Lab Report-08

Course title: Digital Image Processing Laboratory

Course code: CSE-406

4th Year 1st Semester Examination 2023

Date of Submission: 27 October 2024

Submitted to-

Dr. Morium Akter
Professor

Dr. Md. Golam Moazzam
Professor

Department of Computer Science and Engineering
Jahangirnagar University
Savar, Dhaka-1342

Sl	Class Roll	Exam Roll	Name
01	408	202220	Sudipta Singha

Department of Computer Science and Engineering
Jahangirnagar University
Savar, Dhaka, Bangladesh

Lab Report Title

Gaussian Noise Removal in Python

Introduction

Gaussian noise is a common type of statistical noise that follows a Gaussian distribution. It often occurs during image acquisition or transmission. Removing Gaussian noise can enhance image quality for further processing. One effective technique for noise removal is applying a Gaussian filter, which smoothens the image by averaging the pixel values within a local neighborhood, with more weight given to nearby pixels.

Python code

```python
# gaussian_noise_removal.py

import numpy as np
from PIL import Image
import matplotlib.pyplot as plt
from scipy.ndimage import gaussian_filter

def gaussian_noise_removal(input_path, output_path, sigma=1):
    img = Image.open(input_path)  # Open the image
    img_array = np.array(img)

    # Apply Gaussian filter to remove noise
    denoised_img_array = gaussian_filter(img_array,
       ↪ sigma=sigma)

    # Convert the denoised array to an image
    denoised_img =
       ↪ Image.fromarray(denoised_img_array.astype(np.uint8))

    # Save the denoised image
    denoised_img.save(output_path)

    # Display original and denoised images
    plt.figure(figsize=(10, 5))

    plt.subplot(1, 2, 1)
    plt.imshow(img)
    plt.title("Original Image")
    plt.axis("off")

    plt.subplot(1, 2, 2)
    plt.imshow(denoised_img)
    plt.title("Denoised Image (Gaussian Filter)")
    plt.axis("off")

    plt.tight_layout()
    plt.show()

# Example usage
if __name__ == "__main__":
    input_path = "input.jpg"
    output_path = "denoised_image.jpg"
    gaussian_noise_removal(input_path, output_path, sigma=2)
    print(f"Denoised image saved as {output_path}")
```

