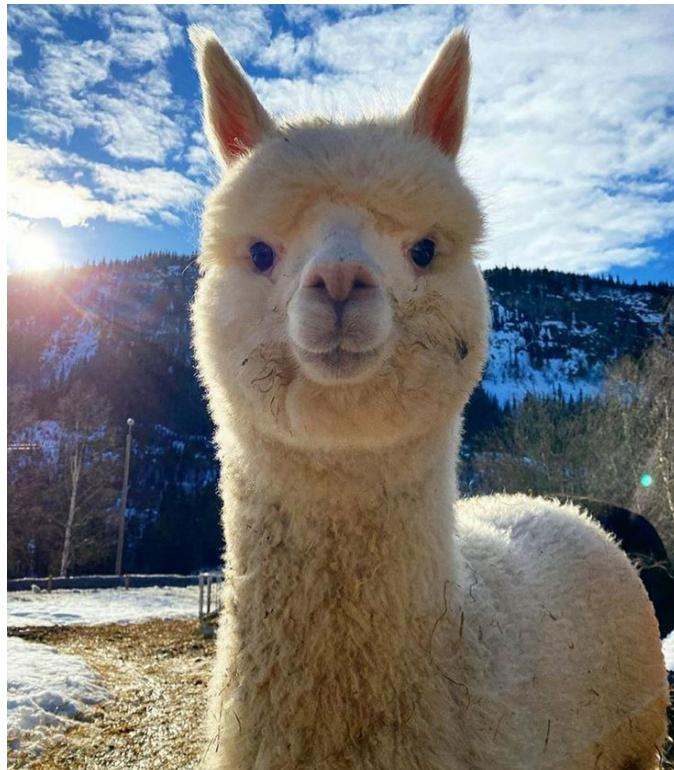


Author: Amirhossein Gholizadeh

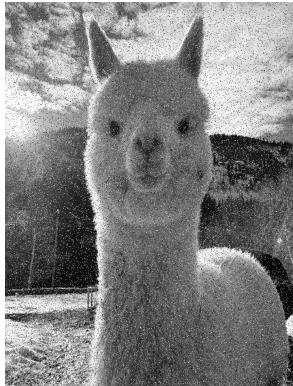
Course: Image Processing

Chapter: 2

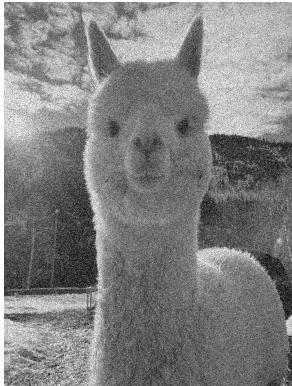
Original Image



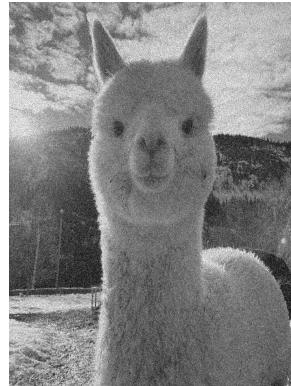
Denoised Images



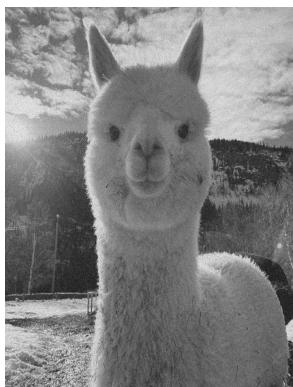
$k=1$



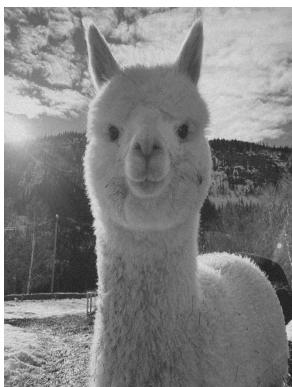
$k=5$



$k=10$



$k=50$



$k=100$



$k=500$

Sample Statistics

Sample Groups	Average	Average Variance
1	126.59023300909149	4897.521901236316
5	126.83193209994272	2958.645411010458
10	126.82020201538559	2463.6412659996254
50	126.73290647847926	2182.3855954470996
100	126.72608921419038	2122.368624948141
500	126.72036704812768	2176.919802433842

