Laplace:

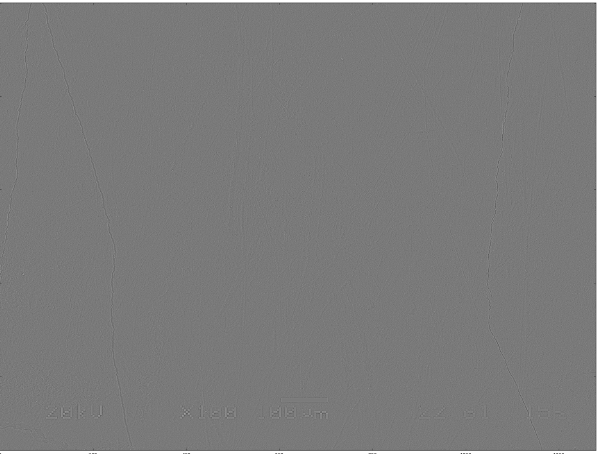
Esta función busca bordes usando el operador de laplace.

from skimage.filters import laplace,prewitt

bord=laplace(img,ksize=3)

plt.figure(figsize=(24, 24))

plt.imshow(bord,cmap=*'gray'*)



Observaciones: Como podemos observar no es un método factible para nuestro propósito por lo que no vamos a continuar usándolo.

Prewitt:

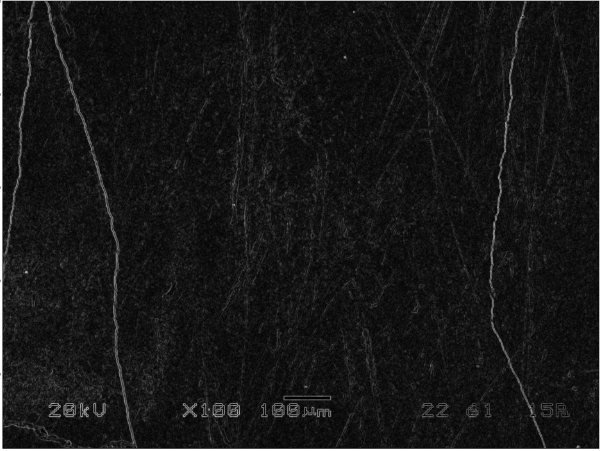
Encuentra los bordes usando la transformada de prewitt.

from skimage.filters import prewitt

bord2=prewitt(img)

plt.figure(figsize=(24, 24))

plt.imshow(bord2, cmap=*'gray'*)



Observaciones: Como podemos ver las grietas en la imagen las detecta bien pero las estrías de dieta son siluetas muy tenues y contiene mucho ruido.

Scharr:

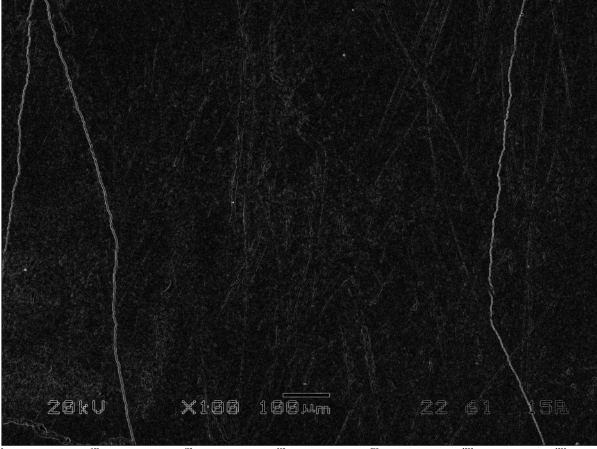
Con este método encontramos los bordes usando la transformadada de scharr

from skimage.filters import scharr

bord3=scharr(img)

plt.figure(figsize=(24, 24))

plt.imshow(bord3, cmap=*'gray'*)



Observaciones: Como podemos ver en la imagen las estrías de dieta son muy tenues pero ligeramente mejores que en la anterior tampoco demasiado pero ligeramente, el método es algo más rápido también pero no obtenemos algo tangible.

Sobel:

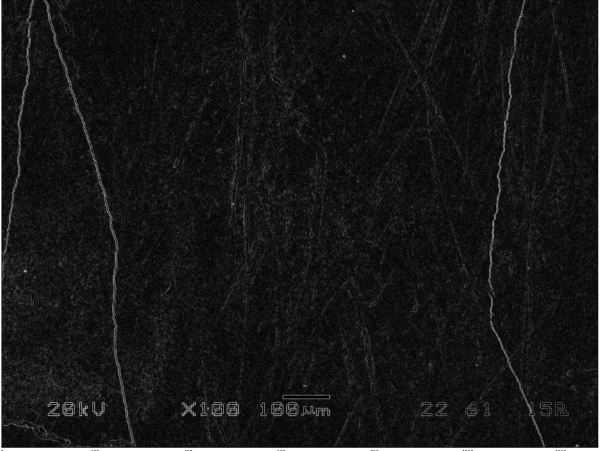
Este método busca bordes usando la transformada de sobel

from skimage.filters import sobel

bord4=sobel(img)

plt.figure(figsize=(24, 24))

plt.imshow(bord4, cmap=*'gray'*)



Observaciones: Con este método tal y como podemos observar en la imagen crea más ruido que los anteriores pero también podemos observar la silueta de las estrías de dieta.

hessian\_matrix + hessian\_matrix\_eigvals:

Este método consiste en obtener la matriz hessiana y después sus autovectores de la matriz hexiana y nos devuelve dos matrices la matriz i1 es la matriz con el eigenvector más largo y la i2 es la matriz con eigenvector más corto.

from skimage.feature import hessian\_matrix, hessian\_matrix\_eigvals

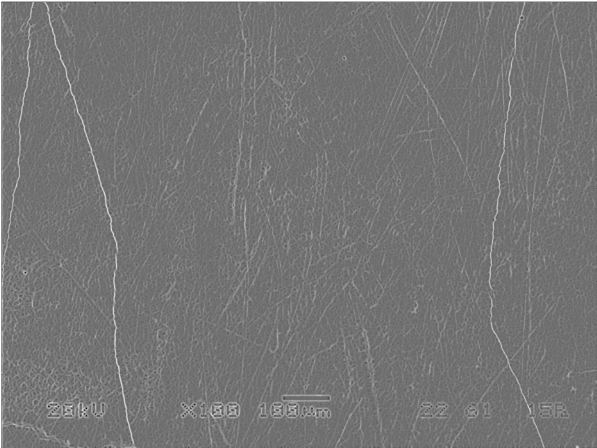
from skimage.color import rgb2grey

hxx, hxy, hyy = hessian\_matrix(img, sigma=1.7,mode=*'wrap'*,cval=0.51)

i1, i2 = hessian\_matrix\_eigvals(hxx, hxy, hyy)

plt.figure(figsize=(24, 24))

plt.imshow(i1,cmap=*'gray'*)



Observaciones: Como podemos observar en la imagen en escala de grises del autovector de los valores más largos las siluetas de las estrías de dieta son las que más se remarcan sobre un tenue fondo gris pero pudiendo ser observadas por lo que este podría ser un punto de partida.

Canny: dos variantes he probado:

Primera:

Consiste en primero de obtener los bordes llamar a una función que elimina el ruido y después al detector de bordes canny para obtener los bordes y los parámetros del canny en como sigma un valor intermedio de 1.4 junto con un umbral bajo muy bajo y un umbral alto también bajo obtenemos una imagen resaltando algunos bordes pero no todo los que queremos ya que no están demasiado marcados.

