Computational Fabrication - Assignment 2

Christian Altrichter

May 22, 2023

1 Pre-fix

In this assignment we were given a pre-implementation of a C++ file that included the following:

- Reading and writing an image.
- Thresholding of an image (converting the image in black and white)

The task now was to implement the following two (plus one) algorithms:

- Dithering
- Error Diffusion
- Extended Error Diffusion

Each of the algorithms has been tested on the two provided images:

- test1.pgm - An image that depicts a women with different gray scale tones. This image has an average intensity of 0.363147.
- test2.pgm - An image that depicts a continuous color change from black to white. This image has an average intensity of 0.500978.

(a) Test1 (b) Test2

The subsequent result of the algorithms has then been saved and stored with a respective abbreviation as followos:

- dithering abbreviation - *"_dith.pgm"*
- error diffusion abbreviation - *"_err.pgm"*
- extenden error diffusion abbreviation - *"_err_etd.pgm"*

2 Dithering

For dithering, I have first initialized a matrix (as seen in class) with the following values:

$$\begin{bmatrix} 2 & 6 & 4 \\ 5 & 0 & 2 \\ 8 & 3 & 7 \end{bmatrix}$$

Then, the matrix has been normalized with the scalar of

$$\frac{1}{9}$$

to convert the weight distribution and the resulting value between 0 and 1. Depending on the matrix average value, the overall color impression of the image will change. Thus, for the matrix above, we receive an average of approximately 0.4. This means that the image's impression will slightly be on the darker side. I then apply to each pixel the matrix with their respective index in the matrix. The results for the above matrix application and with the dithering algorithm are as follows:

(c) Test1 result dithering with 3x3 matrix average approx at 0.4

(d) Test2 result dithering with 3x3 matrix average approx at 0.4

The matrix can then be played with such that pattern and overall intensity can be changed. It ought to be noted that the matrix in the dithering process is not energy preserving. Meaning, should the average of the normalized matrix be above the given input image's average intensity (thus, closer to 1 and therefore the value of 0) the darker the matrix will get. Vice versa, should the average of the matrix be below the images average intensity, then the lighter the image will get. *(Note: I have discussed this assumption with Prof. Didyk. As shown in the results, I believe this assumption to be true. I am still awaiting an answer regarding my question as he argues that the logic is not fully consistent, however, could not precisely argue why and how. Thus, please take the aforementioned statement with caution.)* Here, I have chosen the following two cases:

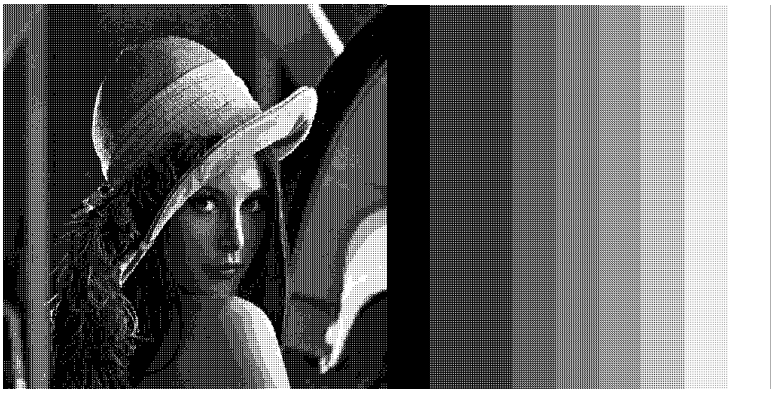
- One case where the overall intensity (thus, the average of the matrix is not centered around the mean intensity of the image)
- The size / pattern of the matrix can be changed.

2.1 Overall changed intensity

The changed matrix remains a 3x3 matrix but its average normalized intensity is now approximately 0.6. Thus, as described above due to its property of not preserving energy, the pictures are expected to get darker. The applied matrix is before normalization is:

$$\begin{bmatrix} 7 & 6 & 4 \\ 5 & 6 & 1 \\ 8 & 3 & 7 \end{bmatrix}$$

The results (including the normalized matrix) for the above matrix applied with the dithering algorithm are follows:



(e) Test1 result dithering with 3x3 matrix average at approx. 0.6

(f) Test2 result dithering with 3x3 matrix average at approx. 0.6

Interestingly, it can be seen that the matrix averagewhich is around 0.6 can be directly seen in a ticker band of values that evolve around this intensity in *image (f)*. Therefore, I assume my beforehand statement to hold true.

2.2 Overall changes size / pattern

The changed matrix is now changed to a 4x4 matrix but its average normalized intensity is now approximately 0.45. The applied matrix is:

$$\begin{bmatrix} 2 & 16 & 3 & 16 \\ 10 & 6 & 11 & 7 \\ 4 & 14 & 1 & 15 \\ 12 & 8 & 9 & 5 \end{bmatrix}$$

Here, to normalize evidently we multiply by the scalar

$$\frac{1}{16}$$

. The output of this matrix application is as follows:

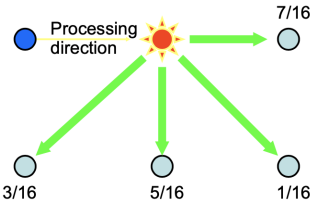
(g) Test1 result dithering with 4x4 matrix average at approx. 0.45

(h) Test2 result dithering with 4x4 matrix average at approx. 0.45

As can be seen especially within the continuous color change, a larger matrix leads to a much more fine grained result in the color distinction and preserved the image quality better. This is also corraborated by the *image g* in which we loose sight of distinct patterns compared to *images c & e*.

3 Error Diffusion

With respect to the error diffusion, I have applied the seen concepts in class by distributing the error as follows:



(i) Applied error diffusion to neighboring pixels

Here, it ought to be noted that when distributing the error, I have used the modulo to avoid getting an index out of bounds. Alternatively, this could have been replaced by multiple if-statements. as the value of a pixel is only computed by the current error, the modulo will not impact any future errors of pixels, but will rather only impact the error of pixels that have already been seen. Thus, using modulo here is applicable and valid. The results on the test images are as follows:

(j) Test1 result error diffusion with threshold at 0.5

(k) Test2 result error diffusion 0.5

Already compared to the dithering process, we can observe that the images quality and details are much better preserved. Altercations to the simplistic approach are the weight distribution and the threshold. Clearly, no distinct pattern can be directly identified compared to the images before. However, when taking a closer look, it becomes obvious that the error diffusion has more error distribution in the vertical direction than the horizontal direction. This can be seen in the *images j and k*. When taking a closer look, we can see a slight pattern in the vertical direction.

3.1 Error diffusion with changed threshold

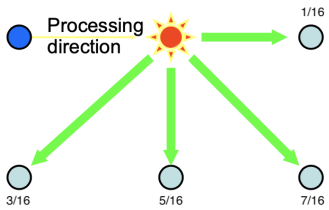
When setting the threshold to the approximate image intensity we get the following:

(l) Test1 result error diffusion with threshold at 0.36

As the image test2.pgm has an approximate image intensity of 0.5, we have already shown its effect in the previous iteration. It is interesting to observe that the thresholding change of 0.15 did not have a substantial impact on the image.

3.2 Error diffusion changed weight distribution

When changing the error diffusion according to the following:



(m) Applied error diffusion to neighboring pixels

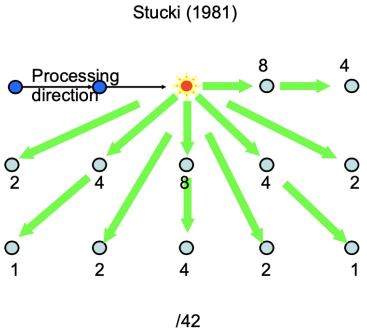
We get the following image results:

(n) Test1 result error diffusion with threshold at 0.5

(o) Test2 result error diffusion 0.5

4 Bonus: Extended Error Diffusion

Lastly, I have applied the extended error diffusion as seen in class according to and based on *Stucki*.



(p) Applied error diffusion to neighboring pixels - *Stucki* (1981)

The results of the images are as follows:

(q) Test1 result extended error diffusion based on *Stucki* (1981)

(r) Test2 result extended error diffusion based on *Stucki* (1981)

As can be observed, the more detailed the error diffusion gets, the better the resulting image.