

# Procesamiento de imágenes digitales

## IMÁGENES MÉDICAS - Práctica 2 - 2024

Maximiliano Gatto

Instituto Balseiro (UNCuyo - CNEA) - Bariloche, Río Negro, Argentina

[maximiliano.gatto@ib.edu.ar](mailto:maximiliano.gatto@ib.edu.ar)

28 de febrero de 2024

## 1. Introducción

En esta práctica, se estudiaron imágenes en escala de grises (formato .pgm, estándar para imágenes médicas) y se implementaron diferentes algoritmos para el procesamiento de las mismas con el objetivo de resaltar diferentes aspectos de la imagen y la eliminación de ruido. Todos los algoritmos se implementaron en el lenguaje de programación Python, utilizando las librerías numpy, matplotlib y openCV, cuyos códigos se encuentran un repositorio en este link.

Los algoritmos que se implementaron en este trabajo se enumeran a continuación:

1. **Contraste:** Realización de histogramas de la imagen, *contrast stretching*, transformacion de *threshold*, transformación gamma y ecualización.
2. **Interpolaciones:** Vecino más cercano, bilineal y bicúbica.
3. **Filtros:** Filtros pasa bajos con kernel, filtros pasa altos (*Unsharp* y *Highboost*) y filtros en espacio de Fourier.

## 2. Resultados

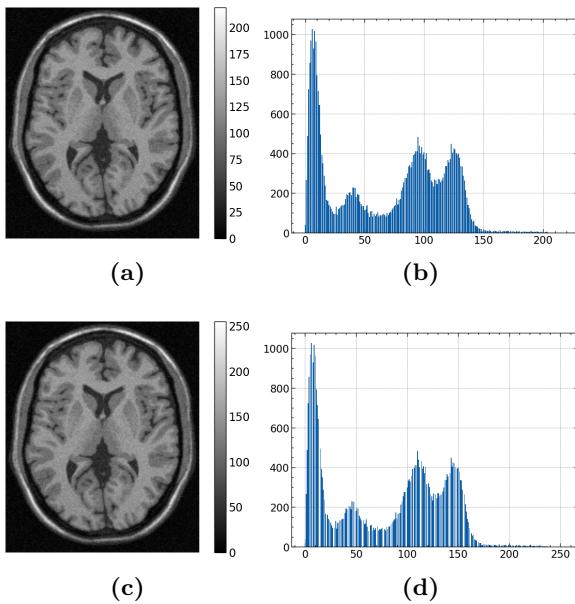
### Ejercicio 1

Se generó el histograma de la `ImagenA.pgm` y se optimizó el rango dinámico mediante el algoritmo *Contrast Streching*. Este consiste en estirar el rango dinámico de la imagen para que los valores de los píxeles se encuentren en un rango óptimo. Para ello, se buscan los valores mínimo y máximo de la imagen, asignándolos a los límites deseados (en este caso, 0 y 255, correspondientes a la cuantificación de la intensidad de la señal en 8 bits). Las intensidades intermedias fueron mapeadas de manera lineal entre estos valores. En la figura 1 se observa tanto la imagen original como la procesada, junto con respectivos histogramas.

Notar que la escala de grises de la imagen original(Figura 1a) tiene un valor máximo de 219, mientras que la imagen procesada (Figura 1c) tiene un valor máximo de 255. Si bien en los histogramas de las figuras 1b y 1d no se observa a simple vista una diferencia significativa, la observación anterior corrobora el correcto funcionamiento del algoritmo.

### Ejercicio 2

En base a la `ImagenA.pgm`, inicialmente se llevó a cabo el algoritmo de ecualización del

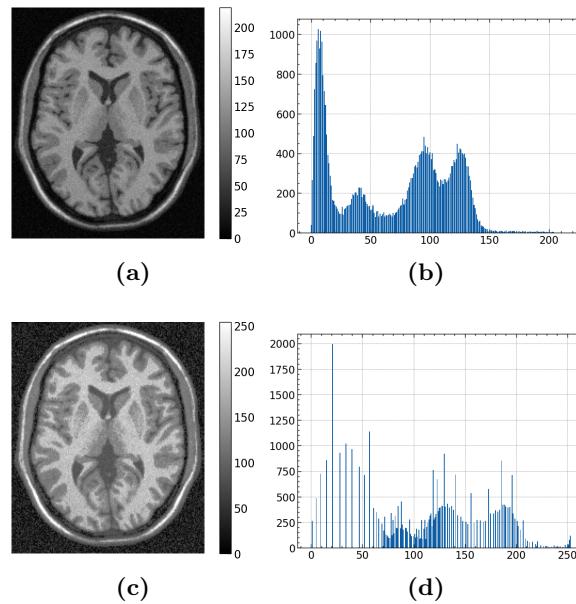


**Figura 1:** imagen original (a) y su histograma (b). Imagen procesada mediante el algoritmo de contrast stretching (c) y su histograma (d).

histograma. Al igual que en el ejercicio anterior, este algoritmo busca optimizar el rango dinámico de la imagen, redistribuyendo las intensidades de los píxeles de manera que el histograma resultante sea uniforme. En la Figura 2 se observa tanto la imagen original como la ecualizada, conjunto con sus respectivos histogramas.