from skimage.restoration import denoise\_tv\_bregman

from skimage.feature import canny

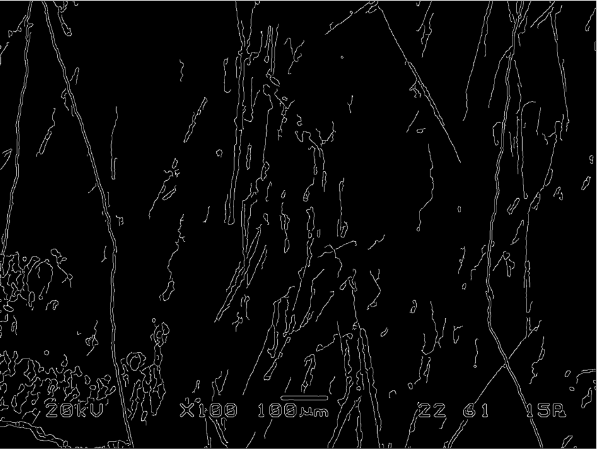
image = img

denoised = denoise\_tv\_bregman(image, 1)

edges = canny(denoised, 1.4, 0.01, 0.1)

plt.figure(figsize=(24, 24))

plt.imshow(edges,cmap=*'gray'*)



Observaciones: Desde esta imagen con bastante ruido ya podemos observar que las más marcadas son detectadas pero no se consiguen diferenciar demasiado bien, pero en comparación con los demás métodos tiene de las mejores salidas aun no siendo buena de ir en esta línea tendiéramos que usar esto.

Segunda:

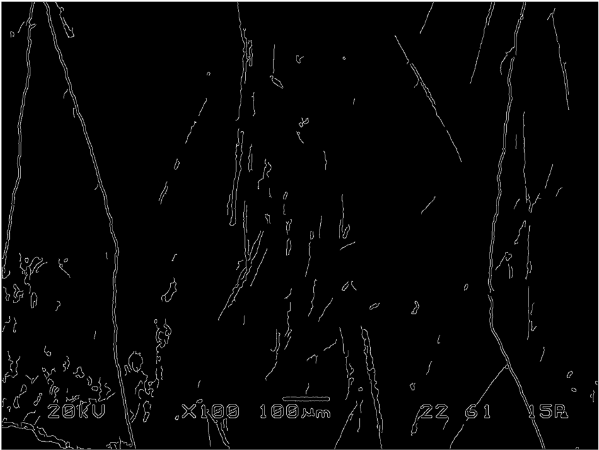
La segunda opción ha sido usar un detector de bordes canny solamente modificando sus parámetros pero en esta opción los valores de los umbrales deben ir sin normalizar entre 0 y 1 sino entre 0 y 255.

from skimage.feature import canny

edges1 = feature.canny(img,sigma,low\_threshold,high\_threshold)

plt.figure(figsize=(24, 24))

plt.imshow(edges1, cmap=*'gray'*)



Observaciones: Esta imagen detecta menos ruido que con la otra tentativa pero sigue siendo deficiente en cuanto a las líneas ya que aparte de detectar pocas detecta las que realmente no son estrías de dieta. Pero de ir en alguna línea seria por este camino.

Roberts:

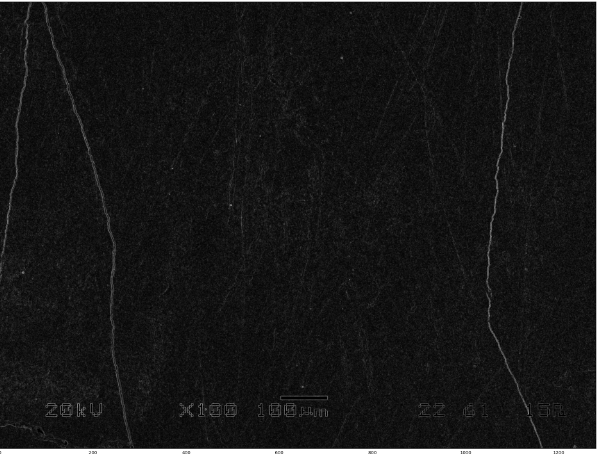
Esta función encuentra los bordes usando la operación cruzada de Robert

from skimage.filters import roberts,roberts\_neg\_diag

edge\_roberts = roberts(img)

plt.figure(figsize=(24, 24))

plt.imshow(edge\_roberts,cmap=*'gray'*)



Observaciones: Este método produce mucho ruido y las estrías son líneas demasiado tenues.

