se obtuvo la siguiente imagen:

Imagen Filtrada con máscara circular



NOTA: Despues de investigar, se encontró que hay una manera de remover el azul de una imagen con la librería cv2. Con esta metodología obtuvimos el siguiente resultado:

Imagen con filtro para el azul



Sin embargo, después de hacer algunos procesos con esta imagen, es claro que no podemos hacer uso de ésta puesto que aún tiene mucho azul alrededor y los métodos de thresholding lo detectan bastante. Aún así se reporta en la sección de resultados.

2. (3.0 puntos) Con la imagen FrotisCentral.png genere 3 versiones de ella: una en RGB (ImRGB), note que esta imagen es propiamente el archivo obtenido en el párrafo anterior, otra en HSV (ImHSV) y la tercera en tonos de gris (ImG). Haga segmentaciones en las 3 imágenes para identificar: fondo, linfoblastos (células que tienen una pigmentación morada) y células hematopéyicas sanas (círculos donde el centro tiene una coloración parecida al fondo) muy probablemente solo pueda segmentar los contornos de este último tipo de células, a esa segmentación aplíquele operaciones morfológicas para conseguir una segmentación completa de las células hematopéyicas normales.

Para la transformación se hizo uso, de nuevo, de la librería cv2 y se realizó el siguiente proceso para el de escala de grises:

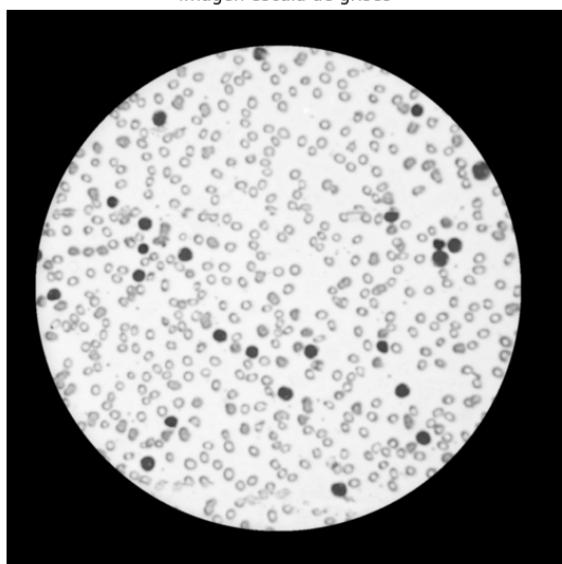
Aplicación de un filtro Gaussiano para suavizar la imagen

Umbralización de la imagen con OTSU para obtener automáticamente un valor de threshold óptimo. Este valor lo que hace es segmentar todos los linfoblastos puesto que estos son más oscuros que todas las demás células y el fondo.

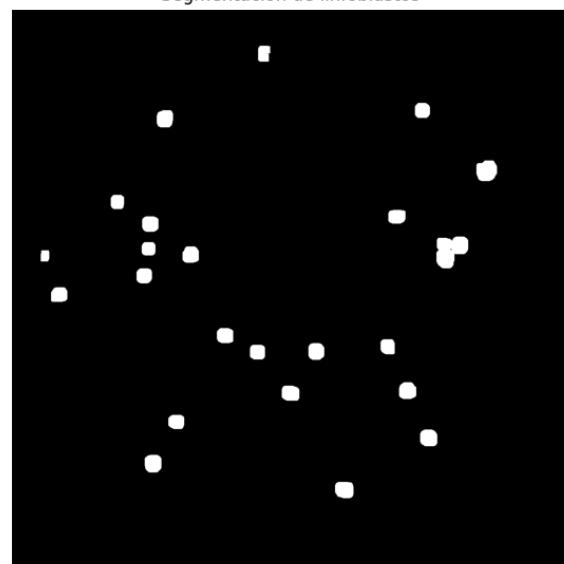
Luego aplicamos la operación de apertura y cerradura para eliminar imperfecciones o puntos blancos mal reconocidos.