Input

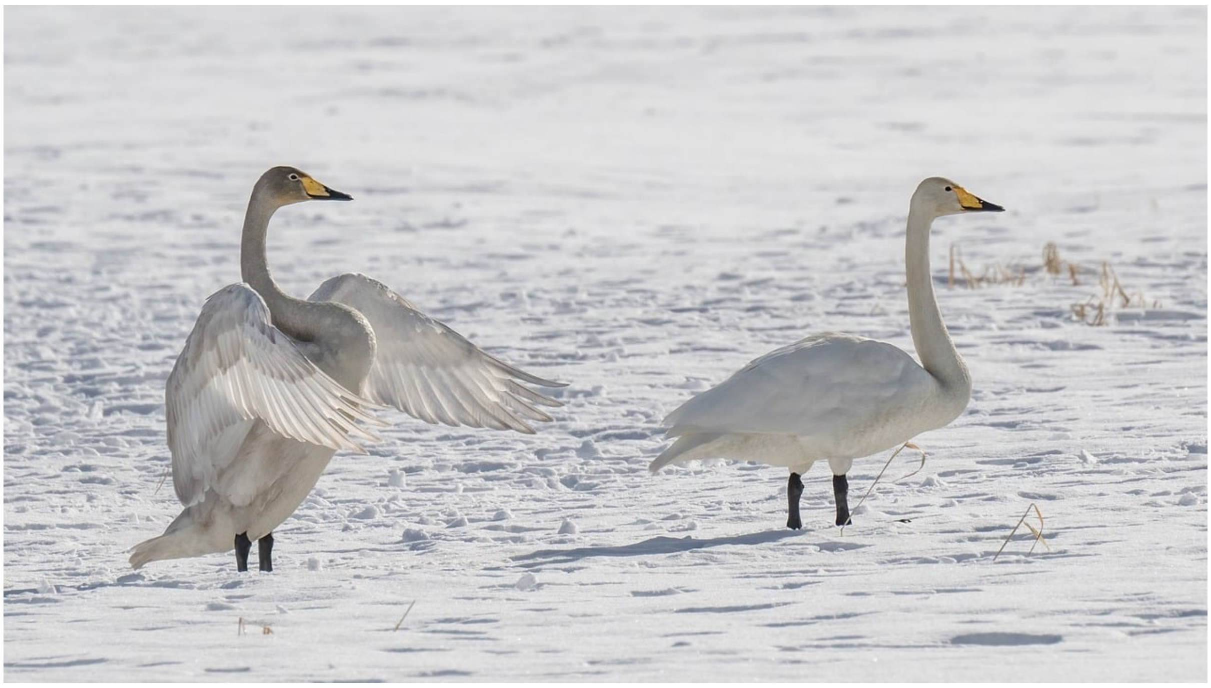


Figure 1:

Output

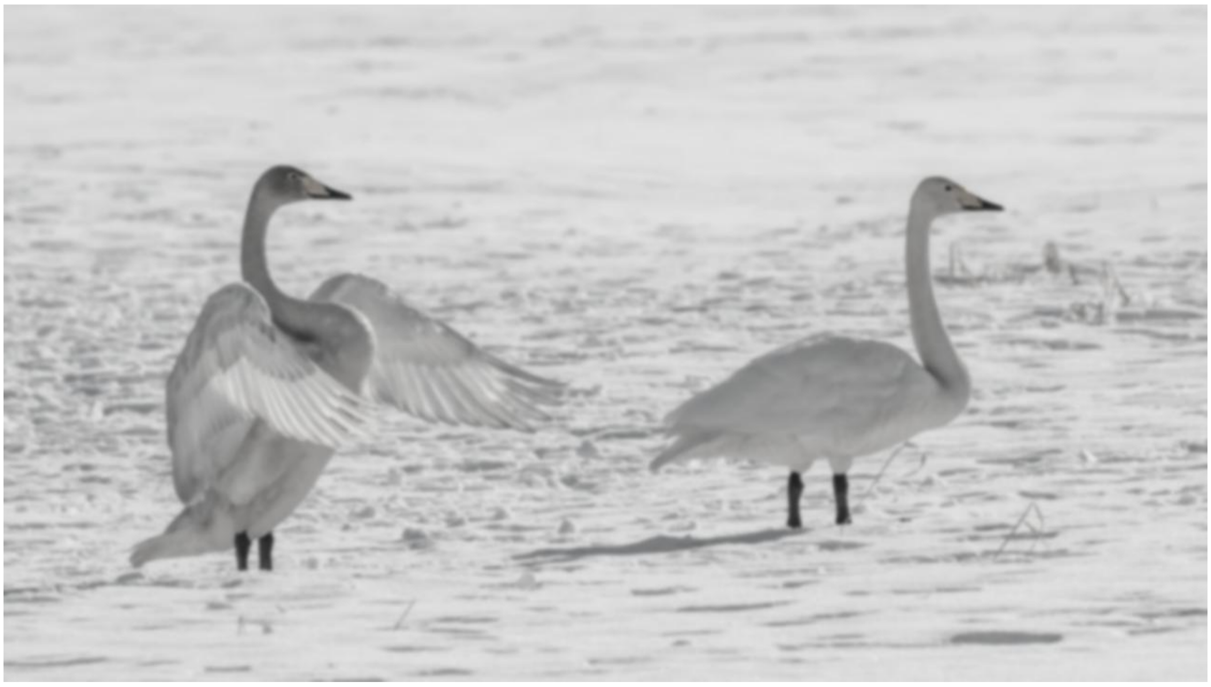


Figure 2:

Lab Report Title

Gaussian, Raleigh, and Erlang Noise Removal in Python

Introduction

In image processing, different types of noise can affect image quality. The Gaussian, Raleigh, and Erlang distributions are often used to model various noise patterns. While Gaussian noise is widely encountered, Raleigh and Erlang noise are less common but may appear in specific scenarios such as radar or communication systems. The task is to apply noise removal techniques suited to each distribution for improving image quality. In this case, we focus on using filters for Gaussian noise removal, while Raleigh and Erlang noise would typically require specialized filtering techniques.

