CV Section Assignment 1 Documentation

Overview

This assignment implements various image processing techniques and filters using Python's OpenCV library. The tasks include:

1. Histogram Equalization
2. Contrast Stretching
3. Applying Averaging, Gaussian, and Median Filters

Prerequisites

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

Functions Description

1. Read Image

Reads an input image from the specified file path.

```
def read_image(filepath):
    image = cv2.imread(filepath, cv2.IMREAD_COLOR)
    if image is None:
        raise FileNotFoundError(f"Image not found at {filepath}")
    return image
```

2. Histogram Equalization and Contrast Stretching

2.1 Histogram Equalization

Enhances the image contrast by redistributing pixel intensity values.

```
def histogram_equalization(image):
    if len(image.shape) > 2:
        image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    equalized_image = cv2.equalizeHist(image)
    return equalized_image
```

2.2 Contrast Stretching

Improves image contrast by stretching the intensity range.

```
def contrast_stretching(image):
    if len(image.shape) > 2:
        image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    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
```

3. Filters

3.1 Averaging Filter

Applies a simple averaging filter using a specified kernel size.

```
def apply_averaging_filter(image, kernel_size=(5, 5)):
    return cv2.blur(image, kernel_size)
```

3.2 Gaussian Filter

Applies a Gaussian filter to reduce image noise and detail.

```
def apply_gaussian_filter(image, kernel_size=(5, 5), sigma=1):
    return cv2.GaussianBlur(image, kernel_size, sigma)
```

3.3 Median Filter

Applies a median filter to preserve edges while reducing noise.

```
def apply_median_filter(image, kernel_size=5):
    return cv2.medianBlur(image, kernel_size)
```

Steps to Execute

1. Place the input image (`Grayscale Photography Of Woman.jpg`) in the same directory as the script.
2. Run the script `C.V. Sec. Assignment 1.py`.
3. The output will generate processed images and save them as `Output Comparison.png`.

Example Visualization

Displayed Images

- Original
- Histogram Equalized

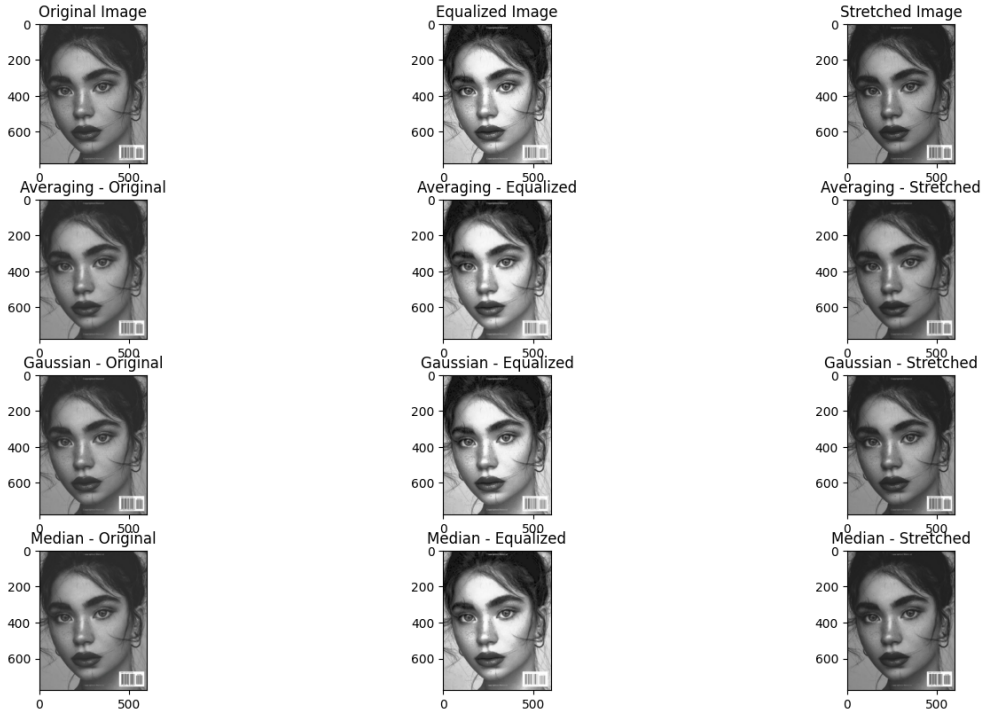
- Contrast Stretched
- Filters applied on each of the above:
    - Averaging
    - Gaussian
    - Median

Sample Visualization Code

```
plt.figure(figsize=(15, 12))

# Original Images
plt.subplot(4, 3, 1), plt.imshow(cv2.cvtColor(image, cv2.COLOR_BGR2RGB)),
plt.title('Original Image')
plt.subplot(4, 3, 2), plt.imshow(cv2.cvtColor(equalized_image,
cv2.COLOR_BGR2RGB)), plt.title('Equalized Image')
plt.subplot(4, 3, 3), plt.imshow(cv2.cvtColor(stretched_image,
cv2.COLOR_BGR2RGB)), plt.title('Stretched Image')

# Averaging Filter Results
plt.subplot(4, 3, 4), plt.imshow(cv2.cvtColor(averaged_original,
cv2.COLOR_BGR2RGB)), plt.title('Averaging - Original')
plt.subplot(4, 3, 5), plt.imshow(cv2.cvtColor(averaged_equalized,
cv2.COLOR_BGR2RGB)), plt.title('Averaging - Equalized')
plt.subplot(4, 3, 6), plt.imshow(cv2.cvtColor(averaged_stretched,
cv2.COLOR_BGR2RGB)), plt.title('Averaging - Stretched')

# Gaussian Filter Results
plt.subplot(4, 3, 7), plt.imshow(cv2.cvtColor(gaussian_original,
cv2.COLOR_BGR2RGB)), plt.title('Gaussian - Original')
plt.subplot(4, 3, 8), plt.imshow(cv2.cvtColor(gaussian_equalized,
cv2.COLOR_BGR2RGB)), plt.title('Gaussian - Equalized')
plt.subplot(4, 3, 9), plt.imshow(cv2.cvtColor(gaussian_stretched,
cv2.COLOR_BGR2RGB)), plt.title('Gaussian - Stretched')

# Median Filter Results
plt.subplot(4, 3, 10), plt.imshow(cv2.cvtColor(median_original,
cv2.COLOR_BGR2RGB)), plt.title('Median - Original')
plt.subplot(4, 3, 11), plt.imshow(cv2.cvtColor(median_equalized,
cv2.COLOR_BGR2RGB)), plt.title('Median - Equalized')
plt.subplot(4, 3, 12), plt.imshow(cv2.cvtColor(median_stretched,
cv2.COLOR_BGR2RGB)), plt.title('Median - Stretched')

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

Input Example

Output Example



The comparison of all processed images is saved in `Output Comparison.png`. The images highlight the effectiveness of each filter and transformation on the original, equalized, and stretched images.

Some notable side effects:

1. Histogram Equalization Over-Enhancement: Can lead to over-enhanced images with unnatural contrasts, especially in cases of excessive noise. Loss of Detail: Fine details in darker or lighter regions may be lost due to the redistribution of intensity values.
2. Contrast Stretching Amplified Noise: Noise in low-contrast regions can become more noticeable. Color Distortion: If applied to colored images without converting to grayscale, the colors may appear unnatural.
3. Averaging Filter Blurring of Edges: Reduces sharpness and detail in the image by averaging pixel values, leading to a loss of edge clarity. Artifacts: Introduces halo artifacts around high-contrast edges.
4. Gaussian Filter Loss of Fine Details: Similar to the averaging filter, Gaussian filtering smooths out fine details, which may not be desirable in all scenarios. Computational Cost: Slightly more computationally intensive than averaging filters, especially with larger kernel sizes.
5. Median Filter Edge Preservation Trade-Off: While effective at removing noise, it can alter or distort finer textures in areas of uniform noise. Processing Overhead: More computationally expensive than averaging or Gaussian filters, especially for larger kernels.

By Ahmed Ibrahim Metwally Negm - ID: 322223887

Supervisors: Dr. Shimaa Othman, Eng. Yosr

Thank you, ✨ 🤍.