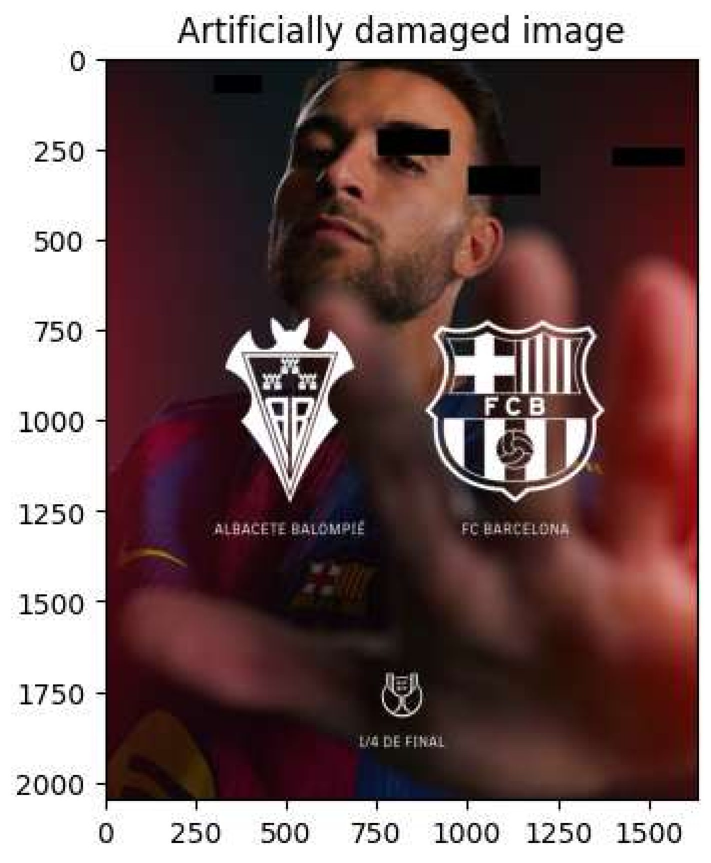
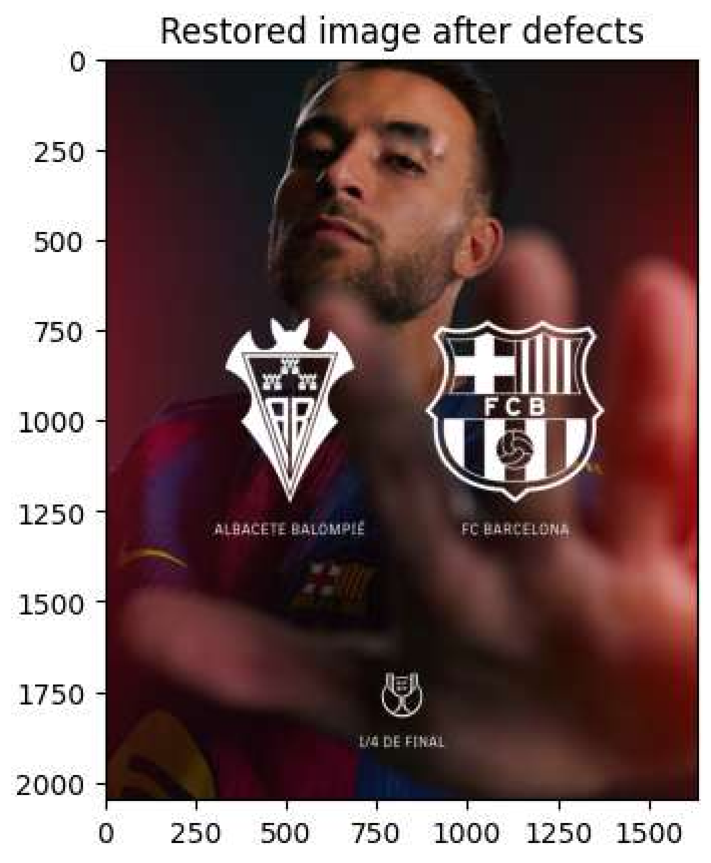
```
Commencez à coder ou à générer avec l'IA.
```

TP: Restauration, Filtrage, Segmentation, YOLOv8

1. Restauration et amélioration d'images

```
import numpy as np
import matplotlib.pyplot as plt
from skimage import io, restoration

test = io.imread("/content/test.jpg")
plt.imshow(test)
plt.title("Original image")
plt.show()
def make_mask(image):
    mask = np.zeros(image.shape[:-1])
    mask[250:300, 1400:1600] = 1
    mask[50:100, 300:433] = 1
    mask[300:380, 1000:1200] = 1
    mask[200:270, 750:950] = 1
    return mask.astype(bool)

mask = make_mask(test)
image_defect = test * ~mask[..., np.newaxis]

plt.imshow(image_defect)
plt.title("Artificially damaged image")
plt.show()

restored_image = restoration.inpaint_biharmonic(
    image_defect, mask=mask, channel_axis=-1
)

plt.imshow(restored_image)
plt.title("Restored image after defects")
plt.show()
```



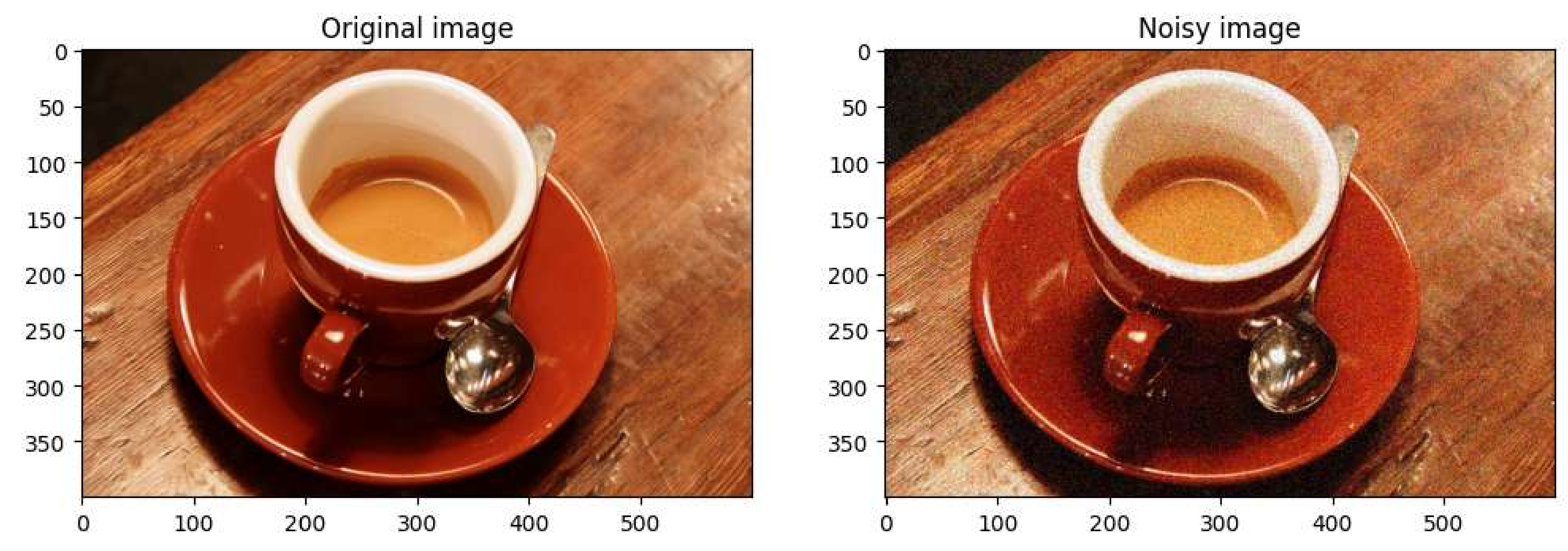


2. Filtrage

2.1. Bruit (noise)

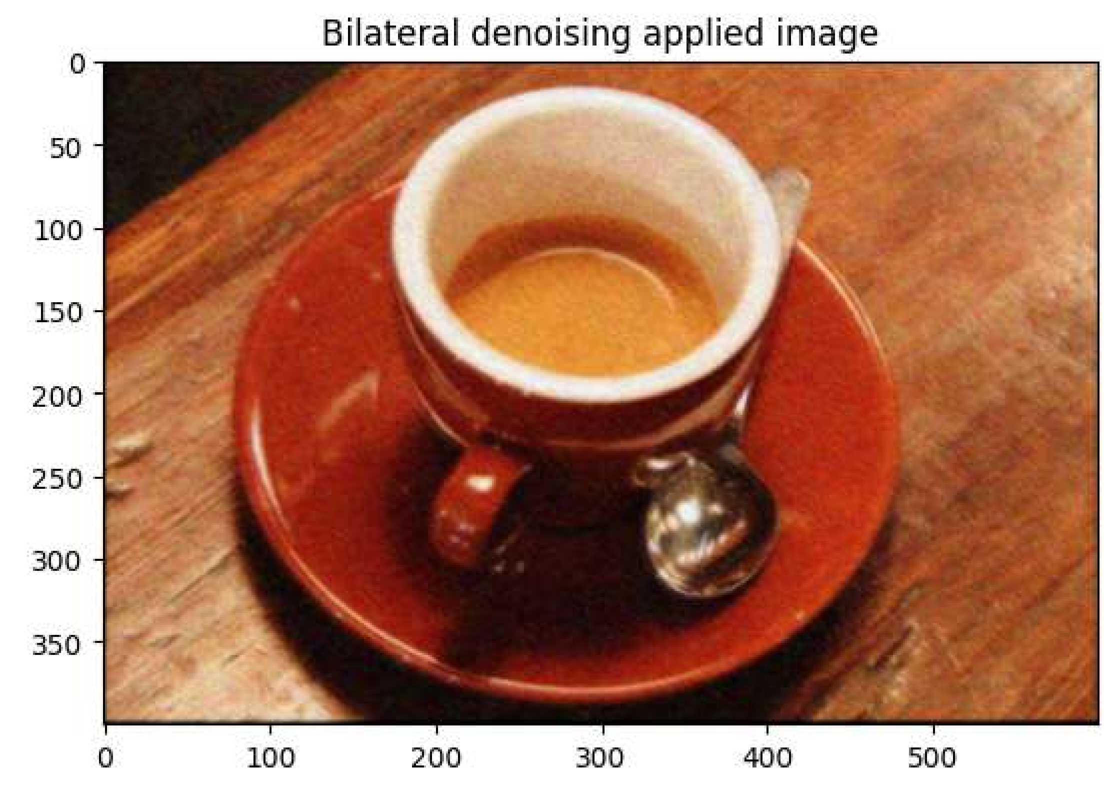
2.1. Ajout de bruit

```
import matplotlib.pyplot as plt
from skimage.util import random_noise
from skimage import data

coffee = data.coffee()
noisy_img = random_noise(coffee)

fig, (ax1, ax2) = plt.subplots(ncols=2, figsize=(12, 6))

ax1.imshow(coffee)
ax1.set_title("Original image")

ax2.imshow(noisy_img)
ax2.set_title("Noisy image")

plt.show()
```



2.1.2. Réduction de bruit

```
from skimage.restoration import denoise_bilateral

denoised_coffee_bilateral = denoise_bilateral(noisy_img, channel_axis=-1)

plt.imshow(denoised_coffee_bilateral)
plt.title("Bilateral denoising applied image")
plt.show()
```