Python code

```python
# gaussian_raleigh_erlang_noise_removal.py

import numpy as np
from PIL import Image
import matplotlib.pyplot as plt
from scipy.ndimage import gaussian_filter
from scipy.stats import rayleigh, erlang

def gaussian_noise_removal(img_array, sigma=1):
    """ Gaussian noise removal using Gaussian filter """
    return gaussian_filter(img_array, sigma=sigma)

def raleigh_noise_removal(img_array, shape=1, scale=1):
    """ Placeholder function for Raleigh noise removal """
    # Custom filter or denoising technique for Raleigh noise
      ↪ can be implemented here
    return gaussian_filter(img_array, sigma=2)

def erlang_noise_removal(img_array, shape=2, scale=1):
    """ Placeholder function for Erlang noise removal """
    # Custom filter or denoising technique for Erlang noise
      ↪ can be implemented here
    return gaussian_filter(img_array, sigma=3)

def apply_noise_removal(input_path, output_path,
   ↪ noise_type="gaussian"):
    img = Image.open(input_path)  # Open the image
    img_array = np.array(img)

    # Apply denoising based on the selected noise type
    if noise_type == "gaussian":
        denoised_img_array =
          ↪ gaussian_noise_removal(img_array)
    elif noise_type == "raleigh":
        denoised_img_array = raleigh_noise_removal(img_array)
    elif noise_type == "erlang":
        denoised_img_array = erlang_noise_removal(img_array)
    else:
        print("Invalid noise type!")
        return

    # Convert the denoised array to an image
    denoised_img =
      ↪ Image.fromarray(denoised_img_array.astype(np.uint8))

    # Save the denoised image
    denoised_img.save(output_path)

    # Display original and denoised images
    plt.figure(figsize=(10, 5))

    plt.subplot(1, 2, 1)
    plt.imshow(img)
    plt.title("Original Image")
    plt.axis("off")

    plt.subplot(1, 2, 2)
    plt.imshow(denoised_img)
    plt.title(f"Denoised Image ({noise_type.capitalize()}
      ↪ Filter)")
    plt.axis("off")

    plt.tight_layout()
    plt.show()

# Example usage
if __name__ == "__main__":
    input_path = "input.jpg"
    output_path = "denoised_image_gaussian.jpg"
    apply_noise_removal(input_path, output_path,
      ↪ noise_type="gaussian")
    apply_noise_removal(input_path, output_path,
      ↪ noise_type="raleigh")
    apply_noise_removal(input_path, output_path,
      ↪ noise_type="erlang")
    # You can replace "gaussian" with "raleigh" or "erlang"
      ↪ to apply respective filters
    print(f"Denoised image saved as {output_path}")
```