Después de esto lo que pasaba era que esta segmentación no cubría completamente los linfoblastos, lo que daba como resultados partículas células sanas “falsas” que corresponden al contorno de estos linfoblastos, así que se decidió aplicar la operación de dilatación para cubrir toda la célula. Este proceso nos dio la siguiente imagen:

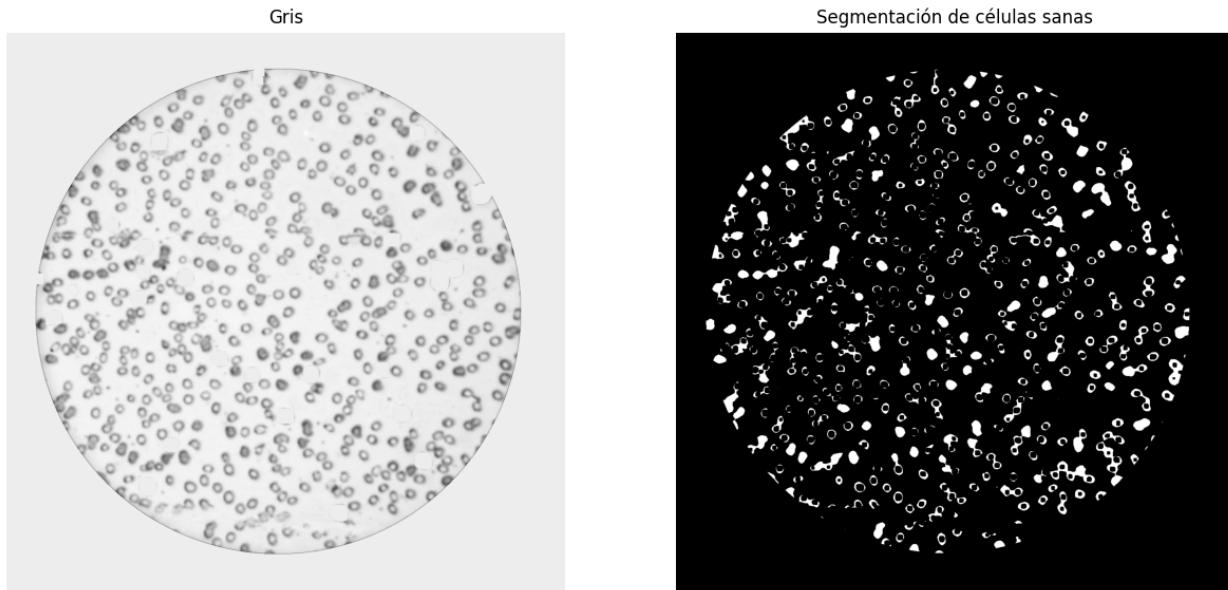
Imagen escala de grises



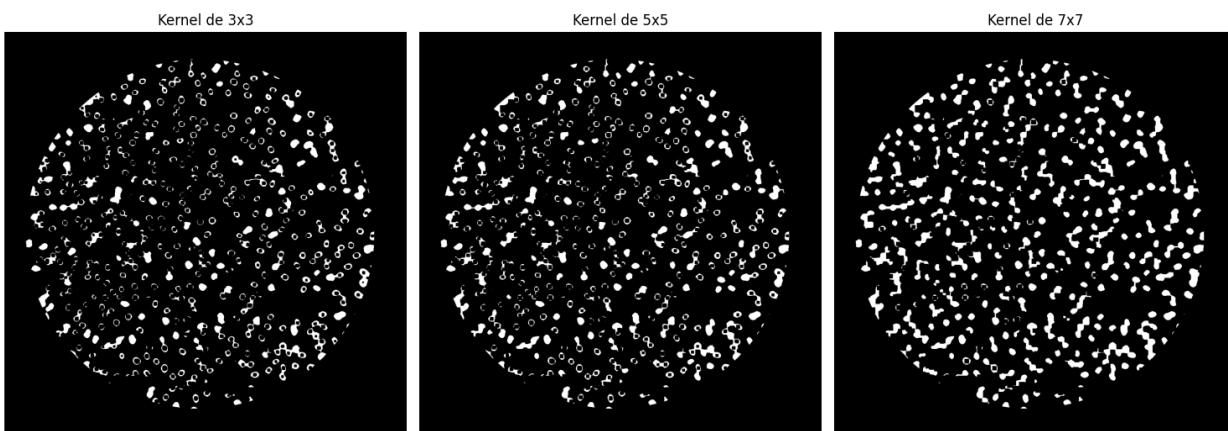
Segmentación de linfoblastos



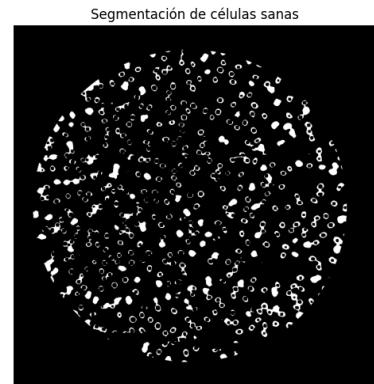
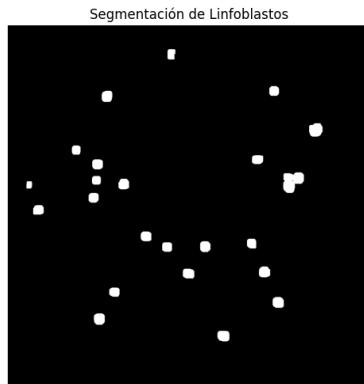
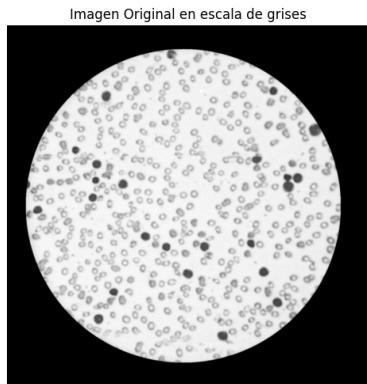
Para obtener la segmentación de las células sanas, lo que se hizo fue eliminar los linfoblastos de la imagen original, y volver a umbralizar, esto es posible porque la segmentación de linfoblastos es una máscara que con la que se pueden eliminar estas células, el método para eliminarlas fue sumando la imagen original con la segmentada (así habrá