Gabor:

Es un filtro linear con un kernel gausiano que es modulado por un onda sinusoidal plana. Principalmente se usa en visión artificial de clasificación y detección de bordes.

Obtenemos un par de imágenes

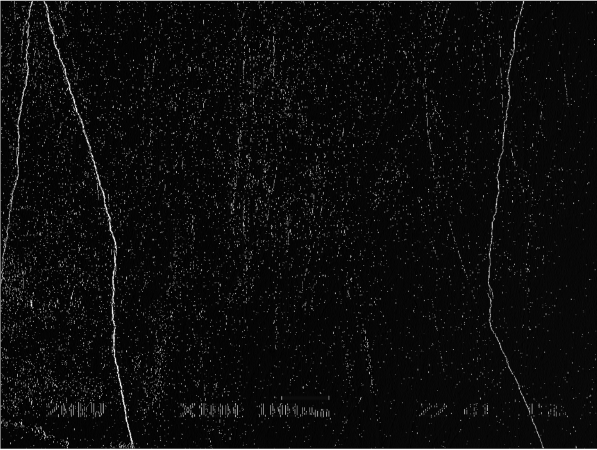
from skimage.filters import gabor,gabor\_kernel

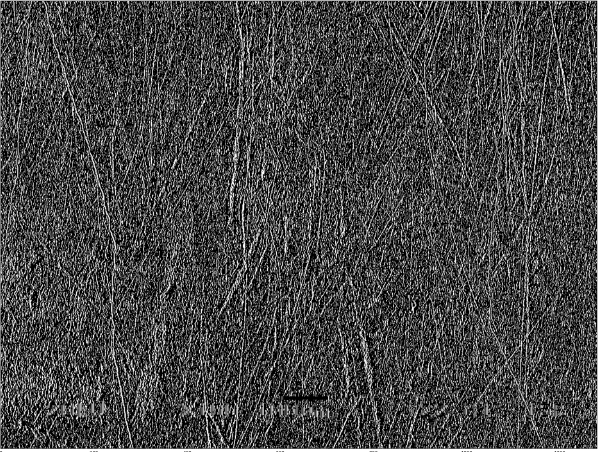
filt\_real, filt\_imag = gabor(img, frequency=0.59)

plt.figure(figsize=(24, 24))

plt.imshow(filt\_real,cmap=*'gray'*)

Imagen de filtro real:



imagen de filtro imaginario.

Observaciones: Este método lo eh probado porque en la obtención de las líneas de sangre en los ojos es lo que se usa para ello pero al no ser líneas continuas y no seguir un patrón no he conseguido buenos resultados. En el filtro real no es tan malo pero en el filtro imaginario es ruido puro.

Deteccion por umbrales:

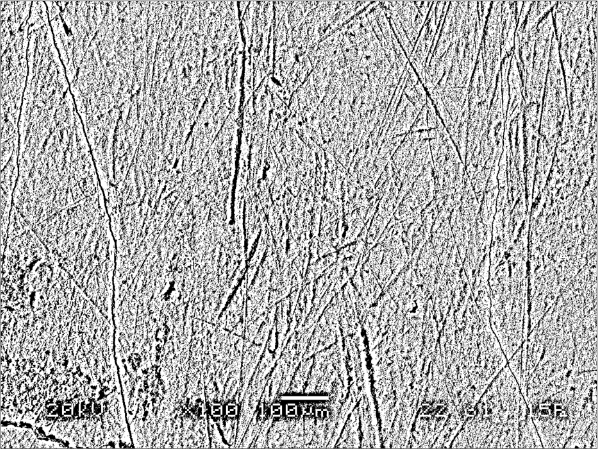
Hemos usado una función que a partir de los umbrales alto y bajo detecta los bordes de la imagen.

from skimage.filters import threshold\_adaptive

th=threshold\_adaptive(img,21,*"mean"*,8,*"mirror"*)

plt.figure(figsize=(24, 24))

plt.imshow(th,cmap=*'gray'*)



Observaciones:

Obtenemos demasiado ruido en la imagen. Por lo que no es muy apta pero también vemos siluetas de las estrías.