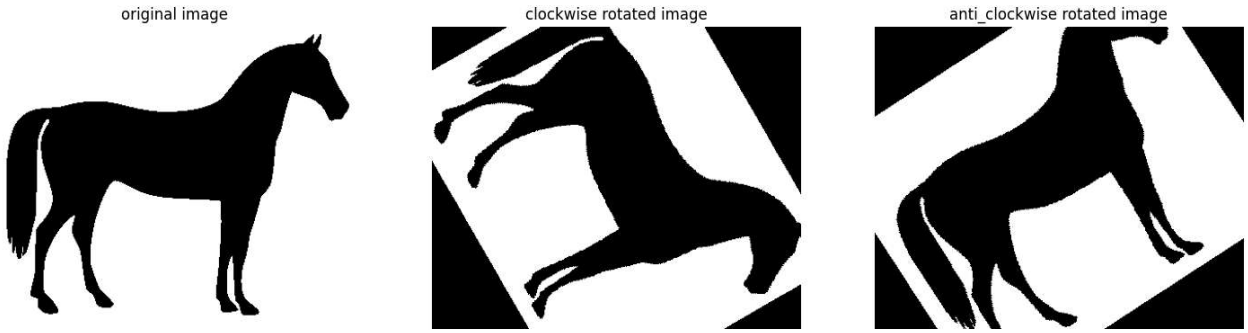
1.Transformations

1.1.Rotation

```
import matplotlib.pyplot as plt
import numpy as np
from skimage import io, data
from skimage.transform import rotate



image = data.horse()

clockwise = rotate(image, angle=-60)
anti_clockwise = rotate(image, angle=33)


fig, (ax1, ax2, ax3) = plt.subplots(ncols=3, figsize=(18, 6))

ax1.imshow(image, cmap='gray')
ax1.set_title("original image")
ax1.axis("off")

ax2.imshow(clockwise, cmap='gray')
ax2.set_title("clockwise rotated image")
ax2.axis("off")

ax3.imshow(anti_clockwise, cmap='gray')
ax3.set_title("anti_clockwise rotated image")
ax3.axis("off")

plt.show()
```



1.2.Remise à l'échelle

```
import matplotlib.pyplot as plt
import numpy as np
from skimage import io
from skimage.transform import rescale



image = io.imread('/content/Photo.png')


scaled_image = rescale(image, scale=3/4, channel_axis=-1)

fig, (ax1, ax2) = plt.subplots(ncols=2, figsize=(18, 6))

ax1.imshow(image)
ax1.set_title("original image")

ax2.imshow(scaled_image)
ax2.set_title("scaled image")
```

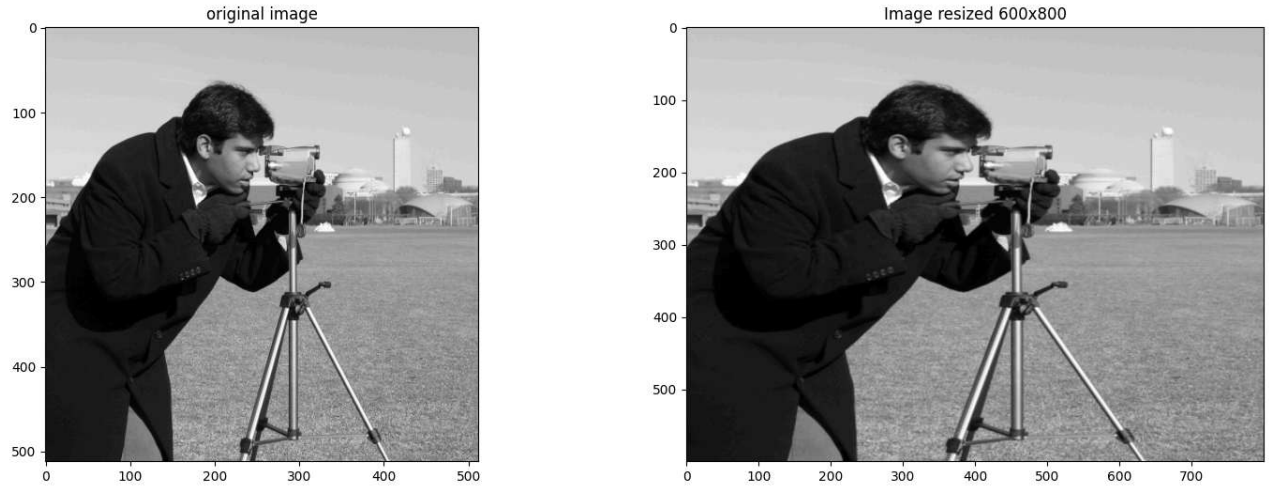
```
factor_10_aa = rescale(image, scale=1/10, anti_aliasing=True, channel_axis=-1)
factor_10_no_aa = rescale(image, scale=1/10, anti_aliasing=False, channel_axis=-1)


fig, (ax1, ax2) = plt.subplots(ncols=2, figsize=(18, 6))

ax1.imshow(factor_10_aa)
ax1.set_title("anti-aliasing")

ax2.imshow(factor_10_no_aa)
ax2.set_title("no anti-aliasing")

plt.show()
```