huecos blancos en los linfoblastos) y llenar estos huecos blancos con el color promedio del fondo. Después de umbralizar de nuevo, se obtuvo lo siguiente:



Solo podemos segmentar los contornos de las algunas células sanas, por tanto, tenemos que aplicar alguna operación morfológica para cerrar estos contornos, por desgracia, la operación de cerradura no funcionó, intentamos con kernel de 3x3, 5x5, 7x7, 9x9 y 11x11. Pero con algunos kernels bajos no alcanzaban a cerrar muchas células y con otros altos pegaba células. A continuación algunos ejemplos:

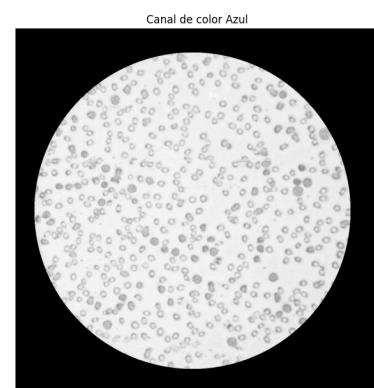
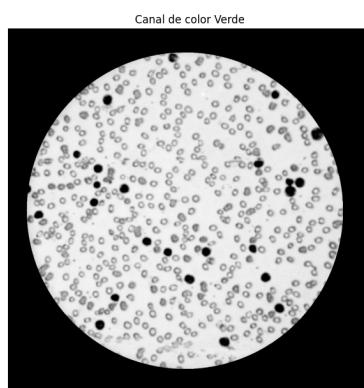
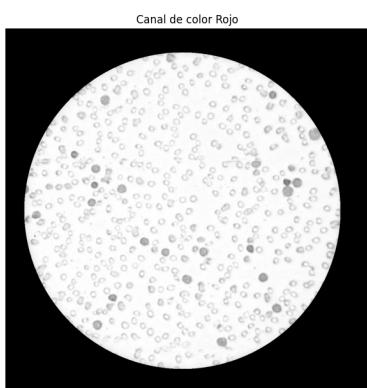


Aun así se tomará la imagen del kernel de 3x3 para los siguientes ejercicios. Entonces, resultados para la escala de grises es la siguiente:

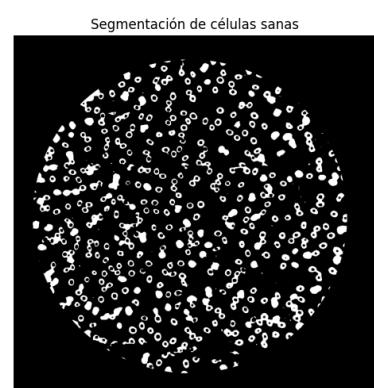
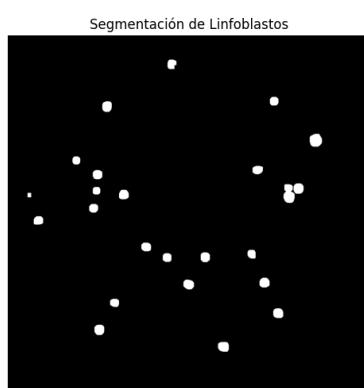
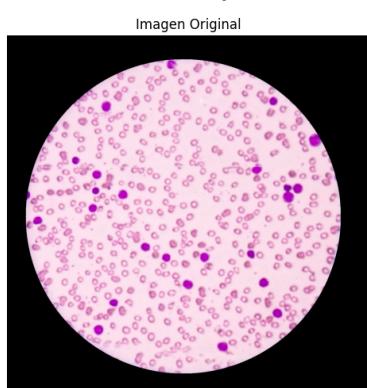