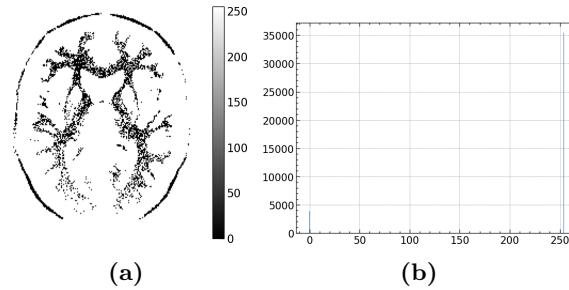
Se puede observar a simple vista que la imagen ecualizada (Figura 2c) es más brillante que la original (Figura 2a), en donde la escala de grises indica que la primera alcanza mayores valores de intensidad. En el histograma de la Figura 2d se observa que las intensidades de los píxeles se encuentran más redistribuidas respecto al histograma de la Figura 2b. Por ejemplo se puede ver que la concentración de los valores de intensidad de los píxeles que se encontraban en el rango de  $\approx [0, 50]$  en la imagen original, se encuentran ahora en el rango de  $\approx [0, 100]$  en la imagen ecualizada. Esto comprueba el correcto funcionamiento del algoritmo de ecualización del histograma.

Diferentes transformaciones  $s = T(r)$  pueden ser aplicadas a la imagen para mejorar su calidad, donde  $s$  y  $r$  denotan la intensidad del



**Figura 2:** imagen original (a) y su histograma (b). Imagen ecualizada (c) y su histograma (d).

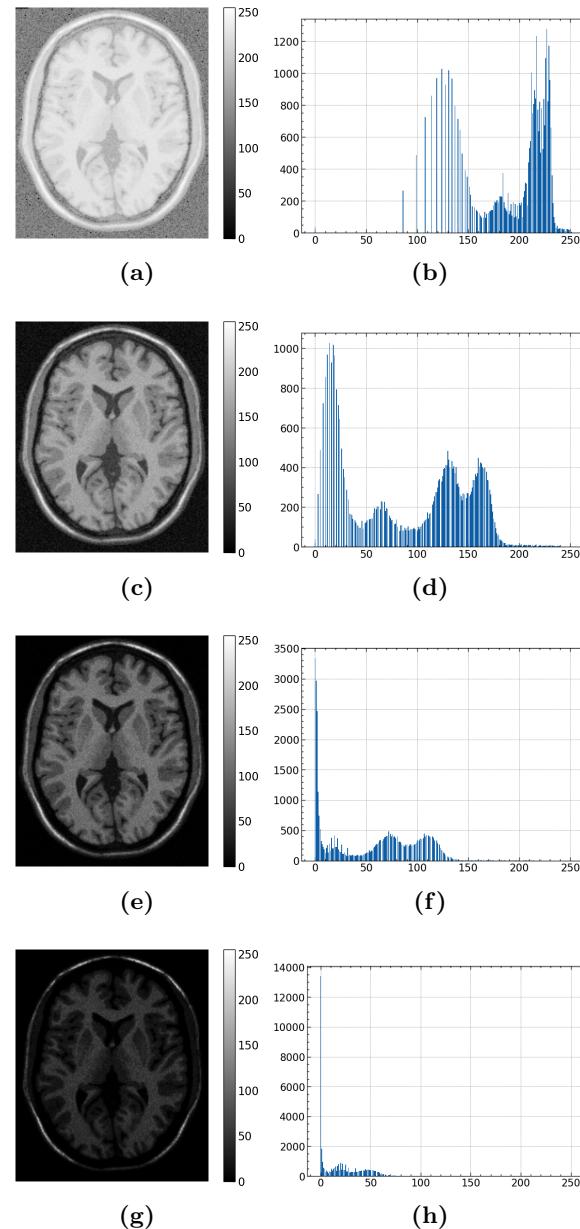
pixel de la imagen nueva y vieja respectivamente. Una de ellas es la *transformación de threshold*, en donde los valores de intensidad que no superen un cierto umbral son mapeados a 0, y los que lo superen son mapeados a 255. En la Figura 3 se observa la imagen procesada mediante el algoritmo de *threshold*, con un umbral de 128.



**Figura 3:** imagen con una transformación de threshold con umbral de 128 (a) y su histograma (b).

Este algoritmo es útil para segmentar la imagen en dos regiones, ya sea para visualizar alguna zona de interés o para utilizar de máscara en otro algoritmo, entre otros. En la Figura 3b se observa que la imagen procesada tiene un histograma con dos picos, uno en 0 y otro en 255. Esto confirma el correcto funcio-

namiento del algoritmo de *threshold*.



