State University of Campinas
Institute of Computer Science
MO443 - Professor: Hélio Pedrini

Project 2

May 10, 2021

Pedro Henrique Vaz Valois
p265676@dac.unicamp.br

***Abstract*—Digital images are abundant in the current age. Be it as means of modern art or source of data, they have a high degree of importance in modern society. Thus, the need for processing, transforming and understanding such images has risen and a multitude of techniques were developed for these tasks. In this project, we analyze and transform monochromatic figures in the Fourier space using the Fast Fourier Transform algorithm.**

I. Background

A monochromatic image is defined as a $M \times N$ matrix

$$I = (I_{mn}), 0 \leq m \leq M-1, 0 \leq n \leq N-1$$

where M is the image pixel width, N is the image pixel height and I_{mn} indicates the intensity in the $[0, 255]$ range for the pixel in position (m, n).

The convolution of an image I with a kernel k is given by $C = I * k$, that is simplified to a multiplication in the Fourier space

$$\begin{aligned} C = I * k &\implies \mathscr{F}(C) = \mathscr{F}(I)\mathscr{F}(k) \\ &\implies C = \mathscr{F}^{-1}(\mathscr{F}(I)\mathscr{F}(k)) \end{aligned}$$

where $\mathscr{F}$ represents the Fourier transform and $\mathscr{F}^{-1}$ its inverse.

With that in mind, this project uses two families of filters: ideal and Butterworth, with both defined in the Fourier space.

Finally, we use compression mechanism based on occurrence in Fourier space, in which an image is compressed by setting the intensity of uncommon frequencies to 0 in the Fourier space.

In that sense, it is possible to measure the amount of information that is lost by compression through Shannon's entropy equation

$$H = -\sum p_k log(p_k),$$

where $p_k = \frac{n_k}{n}$, n_k being the count of pixels with intensity k and n the total number of pixels.

II. Implementation

All code for this project has been developed and tested with `Python 3.7.9`, `numpy 1.20.2`, `scipy 1.6.3` and `matplotlib 3.4.1`.

The software that implements the methods discussed in the last section is contained in the `script.py` file. It depends on `numpy` to perform the vectorized operations, `scipy` for image rotation and `matplotlib` for reading and saving image files. Moreover, it uses python standard library's `os` module for checking if the output file already exists and the `argparse` module for parsing command line arguments.

Images and kernels are represented in the code as 2D `numpy` arrays in order to gain performance and also to simplify most math operations. All fourier space representations had its zero frequency centered at the middle of the image using the `numpy.fft.fftshif` function.

The `argparse` module provides a help text for using the script. By typing `python script.py --help`, a helper message is shown in the console.

The code was not designed to deal with non-monochromatic images, such as RGB files, and no tests were performed with such kind of data.

III. Results and Discussion

All tests were conducted on a PNG 512×512 monochromatic image of a butterfly. The original image and its fourier spectrum are depicted in figure 1 and the command line used to produce it is shown in listing 1

```
python script.py butterfly.png butterfly_fourier.png
```

Listing 1: Fourier transform of image

A. Filter convolution

The filters used are defined in the frequency space and they are divided in 4 kinds that can be chosen with the `-f` argument: `low`, `mid`, `rejectmid` or `high`.

1) Ideal filter: The ideal filter is defined as a circular range of frequencies either to be passed or rejected. In our implementation, it can be selected by setting `-fn ideal` and the size of the range can be defined by a float factor passed by `-ff`.

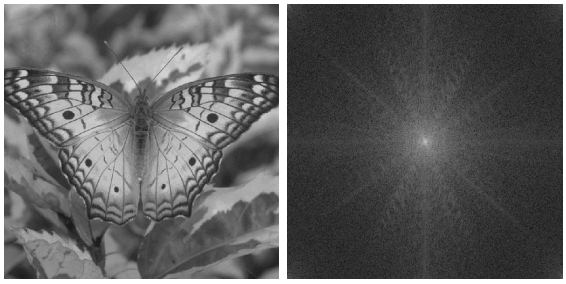

Figure 1: Original butterfly image and fourier representation centered in the middle