Input

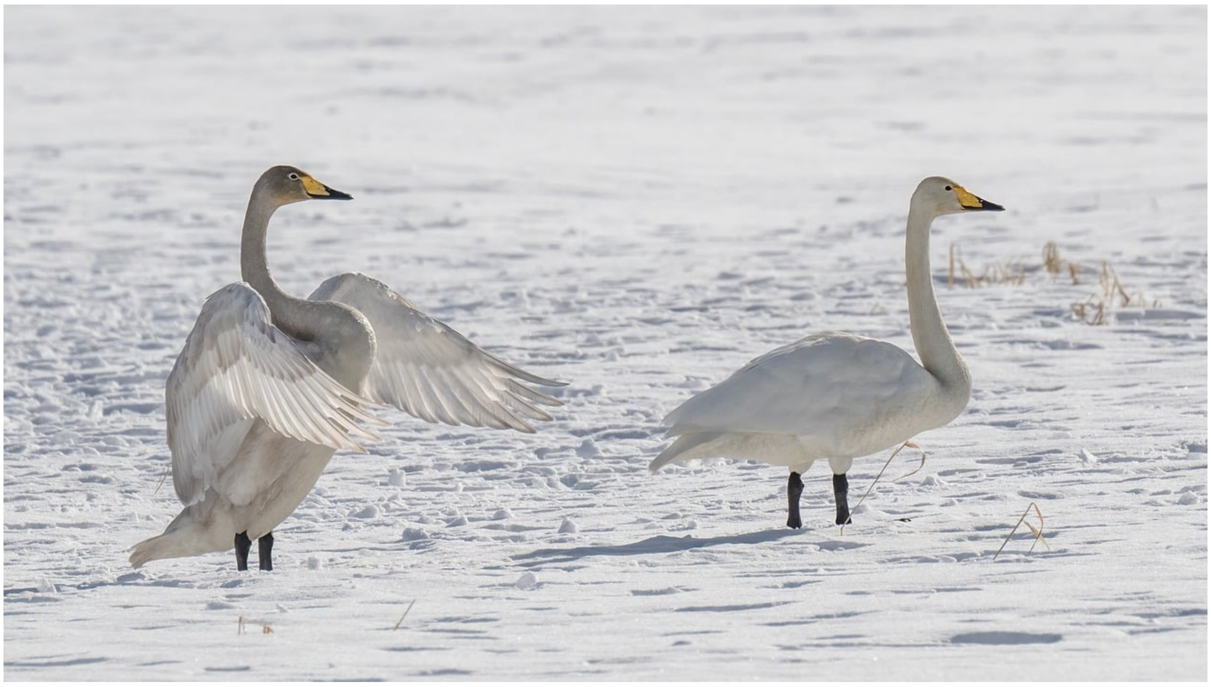
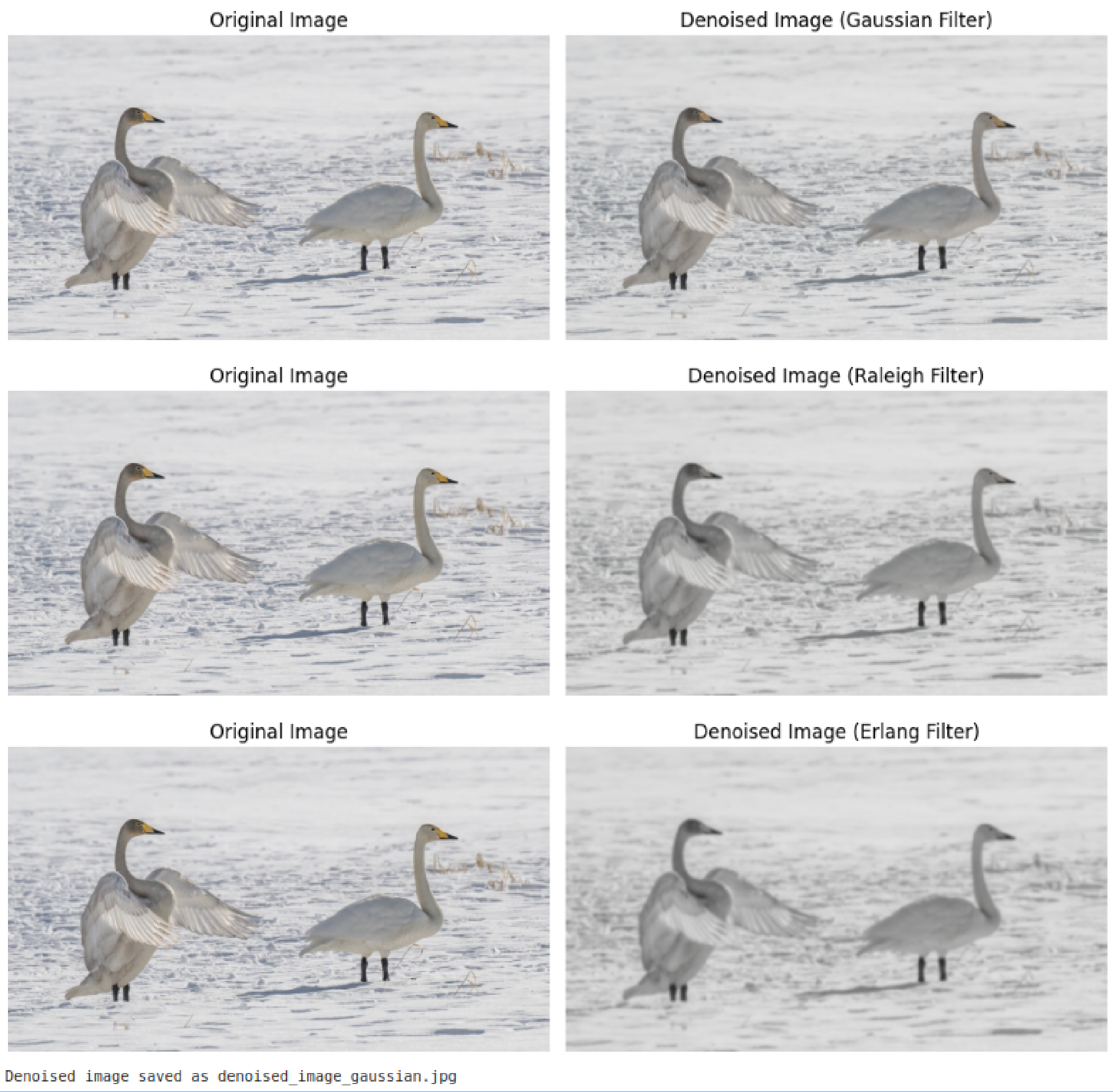