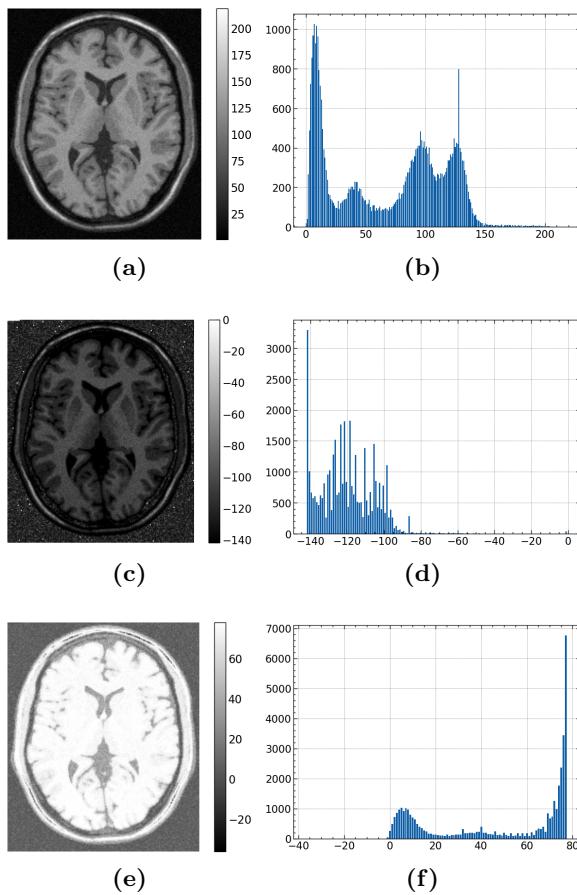
Otro tipo de transformación que se puede aplicar a la imagen es la *transformación gamma*. Se define mediante la ecuación  $s = c \cdot r^\gamma$ , donde  $c$  es una constante para ajustar los valores de las intensidades y  $\gamma$  es el parámetro de la transformación (para que los valores de salida estén entre 0 y 255,  $c = 255/\max\{\text{imagen}\}$ ). Notar que para  $\gamma < 1$  se “dilatan” las intensidades, en cambio para  $\gamma > 1$  se “contraen”. En la Figura 4 se observa la imagen procesada mediante el algoritmo de *transformación gamma*, para diferentes valores de  $\gamma$ .

**Figura 4:** imagen procesada mediante una transformación gamma conjunto con sus histogramas para diferentes valores de  $\gamma$ : (a-b)  $\gamma = 0.2$ , (c-d)  $\gamma = 0.8$ , (e-f)  $\gamma = 1.5$ , (g-h)  $\gamma = 3$ .

En la Figura 4 se observa que para  $\gamma = 0.2$  (Figura 4a) la imagen se encuentra más brillante que la original, mientras que a medida que se incrementa el valor de  $\gamma$  hasta  $\gamma = 3$  (ver figuras 4c, 4e y 4g) la imagen se encuentra más oscura. Los histogramas de las figuras 4b, 4d, 4f y 4h confirman esta observación debida a la dilatación y contracción de las intensi-

dades de los píxeles, discutidas en el párrafo anterior.

Finalmente, se realizó la diferencia entre la imagen original y las imágenes procesadas mediante el algoritmo de *threshold* y la transformación *gamma* (se eligió  $\gamma = 0.2$  y  $3$ ). En la Figura 5 se observan las imágenes resultantes.



**Figura 5:** imágenes y sus respectivos histogramas de la diferencia entre la imagen original y la imagen transformada mediante las siguientes operaciones: (a-b) transformación de *threshold*, (c-d) transformación *gamma* con un parámetro  $\gamma = 0.2$ , y (e-f) transformación *gamma* con un parámetro  $\gamma = 3$ .

En la Figura 5a, se aprecia que la imagen resultante presenta un histograma (consultar Figura 5b) en el cual se han eliminado las intensidades superiores a 128 debido al umbral de la transformación de *threshold*. En la Figura 5c, la imagen resulta más opaca, y su histograma (ver Figura 5d) muestra que las