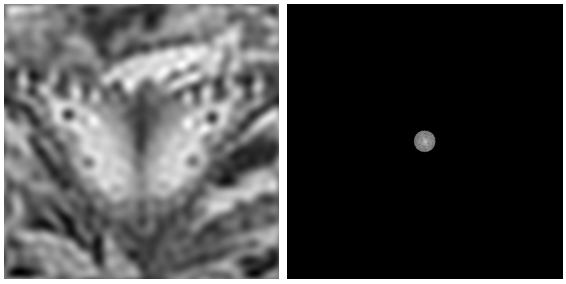

Figure 2: Low pass ideal filter and fourier representation with factor = 20

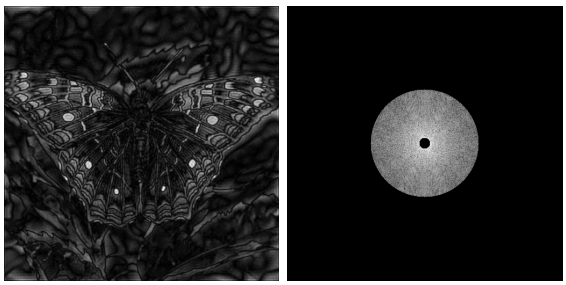

Figure 3: Mid pass ideal filter and fourier representation with factor = 10

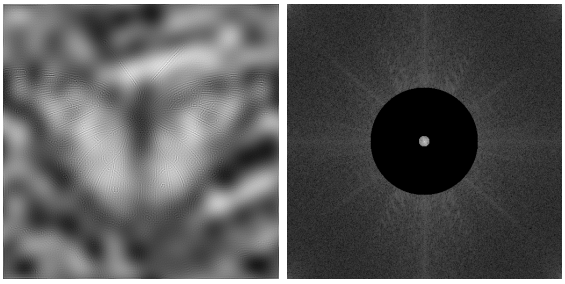

Figure 4: Reject mid pass ideal filter and fourier representation with factor = 10

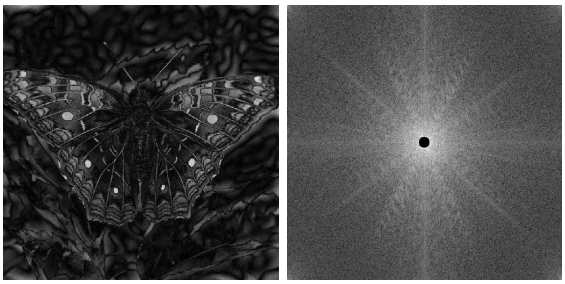

Figure 5: High pass ideal filter and fourier representation with factor = 10

On figure 2, we show the effect of the low pass filter where it is possible to see that the overall look of the image is kept, but small details are gone. On figure 3, we remove low and high frequencies, as visible in the Fourier spectrum, reducing the image color contrast in the sense most of it tends to black while highlighting the edges. At figure 4, we reject the frequencies used in the mid pass, resulting in an image that is mostly blurry but that keeps the small details of the strips of the butterfly. Finally, on figure 5, we make a high pass filter by removing the low frequencies and produce an image similar to the mid pass case, but where the edges are not too much highlighted.

2) Butterworth filter: The Butterworth filter is defined as a circular continuous distribution of frequencies either to be passed or rejected. In our implementation, it can be selected by setting `-fn butterworth` and the size of the circle can be defined by a float factor passed by `-ff`.

On figure 6, we show the effect of the low pass filter where intricate details are removed, appearing as a layer of fog sitting on top of the original figure. On figure 7, we use a thin circle to filter out low and high frequencies, as visible in the Fourier spectrum, what keeps the inner details of the butterfly while erasing all the rest. At figure 8, we reject circle from the mid pass, resulting in an image that removes specific details from the butterfly, but that overall looks like the original figure. Finally, on figure 9, we make a high pass filter that highlights the edges and details of the butterfly and the leaves at the bottom.

B. Compression

The compression algorithm filters frequencies with low occurrence out of the Fourier spectrum. As seen in figure 10, this greatly reduces the level of detail in the image, by giving it a sense of low granularity. This effect might resemble a low pass filter for high factors, as the close-to-zero frequencies

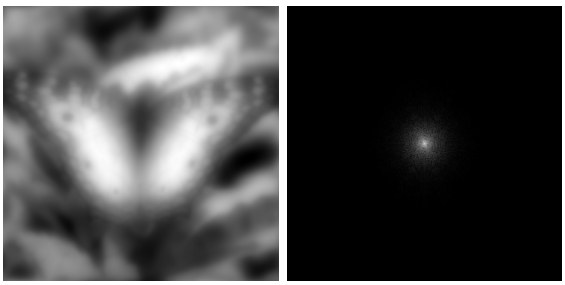

Figure 6: Low pass Butterworth filter and fourier representation with factor = 5

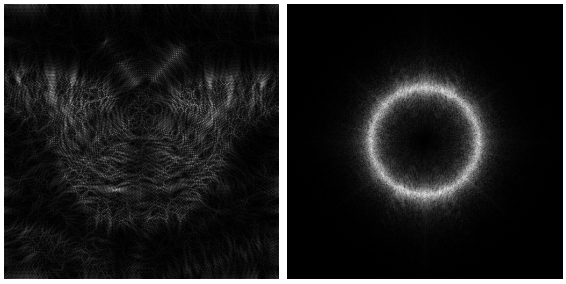

Figure 7: Mid pass Butterworth filter and fourier representation with factor = 100

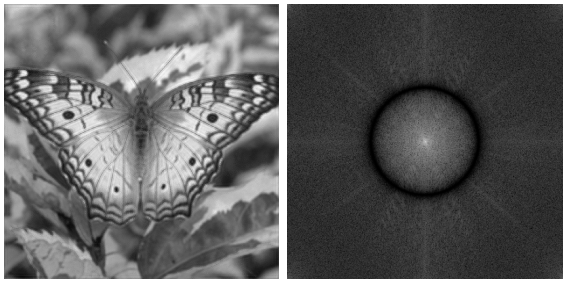

Figure 8: Reject Mid pass Butterworth filter and fourier representation with factor = 100

have higher count than the others. On the other hand, using a lower factor, it is possible to see that high frequencies are not removed while some low frequencies are in specific locations, evidencing the difference between filters and compression, what makes it possible to use both simultaneously.

The idea of lossy compression is closely related with the idea of removing undesired information on data. Therefore, it is possible to measure the effects of compression with Shannon entropy equation. In figure 11, we see that the compression slowly reduces the image entropy, until the evolution starts to become more sudden after factor = 2000 and a high drop appears beyond factor = 4000. This shows that the compression keeps most of the information until it is impossible not to and, comparing it with figure 10, it is visible that the butterfly silhouette is still present until factor = 1000 whereas factor = 2000 could be taken as blur.

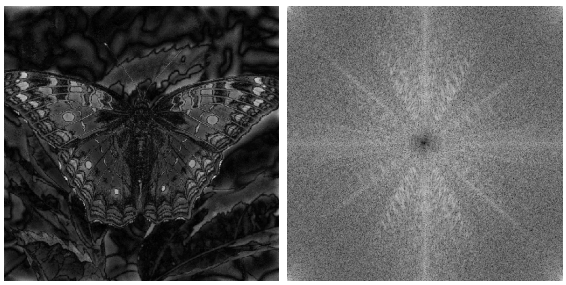

Figure 9: High pass Butterworth filter and fourier representation with factor = 100

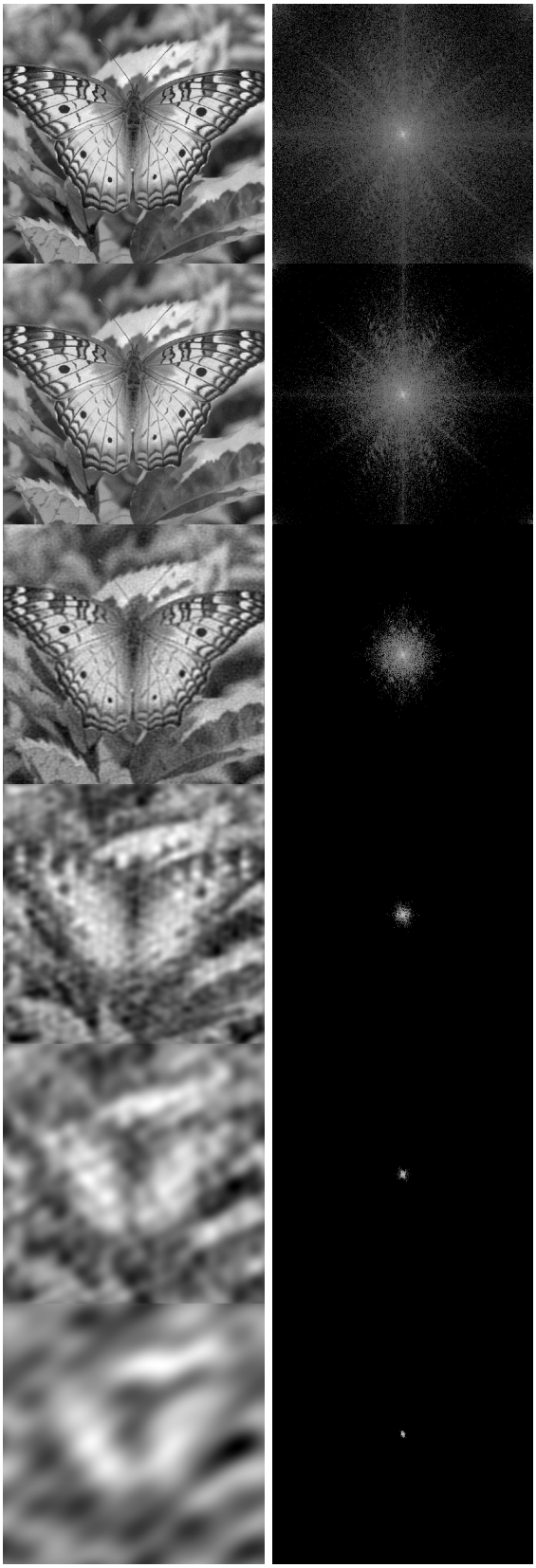

Figure 10: Image compression and fourier representation with factors 10, 25, 100, 500, 1000 and 2000 from top to bottom. It is possible to see that both high and low frequencies are kept in the compressed image with factor = 10 and 25

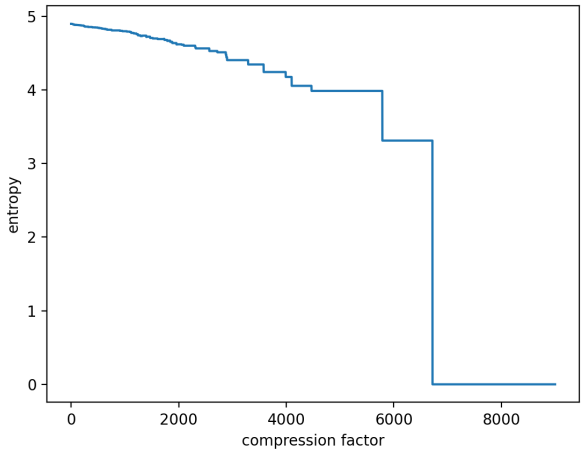

Figure 11: Evolution of image entropy versus image compression on butterfly.png

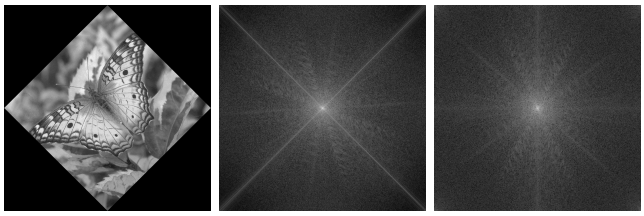

Figure 12: Image rotation by 45 degrees (left), rotated fourier representation (middle) and original fourier spectrum (right)

C. Rotation

The rotation of an image changes the directions of all displayed information and rotating by 45 degrees adds 4 black triangles in the borders of the image. These changes directly impact the result of the fourier transform, as visible on figure 12: Two strong diagonal lines are created and the vertical line from the original spectrum disappears. Notice that these diagonals cross the center with a 45 degree angle, just like the angle used in the image rotation.

IV. Conclusion

In this project, Fourier transform was applied as means of optimizing, simplifying and analyzing some common image transformations. The image frequency spectrum allows for convolving figures with kernels by directly filtering frequencies in or out. Moreover, it is possible to compress the amount of information by nullifying uncommon frequencies. Also, all transformations were developed with a vectorized approach, thus leading to fast executions on all tests.