Figure 3:

Output


Original Image
Denoised Image (Gaussian Filter)
Original Image
Denoised Image (Raleigh Filter)
Original Image
Denoised Image (Erlang Filter)

```
Denoised image saved as denoised_image_gaussian.jpg
```

Figure 4:

Lab Report Title

Periodic Noise Removal in Python

Introduction

Periodic noise is a type of noise that has a repeating pattern, often arising due to various environmental or sensor-related factors. It typically appears as repetitive interference in the frequency domain. To remove periodic noise, techniques like Fourier Transform and Notch Filter can be applied. The Fourier Transform converts an image from the spatial domain to the frequency domain, where periodic noise can be identified and removed by filtering out specific frequencies corresponding to the noise.

Python code

```
# periodic_noise_removal.py

import numpy as np
from PIL import Image
import matplotlib.pyplot as plt
from scipy.fft import fft2, ifft2, fftshift

def periodic_noise_removal(input_path, output_path,
   ↪ radius=30):
    img = Image.open(input_path)  # Open the image
    img_array = np.array(img)

    # Convert to grayscale if the image is colored
    if img_array.ndim == 3:
        img_array = img_array.mean(axis=-1).astype(np.uint8)

    # Apply Fourier Transform
    f = fft2(img_array)
    fshift = fftshift(f)

    # Create a mask to filter out the periodic noise in the
       ↪ frequency domain
    rows, cols = img_array.shape
    center = (rows // 2, cols // 2)
    mask = np.ones((rows, cols))

    # Set a circular region to zero in the frequency domain
       ↪ (notch filter)
    x, y = np.ogrid[:rows, :cols]
    mask_area = (x - center[0])**2 + (y - center[1])**2 <=
       ↪ radius**2
    mask[mask_area] = 0

    # Apply the mask to filter out the periodic noise
    fshift_filtered = fshift * mask

    # Perform inverse Fourier Transform to get the filtered
       ↪ image back
    f_ishift = np.fft.ifftshift(fshift_filtered)
    img_back = np.abs(ifft2(f_ishift))

    # Convert back to uint8 and save the output image
    img_back = np.uint8(np.clip(img_back, 0, 255))
    denoised_img = Image.fromarray(img_back)
    denoised_img.save(output_path)

    # Display original and denoised images
    plt.figure(figsize=(10, 5))

    plt.subplot(1, 2, 1)
    plt.imshow(img_array, cmap='gray')
    plt.title("Original Image")
    plt.axis("off")

    plt.subplot(1, 2, 2)
    plt.imshow(img_back, cmap='gray')
    plt.title("Denoised Image (Periodic Noise Removal)")
    plt.axis("off")

    plt.tight_layout()
    plt.show()

# Example usage
if __name__ == "__main__":
    input_path = "input.jpg"
    output_path = "denoised_image_periodic_noise.jpg"
    periodic_noise_removal(input_path, output_path,
       ↪ radius=30)
    print(f"Denoised image saved as {output_path}")
```

