Digital Image Processing
Spring 2025
Lab:5 Intensity Transformation II (Histogram Processing)

Name	
Student ID	
Date	

Objectives

- Perform histogram equalization to enhance image contrast.
- Implement histogram matching to transform an image's histogram to resemble another image's histogram.
- Explore how histogram equalization can be applied to both grayscale and color images.

Prerequisites

Make sure you have the following Python libraries installed:

- `opencv-python` (OpenCV)
- `matplotlib`
- `numpy`
- `scikit-image`

You can install them using following command in command prompt or bash:

pip install opencv-python matplotlib numpy scikit-image

scikit-image overview

scikit-image is an open-source Python library designed for **image processing**. It is built on top of other scientific libraries like **NumPy**, **SciPy**, and **matplotlib**. It provides easy-to-use functions for a variety of image processing tasks, making it an excellent tool for students and professionals alike.

Key Features of scikit-image:

1. **Simple Interface**: Functions are easy to use and designed to work with common image data formats like NumPy arrays.
2. **Image I/O**: You can easily load, save, and display images in various formats.
3. **Filtering and Enhancement**: Provides filters for image denoising, sharpening, edge detection, and contrast enhancement.
4. **Segmentation**: Allows you to perform tasks like thresholding, watershed segmentation, and region-based segmentation.

5. **Geometric Transformations**: Offers functions for resizing, rotating, and warping images.
6. **Color Space Conversions**: Converts between different color spaces (e.g., RGB to grayscale).
7. **Histogram Operations**: Facilitates histogram-based processing such as histogram equalization and matching.
8. **Morphological Operations**: Tools for tasks like dilation, erosion, and skeletonization.

Example Functions from scikit-image:

1. **Histogram Matching (`exposure.match_histograms`)**: Matches the histogram of one image to that of another.
2. **Thresholding (`filters.threshold_otsu`)**: Automatically calculates an optimal threshold for converting grayscale images to binary.
3. **Geometric Transformations (`transform.rotate`, `transform.resize`)**: Functions for rotating and resizing images.
4. **Feature Detection (`feature.canny`)**: Detects edges using Canny edge detection.

Why Use scikit-image?

- **Performance**: It is built on NumPy, ensuring fast computation on arrays.
- **Broad Functionality**: It provides functions for most image processing tasks, including advanced operations like image segmentation and feature extraction.
- **Integration with Other Libraries**: Works well with other scientific Python libraries like pandas, matplotlib, and OpenCV, which makes it very versatile.

For example, in this lab, we used the function `exposure.match_histograms()` from **scikit-image** to match histograms between two images, which is a common task in image enhancement.

Digital Image Processing
Spring 2025
Lab:5 Intensity Transformation II (Histogram Processing)

Part 1: Histogram Equalization

Histogram equalization improves contrast in images by redistributing the intensity values to make the histogram as flat as possible.

```
import cv2
import matplotlib.pyplot as plt

# Read a grayscale image
image = cv2.imread('1.png', cv2.IMREAD_GRAYSCALE)

# Apply histogram equalization
equalized_image = cv2.equalizeHist(image)

# Display the original and equalized images
plt.figure(figsize=(10,5))

# Original image and its histogram
plt.subplot(2, 2, 1)
plt.imshow(image, cmap='gray')
plt.title('Original Image')
plt.axis('off')

# Function Explanation: ravel() flattens a multi-
dimensional array into 1D.
# In this case, it flattens the 2D image matrix
into a 1D array to plot its histogram.
plt.subplot(2, 2, 2)
plt.hist(image.ravel(), 256, [0,256])
plt.title('Histogram (Original)')

# Equalized image and its histogram
plt.subplot(2, 2, 3)
plt.imshow(equalized_image, cmap='gray')
```

```
plt.title('Equalized Image')
plt.axis('off')

plt.subplot(2, 2, 4)
plt.hist(equalized_image.ravel(), 256, [0,256])
plt.title('Histogram (Equalized)')

plt.show()
```

Explanation of `ravel()`:

The `ravel()` function from NumPy flattens a multi-dimensional array into a one-dimensional array. In the case of images, it is used to convert the 2D image matrix into a 1D array for easy histogram plotting.

Task 1:

1. Apply histogram equalization to an image and observe the effect on contrast.
2. Compare the histograms before and after equalization.
3. Discuss why histogram equalization improves contrast in images with poor lighting.

Part 2: Histogram Equalization on Color Images

Step 2: Applying Histogram Equalization to Color Channels Separately

When working with color images, histogram equalization can be applied to individual color channels (Red, Green, Blue) separately.

```
import cv2
import matplotlib.pyplot as plt
# Read a color image
color_image = cv2.imread('1.png')

# Split the image into R, G, B channels
(b, g, r) = cv2.split(color_image)

# Equalize each channel separately
```

```
equalized_b = cv2.equalizeHist(b)
equalized_g = cv2.equalizeHist(g)
equalized_r = cv2.equalizeHist(r)

# Merge the equalized channels
equalized_color_image = cv2.merge([equalized_b,
equalized_g, equalized_r])

# Display the original and equalized images
plt.figure(figsize=(10,5))

plt.subplot(1, 2, 1)
plt.imshow(cv2.cvtColor(color_image,
cv2.COLOR_BGR2RGB))
plt.title('Original Color Image')
plt.axis('off')

plt.subplot(1, 2, 2)
plt.imshow(cv2.cvtColor(equalized_color_image,
cv2.COLOR_BGR2RGB))
plt.title('Equalized Color Image (Channels
Separately)')
plt.axis('off')

plt.show()
```

Task 2:

1. Perform histogram equalization on the color channels separately (Red, Green, Blue).
2. Observe the changes in the color balance.

Digital Image Processing
Spring 2025
Lab:5 Intensity Transformation II (Histogram Processing)

Part 3: Histogram Matching (Specification)

Histogram matching transforms the histogram of an input image to match a target histogram from another image. This is useful for matching brightness and contrast between two images.

Step 3: Applying Histogram Matching

```
import cv2
import numpy as np
from skimage import exposure
import matplotlib.pyplot as plt

# Read two grayscale images
image_source = cv2.imread('1.png',
cv2.IMREAD_GRAYSCALE)
image_target = cv2.imread('bright.png',
cv2.IMREAD_GRAYSCALE)

# Function Explanation: exposure.match_histograms()
matches the histogram of the source image
# to that of the target image. It adjusts pixel
values to make the intensity distribution
# of the source image resemble the target image's
histogram.

# Perform histogram matching
matched_image =
exposure.match_histograms(image_source,
image_target)

# Display the source, target, and matched images
plt.figure(figsize=(15,5))
```

Digital Image Processing
Spring 2025
Lab:5 Intensity Transformation II (Histogram Processing)

```
# Source image and its histogram
plt.subplot(3, 3, 1)
plt.imshow(image_source, cmap='gray')
plt.title('Source Image')
plt.axis('off')

plt.subplot(3, 3, 2)
plt.hist(image_source.ravel(), 256, [0,256])
plt.title('Histogram (Source)')

# Target image and its histogram
plt.subplot(3, 3, 4)
plt.imshow(image_target, cmap='gray')
plt.title('Target Image')
plt.axis('off')

plt.subplot(3, 3, 5)
plt.hist(image_target.ravel(), 256, [0,256])
plt.title('Histogram (Target)')

# Matched image and its histogram
plt.subplot(3, 3, 7)
plt.imshow(matched_image, cmap='gray')
plt.title('Matched Image')
plt.axis('off')

plt.subplot(3, 3, 8)
plt.hist(matched_image.ravel(), 256, [0,256])
plt.title('Histogram (Matched)')

plt.show()
```

Explanation of `exposure.match_histograms()`:

The `exposure.match_histograms()` function from `skimage` adjusts the pixel intensities of a source image so that its histogram matches that of a target image. It is useful for ensuring visual consistency between images in terms of contrast and brightness.

Task 3:

1. Choose two images, one as the source and the other as the target.
2. Apply histogram matching to make the source image's histogram match the target image's histogram.
3. Explain how histogram matching affects the visual appearance of the source image.

Part 4: Combining Histogram Equalization and Matching

Task 4:

1. First, equalize the histogram of a source image.
2. Then, perform histogram matching on the equalized image using a target image.
3. Discuss how equalization followed by matching alters the appearance of the original image.

Submission

Submit the following:

1. Screenshots of the images before and after histogram equalization and matching.
2. The code for each task.
3. A brief report explaining the effects of histogram equalization and matching on image contrast and brightness distribution.

Conclusion

In this lab, you:

- Applied histogram equalization to both grayscale and color images.
- Used the `ravel()` function to flatten image arrays for histogram plotting.

Digital Image Processing
Spring 2025
Lab:5 Intensity Transformation II (Histogram Processing)

- Used `exposure.match_histograms()` to match the intensity distributions of two images.
- Combined equalization and matching to improve the appearance of images in various lighting conditions.