Addis Ababa University

Master's in Artificial Intelligence

Digital Image Processing (DIP) Laboratory Manual

Prepared by: Dr. Natnael Argaw Wondimu

Environment: GNU Octave

Preface

This manual is prepared in a workbook style for AI Master's students at Addis Ababa University. It provides a hands-on introduction to Digital Image Processing using GNU Octave, covering basic to advanced topics. Each lab includes objectives, theoretical background, procedures, Octave code, checkpoints, try-it-yourself prompts, and collaborative assignments. The manual is intended to build foundational knowledge in Digital Image Processing and serve as a base for advanced studies in Computer Vision.

Chapter 5: Image Filtering and Correlation (GNU Octave)

Objective

To understand and implement image filtering and correlation operations in spatial domain using GNU Octave. This includes smoothing, sharpening, and noise reduction techniques.

1. What is Image Filtering?

Description: Image filtering involves the modification of an image by applying a kernel or mask over its pixels. It is used for tasks such as blurring, sharpening, and edge detection.

2. Preparing the Environment

2.1 Load Required Package and Image

Code Snippet:

```
pkg load image;
img = imread('cameraman.tif');
img = im2double(img);
imshow(img);
title('Original Image');
```

Output Description: Loads and displays the grayscale image in double precision for better processing.

3. Averaging (Box) Filter

3.1 What is it?

Description: A simple smoothing filter that averages pixel values in a local neighborhood.

3.2 Applying a 3x3 Averaging Filter

Code Snippet:

```
kernel = ones(3,3)/9;
avg_filtered = imfilter(img, kernel);
imshow(avg_filtered);
title('Averaging Filter (3x3)');
```

Output Description: Image appears smoother with reduced noise and softened edges.

4. Gaussian Filtering

4.1 What is it?

Description: A smoothing filter using a Gaussian kernel to reduce noise while preserving edges better than box filters.

4.2 Applying Gaussian Filter

Code Snippet:

```
gauss_filtered = imgaussfilt(img, 1);
imshow(gauss_filtered);
title('Gaussian Filter (Sigma = 1)');
```

Output Description: Smoother result than averaging, with less distortion of image structures.

5. Median Filtering

5.1 What is it?

Description: A non-linear filter that replaces each pixel with the median value in its neighborhood. Excellent for salt-and-pepper noise removal.

5.2 Applying Median Filter

Code Snippet:

```
median_filtered = medfilt2(img);
imshow(median_filtered);
title('Median Filtered Image');
```

Output Description: Preserves edges while removing outlier noise.

6. Sharpening with Laplacian Filter

6.1 What is it?

Description: Laplacian filtering enhances edges by highlighting regions of rapid intensity change.

6.2 Applying Laplacian Filter

Code Snippet:

```
laplacian_kernel = [0 -1 0; -1 4 -1; 0 -1 0];
laplacian_img = imfilter(img, laplacian_kernel);
sharpened_img = img + laplacian_img;
imshow(sharpened_img);
title('Sharpened Image (Laplacian)');
```

Output Description: Enhances object edges, resulting in a crisper image.

7. Image Correlation

7.1 What is Correlation?

Description: Correlation compares two image regions (or image and filter) to detect similarities. It's commonly used for template matching and feature detection.

7.2 Performing 2D Correlation

Code Snippet:

```
corr_output = conv2(img, kernel, 'same');
imshow(corr_output, []);
title('Correlation Output');
```

Output Description: Shows correlation result between the image and a chosen kernel. High intensity in matched areas.

8. Custom Filter Example: Edge Detection

Code Snippet:

```
sobel_x = [-1 0 1; -2 0 2; -1 0 1];
sobel_y = [-1 -2 -1; 0 0 0; 1 2 1];
gx = imfilter(img, sobel_x);
gy = imfilter(img, sobel_y);
gradient_magnitude = sqrt(gx.^2 + gy.^2);
imshow(gradient_magnitude, []);
title('Edge Detection using Sobel');
```

Output Description: Displays edges detected in horizontal and vertical directions using gradient magnitude.

9. Summary

- **Smoothing filters** reduce noise but may blur details.
- **Sharpening filters** enhance edges and fine structures.
- **Correlation** can help detect template matches or features.
- **Median filters** are best for non-linear noise (salt and pepper).

Suggested Exercises

1. Apply filters of different sizes (5x5, 7x7) and compare results.
2. Implement unsharp masking manually.
3. Test filtering on noisy images and evaluate PSNR or SSIM.
4. Create and correlate a small patch against a larger image.