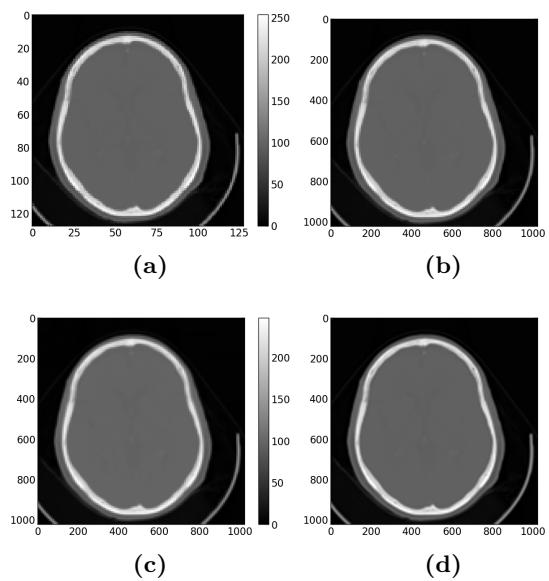
intensidades superiores se agruparon en valores inferiores. Por otro lado, la imagen de la Figura 5e es más brillante, y su histograma (ver Figura 5f) indica que las intensidades inferiores se agruparon en valores superiores. Este último efecto se debe a que la transformación *gamma* con  $\gamma = 0.2$  dilata las intensidades hacia valores mayores, mientras que la transformación *gamma* con  $\gamma = 3$  contrae las intensidades hacia valores inferiores. En consecuencia, al realizar la diferencia, la primera muestra intensidades más bajas, y la segunda, intensidades más altas. Cabe aclarar que los valores negativos se deben a que se presentan los resultados crudos de la diferencia, sin aplicar el mapeo de intensidades a valores entre 0 y 255.

### Ejercicio 3

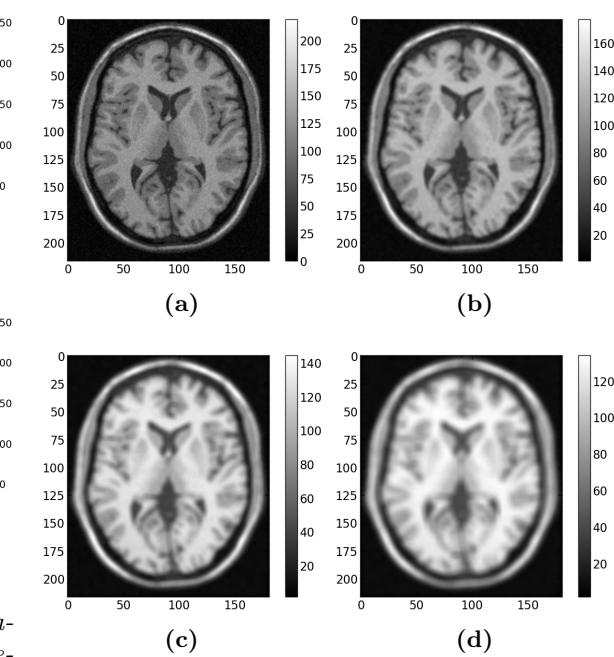
Se exploraron diferentes algoritmos de interpolación para la `ImagenC.pgm`, para llevarla desde su tamaño original  $128 \times 128$  hasta un tamaño de  $1024 \times 1024$ . Inicialmente, se implementó el algoritmo de interpolación del vecino más cercano. Este algoritmo consiste en asignar a cada píxel de la imagen nueva el valor del píxel más cercano de la imagen original. Luego, se implementó el algoritmo de interpolación bilineal, en el cual la intensidad resultante es una combinación lineal de los valores de los píxeles vecinos de la imagen original. Finalmente, se implementó el algoritmo de interpolación bicúbica mediante la función `cv2.resize` del modulo `openCV` de Python.

En la Figura 6 se observan las imágenes resultantes de los algoritmos de interpolación.

En la Figura 6 se observa que la imagen resultante de la interpolación del vecino más cercano (Figura 6b) presenta una gran cantidad de artefactos, como el efecto de “pixeado”, el cual agranda el tamaño de los píxeles, especialmente en los bordes. Por otro lado, en la imagen resultante de la interpolación bilineal (Figura 6c) se reducen estos efectos, pero se sigue observando un efecto de pixeado en los bordes. Finalmente, la imagen resultante de la interpolación bicúbica (Figura 6d) pre-



**Figura 6:** (a) imagen original de  $124 \times 124$ . Imágenes con tamaño de  $1024 \times 1024$  procesadas mediante los algoritmos de interpolación: (b) vecino más cercano, (c) bilineal y (d) bicúbica.

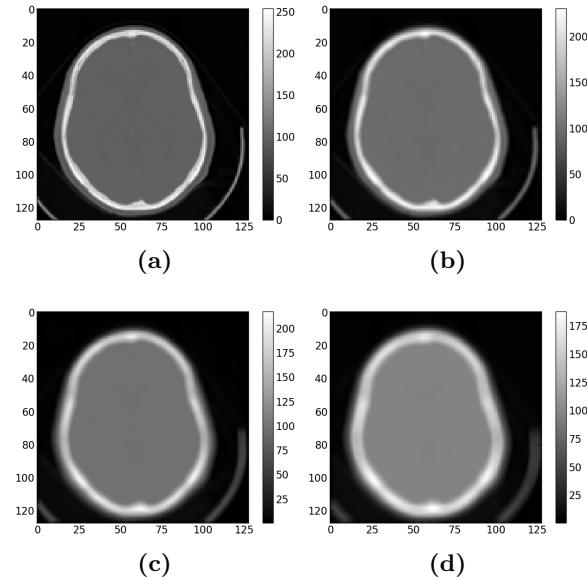


**Figura 7:** (a)ImagenA original. Imágenes filtradas con kernels de (b)  $3 \times 3$ , (c)  $5 \times 5$  y (d)  $7 \times 7$ .

senta la menor cantidad de artefactos y un mayor efecto de suavizado en los bordes, disminuyendo el pixeleaseo.