1.3.Redimensionnement

```
import matplotlib.pyplot as plt
import numpy as np
from skimage import io, data
from skimage.transform import resize

image = data.camera()

image_resized = resize(image, output_shape=(600, 800), anti_aliasing=True)

print(f"Dimensions originales : {image.shape}")
print(f"Dimensions après resize : {image_resized.shape}")

fig, (ax1, ax2) = plt.subplots(ncols=2, figsize=(18, 6))
```

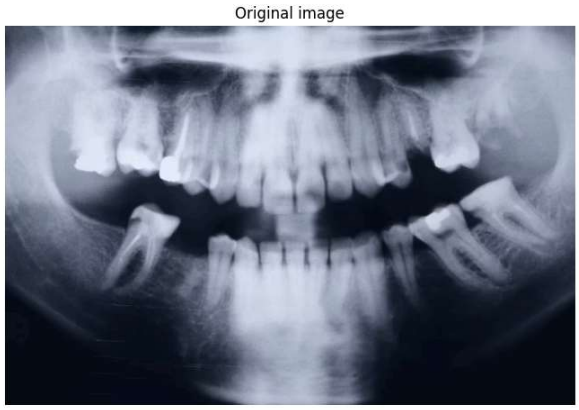
```
ax1.imshow(image, cmap='gray')
ax1.set_title("original image")

ax2.imshow(image_resized, cmap='gray')
ax2.set_title("Image resized 600x800")

plt.show()
```

```
Dimensions originales : (512, 512)
Dimensions après resize : (600, 800)
```



2.Amélioration du contraste

```
import matplotlib.pyplot as plt
import numpy as np
from skimage import io
from skimage.color import rgb2gray

xray = io.imread('/content/Xray.png')

if xray.shape[-1] == 4:
    xray = xray[:, :, :3]

xray_gray = rgb2gray(xray)

fig, (ax1, ax2) = plt.subplots(ncols=2, figsize=(18, 6))

ax1.imshow(xray)
ax1.set_title("Original image")
ax1.axis("off")

ax2.imshow(xray_gray, cmap="gray")
ax2.set_title("Image en gris")
ax2.axis("off")

plt.show()
```

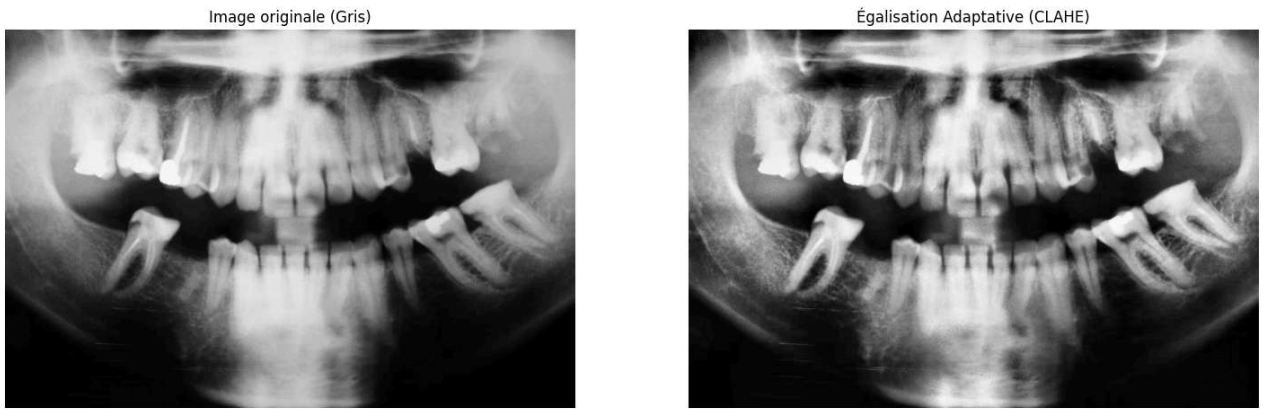


L'égalisation classique

```
import matplotlib.pyplot as plt
import numpy as np
from skimage import io, color
from skimage.exposure import equalize_hist

xray = io.imread('/content/Xray.png')

if xray.shape[-1] == 4:
    xray = xray[:, :, :3]

xray_gray = color.rgb2gray(xray)

enhanced = equalize_hist(xray_gray)

fig, (ax1, ax2) = plt.subplots(ncols=2, figsize=(18, 6))

ax1.imshow(xray_gray, cmap="gray")
ax1.set_title("Image originale (Gris)")
ax1.axis("off")

ax2.imshow(enhanced, cmap="gray")
ax2.set_title("Égalisation Classique")
ax2.axis("off")

plt.show()
```

L'égalisation adaptative

```
import matplotlib.pyplot as plt
import numpy as np
from skimage import io, color
from skimage.exposure import equalize_adapthist

xray = io.imread('/content/Xray.png')

if xray.shape[-1] == 4:
    xray = xray[:, :, :3]

xray_gray = color.rgb2gray(xray)


enhanced_adaptive = equalize_adapthist(xray_gray, clip_limit=0.01)

fig, (ax1, ax2) = plt.subplots(ncols=2, figsize=(18, 6))

ax1.imshow(xray_gray, cmap="gray")
ax1.set_title("Image originale (Gris)")
ax1.axis("off")

ax2.imshow(enhanced_adaptive, cmap="gray")
ax2.set_title("Égalisation Adaptative (CLAHE)")
ax2.axis("off")

plt.show()
```

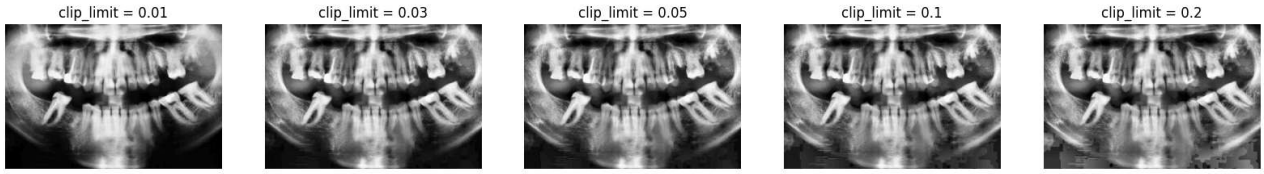


Double-cliquez (ou appuyez sur Entrée) pour modifier

Tester différentes valeurs de clip_limit

```
import matplotlib.pyplot as plt
import numpy as np
from skimage import io, color
from skimage.exposure import equalize_adapthist

xray = io.imread('/content/Xray.png')
if xray.shape[-1] == 4:
    xray = xray[:, :, :3]
xray_gray = color.rgb2gray(xray)

values = [0.01, 0.03, 0.05, 0.1, 0.2]

fig, axes = plt.subplots(ncols=5, figsize=(20, 5))

for i, v in enumerate(values):

    enhanced = equalize_adapthist(xray_gray, clip_limit=v)

    axes[i].imshow(enhanced, cmap="gray")
    axes[i].set_title(f"clip_limit = {v}")
    axes[i].axis("off")

plt.show()
```

```
for i, v in enumerate(values):
  print(i,v)
```




```
0 0.01
1 0.03
2 0.05
3 0.1
4 0.2
```

Exemple d'égalisation d'histogramme et d'étirement du contraste

```
import matplotlib.pyplot as plt
import numpy as np
from skimage import data, img_as_float
from skimage import exposure

def plot_img_and_hist(image, axes, bins=256):
    image = img_as_float(image)
    ax_img, ax_hist = axes
    ax_cdf = ax_hist.twinx()


    ax_img.imshow(image, cmap=plt.cm.gray)
    ax_img.set_axis_off()


    ax_hist.hist(image.ravel(), bins=bins, histtype='step', color='black')
    ax_hist.ticklabel_format(axis='y', style='scientific', scilimits=(0, 0))
    ax_hist.set_xlabel('Pixel intensity')
    ax_hist.set_xlim(0, 1)
    ax_hist.set_yticks([])


    img_cdf, bins = exposure.cumulative_distribution(image, bins)
    ax_cdf.plot(bins, img_cdf, 'r')
    ax_cdf.set_yticks([])

    return ax_img, ax_hist, ax_cdf


img = data.moon()

p2, p98 = np.percentile(img, (2, 98))
img_rescale = exposure.rescale_intensity(img, in_range=(p2, p98))

img_eq = exposure.equalize_hist(img)

img_adapteq = exposure.equalize_adapthist(img, clip_limit=0.03)


fig = plt.figure(figsize=(20, 10))
axes = np.zeros((2, 4), dtype=object)


axes[0, 0] = fig.add_subplot(2, 4, 1)
axes[1, 0] = fig.add_subplot(2, 4, 5)

axes[0, 1] = fig.add_subplot(2, 4, 2)
axes[1, 1] = fig.add_subplot(2, 4, 6)

axes[0, 2] = fig.add_subplot(2, 4, 3)
axes[1, 2] = fig.add_subplot(2, 4, 7)

axes[0, 3] = fig.add_subplot(2, 4, 4)
axes[1, 3] = fig.add_subplot(2, 4, 8)


ax_img, ax_hist, ax_cdf = plot_img_and_hist(img, axes[:, 0])
ax_img.set_title('Low contrast image')
ax_hist.set_ylabel('Number of pixels')
```

```
ax_img, ax_hist, ax_cdf = plot_img_and_hist(img_rescale, axes[:, 1])
ax_img.set_title('Contrast stretching')

ax_img, ax_hist, ax_cdf = plot_img_and_hist(img_eq, axes[:, 2])
ax_img.set_title('Histogram equalization')

ax_img, ax_hist, ax_cdf = plot_img_and_hist(img_adapteq, axes[:, 3])
ax_img.set_title('Adaptive equalization')

fig.tight_layout()
plt.show()
```