Input

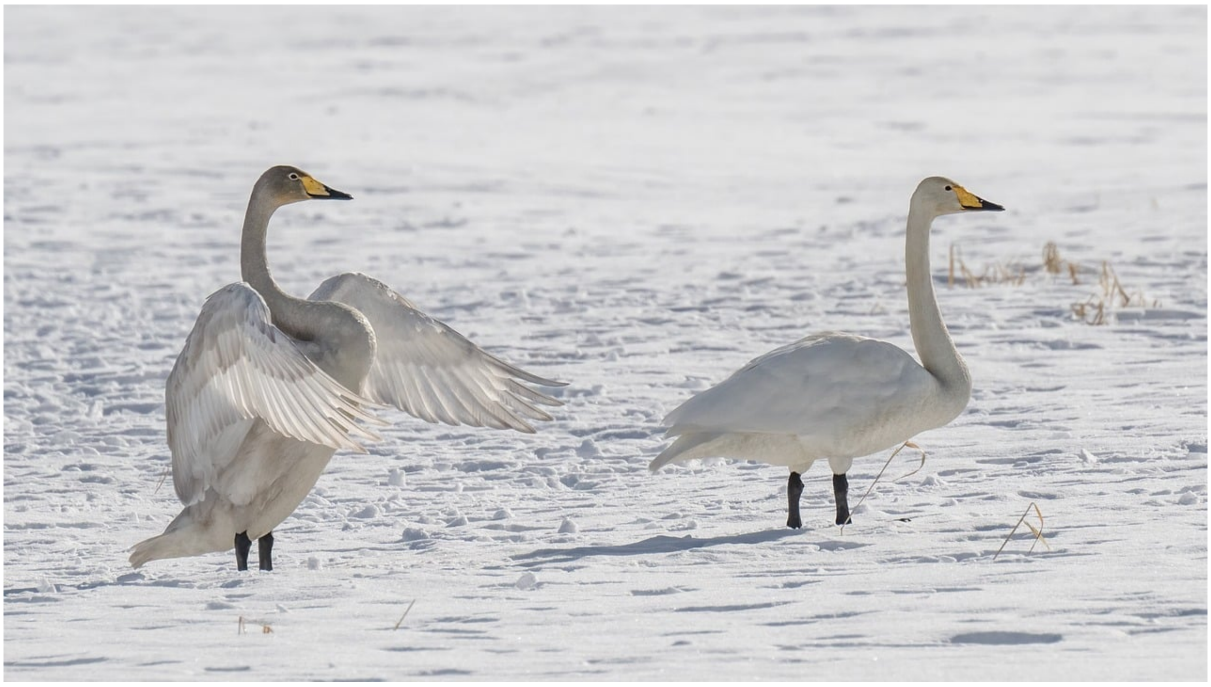


Figure 5:

Output

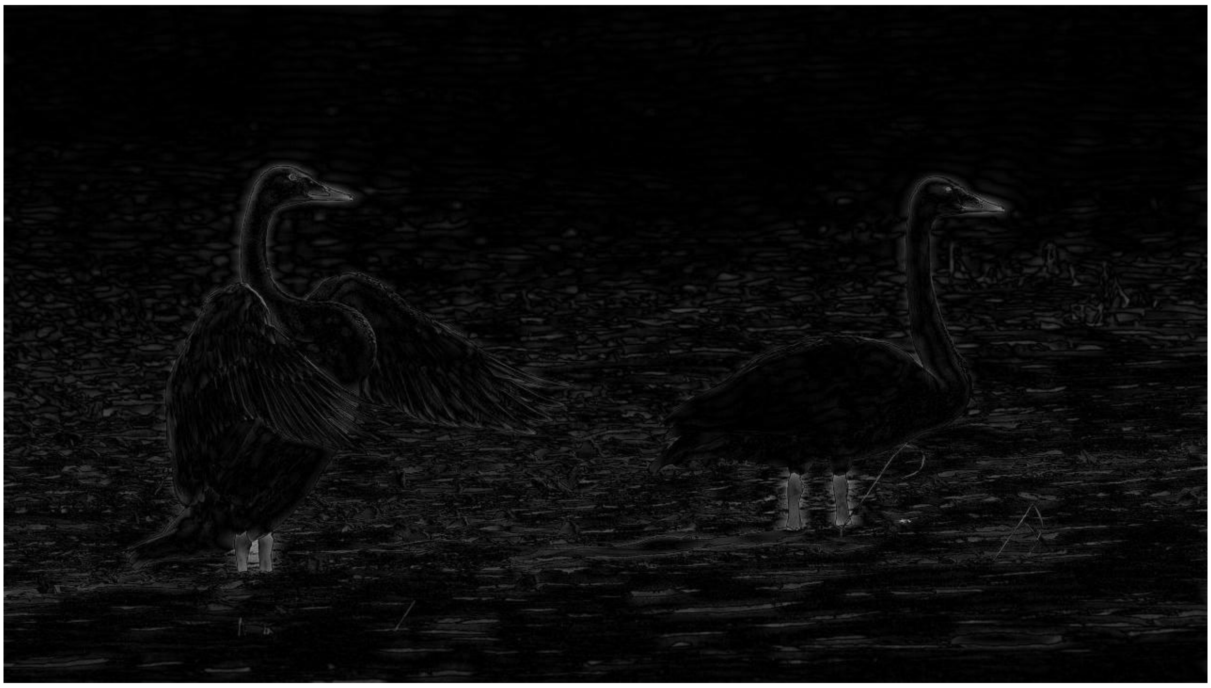


Figure 6:

Lab Report Title

Salt and Pepper Noise Removal in Python

Introduction

Salt and pepper noise is a type of impulse noise that appears as random white and black pixels scattered throughout the image. This type of noise can be removed using various filtering techniques, with the Median filter being one of the most effective methods. The median filter works by replacing each pixel's value with the median value of its neighboring pixels, effectively removing both salt and pepper noise without blurring the edges significantly.

Python code

```
# salt_and_pepper_noise_removal.py

import numpy as np
from PIL import Image
import matplotlib.pyplot as plt
from scipy.ndimage import median_filter

def salt_and_pepper_noise_removal(input_path, output_path,
   ↪ size=3):
    img = Image.open(input_path)  # Open the image
    img_array = np.array(img)

    # Apply median filter to remove salt and pepper noise
    denoised_img_array = median_filter(img_array, size=size)

    # Convert the denoised array to an image
    denoised_img =
       ↪ Image.fromarray(denoised_img_array.astype(np.uint8))

    # Save the denoised image
    denoised_img.save(output_path)

    # Display original and denoised images
    plt.figure(figsize=(10, 5))

    plt.subplot(1, 2, 1)
    plt.imshow(img_array)
    plt.title("Original Image")
    plt.axis("off")

    plt.subplot(1, 2, 2)
    plt.imshow(denoised_img)
    plt.title("Denoised Image (Salt and Pepper Removal)")
    plt.axis("off")

    plt.tight_layout()
    plt.show()

# Example usage
if __name__ == "__main__":
    input_path = "input.jpg"
    output_path = "denoised_image_salt_pepper.jpg"
    salt_and_pepper_noise_removal(input_path, output_path,
      ↪ size=3)
    print(f"Denoised image saved as {output_path}")
```

