CV Lecture Assignments From 1 to 6 Documentation

Overview

This document covers six image processing techniques used in computer vision and image manipulation. These techniques include connected region labeling, image negative transformation, bit plane slicing, contrast stretching and contraction, gray level slicing, and histogram equalization. The task focuses on applying these methods to process grayscale images and visualize the results for better understanding of image transformations.

Prerequisites

- Python 3.x
- OpenCV (`cv2`)
- NumPy
- Matplotlib

Functions Description

1. Labeling of Connected Regions

This function labels the connected regions in a binary image using connected component analysis. The image is first converted to binary, and then the connected components are labeled.

```
def label_connected_regions(image):
    # Convert the image to binary
    _, binary_image = cv2.threshold(image, 127, 255, cv2.THRESH_BINARY)
    # Find the connected components
    num_labels, labels = cv2.connectedComponents(binary_image)
    return num_labels, labels
```

2. Image Negative

This function computes the negative of a grayscale image by subtracting each pixel value from 255.

```
def image_negative(image):
    negative_image = 255 - image
    return negative_image
```

3. Bit Plane Slicing

This function extracts a specified bit plane from a grayscale image. The bit plane is then multiplied by 255 to convert the binary values into visible ones.

```
def bit_plane_slicing(image, plane):
    # Extract the specified bit plane (0-7)
    bit_plane = (image >> plane) & 1
    bit_plane = bit_plane * 255
    return bit_plane.astype(np.uint8)
```

4. Contrast Stretching and Contrast Contraction

These functions adjust the contrast of an image. Contrast stretching increases the image contrast, while contrast contraction reduces it by scaling the pixel values between 0 and 255.

```
def contrast_stretching(image):
    min_val, max_val = np.min(image), np.max(image)
    stretched_image = ((image - min_val) / (max_val - min_val) *
255).astype(np.uint8)
    return stretched_image

def contrast_contraction(image):
    min_val, max_val = np.min(image), np.max(image)
    contracted_image = ((image - min_val) / (max_val - min_val) * 255 *
0.5).astype(np.uint8)
    return contracted_image
```

5. Gray Level Slicing

This function slices the grayscale image at specified intensity ranges, setting the pixel values within that range to 255 (white) and the rest to 0 (black).

```
def gray_level_slicing(image, low, high):
    sliced_image = np.copy(image)
    sliced_image[(image >= low) & (image <= high)] = 255
    sliced_image[~((image >= low) & (image <= high))] = 0
    return sliced_image
```

6. Histogram Equalization

This function performs histogram equalization on the image, which enhances the contrast of the image by redistributing the pixel intensities.

```
def histogram_equalization(image):
    equalized_image = cv2.equalizeHist(image)
    return equalized_image
```

Steps to Execute

1. Place the input grayscale image (`Grayscale Photography Of Woman.jpg`) in the same directory as the script.
2. Run the script to apply the image processing techniques.
3. The output will display the results of connected region labeling, image negative, bit plane slicing, contrast stretching, contrast contraction, gray level slicing, and histogram equalization side by side.

Example Visualization

Displayed Images

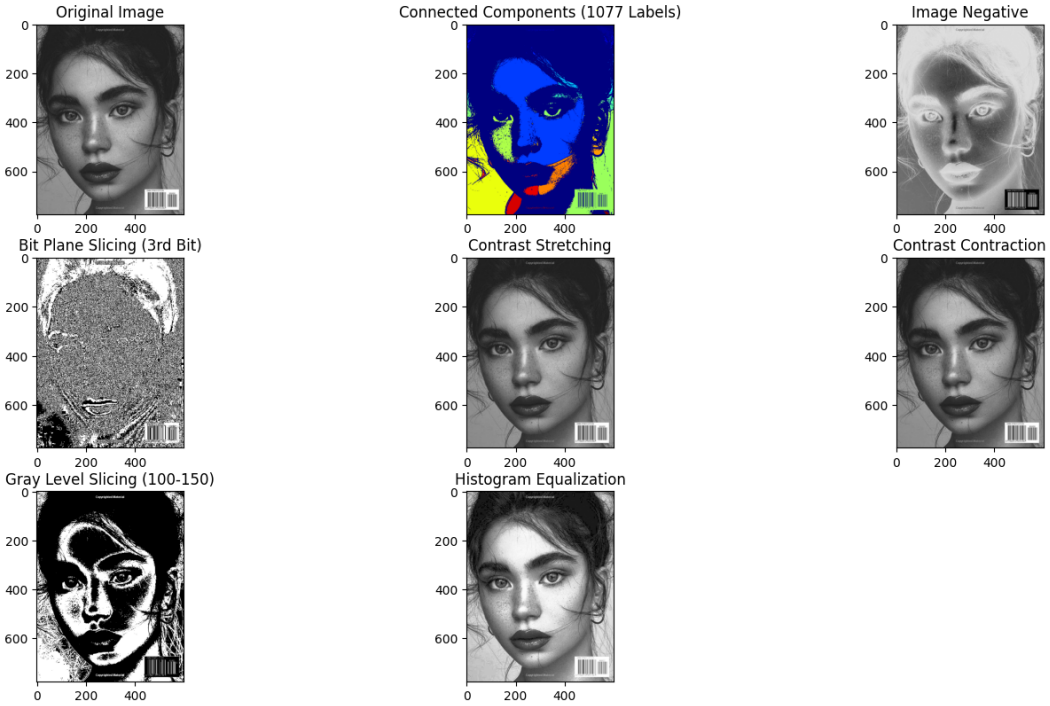
- **Original Image**
- **Connected Components Labeling**
- **Image Negative**
- **Bit Plane Slicing (3rd Bit)**
- **Contrast Stretching**
- **Contrast Contraction**
- **Gray Level Slicing (100-150)**
- **Histogram Equalization**

Sample Visualization Code

```
plt.figure(figsize=(12, 10))

# Original Image
plt.subplot(3, 3, 1), plt.imshow(image, cmap='gray'), plt.title('Original Image')

# Labeling of Connected Regions
plt.subplot(3, 3, 2), plt.imshow(labeled_image, cmap='jet'), plt.title(f'Connected
Components ({num_labels} Labels)')

# Image Negative
plt.subplot(3, 3, 3), plt.imshow(negative_image, cmap='gray'), plt.title('Image
Negative')

# Bit Plane Slicing
plt.subplot(3, 3, 4), plt.imshow(bit_plane_image, cmap='gray'), plt.title('Bit
Plane Slicing (3rd Bit)')

# Contrast Stretching
plt.subplot(3, 3, 5), plt.imshow(stretched_image, cmap='gray'),
plt.title('Contrast Stretching')

# Contrast Contraction
plt.subplot(3, 3, 6), plt.imshow(contracted_image, cmap='gray'),
plt.title('Contrast Contraction')

# Gray Level Slicing
plt.subplot(3, 3, 7), plt.imshow(sliced_image, cmap='gray'), plt.title('Gray Level
Slicing (100-150)')
```

```
# Histogram Equalization
plt.subplot(3, 3, 8), plt.imshow(equalized_image, cmap='gray'),
plt.title('Histogram Equalization')

plt.tight_layout()
plt.show()
```

Output Example



Comparison of Techniques

1. Labeling of Connected Regions

- **Purpose**: Identifies and labels distinct connected regions in a binary image.
- **Strengths**: Useful for object detection, segmentation, and counting.
- **Weaknesses**: The quality of the labeling depends on the binarization threshold and noise in the image.

2. Image Negative

- **Purpose**: Inverts the pixel intensities, turning light areas dark and dark areas light.
- **Strengths**: Simple and effective for creating artistic or contrast-enhancing effects.
- **Weaknesses**: May not be meaningful for all types of analysis.

3. Bit Plane Slicing

- **Purpose**: Isolates specific bit planes to analyze low or high-level information in the image.
- **Strengths**: Helps to identify significant features in images like edges or textures.
- **Weaknesses**: Requires a deep understanding of image encoding.

4. Contrast Stretching and Contrast Contraction

- **Purpose**: Enhances or reduces the image contrast, improving visual quality.
- **Strengths**: Contrast stretching enhances detail in the image; contrast contraction can be used for noise reduction.
- **Weaknesses**: Overstretching can lead to unnatural-looking images, while contraction may reduce detail.

5. Gray Level Slicing

- **Purpose**: Highlights certain intensity ranges while suppressing others.
- **Strengths**: Useful for isolating specific intensity ranges (e.g., bright or dark features).
- **Weaknesses**: Limited flexibility since it only affects pixel intensity ranges.

6. Histogram Equalization

- **Purpose**: Enhances the image contrast by redistributing pixel intensities.
- **Strengths**: Improves visual quality by using the entire intensity range.
- **Weaknesses**: May lead to over-enhancement or loss of details in some cases.

By Ahmed Ibrahim Metwally Negm - ID: 322223887

Supervisors: Dr. Shimaa Othman

Thank you, ✨ 🤍