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 10: Feature Extraction and Recognition (GNU Octave)

Objective

To understand and implement basic feature extraction and object recognition techniques using GNU Octave, including edge, corner, and keypoint detection methods.

1. What is Feature Extraction?

Description: Feature extraction involves identifying important visual elements from an image, such as edges, corners, blobs, or textures. These features are then used for classification, matching, or recognition tasks.

2. Edge Detection as Features

2.1 Canny Edge Detector

Code Snippet:

```
pkg load image;
img = im2double(imread('cameraman.tif'));
edges = edge(img, 'canny');
imshow(edges);
title('Canny Edge Detection');
```

Output Description: Highlights strong edges within the image. Edge maps are used for object boundaries.

3. Corner Detection

3.1 Harris Corner Detector

Code Snippet:

```
corners = cornermetric(img);
imshow(corners, []);
title('Harris Corner Strength Map');
```

Output Description: Generates a heatmap showing locations of corner responses.

3.2 Marking Detected Corners

Code Snippet:

```
corner_thresh = corners > 0.01 * max(corners(:));
[r, c] = find(corner_thresh);
imshow(img); hold on;
plot(c, r, 'r*');
title('Detected Corners');
```

Output Description: Shows red asterisks at detected corner points on the image.

4. Blob Detection using LoG (Laplacian of Gaussian)

Code Snippet:

```
log_filter = fspecial('log', [5 5], 0.5);
blob_img = imfilter(img, log_filter, 'replicate');
imshow(blob_img, []);
title('Blob Detection via LoG');
```

Output Description: Detects blob-like structures in the image, useful for spotting regions with texture.

5. Feature Matching and Recognition

5.1 Feature Descriptor (Simplified - Intensity Patch)

Code Snippet:

```
patch1 = img(30:39, 40:49);  % Patch from reference image
imshow(patch1);
title('Template Patch');
```

5.2 Template Matching (Correlation-Based)

Code Snippet:

```
corr_result = normxcorr2(patch1, img);
[max_corr, idx] = max(corr_result(:));
[y, x] = ind2sub(size(corr_result), idx);

imshow(img);
hold on;
rectangle('Position', [x-9, y-9, 10, 10], 'EdgeColor', 'g');
title('Matched Template Location');
```

Output Description: Locates the region in the image that best matches the selected patch using normalized cross-correlation.

6. Summary

- **Edges** and **corners** are primary low-level features.
- **Blobs** detect regions of interest with texture.
- **Feature matching** allows recognition and localization.

Suggested Exercises

1. Apply corner detection on different standard images.
2. Try matching patches from rotated or scaled versions of the image.
3. Explore using SIFT/ORB (if external toolboxes are available).
4. Develop a basic recognition pipeline using multiple detected features.