main.py

```
1 """
2 Image Processing Project: Noise Analysis and Denoising
3
4 This project aims to analyze the effects of salt-and-pepper noise on an image and perform denoising operations
5 using an averaging technique. The project involves the following steps:
6
7 1. Read an original image.
8 2. Generate multiple noisy samples by adding varying amounts of salt-and-pepper noise.
9 3. Group the noisy samples into different sample sizes (1, 5, 10, 50, 100, 500).
10 4. For each sample size, calculate the average and variance of the noisy samples.
11 5. Denoise the samples by averaging them within each sample group.
12 6. Save the denoised images and the calculated statistics to a CSV file.
13 7. Generate a PDF report with the original image, denoised images, and sample statistics.
14
15 Usage:
16     1. Ensure that the required dependencies are installed.
17     2. Place the original image file named "original.jpg" in the project directory.
18     3. Run the main.py script.
19     4. The noisy samples will be generated in the "noisy_samples" folder.
20     5. The denoised images will be saved in the "result" folder.
21     6. The CSV file with statistics will be saved as "result/results.csv".
22     7. The PDF report will be saved as "result/results.pdf".
23
24 Note: This project is part of an Image Processing class exercise and is intended for educational purposes.
25
26 Author: Amirhossein Gholizadeh
27 """
28
29 import cv2
30 import numpy as np
31 import os
32 import csv
33 from noise_gen import add_salt_and_pepper_noise
34 from noise_var_calc import calculate_average_and_variance
35 from report_gen import generate_pdf_report
36
37 def main() -> None:
38     # Reading the image
39     image_path = "original.jpg"
40     image = cv2.imread(image_path)
41
42     # Create directories to save noisy samples and results
43     os.makedirs("noisy_samples", exist_ok=True)
44     os.makedirs("result", exist_ok=True)
45
46     # Generate noisy samples
47     samples = 500
48     print(f"Generating {samples} noisy images...")
49     for i in range(samples):
50         noise_amount = np.random.uniform(0.01, 0.5)
51         noisy_samples = add_salt_and_pepper_noise(image, amount=noise_amount)
52         cv2.imwrite(f"noisy_samples/noisy_sample_{i}.jpg", noisy_samples)
53
54     # Select sample groups and calculate statistics
55     sample_sizes = [1, 5, 10, 50, 100, 500]
56     results = []
57
58     print("Averaging operation started...")
59     # For each value of k (sample group size), perform the following steps.
60     for k in sample_sizes:
61         #Read the corresponding number of noisy samples (based on k). Each sample is loaded as a grayscale image.
62         selected_samples = [cv2.imread(f"noisy_samples/noisy_sample_{i}.jpg", cv2.IMREAD_GRAYSCALE) for i in range(k)]
63         # Calculate the average and variance for this group of noisy samples.
64         avg, var = calculate_average_and_variance(np.mean(selected_samples, axis=0))
65         results.append((k, avg, var))
66
67         # Create resulting image for this sample group by averaging the selected samples
68         resulting_image = np.mean(selected_samples, axis=0) # Based on the formula: g(x, y) = (1/k) * Σ(i=1 to k) gi(x, y)
69         cv2.imwrite(f"result/result_for_{k}_samples.jpg", resulting_image)
70         print(f"Saved the results for {k} samples in \"result\" folder")
71
72     # Save results to a csv file
73     csv_filename = "result/results.csv"
74     # Write the sample group size, average, and average variance for each group.
75     with open(csv_filename, "w", newline="") as csvfile:
76         writer = csv.writer(csvfile)
77         writer.writerow(["Sample Groups", "Average", "Average Variance"])
78         for row in results:
79             writer.writerow(row)
80
81     generate_pdf_report(sample_sizes)
82     print(f"Finished the operation successfully. To see the results please check the \"result\" folder")
83
84 if __name__ == "__main__":
85     main()
```

noise_gen.py

```
● ● ●
1 import numpy as np
2
3 def add_salt_and_pepper_noise(image, amount) -> list:
4     """
5         Adds salt-and-pepper noise to an image.
6         :param image: Input image
7         :param amount: Proportion of image pixels to replace with noise
8         :return: Image with salt-and-pepper noise
9     """
10
11     # Converts the input image to a floating-point format (values between 0.0 and 1.0).
12     # Divides each pixel value by 255 to normalize it to the [0, 1] range.
13     image_float = np.float32(image) / 255.0
14
15     # Creates random values for each pixel in the image.
16     # These random values determine whether a pixel becomes "salt" (white) or "pepper" (black).
17     random_values = np.random.rand(*image.shape[:2])
18
19     # Set pixels to 1 (salt) or 0 (pepper) based on random values
20     image_float[random_values < (amount / 2)] = 0 # If a random value is less than half of the specified amount, set the pixel value to 0 (black) for "pepper."
21     image_float[random_values > (1 - amount / 2)] = 1 # If a random value is greater than (1 - half of the amount), set the pixel value to 1 (white) for "salt."
22
23     # Converts the modified floating-point image back to an 8-bit format (values between 0 and 255).
24     noisy_image = np.uint8(image_float * 255)
25
26     return noisy_image
```

noise_var_calc.py



```
1 import numpy as np
2
3 def calculate_average_and_variance(image) -> tuple[float, float]:
4     """
5         Calculates average and variance for an image based on these formulas:
6         -  $E[g(x, y)] = f(x, y)$ 
7         -  $\sigma^2 g(x, y) = (1/k) * \sigma^2 \eta(x, y)$ 
8     :param image: Input image
9     :return: Average and variance
10    """
11    # Computes the total number of pixels in the image.
12    # image.shape[0] gives the height (number of rows), and image.shape[1] gives the width (number of columns).
13    total_pixels = image.shape[0] * image.shape[1]
14
15    # Computes the average pixel value (mean) for the entire image.
16    average_value = np.sum(image) / total_pixels
17    # Computes the sum of squared differences between each pixel value and the average.
18    squared_diff_sum = np.sum((image - average_value) ** 2)
19    # Computes the variance of pixel values in the image.
20    variance_value = squared_diff_sum / total_pixels
21    return average_value, variance_value
```

