Lab Report-09

Course title: Digital Image Processing Laboratory

Course code: CSE-406

4th Year 1st Semester Examination 2023

Date of Submission: 03 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

Circle Detection Using Hough Transform in Python

Introduction

The Hough Transform is a popular technique used in image processing for detecting shapes like lines, circles, and other parametric curves. For circle detection, the Hough Circle Transform algorithm is employed, where circles are represented in a parameterized form using their center (x, y) coordinates and radius (r). The algorithm works by transforming the image into a parameter space and accumulating votes for potential circle centers and radii. This method is particularly useful for detecting circular shapes in noisy images.

Python code

```python
# circle_detection_hough_transform.py

import cv2
import numpy as np
import matplotlib.pyplot as plt

def detect_circles(input_path, output_path, min_radius=10,
   ↪ max_radius=100):
    # Read the image
    img = cv2.imread(input_path, cv2.IMREAD_COLOR)

    # Convert to grayscale
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

    # Apply Gaussian blur to reduce noise
    blurred = cv2.GaussianBlur(gray, (5, 5), 0)

    # Detect circles using Hough Transform
    circles = cv2.HoughCircles(
        blurred,
        cv2.HOUGH_GRADIENT,
        dp=1.2,
        minDist=20,
        param1=50,
        param2=30,
        minRadius=min_radius,
        maxRadius=max_radius
    )

    if circles is not None:
        # Convert coordinates and radius to integers
        circles = np.uint16(np.around(circles))

        # Draw the circles on the original image
        for circle in circles[0, :]:
            center = (circle[0], circle[1])  # center of the
               ↪ circle
            radius = circle[2]  # radius of the circle
            cv2.circle(img, center, radius, (0, 255, 0), 2)
               ↪ # draw the circle
            cv2.circle(img, center, 2, (0, 0, 255), 3)  #
               ↪ draw the center

        # Save the output image
        cv2.imwrite(output_path, img)

    # Display the result
    plt.figure(figsize=(10, 5))
    plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
    plt.title("Detected Circles Using Hough Transform")
    plt.axis("off")
    plt.show()

# Example usage
if __name__ == "__main__":
    input_path = "input.jpg"  # Input image path
    output_path = "output_circles.jpg"  # Output image path
    detect_circles(input_path, output_path, min_radius=10,
       ↪ max_radius=100)
    print(f"Circles detected and saved as {output_path}")
```