El proceso para la imagen RGB fue muy parecido:

1. Visualizar y escoger con qué canal del RGB trabajar:

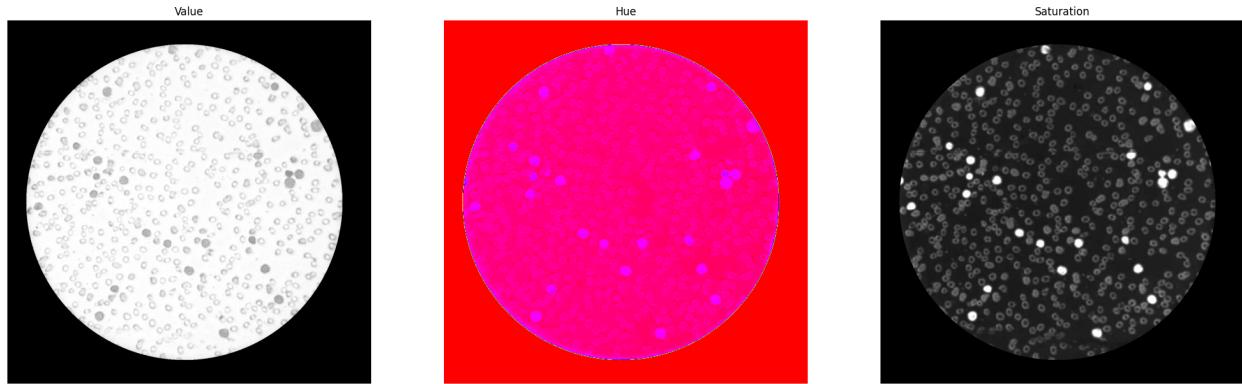


2. Se escogió el canal verde, fue una sorpresa, se esperaba que fuera un canal rojo o azul, sin embargo, este fue el canal en el que mejor se distinguen tanto las células, como los linfoblastos.
3. Después de escoger el canal se aplicó exactamente el mismo proceso que se usó en la escala de grises sobre este canal. Los resultados fueron similares al anterior pero en esta ocasión algunos linfoblastos se ven más separados:

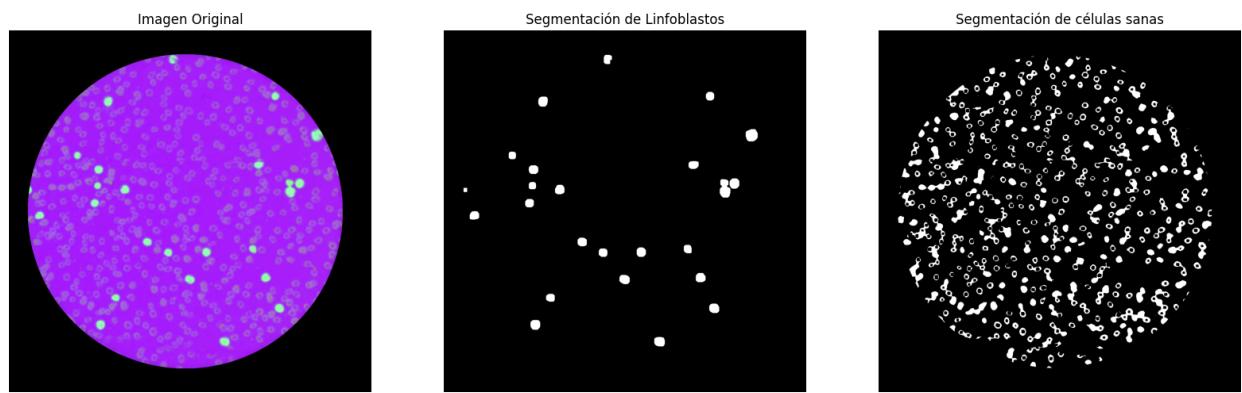


Para HSV se hizo exactamente lo mismo que para RGB:

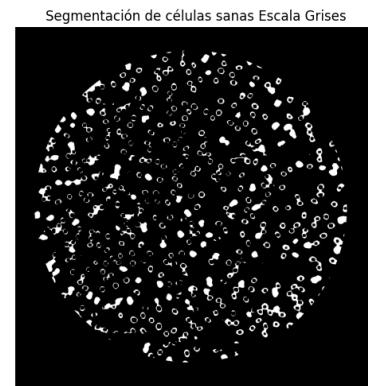
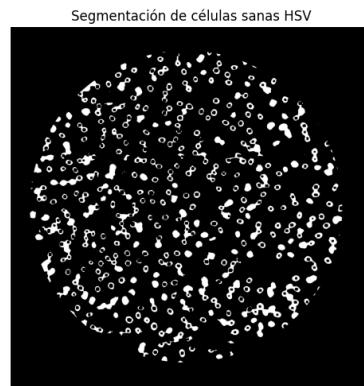
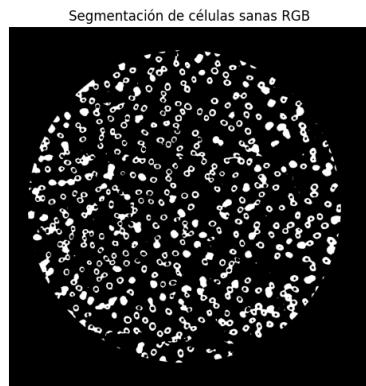
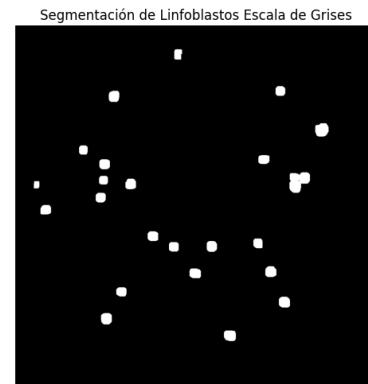
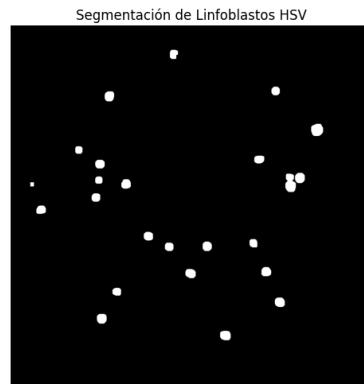
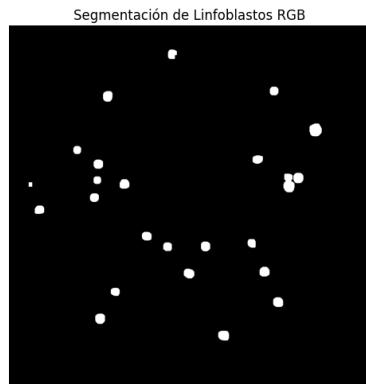
1. Se escogió un canal apropiado:



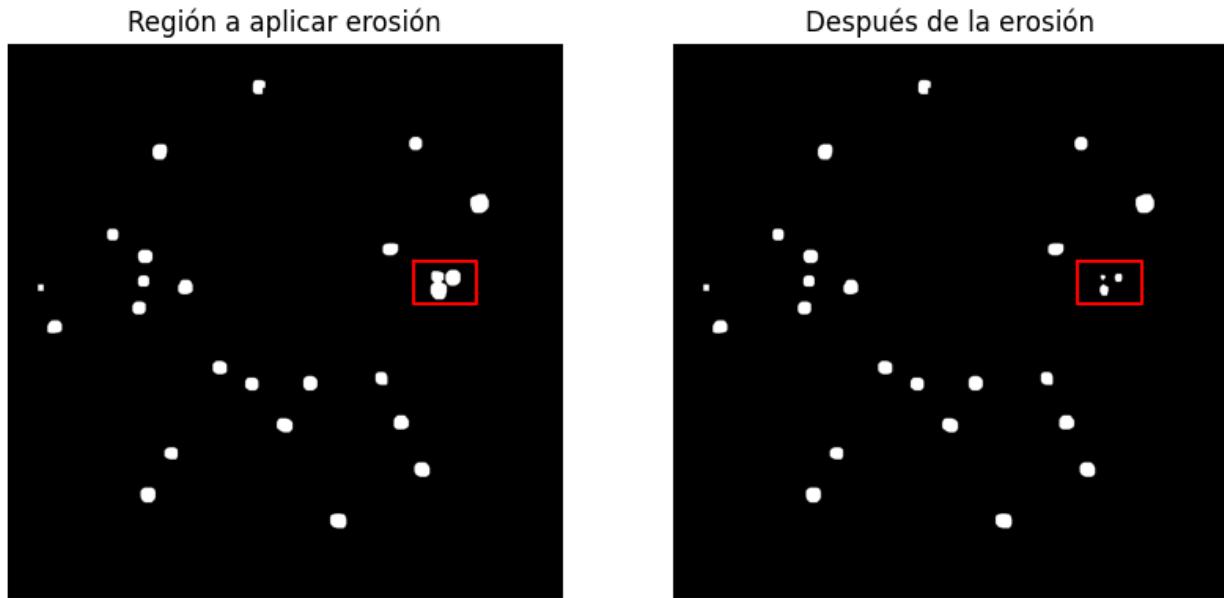
2. En este caso el mejor canal es el de saturación.
3. Se le aplicó un pre-proceso para luego hacer uso del mismo algoritmo que usamos en los casos anteriores. Los resultados fueron los siguientes:



Después se compararon los tres resultados y se escogió el mejor:

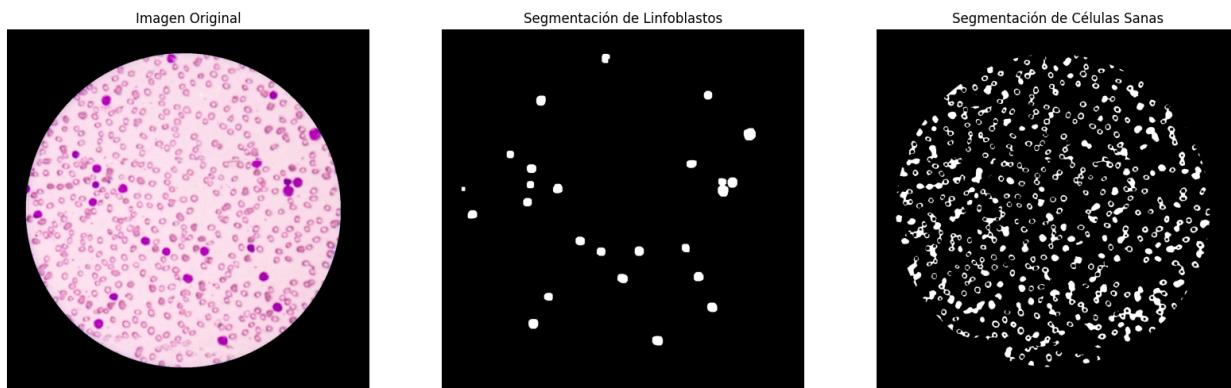


Se definió que el mejor fue el HSV para ambos casos, tanto linfoblastos como HSV. Como se puede observar, hay un par de linfoblastos que están muy juntos entre sí, podemos aplicar erosión sobre esa región específica para separarlas, (si lo hacemos sobre toda la imagen global podrían desaparecer algunos linfoblastos) sólo hay que averiguar primero qué píxeles lo rodean. Después de eso se aplicará la operación y esto es lo que se obtuvo.



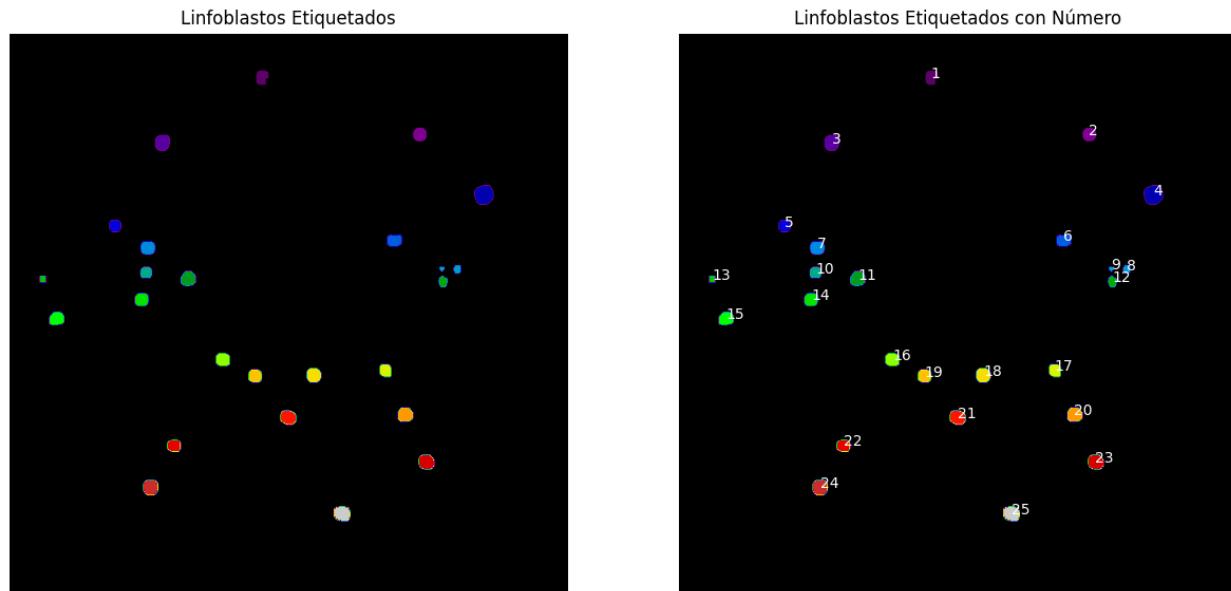
3. (2.0 puntos) Eligiendo la mejor segmentación, obtenga 2 imágenes: la primera imagen es la segmentación de linfoblastos, los píxeles indicarán 1 cuando sean parte de un linfoblasto y 0 cuando no, la llamaremos linfoblastos. La segunda imagen denominaremos hematonorm y tendrá 1's en los píxeles que pertenecen a las células sanas y 0's donde sean fondo o linfoblastos.

Estos son los resultados:



(1.5 puntos) Usando la imagen linfoblastos: cuente cuántas células hay, etiquetarlas y para cada etiqueta forme una tabla:

El etiquetado y cálculo de propiedades se hizo con la librería **skimage** específicamente las funciones **label** y **regionprops**. Aquí el etiquetado con números:

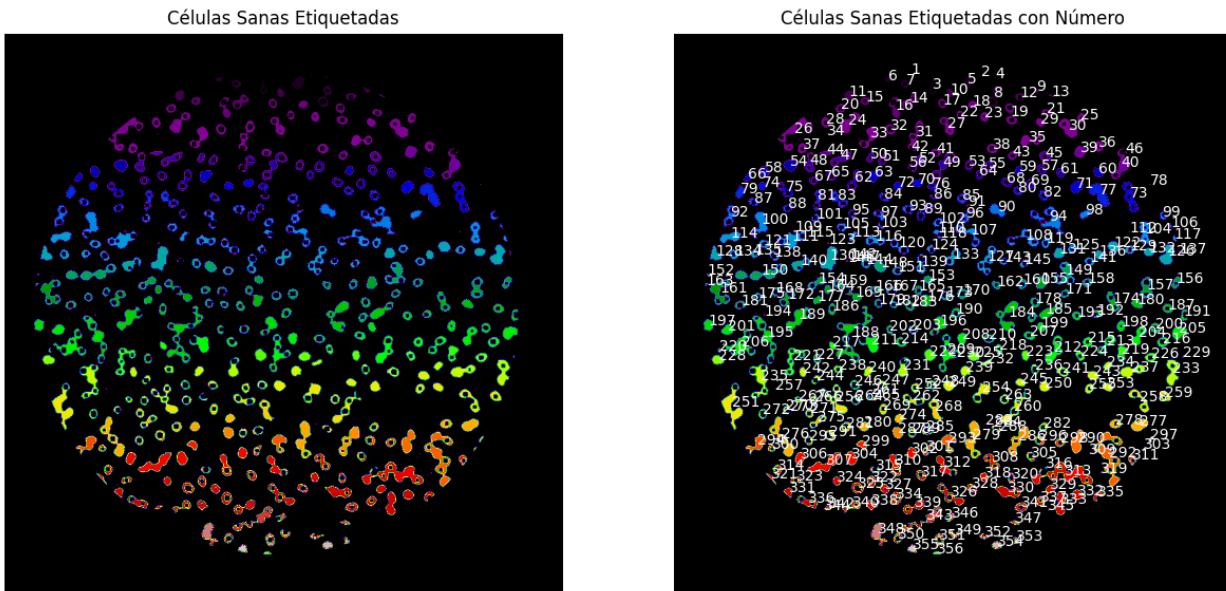


En la sección de resultados podrá observar la tabla en .csv y Excel, no se anexará aquí puesto que las tablas de linfoblastos y células sanas son muy grandes. Aún así se anexará una pequeña versión llamada "head".

Etiqueta	Área	Perímetro	Circularidad	Exentricidad	Centroide_x	Centroide_y
1	811	110.041631	0.841621	0.601327	587.061652	103.302096
2	800	101.597980	0.973934	0.255087	954.603750	236.058750
3	1064	119.012193	0.943993	0.442053	355.564850	255.211466
4	1588	147.154329	0.921540	0.306787	1103.963476	376.494332
5	639	90.870058	0.972454	0.280345	245.384977	448.992175

(1.5 puntos) Usando la imagen hematonorm: cuente cuantas células hay, etiquételas y para cada etiqueta forme la siguiente tabla:

De igual manera se hizo para las células sanas y estos son los resultados:



Y la tabla:

	Área	Perímetro	Circularidad	Exentricidad	Centroide_x	Centroide_y
Etiqueta						
1	14	12.000000	1.221730	0.995285	551.642857	93.571429
2	103	49.006097	0.538948	0.736773	714.922330	98.766990
3	413	138.704581	0.269760	0.907748	602.186441	127.835351
4	75	34.935029	0.772234	0.858114	748.533333	102.546667
5	584	93.053824	0.847528	0.746852	683.943493	115.787671

preguntas

6. (0.5 puntos) *El conteo que obtuvo en los puntos 4 y 5, ¿coincide con lo que usted obtiene al hacerlo manualmente?, si la respuesta es no, ¿Qué agregaría a su código para que sí coincida?*

La respuesta es sí en el caso de los linfoblastos y no en el caso de las células sanas, esto debido a que muchas células estaban un poco juntas y con la umbralización se terminaron uniendo formando una sola célula, esto tal vez de pueda arreglar aplicando operaciones morfológicas diferentes a cada sub espacio de la imagen (Puesto que si aplicamos una operación a toda la imagen, algunas células desaparecen, otras se dividen en dos, otras se juntan, etc...), por ejemplo, dividir la imagen en cuadrantes y en cada cuadrante aplicar una operación morfológica que más convenga, cerradura,

apertura, erosión, dilatación, etc... Pero esto requiere de muchísimo esfuerzo y de analizar cada cuadrante por separado.

Otra cosa que podría funcionar sería aplicar un modelo de inteligencia artificial entrenado para detectar este tipo de células, tal y como se muestra en este artículo: <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6161200/> puede que sea más retador, sin embargo, ya existen datasets grandes con imágenes de linfoblastos y una red neuronal pre-entrenada, como YOLO (<https://towardsdatascience.com/yolo-you-only-look-once-real-time-object-detection-explained-492dc9230006>) puede ser entrenada para detectar estos, aún así se requiere trabajo, pero tal vez podría dar resultados más potentes y automáticos que en este proyecto, por ejemplo, el modelo podría reconocer que dos linfoblastos están juntos y detectar ambos, en lugar de solo uno, como lo hizo el algoritmo que desarrollamos en este proyecto y que evitamos haciendo erosión en una zona en particular, además también podría ignorar el aro azul en lugar de recortarlo como en este caso.

7. (0.5 puntos) Compare las tablas obtenidas en el punto 4 y 5 . ¿Hay algún parámetro de los medidas que permite distinguir a los linfoblastos de las células sanas? ¿Cuál? ¿Cómo haría el clasificador?

En la imagen que obtuvimos (y por lo que se investigó, en general), los linfoblastos son mucho más oscuros que las células sanas, entonces con una umbralización y ayuda de OTSU deberíamos de tener segmentados los linfoblastos. Si esto no llegara a funcionar, entonces lo que pensaría sería comparar los tamaños, es decir, las áreas, las áreas de los linfoblastos son más grandes en promedio, lo mismo con la circularidad, la circularidad en los linfoblastos es de casi 1 en promedio, mientras que el de las células sanas es más variable.

También, sabemos que en general, las células sanas tienen un hueco en el medio, si se requiere saber si una célula es sana o no, se podría detectar si tiene un hueco en el centro o no. Esto se podría hacer primero calculando el centroide de esa célula, preguntarnos su circularidad, si la circularidad es cercana a 1, entonces podría ser tanto un linfoblasto como una célula sana, sino entonces es muy probable que sea una célula sana, por tanto, se compararía el área de la célula, con el área de un círculo centrado en el centroide de esta, con radio de tal manera que cubra toda la célula, si el área de la célula es casi igual que el área del círculo digamos en un 80 o 90 por ciento, entonces es un linfoblasto, si cumple con menos de eso, entonces es una célula sana (porque significa que tiene un hoyo y su área debería de ser mucho menor).

CONCLUSIÓN

En este proyecto de procesamiento digital de imágenes, se aplicaron diversas técnicas y algoritmos para segmentar y contar células. Mediante el uso de operaciones de preprocessamiento, filtrado, segmentación y morfología, se lograron mejoras significativas en la calidad de las imágenes, como recortar el área del círculo de la imagen y la identificación de objetos de interés. El algoritmo de Otsu fue particularmente efectivo para la binarización automática. Si bien se obtuvieron resultados positivos, existen áreas que podemos mejorar y posibles aplicaciones futuras. En resumen, este proyecto destaca el potencial y la utilidad del procesamiento digital de imágenes en diversas áreas, en este caso: células observadas en un frotis extraído de médula ósea de un paciente.

BIBLIOGRAFÍA

- [1] R. C. Gonzalez, Digital Image Processing Using MATLAB, 2nd ed. Gatesmark Pub, 2009.
- [2]<https://konfuzio.com/es/cv2/#:~:text=El%20m%C3%B3dulo%20cv2%20es%20el,funciones%20m%C3%A1s%20utilizadas%20en%20cv2.>