report_gen.py

```
 1  from reportlab.lib.pagesizes import landscape, A5
 2  from reportlab.platypus import SimpleDocTemplate, Image, Paragraph, Spacer, Table, TableStyle
 3  from reportlab.lib.styles import getSampleStyleSheet
 4  from reportlab.lib.units import inch
 5  import csv
 6
 7  def add_snapshot():
 8      from PyPDF2 import PdfReader, PdfWriter
 9      # Open the existing PDF
10      existing_pdf = PdfReader("result/report.pdf")
11      pdf_writer = PdfWriter()
12
13      # Add all pages from the existing PDF to the writer
14      for page in existing_pdf.pages:
15          pdf_writer.add_page(page)
16
17      # Open the PDF file to append
18      pdf_to_append = PdfReader("snapshots/snapshots.pdf")
19
20      # Append pages from the new PDF to the writer
21      for page in pdf_to_append.pages:
22          pdf_writer.add_page(page)
23
24      # Write out the combined PDF
25      with open("result/report.pdf", "wb") as out_file:
26          pdf_writer.write(out_file)
27
28  def generate_pdf_report(samples) -> None:
29      """
30          Generate a PDF report with the original image, denoised images, and sample statistics.
31
32          Args:
33              samples (list): List of sample sizes used for denoising.
34
35          This function creates a PDF report using the ReportLab library. The report includes the following elements:
36              - Original image
37              - Denoised images for each sample size, grouped in rows of 3
38              - Titles for each denoised image indicating the sample size
39
40          The PDF report is saved as "result/report.pdf".
41      """
42
43      print("Generating pdf report...")
44
45      # Set up styles
46      styles = getSampleStyleSheet()
47      title_style = styles["Heading2"]
48      subtitle_style = styles["Heading3"]
49
50      # Set up the PDF document
51      doc = SimpleDocTemplate("result/report.pdf", pagesize=landscape(A5), topMargin=0*inch, bottomMargin=0*inch)
52      elements = []
53
54      # Add the original image with title
55      original_image = Image("original.jpg", width=3.5*inch, height=4*inch)
56      elements.append(Paragraph("Author: Amirhossein Gholizadeh"))
57      elements.append(Paragraph("Student Num: 4021119008"))
58      elements.append(Paragraph("Course: Image Processing"))
59      elements.append(Paragraph("Chapter: 2"))
60      elements.append(Paragraph("Original Image", title_style))
61      elements.append(original_image)
62
63      # Add a spacer for bottom margin
64      #elements.append(Spacer(0, 1*inch))
65
66      # Add the denoised images with titles
67      image_files = ["result/result_for_{i}_samples.jpg" for i in samples]
68      titles = ["k=1", "k=5", "k=10", "k=20", "k=50", "k=100"]
69      table_data = []
70      for i in range(0, len(image_files), 3):
71          # Create a new row for the table
72          row = []
73          for j in range(3):
74              # Check if there is an image file for this position
75              if i + j < len(image_files):
76                  # Create an Image object for the denoised image file
77                  image = Image(image_files[i + j], width=1.5*inch, height=2*inch)
78                  # Create a Paragraph object for the title
79                  title = Paragraph(titles[i + j], subtitle_style)
80                  # Append the image and title as a list to the row
81                  row.append((image, title))
82          # Append the row to the table data
83          table_data.append(row)
84
85      # Create a Table object with the table data and set column widths
86      table = Table(table_data, colWidths=[2.5*inch, 2.5*inch, 2.5*inch])
87      # Set the vertical alignment for all cells in the table
88      table.setStyle(TableStyle([('VALIGN', (0, 0), (-1, -1), 'MIDDLE')]))
89      elements.append(Spacer(0, 0.3*inch))
90      elements.append(Paragraph("Denoised Images", title_style))
91      elements.append(Spacer(0, 0.3*inch))
92      elements.append(table)
93
94      # Add the sample statistics table from CSV file
95      with open('result/results.csv', 'r') as csv_file:
96          csv_reader = csv.reader(csv_file)
97          csv_data = list(csv_reader)
98
99          # Create a table from the CSV data
100         stats_table = Table(csv_data, colWidths=[inch, 1.5*inch, 1.5*inch])
101         stats_table.setStyle(TableStyle([('ALIGN', (0, 0), (-1, -1), 'CENTER')]))
102         elements.append(Spacer(0, 0.3*inch))
103         elements.append(Paragraph("Sample Statistics", title_style))
104         elements.append(Spacer(0, 0.3*inch))
105         elements.append(stats_table)
106
107     # Build the PDF document
108     doc.build(elements)
109     add_snapshot()
110     print("PDF file 'report.pdf' generated successfully.")
```

Requirements:

chardet==5.2.0
numpy==1.26.4
opencv-python==4.9.0.80
pillow==10.3.0
PyPDF2==3.0.1
reportlab==4.2.0