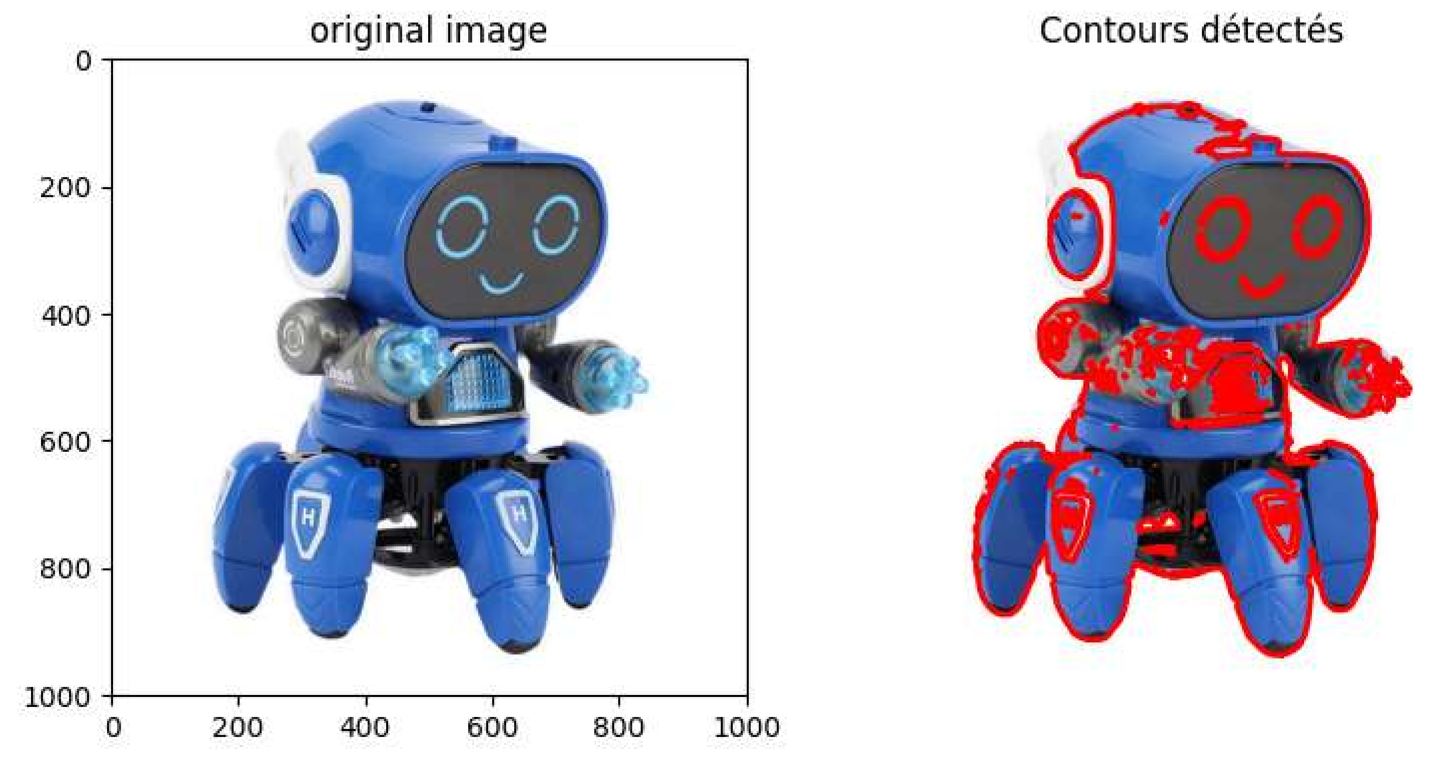


2.2. Détection de Contour

Exemple 1

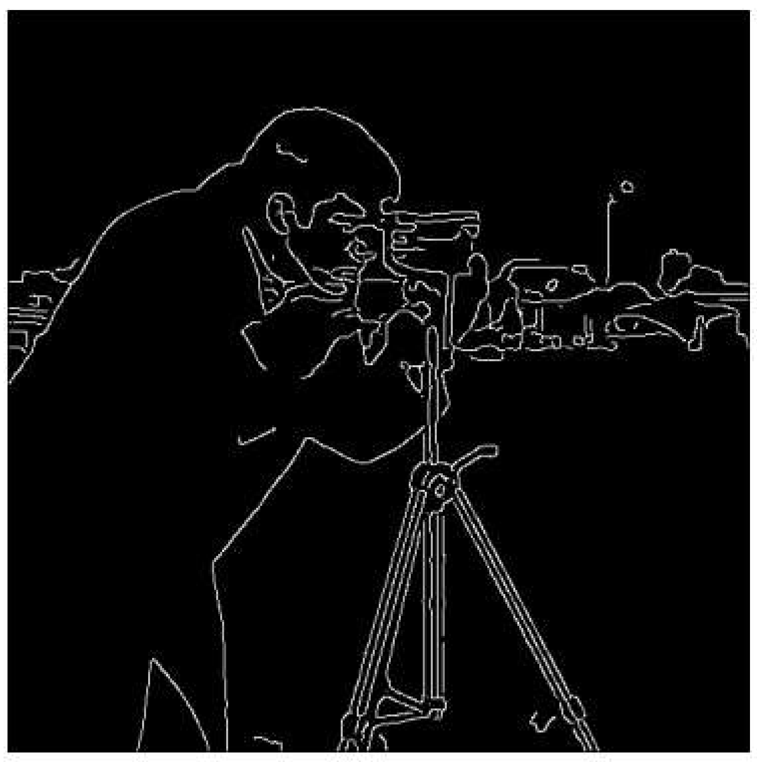
```
import matplotlib.pyplot as plt
import numpy as np
from skimage import io, data, color
from skimage.color import rgb2gray
from skimage.measure import find_contours
from skimage.filters import threshold_otsu
from skimage.restoration import denoise_tv_chambolle

image = io.imread('/content/robot.jpg')


robot_gray = rgb2gray(image)
thresh = threshold_otsu(robot_gray)
robot_binary = robot_gray > thresh
robot_contours = find_contours(robot_binary)

for contour in robot_contours[:5]:
    print(contour.shape)

def mark_contours(image):
    gray_image = rgb2gray(image)
    thresh = threshold_otsu(gray_image)
    binary_image = gray_image > thresh
    contours = find_contours(binary_image)
    return contours

def plot_image_contours(image):
    fig, (ax1, ax2) = plt.subplots(ncols=2, figsize=(9, 6))
    ax1.imshow(image, cmap=plt.cm.gray)
    ax2.imshow(image, cmap=plt.cm.gray)

    for contour in mark_contours(image):
        ax2.plot(contour[:, 1], contour[:, 0], linewidth=2, color="red")

    ax1.set_title("original image")
    ax2.set_title("Contours détectés")
    ax2.axis("off")

plot_image_contours(image)
```

```
(6193, 2)
(5, 2)
(7, 2)
(5, 2)
(7, 2)
```



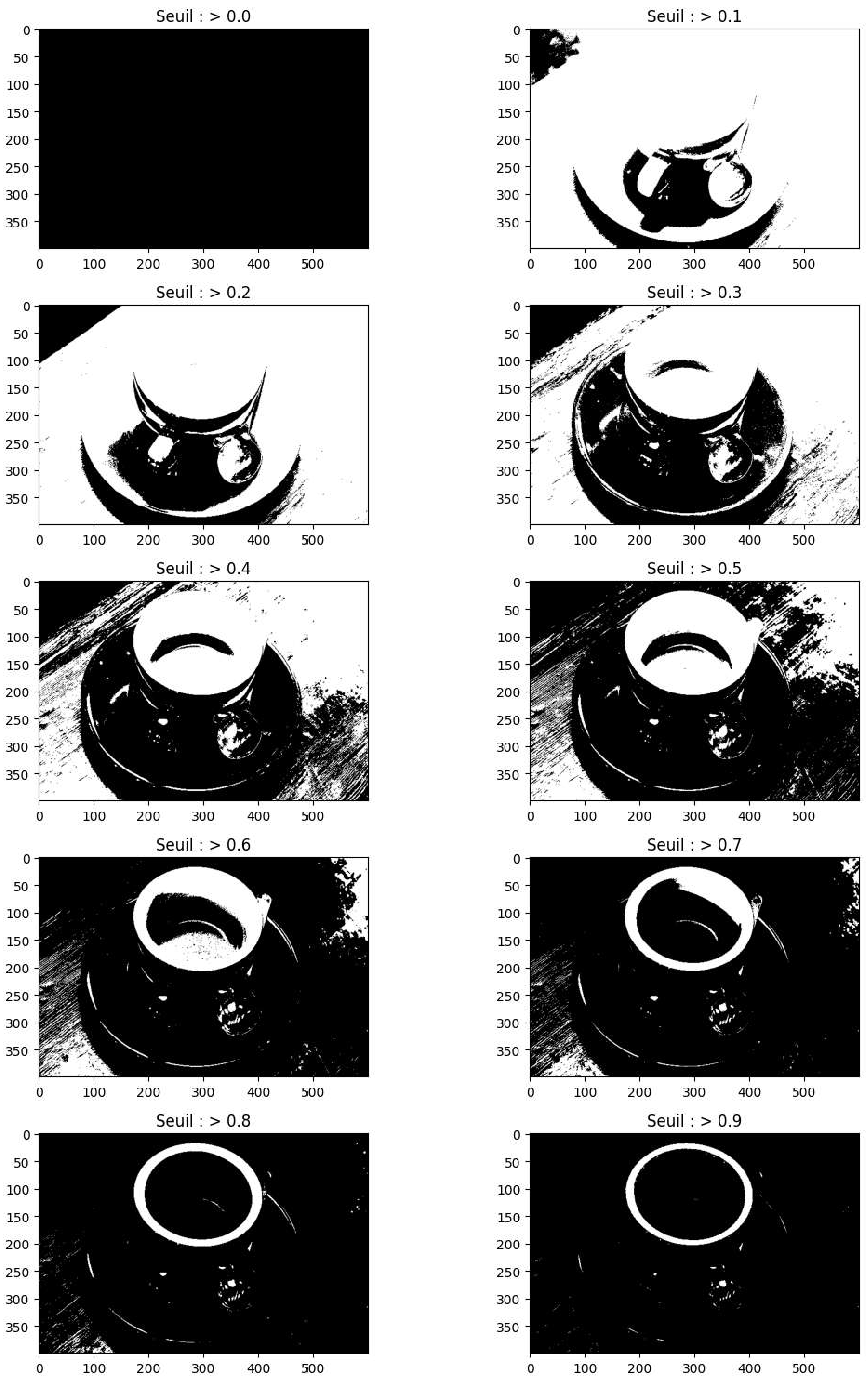
Exemple 2

```
from skimage import feature, data
import matplotlib.pyplot as plt

img = data.camera()

edges = feature.canny(img, sigma=2)

plt.imshow(edges, cmap='gray')
plt.axis('off')
plt.show()
```



3. Segmentaion

```
from skimage import data
from skimage.color import rgb2gray
import matplotlib.pyplot as plt

coffee = data.coffee()
gray_coffee = rgb2gray(coffee)

plt.figure(figsize=(15, 15))

for i in range(10):
    binarized_gray = (gray_coffee > i * 0.1) * 1

    plt.subplot(5, 2, i + 1)
    plt.title("Seuil : > " + str(round(i * 0.1, 1)))
    plt.imshow(binarized_gray, cmap='gray')
    plt.tight_layout()

plt.show()
```



4. Application : Introduction à YOLOv8

```
from ultralytics import YOLO
from skimage import io
import matplotlib.pyplot as plt
import numpy as np

model = YOLO('yolov8s.pt')

image = io.imread('/content/people.jpg')

results = model.predict(image)

firstResult = results[0]
firstResult.save(filename='predictFile.png')

resultImage = io.imread('predictFile.png')
resultImage=resultImage[:,:,::-1]

plt.imshow(resultImage)
plt.axis('off')
plt.show()
```

```
0: 448x640 7 persons, 1099.7ms
Speed: 10.4ms preprocess, 1099.7ms inference, 1.4ms postprocess per image at shape (1, 3, 448, 640)
```