Image Processing

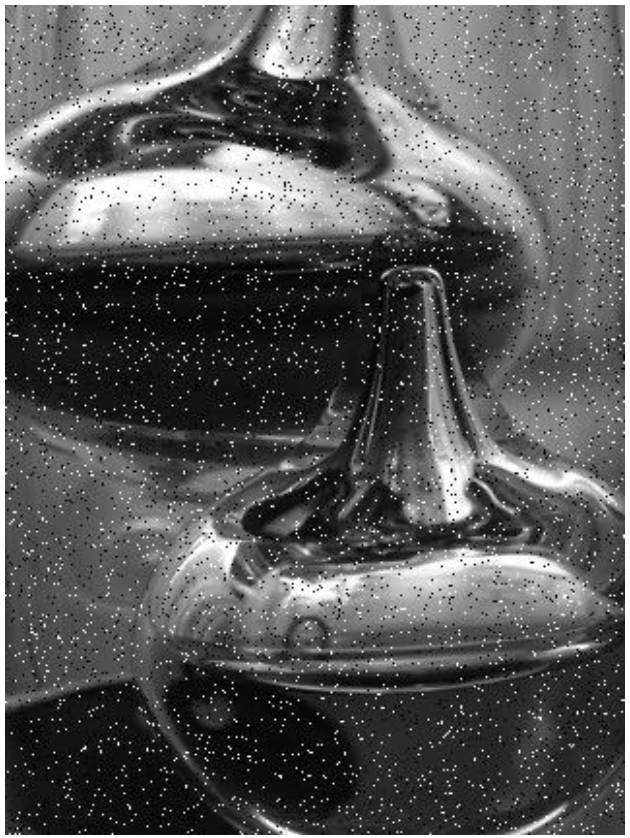
Lab (4) Notes

Autumn 2024-25

Smoothing (or blurring) Filters

Filter	Purpose
Median	Reduces salt-and-pepper* noise by replacing each pixel with the median of its neighborhood.
Mean	Averages neighboring pixels to create a simple blurring effect.
Gaussian	Reduces high-frequency noise, creating a general blurring effect.
Non-Local Means	Reduces noise while preserving details, suitable for low noise-to-signal ratio images.
Bilateral	Smooths while preserving edges, reducing noise without blurring edges.

Salt-and-pepper noise is a type of image noise where random pixels are either very bright or very dark.
This noise appears as small white and black dots scattered throughout the image.



An image with salt-and-pepper noise

Hands On: Median Filter

```
from skimage import data, io
from skimage import filters
import matplotlib.pyplot as plt
image = io.imread('Noise_salt_and_pepper.png', as_gray=True)
smoothed_image = filters.median(image)

titles = ['Original Image', 'Median Filtered Image']
fig, ax = plt.subplots(1, 2)
ax[0].imshow(image, cmap='gray')
ax[1].imshow(smoothed_image, cmap='gray')
for i in range(len(ax)):
    ax[i].axis('off')
    ax[i].set_title(titles[i])
plt.show()
```

Original Image

Median Filtered Image

Hands On: Mean, Gaussian Filters

```
from skimage import data, io, filters
from skimage.morphology import square
import matplotlib.pyplot as plt

footprint = square(3)
image = io.imread('Noise_salt_and_pepper.png', as_gray=True)

smoothed_median = filters.median(image, footprint)
smoothed_mean = filters.rank.mean(image, footprint) # Image should be uint8
smoothed_gaussian = filters.gaussian(image, sigma=2)

# Plotting
```

Original

Median (3)

Best performance with salt-and-pepper noise

Mean (3)

Gaussian (Sigma=2)

Edge Detection Filters

Filter	Purpose
Sobel	Highlights edges by calculating the gradient of image intensity in both horizontal and vertical directions.
Scharr	Similar to Sobel but provides better edge detection accuracy; especially useful for high-quality edges.
Prewitt	Detects edges by approximating the gradient, emphasizing vertical and horizontal edges.
Roberts	Detects edges using diagonal gradients, often resulting in sharper edges but more sensitive to noise.
Canny	A multi-step edge detection process (gradient calculation, non-maximum suppression, and hysteresis thresholding), producing highly accurate edges.

Hands On: Edge detection

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

image = rgb2gray(data.coffee())

sobel_edges = filters.sobel(image)
scharr_edges = filters.scharr(image)
prewitt_edges = filters.prewitt(image)
roberts_edges = filters.roberts(image)
canny_edges = feature.canny(image, sigma=1)
```

Original Image

Sobel Filter

Scharr Filter

Prewitt Filter

Roberts Filter

Canny Edge Detection

More on Edge filters

Sobel, Scharr, Prewitt: These filters work by detecting the gradient in intensity, producing a bright line where the gradient is steepest.

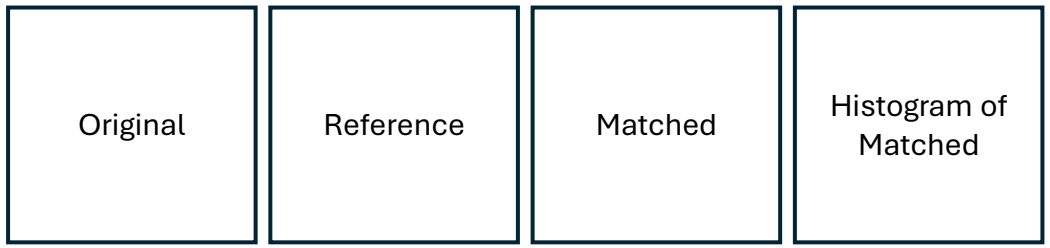
Roberts: Detects diagonal edges but can be more sensitive to noise.

Canny: The Canny method is a more complex edge detector, to produce a cleaner edge image.

Each filter has strengths suited to different applications, from simple edge highlights (Sobel) to more detailed edge maps (Canny).

Quiz: 5 questions 1 degree each.

1. Convert an RGB image into *grayscale* image. ***Don't plot***
2. Print the *shape*, *number of intensity levels* of both RGB and grayscale image.
3. Calculate the storage *size* of both images in *bits*.
4. Apply *histogram matching* on two grayscale images.
5. Use *exposure* to find the histogram of the matched image.



Thank You

Types of Noise

https://medium.com/@Coursesteach/computer-vision-part-14-common-types-of-noise-7e6507cc763c

https://en.wikipedia.org/wiki/Image_noise