Iman Chaudhry 210650820

A1: Interpolation, Point Operations, Histogram Methods

Introduction

This assignment explores image interpolation, point operations, and histogram methods using Python and OpenCV. The images "lena", "cameraman", and "chest_x-ray1/chest_x-ray2" were processed through both custom implementations and OpenCV functions, providing practical experience in image processing.

Image Interpolation

The "lena" image was rescaled to ¼ of its original size, then upscaled back to its original dimensions using different interpolation methods. The nearest neighbor and bilinear methods were implemented from scratch and using OpenCV, while bicubic interpolation used only OpenCV. Three methods applied to this image were *Nearest Neighbor* interpolation, *Bilinear* interpolation, and *Bicubic* interpolation. *Nearest Neighbor* selects the nearest pixel without smoothing, resulting in fast processing but pixelated images ("lena_nearest_scratch.png"). OpenCV's built-in version ("lena_nearest_cv.png") gave similar results. The *Bilinear* interpolation computes a weighted average from the four nearest pixels, producing smoother results compared to nearest neighbor but with slight blurring ("lena_bilinear_scratch.png"). OpenCV's version ("lena_bilinear_cv.png") was smoother with better quality. *Bicubic* interpolation uses a 4x4 pixel grid and cubic polynomials, generating the smoothest result of the aforementioned ("lena_bicubic_cv.png").

Point Operations

Point operations were applied to the "cameraman" image by scratch. The three methods applied were the *Negative* transformation*, Power-Law* transformation*, and Bit-Plane Slicing*. For the *Negative* transformation, the pixels were inverted and produced a higher contrast image ("cameraman_negative.png"). *Power-Law* transformation, also known as gamma correction, enhanced the contrast in darker regions. I used a gamma value of 0.65 to yield the best-looking result after testing images with various gamma values ranging from 0-2 ("cameraman_power.png"). The last transformation was *Bit-Plane Slicing*, which split the image into 8-bit planes, with higher-order bits containing most of the visual information compared to the lower bits, which added finer details ("cameraman_b1.png" through "cameraman_b8.png").

Histogram Processing

Two histogram methods were applied: *Histogram Equalization* and *Histogram Matching*. *Histogram Equalization* redistributes pixel intensities to improve contrast, which significantly enhances the visibility in the "einstein" image ("einstein_equalized.png"). In *Histogram Matching*, the histogram of "chest_x-ray1" was adjusted to match "chest_x-ray2", ensuring consistent contrast ("chest_x-ray3.png").

Results and Discussion

For the first task of image interpolation, the *Nearest Neighbor* interpolation (from scratch and OpenCV) produced blocky, pixelated images ("lena_nearest_scratch.png" and "lena_nearest_cv.png") due to the simplicity of the method. *Bilinear* interpolation smoothed out transitions but introduced slight blurring, particularly in the version created from scratch ("lena_bilinear_scratch.png" vs. "lena_bilinear_cv.png"). *Bicubic* interpolation, using OpenCV, resulted in the smoothest, best quality images compared, and while still blurry, looked better than the images produced using the other two methods ("lena_bicubic_cv.png").

For the second task of point operations, the *Negative* transformation inverted the "cameraman" image, producing a sharper contrast that highlighted features in the negative image. *Power-Law* transformation using gamma correction significantly enhanced the contrast in darker regions of the image, and *Bit-Plane Slicing* provided insight into the individual contribution of each bit plane (1-8) in the overall image structure.

Through the last task of histogram processing: *Histogram Equalization* greatly enhanced the contrast in the "einstein" image, making features in both dark and bright regions more prominent. *Histogram Matching* successfully aligned the contrast of the chest X-rays, making the comparison between "chest_x-ray1" and "chest_x-ray2" more accurate and uniform.

Challenges

Some implementation challenges were encountered, especially during the histogram processing task. It was hard at first to figure out how to calculate the histogram and CDF without built-in functions. I had to create a custom histogram through a function that iterates through each pixel in the flattened "einstein" image to count intensity, and then compute the CDF through another function that used cumulative summation. Another challenge was to redistribute the pixel intensities based on the CDF to enhance contrast. For this, I had to normalize the CDF to the 0-255 range and map the old pixels to the new equalized values, and used the numpy library to do so.

Conclusion

This assignment demonstrated the trade-offs between different image interpolation methods, as well as the benefits of point operations and histogram processing for enhancing images. *Nearest Neighbor* interpolation is fast but produces lower-quality images, while *Bilinear* and *Bicubic* methods offer better results at the cost of computation. Point operations such as negative transformation and power-law adjustment are useful for image enhancement, while histogram equalization and matching ensure consistent contrast across images. The project improved the understanding of both manual implementations and the use of OpenCV for optimized results.

Appendices

Appendix A: Image Interpolation


lena_nearest_scratch


lena_nearest_scratch_downscaled


lena_nearest_cv


lena_nearest_cv_downscaled


lena_bilinear_scratch


lena_bilinear_scratch_downscaled


lena_bilinear_cv


lena_bilinear_cv_downscaled


lena_bicubic_cv


lena_bicubic_cv_downscaled

Appendix B: Point Operations

cameraman_negative

cameraman_power

cameraman_b1

cameraman_b2

cameraman_b3

cameraman_b4

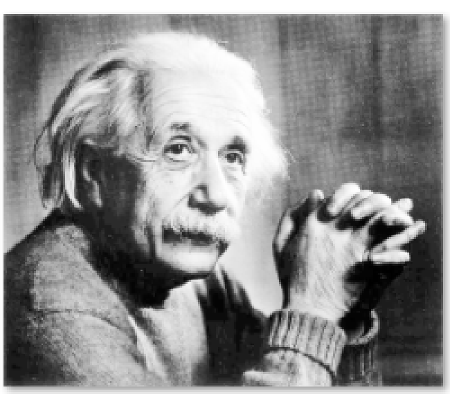
cameraman_b8

cameraman_b7

cameraman_b6

cameraman_b5

Appendix C: Histogram Processing



einstein_equalized

chest_x-ray3