Input

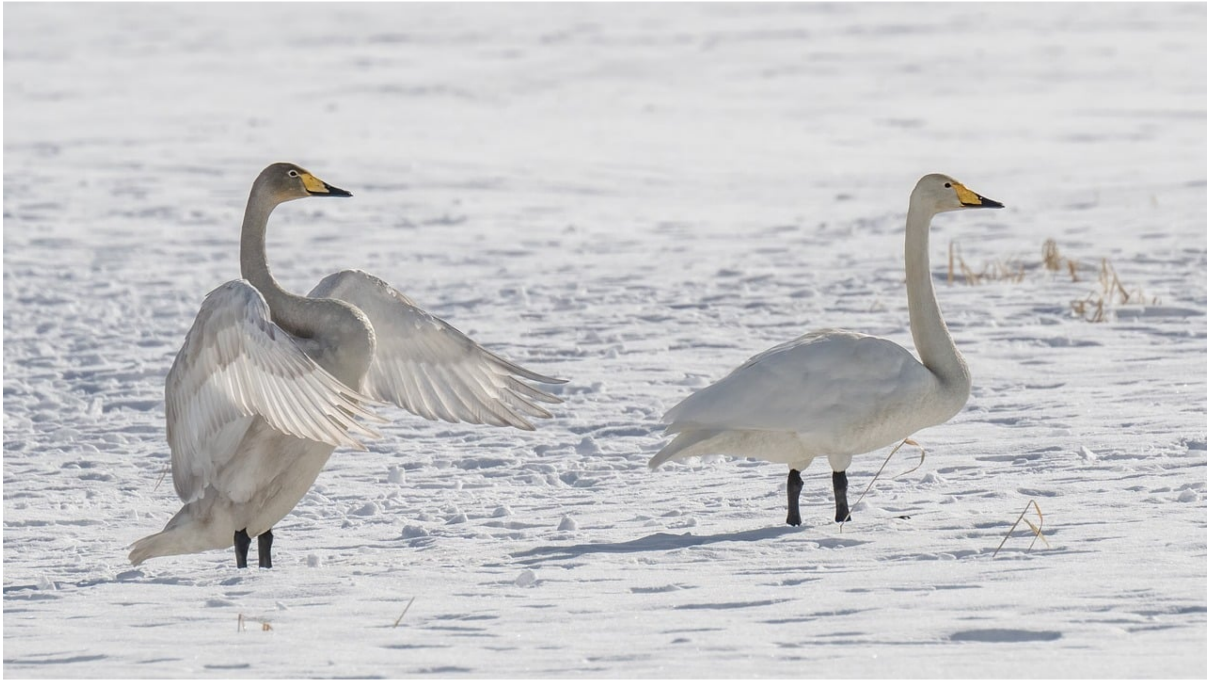

Figure 1:

Output

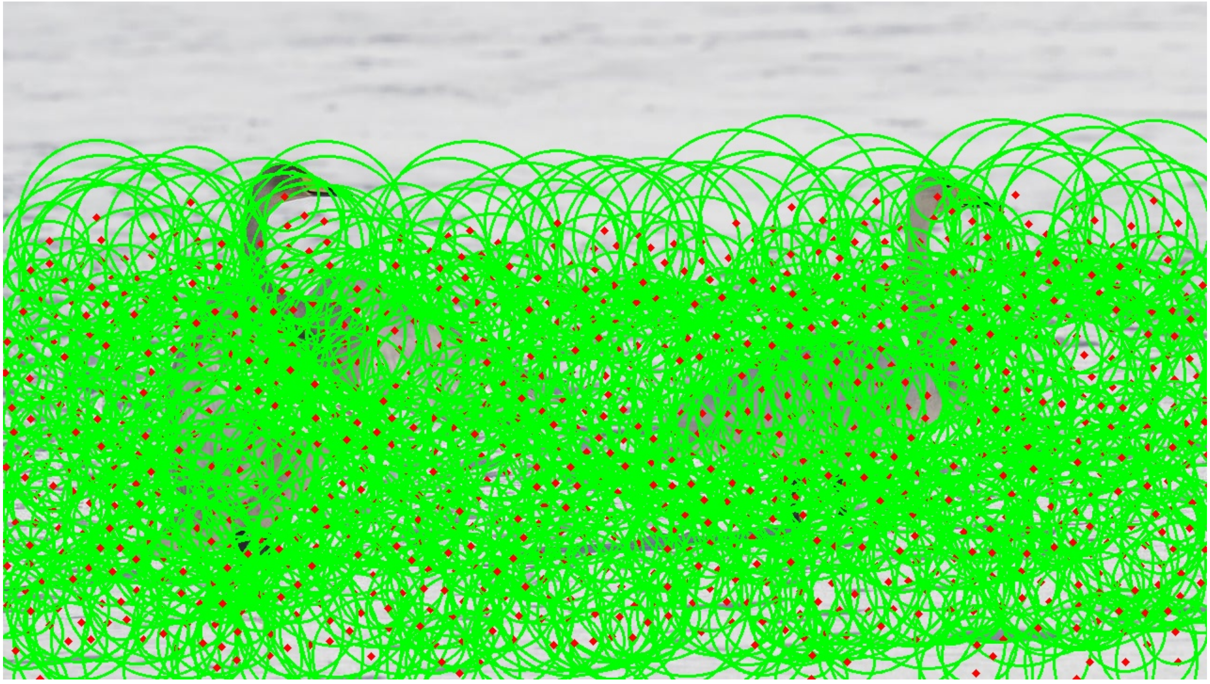

Figure 2:

Lab Report Title

Line Detection Using Hough Transform in Python

Introduction

The Hough Transform is widely used in image processing for detecting lines, circles, and other shapes in images. For line detection, the Hough Line Transform maps each point in the image space to a line in the Hough parameter space. This transform works by identifying points that align along a straight path, making it effective for detecting lines even in noisy images. It is especially useful in applications where lines or edges need to be identified, such as road lane detection or object contour detection.