#### Ejercicio 4

Se implementan filtros pasabajos para las imágenes `ImagenA.pgm` e `ImagenC.pgm`. Se utilizaron diferentes kernels para el filtro con el objetivo de comparar los resultados. Para la implementación de un filtro pasabajo, el kernel que se utiliza es una matriz de la forma  $\bar{k} = \frac{1}{n^2} \bar{\mathbb{I}}_{n \times n}$ , donde  $\bar{\mathbb{I}}_{n \times n}$  es la matriz identidad de tamaño  $n \times n$ . Para el proceso de filtrado se realiza la convolución discreta de la imagen con el kernel, en donde el valor de cada píxel de la imagen resultante es el promedio de los valores de los píxeles vecinos de la imagen original. En las figuras 7 y 8 se observan las imágenes resultantes de los algoritmos de filtrado para kernels de tamaño  $3 \times 3$ ,  $5 \times 5$  y  $7 \times 7$  para la `ImagenA.pgm` y la `ImagenC.pgm` respectivamente. En cada imagen se implementó el filtro mediante el método “same” que mantiene el tamaño de la imagen original.



**Figura 8:** (a)ImagenC original. Imágenes filtradas con kernels de (b)  $3 \times 3$ , (c)  $5 \times 5$  y (d)  $7 \times 7$ .

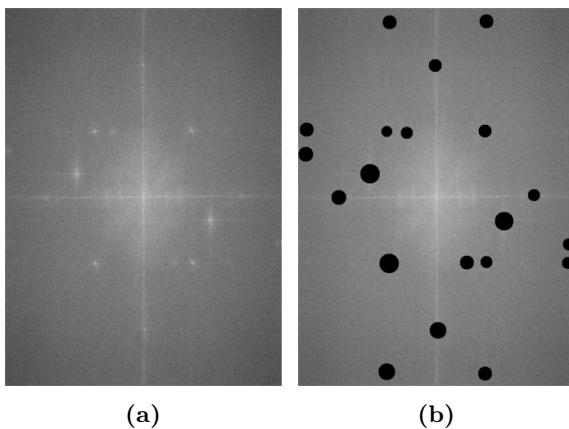
Tanto en las diferentes imágenes de la Figura 7 como en las de la Figura 8, se observa que a medida que se incrementa el tamaño del kernel, la imagen resultante se encuentra más suavizada. Si bien el efecto de suavizado redu-

ce el ruido y pixeado de la imagen, también disminuye la nitidez de los bordes. Por ello es importante elegir un tamaño de kernel adecuado para el filtro, dependiendo de la imagen y el efecto deseado. Además, se visualiza que el método de filtrado “same” mantiene el tamaño de la imagen original. La ventaja frente al método “valid” es que se evita la perdida de información de los bordes de la imagen, que se hace más notorio para kernels de mayor dimensión.

### Ejercicio 5

Se exploró la transformada rápida de Fourier para implementar filtros en el dominio de la frecuencia, con el objetivo de eliminar ruido de la imagen, que presenta cierta periodicidad.

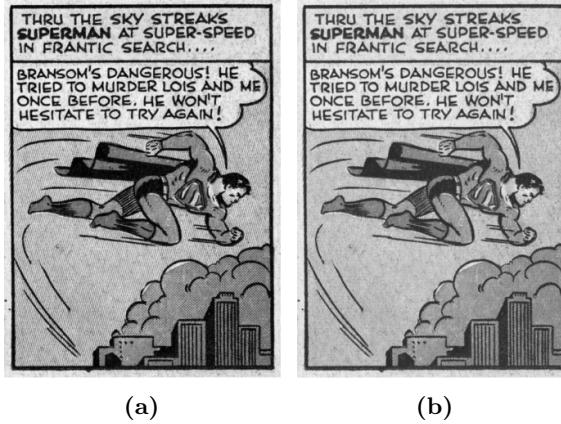
Para ello, se realizó la transformada de Fourier de la imagen `superman.pgm` y se graficó el modulo de la transformada, como se muestra en la Figura 9a. En ella se puede observar que la transformada de Fourier de la imagen presenta picos de intensidad en ciertas zonas, lo que podría indicar presencia de artefactos.



**Figura 9:** (a) modulo de la transformada de Fourier de la imagen `superman`. (b) máscara aplicada sobre la transformada para filtrar el ruido.

Para eliminar estos artefactos observados en la Figura (a), se aplicó una máscara en forma de círculos trazados manualmente, como se muestra en la Figura (b), en cada lugar donde se encontraban estos picos de intensidades. Luego, se estableció el valor de la intensidad

de los píxeles de la máscara en 0, para que al realizar la transformada inversa de Fourier, se elimine el ruido de la imagen. En la Figura 10b se observa la imagen original y la imagen filtrada.



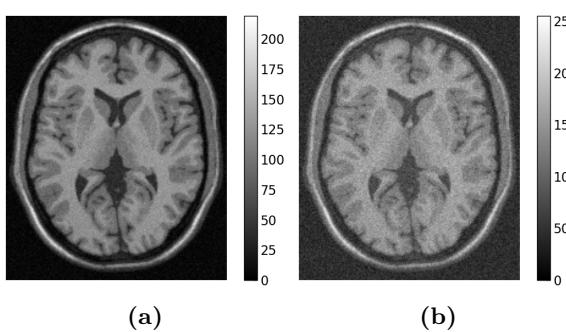
**Figura 10:** (a) imagen original `superman`. (b) imagen filtrada mediante la transformada de Fourier.

En la Figura 10a se visualiza una componente periódica en la textura de la imagen, producto del proceso de impresión de la imagen. Luego de la aplicación del filtro anteriormente mencionado, gran parte de esta componente periódica se elimina, como se observa en la Figura 10b.

### Ejercicio 6

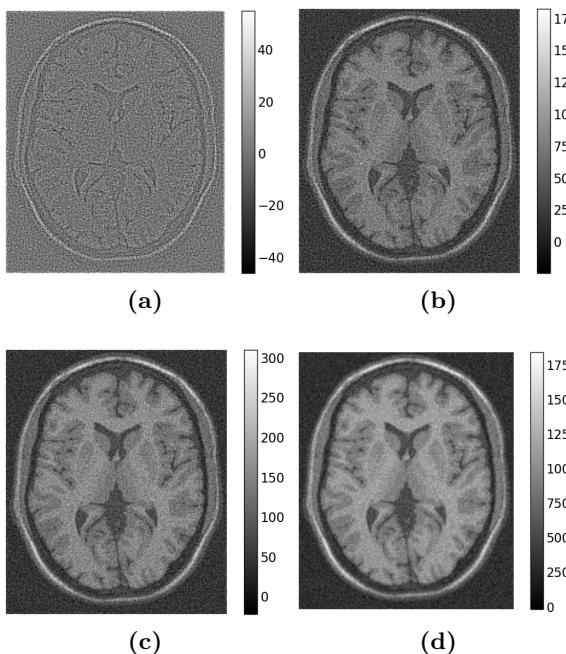
Se agregó ruido gaussiano a la `ImagenA.pgm` para evaluar los filtros *Unsharp* y *Highboost*. El ruido gaussiano se generó mediante la función `numpy.random.normal` de Python, con media 0 y desviación estándar 1, y a los valores obtenidos se los multiplicó con una constante “level”, que cuantifica el nivel de ruido agregado. En la Figura 11 se observa la imagen original y la imagen con ruido gaussiano.

Se probó el funcionamiento de los distintos filtros pasa altos en la imagen ruidosa mostrada en la Figura 11b. Se denota  $f(x, y)$  como la imagen original a filtrar y  $g(x, y)$  como la imagen filtrada mediante un filtro pasa bajos. Un filtro pasa altos se define como  $f_{filtrada}(x, y) = Af(x, y) - g(x, y)$ , donde  $A$  es



**Figura 11:** (a) imagen original *ImagenA*. (b) imagen con ruido gaussiano.

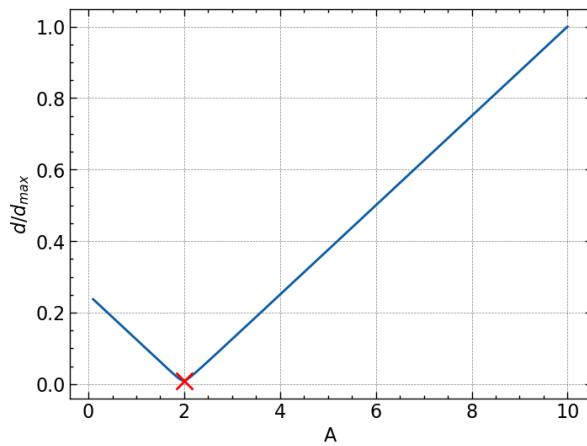
un factor. Cuando  $A = 1$ , corresponde al filtro *unsharp*; para otros valores de  $A$ , se trata del filtro *highboost*. En la Figura 12 se muestra el resultado de aplicar los filtros con diferentes parámetros  $A$ .



**Figura 12:** (a) Imagen filtrada con filtro unsharp. Imágenes filtradas con el filtro highboost con parámetro  $A$ : (b)  $A = 1.5$ , (c)  $A = 2$ , y (d)  $A = 8$ .

Para evaluar la performance del filtro para diferentes valores del parámetro  $A$ , se define una métrica mediante el valor absoluto de la diferencia término a término, es decir

$d = \sum |a_{ij} - b_{ij}|$ ,  $a_{ij}$  y  $b_{ij}$  corresponden al elemento  $i, j$  de la matriz de la imagen filtrada y la original respectivamente. Notar que a menor distancia  $d$ , mayor es la performance del filtro ya que difiere menos de la imagen original. En la Figura 13 se muestra la distancia obtenida para diferentes valores del parámetro  $A$ .



**Figura 13:** resultados obtenidos de la metrica resultante para diferentes valores del parámetro  $A$  del filtro *highboost*.

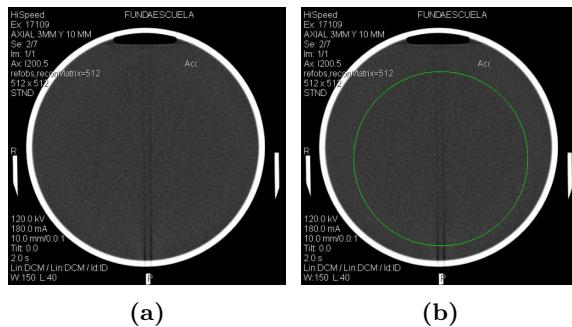
En la gráfica de la Figura 13 se puede observar que la distancia se minimiza para  $A = 2$ . El crecimiento lineal para  $A > 2$  se puede explicar por la forma en que se calcula el filtro *highboost*. A medida que se incrementa el valor de  $A$ , la imagen  $g(x, y)$  (filtrada con p-absabados) es menos significativa ya que el peso de la imagen  $f(x, y)$  original crece proporcionalmente. Por lo tanto, es de esperar que la diferencia aumente proporcional con el parámetro.

## Ejercicio 7

Se utilizaron imágenes adquiridas con el CT HiSpeed con el objetivo de medir la relación señal ruido (*SNR* por sus siglas en inglés) y el ancho de la *point spread function* (*PSF*) (definido por el FWHM).

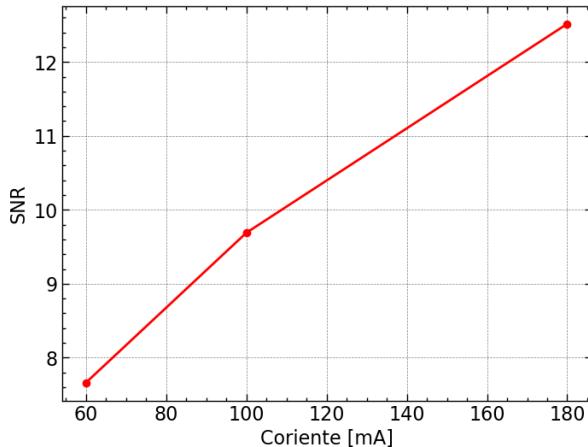
Inicialmente se midió la *SNR*, definida mediante  $SNR = \mu/\sigma$ , donde  $\mu$  es el valor medio de la imagen y  $\sigma$  es la desviación estándar. Para ello, se seleccionó una región de interés *ROI*

en donde se quiere medir el valor medio y el desvío estándar de la imagen. En la Figura 14 se observa la imagen original y la imagen con la *ROI* seleccionada.



**Figura 14:** (a) imagen original adquirida con el CT HiSpeed (AAA0002.pgm). (b) imagen con la región de interés seleccionada.

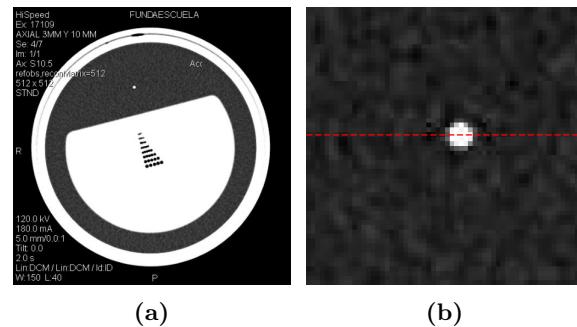
Se mantuvo la misma ROI para cada todos los cortes (2, 3 y 4) dado que se trataba de la misma figura pero obtenida con diferentes corrientes. En la Figura 15 se graficó la relación señal ruido en función de la corriente utilizada para adquirir la imagen.



**Figura 15:** SNR en función de la corriente utilizada para adquirir la imagen.

En la Figura 15 se observa que la SNR aumenta a medida que se incrementa la corriente utilizada para adquirir la imagen. Esto se debe a que a mayor corriente, mayor es la cantidad de fotones que llegan al detector, lo que disminuye el ruido de la imagen.

Por último, se midió la *PSF* sobre el punto definido por los cortes 11 al 14. En la Figura 16 se observa la imagen original y la linea transversal sobre la cual se midió la *PSF*.



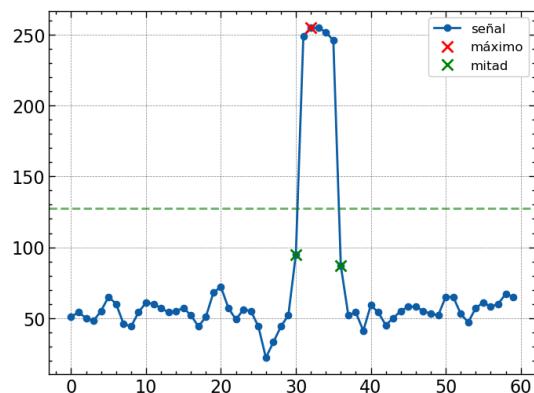
**Figura 16:** (a) imagen original capturada con el CT HiSpeed (AAA0011.pgm). (b) Imagen ampliada con una línea roja que delimita la región seleccionada para medir la PSF.

Se obtuvieron los valores de los pixeles delimitados por la línea roja que se muestra en la Figura 16b para medir la *PSF* sobre el punto blanco del centro. Para ello, primero se identifica el máximo de la señal, dado que se trata de un punto blanco, y se buscan los valores de los pixeles más cercanos que se encuentran a la mitad de la altura del máximo o el más cercano estrictamente menor. En la Figura 17 se muestra, a modo de ejemplo, la señal resultante y los valores necesarios para el FWHM del corte 11.

La linea punteada verde de la Figura 17 indica la mitad del máximo de la señal, y las dos cruces verdes indican los valores de los pixeles más cercanos al máximo, que se encuentran a la mitad de la altura del máximo. El *PSF* se calcula mediante la resta de los valores de estos pixeles. Los valores obtenidos para los diferentes cortes se muestran en la Tabla 1.

Corte	PSF
11	6
12	15
13	7
14	5

**Tabla 1:** Valores de *PSF* obtenidos para los diferentes cortes analizados.



**Figura 17:** Señal resultante y valores necesarios para el FWHM del corte 11. La línea punteada verde representa la mitad del máximo de la señal, y las dos cruces verdes indican los valores de los pixeles más cercanos al máximo, que se encuentran a la mitad de la altura del máximo.

En la Tabla 1 se observa que el valor de la *PSF* es mayor para el corte 12, lo que indica que la resolución espacial es menor en este corte. En esta imagen se observó una especie de anillo concentrólico al punto blanco, lo que indica fenómenos de interferencia, produciendo artefactos que afectan la resolución espacial.

### 3. Conclusiones

En conclusión, se implementaron diversos algoritmos de procesamiento de imágenes, destacando su aplicabilidad en diferentes contextos. La transformación de *threshold* resultó útil para la segmentación de imágenes, mientras que la transformación *gamma* proporcionó un medio versátil para ajustar el contraste, permitiendo dilatar ( $\gamma < 1$ ) o contraer ( $\gamma > 1$ ) el histograma. Además, se verificó que el método de ecualización tiende a generar histogramas más uniformes, lo que permite utilizar todo el rango dinámico de la imagen.

Se implementaron diferentes algoritmos de interpolaciones para aumentar el tamaño de la imagen. Como resultado, el método de vecino más cercano tiende a generar artefactos, como agrandar el tamaño de los pixeles. La interpolación bilineal redujo estos efectos, pe-

ro la interpolación bicúbica presentó la menor cantidad de artefactos y un mayor efecto de suavizado en los bordes, disminuyendo el pixeleado. Sin embargo, es importante tener en cuenta que la interpolación bicúbica es más costosa computacionalmente que la interpolación bilineal.

Para el caso del filtrado se constató que los filtros pasa bajos son útiles para reducir el ruido, aunque con la contrapartida de afectar la nitidez de los bordes. La aplicación de la transformada de Fourier se reveló eficaz en la eliminación de artefactos periódicos. En cuanto a los filtros pasa altos, se estableció que el parámetro óptimo (que presenta una métrica menor) para el filtro *highboost* es  $A = 2$ .

Por último, se midió que la *SNR* adquiridas en el CT HiSpeed aumenta a medida que se incrementa la corriente utilizada para adquirir la imagen, lo cual es esperable dado a que se incrementa la cantidad de fotones que llegan al detector, disminuyendo el ruido de la imagen. Además, se observó que el valor de la *PSF* es mayor para el corte 12, debido a los efectos de interferencia, resultando en una menor resolución espacial.