Image Processing and Pattern Recognition

Assignment O

14.10.2024

Jakob Freiberger

Task 1: Implementation of PSNR

The PSNR was quite straightforward to implement. For the accelerated (no for-loops) version, I relied on numpy’s builtin matrix operations, concretely the matrix subtraction, np.square and np.mean.

Task 2: Implementation of MSSIM

In the MSSIM implementation, I tried to optimize out any for loops as well. I did not fully eliminate them because I could not manage getting the individual patches without iterating over the whole image at least once.  
The formulas themselves were not that difficult to implement, however it seemed that the formula for the mean patch intensity was missing a division operation. This is why I used the mean() instead of the sum() function for the computation (see Figure 1)

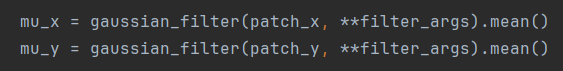


Figure 1: Mean patch intensity computation

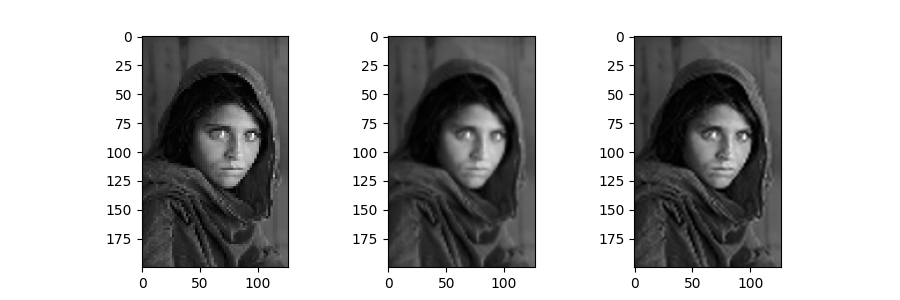


Figure : Rescaled images, from left to right: nearest neighbor, linear and cubic

The rescaled images turned out as shown in Figure 2. In my opinion, the cubic filter does the best job at preserving the original image, as it has more detail than the nearest neighbor and does not blur the image as much as the linear filter. These results are reflected in the scores, as shown in Table 1.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Nearest neighbor | Linear | Cubic |
| MSSIM | 0.792 | 0.883 | 0.914 |
| PSNR | 27.167 | 31.092 | 31.976 |

Table 1: Similarity scores for all three methods

The MSSIM score unambiguously prefers the cubic method, but the PSNR score results in similar numbers for linear and cubic interpolation. Therefore, I think that the MSSIM score is the superior measure in this case.