Python code

```
# line_detection_hough_transform.py

import cv2
import numpy as np
import matplotlib.pyplot as plt

def detect_lines(input_path, output_path, threshold=100):
    # Read the image
    img = cv2.imread(input_path)

    # Convert to grayscale
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

    # Apply Canny edge detector to highlight edges in the
       ↪ image
    edges = cv2.Canny(gray, 50, 150, apertureSize=3)

    # Detect lines using the Hough Line Transform
    lines = cv2.HoughLines(edges, rho=1, theta=np.pi / 180,
       ↪ threshold=threshold)

    # Draw the detected lines on the original image
    if lines is not None:
        for line in lines:
            rho, theta = line[0]
            a = np.cos(theta)
            b = np.sin(theta)
            x0 = a * rho
            y0 = b * rho
            x1 = int(x0 + 1000 * (-b))
            y1 = int(y0 + 1000 * (a))
            x2 = int(x0 - 1000 * (-b))
            y2 = int(y0 - 1000 * (a))
            cv2.line(img, (x1, y1), (x2, y2), (0, 255, 0), 2)

    # Save the output image
    cv2.imwrite(output_path, img)

    # Display the result
    plt.figure(figsize=(10, 5))
    plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
    plt.title("Detected Lines Using Hough Transform")
    plt.axis("off")
    plt.show()

# Example usage
if __name__ == "__main__":
    input_path = "input.jpg"  # Input image path
    output_path = "output_lines.jpg"  # Output image path
    detect_lines(input_path, output_path, threshold=100)
    print(f"Lines detected and saved as {output_path}")
```

Input

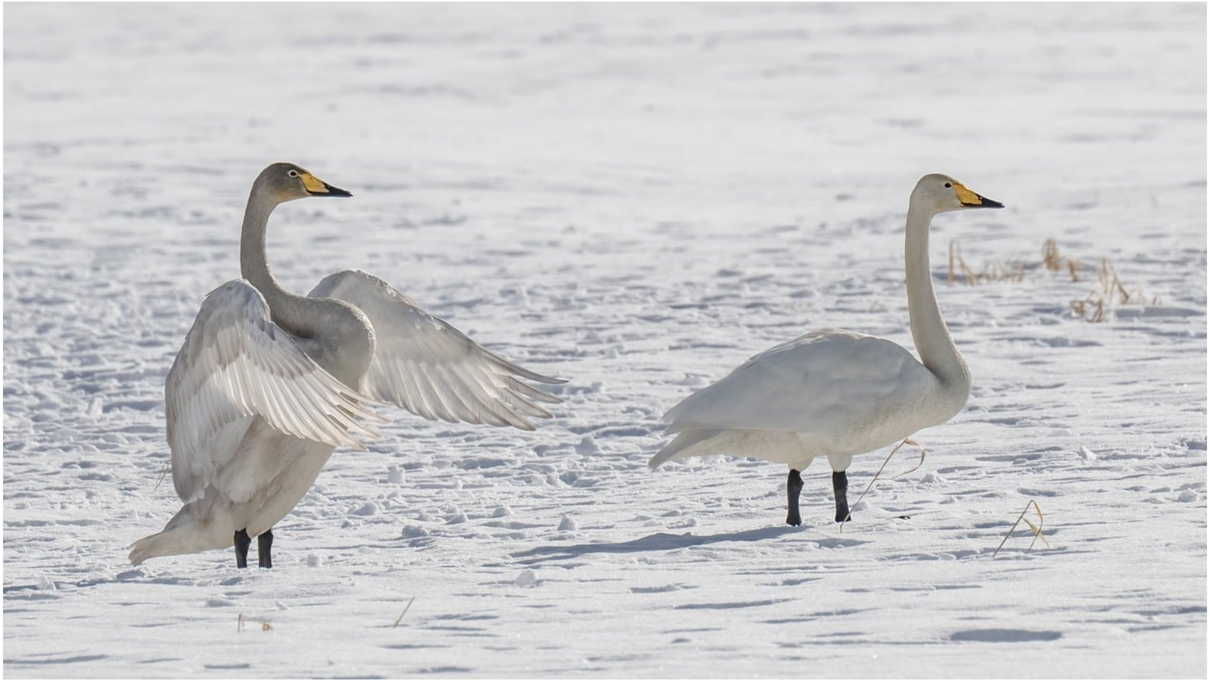

Figure 3:

Output

Figure 4: