CSC\_8628

Harvey Yuan 200774394

12/01/2023

**Aim and Objective**

Image denoising is a fundamental process required for image processing which can be used for many different purposes such as machine learning algorithms, facial recognition systems and object detection.

The aim of this assignment is to create an image denoising algorithm which involves algorithmic development. This is to allow for the algorithm to update arguments to parameters as the input image changes.

**Algorithm**

Diagram

Description automatically generated

Figure Algorithm Flow Chart

Figure 1 demonstrates the flow that the algorithm takes as it denoises the input of a colour RGB image. The first step once the input image is loaded onto into the algorithm, is that the input image gets split between the 3 channels to allow for processing. This results in 3 different images that occupy the separate channels.

Once the image has been split, each resulting image goes through a Fourier transformation into the frequency domain. A hard low pass mask is applied against the frequencies which filters out some of the higher frequencies.

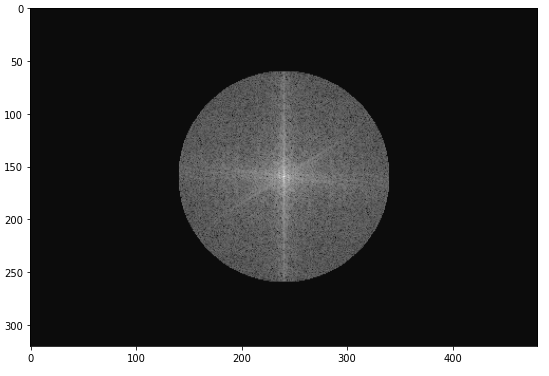


Figure Example of Mask

|  |
| --- |
| # Defining the required radius of the Fourier mask  sig\_est = np.mean(estimate\_sigma(im, multichannel=True))  def rad(sig\_est):    if(sig\_est < 18):      return 100    elif (18<sig\_est<36):      return 75    elif (sig\_est>36):      return 75  r = rad(sig\_est) |

Figure Code snippet for radius choosing

Figure 3 allows for the radius of the mask to be dynamically updated based on the estimation of the standard deviation of noise in the input image. The sigma thresholds were calculated by applying the 75 training images against the complete algorithm at different radii set to different sizes to find their SSIM and MSE scores. The radii for the highest resulting SSIM and lowest MSE scores were selected based on the estimated sigma values of the input. As the algorithm gets applied to more and more inputs, the thresholds and radii values can be updated to improve the scoring metrics.

Once the low pass filter has been applied against the channels, the inverse Fourier transform takes place and is then recombined to receive an image within the RGB colour space. As the output of the low pass filter is smoothed, the image is then put through the unsharp filter to improve the edges.

This sharpened image is then once again split into the 3 channels and a non-local means filter is applied to against them. The non-local means algorithm “replaces the value of a pixel by an average of a selection of other pixels values” [1]. Although the NLM filter has good performance against the two metrics in comparison to some other algorithms such as Wiener filter and Bilateral filters, it has trouble restoring small structures [2] which could destroy texture detail within an image depending on how aggressive the filter is.

After the NLM filter has completed, the channels are recombined one last time to produce a denoised image.

|  |
| --- |
| # Define function for Sigma est and NLM filter  def sig\_est(im):    sig = np.mean(estimate\_sigma(im, multichannel=True))    return sig  def nlm(im, sig\_est):    nl = denoise\_nl\_means(im, h=2.5 \* sig\_est, sigma=sig\_est, fast\_mode=True)    return nl |

Figure Function for NLM filter

In figure 4, h allows for more smoothing between dissimilar patches which defines how aggressively the filter will work. A higher value of h will increase the amount of smoothing. As this value is applied as a multiplier against the sigma estimation, if an image with a high value for sig\_est, the algorithm will work more aggressively to denoise and smooth patches. This could be problematic because if the estimated value is magnitudes higher than another image, it would cause some text details within the image to be destroyed.

**Results**

Table Results from Noise 10 images for denoise Algorithm vs mean and median filters

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Noise 10** | **denoise SSIM** | **denoise MSE** | **mean SSIM** | **mean MSE** | **median SSIM** | **median MSE** |
| **min** | 0.601 | 27.486 | 0.501 | 28.489 | 0.531 | 17.588 |
| **max** | 0.951 | 591.573 | 0.858 | 794.486 | 0.832 | 685.010 |
| **average** | 0.787 | 184.857 | 0.699 | 251.523 | 0.716 | 215.407 |

Table Results from Noise 25 images for denoise Algorithm vs mean and median filters

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Noise 25** | **denoise SSIM** | **denoise MSE** | **mean SSIM** | **mean MSE** | **median SSIM** | **median MSE** |
| **min** | 0.576 | 46.862 | 0.476 | 50.865 | 0.482 | 55.111 |
| **max** | 0.843 | 763.514 | 0.742 | 820.703 | 0.717 | 745.728 |
| **average** | 0.713 | 256.792 | 0.632 | 277.824 | 0.617 | 257.847 |

Table Results from Noise 50 images for denoise Algorithm vs mean and median filters

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Noise 50** | **denoise SSIM** | **denoise MSE** | **mean SSIM** | **mean MSE** | **median SSIM** | **median MSE** |
| **min** | 0.448 | 98.473 | 0.368 | 130.573 | 0.361 | 178.793 |
| **max** | 0.695 | 911.905 | 0.605 | 921.988 | 0.564 | 914.172 |
| **average** | 0.588 | 363.142 | 0.506 | 375.049 | 0.462 | 387.124 |

From the above, the denoise algorithm works well in both the SSIM and MSE metrics in comparison to just mean and median filtering where the min, max and average scores for both metrics outperform the other two filters.

Although the algorithm works well for images with low noise, it starts to reduce in performance once noise starts to increase as evident in the average SSIM score for the algorithm at 0.588. From the image comparisons below, figure 5 and 6 contains images with regions of low texture which the algorithm deals with well. However, in figure 7, the high textures have not been preserved well due to the NLM filter.

Graphical user interface, PowerPoint

Description automatically generated

Figure Image comparison - Noise 50, Denoised Algorithm, Mean and Median

A picture containing text, clock

Description automatically generated

Figure Image comparison - Noise 50, Denoised Algorithm, Mean and Median

A picture containing diagram

Description automatically generated

Figure Image Comparison - Noise 50, Denoised Algorithm, Mean and Median

**Conclusion**

To conclude, the denoising algorithm works well in comparison to the mean and median filters as it includes image smoothing, sharpening of edges and finally an NLM filter. The issue with the algorithm, as mentioned, is that the h value has been applied as a constant in the NLM function which results in image outputs with destroyed text detail. To solve this problem, the h value should be thresholded much like how the mask radii is for the Fourier transform, this should help reduce the amount of text loss in the input images.

# References

|  |  |
| --- | --- |
| [1] | Chris Roat, Fabian Schneider, Gregory Lee, Hande Gözükan, Larry Bradley, Marianne Corvellec, Mark Harfouche, Miles Lucas and Riadh Fezzani, "Non-local means denoising for preserving textures," scikit-image Library. [Online]. [Accessed 09 1 2023]. |
| [2] | Linwei Fan, Fan Zhang, Hui Fan and Caiming Zhang, "Brief review of image denoising techniques," Vis Comput Ind Biomed Art., 2019. |
|  |  |