Input

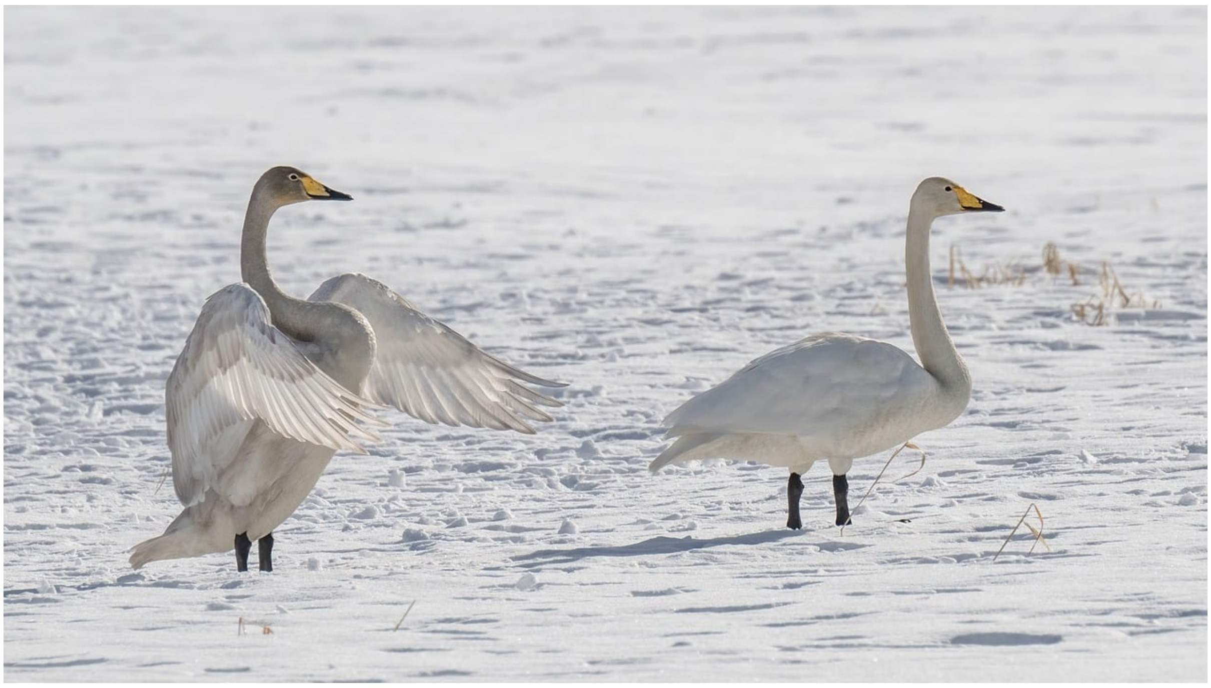


Figure 7:

Output

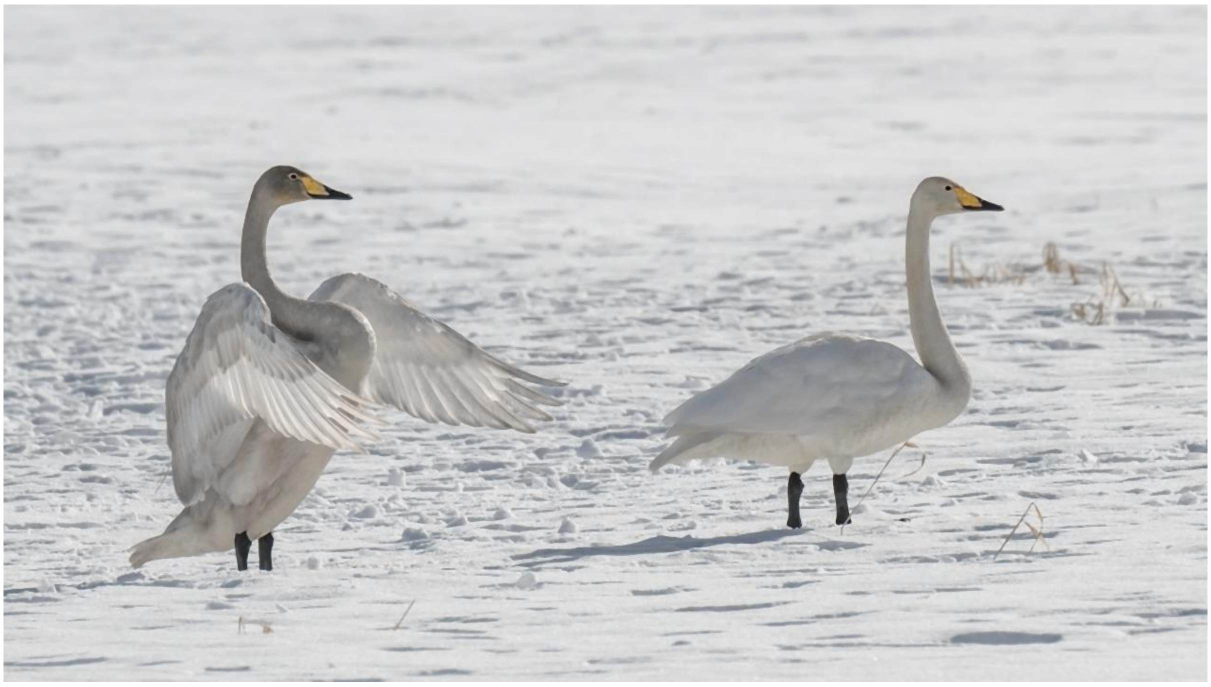


Figure 8: