Data science in fractional Fourier transform

Pedro José Albarracín Vargas

---

1 Introduction

Currently the area of optical information encryption attracts the attention of research centers, this because the contributions presented in the last two decades have shown the reliability, versatility and applicability of optical encryption systems. In an optical encryption scheme is proposed dubbed "double random phase encoding" which involves multiplying by two random phases in the input plane and in the Fourier domain. It can be shown if these random phases are statistically independent white noises then the encrypted image is also a stationary white noise. The random phase key located at the Fourier plane serves as the only key in this encryption scheme but it was not long until the extra degree of freedom offered by the Fractional Fourier transform (FRFT) was utilised as a new key in similar encryption schemes, which shall be discussed shortly. This document mentions the advantage of having at hand an instrument that illustrates, compares data and quickly approximates the construction of FRFT. Nevertheless, despite contributions from data science, many digital realizations of the FRFT still lack properties of the continuous FRFT. Its practical applications are expected to grow significantly in years to come, given that the FRFT offers many advantages over the traditional Fourier analysis. It concludes with a discussion of possible uses. In doing so, this work illustrates how Fourier's approach remains modern and universal and shows how his ideas, essentially derived from physics, then enriched by mathematics, computer science and signal theory, continue to be essential to address current data science challenges.

The outline of this paper is as follows: In the section 2, we present the notion of fractional Fourier Transform (FRFT) and something special function (see [3]). Then, we explain how optical encryption works by double random phase encoding using a simulation with grayscale image. We announce the interpolations (sinc and Lagrange) that achieve an approximation to the fractional. In the section 3, the matlab program is used to elaborate the two interpolations and note the advantages they have for each FRFT order a and for two dimension transform, by an appropriate adjustment to the order of a, it can identify in the plot, the neighborhood where the fading of the disturbance occurs in the image. Finally, we briefly discussed the use of data science in this mathematical topic, the need to improve or sufficiency regarding auxiliary functions or libraries.

2 The Fractional Fourier Transform

Definition 1 (Continuous fractional Fourier Transform) *[3, Definition 4.1.1] The fractional Fourier transform $\mathcal{F}^a$ of order $a \in \mathbb{R}$ is a linear integral operator that maps a given function (signal) $f(x), x \in \mathbb{R}$ onto $f_a(u), u \in \mathbb{R}$ by*

$$f_a(u) = \mathcal{F}^a(f) = \int_{-\infty}^{+\infty} K_a(u', u) f(u') du'$$

where the kernel is defined as follows. Set $\alpha = \dfrac{a\pi}{2}$ *then*

$$K_a(u,u') = A_\alpha exp\{-\pi i \left(2\csc\alpha uu' - (u-u')^2\cot\alpha\right)\}$$

with

$$A_\alpha = \sqrt{1 - i\cot\alpha}.$$

We need to know about chirp functions for some application of fractional Fourier transform. A chirp is a function that sweeps a certain frequency interval $[\omega_0, \omega_1]$ in a certain time interval $[t_0, t_1]$. If the sweep rate is linear, it has the form $\exp\{i\pi(\chi x + \gamma)x\}$ with χ the sweep rate.

Optical implement

The fundamental idea is to encrypt the information as white noise by using two random phase masks, located in the input plane and in the Fourier plane of the first lens. This process is completely reversible only if the Fourier plane mask is known when the information is amplitude, and both masks if the information is complex or phase.
The input image used in the simulation is a 256×256 sized grayscale (levels ranging from 0 to 255). The real value function denotes the original two-dimensional image of Fourier.

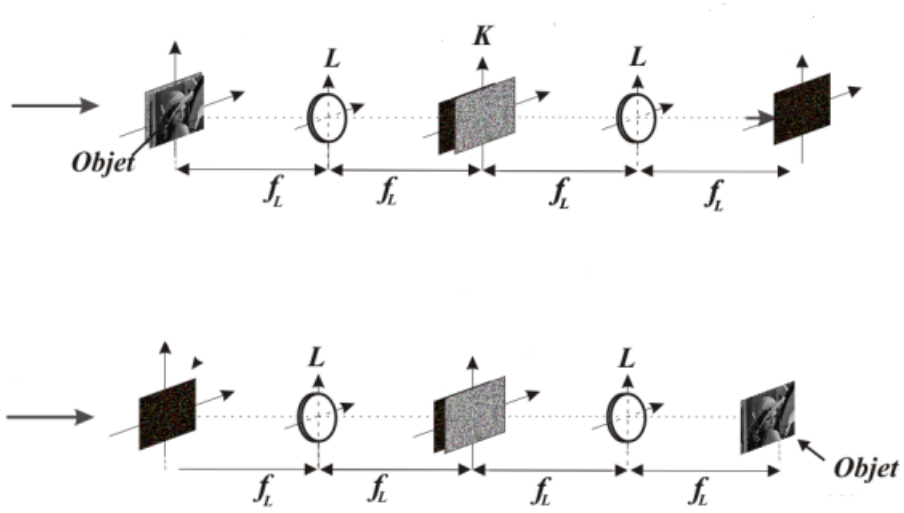

Figure 1: dual phase mask encoding

The original image is multiplied by a random phase function and then computes the Fourier transform. In the next step, the Fourier transform of the image is multiplied by another random phase mask, which is statistically different independent of the previous one. Then the inverse Fourier transform is performed on this image to obey the encrypted image in the dormindio space. The encrypted image is retrieved using CCD camera. During the decryption process (Figure 1), the encrypted image is the multiplied Fourier transform with the conjugate complex of the second random phase function. The image obtained is multiplied by the Fourier inverse transform and the decrypted image is obtained.

In the Figure 2 (a) shows the original image before projecting it onto the lens. Then, in Figure (b) white noise appears due to the first random phase function. In Figure 2 (c) the encryption is terminated by a second function. In case of finding the inverse the encryption with an erroneous key, in Figure 2 (d) holds white noise, but if we consider the appropriate key, the result of decryption process is again the initial image. Therefore, the two random phase images used during the encryption process act as keys to secure the data during decryption.

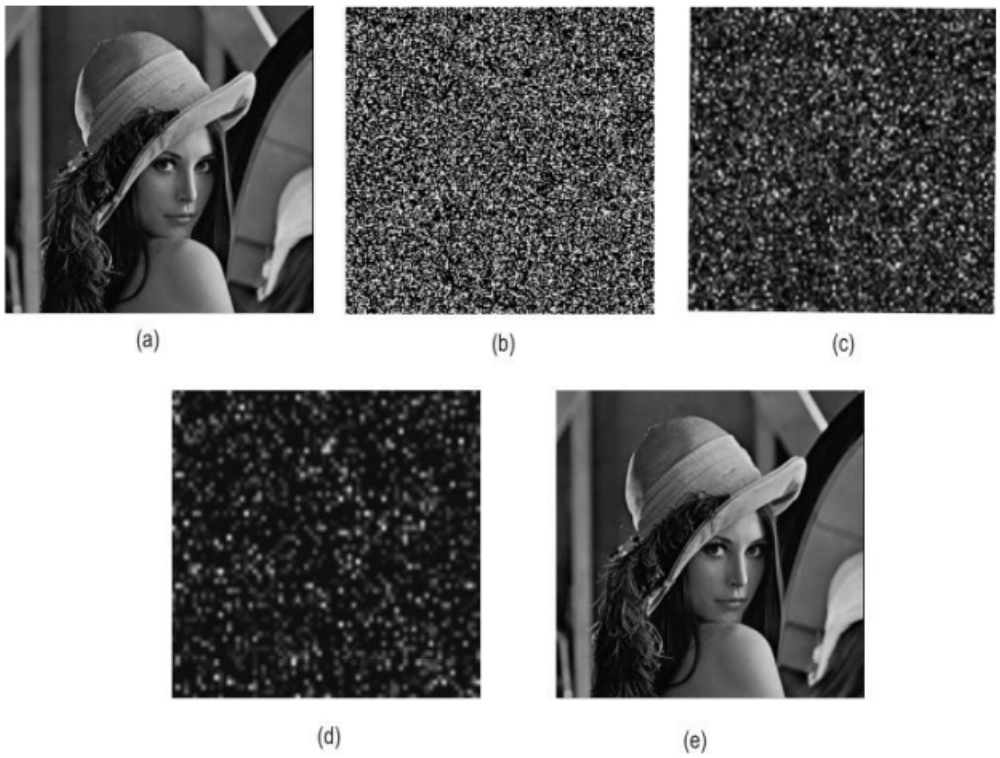

Figure 2: (a) Original image, (b) security key (random phase mask), (c) encrypted image, (d) decrypted image when it uses a different key and (e) decrypted when it uses correct key.

Now, we must consider a tool that allows illustrates the values of fractional transform when a changes, so that it is a good approximation for continuous franctional Fourier transform $\mathcal{F}^a f$. The algorithms considered here are algorithms that approximate the continuous fractional Fourier transform in the sense that they map samples of the signal to samples of the samples of the continuous fractional Fourier transform. Because it uses FFT techniques with complexity $N \log N$, it is a fast algorithm. However, it is not really the fractional analog of the Fast Fourier Transform (FFT) which is a particular fast implementation of the Discrete Fourier Transform (DFT)(see [1]). And although it implements just the FFT when the power $a = 1$ and the inverse FFT when $a = 3$, and although some authors have coined it fast fractional Fourier transforms, we deliberately avoid to use this term, and call it Fast Approximate Fractional Fourier Transform (FAFrFT). For the moment, the fast implementation of the discrete fractional Fourier transform which would be the genuine fast fractional Fourier transform (FFrFT), is yet unknown.

3 How to implement Data Science

The idea of the algorithm is the definition in the form of a convolution in between two chirp multiplications. Using $\cot\alpha - \csc\alpha = -\tan(\alpha/2)$, then

$$f_a(\xi) = A_\alpha e^{-i\pi\tan(\alpha/2)\xi^2} \int_{-\infty}^{+\infty} e^{i\pi\csc\alpha(\xi-x)^2}[e^{-i\pi\tan(\alpha/2)x^2} f(x)]dx.$$

Assuming that the functions can be limited to the interval $[-\Delta/2, \Delta/2]$ in all time–frequency directions. For a sufficiently large percentage of the energy of the signal is contained in that circle, For any signal, this assumption can be justified by choosing Δ sufficiently large. It is in our interest to choose it as small as possible to reduce computational complexity. For convenience, we will require Δ to be an integer. if we want to recover the result from discrete samples, then we should have samples at intervals $1/2\Delta$. Thus, assuming we have $N = \Delta^2$ samples of the original f, then we need at least $2N$ samples of the convolution. So the integral sampled at $\xi_k = k/2\Delta$ can be approximated as

$$f_a(\xi_k) \approx \frac{A_\alpha}{2\Delta} e^{-i\pi x_k^2\tan(\alpha/2)} \sum_{l=-N}^{N-1} g(k-l)h(l), \tag{1}$$

with $x_k = k/2\Delta$, and

$$g(k) = e^{i\pi x_k^2\csc\alpha}, \qquad h(k) = e^{-i\pi x_k^2\tan(\alpha/2)} f(x_k).$$

This gives a discrete approximation of the continuous transform. The convolution has $O(N\log N)$ complexity when using FFT. Therefore, it is called a fast approximate fractional Fourier transform. To understand the algorithm, we will first need a multiplication of f with a chirp function,

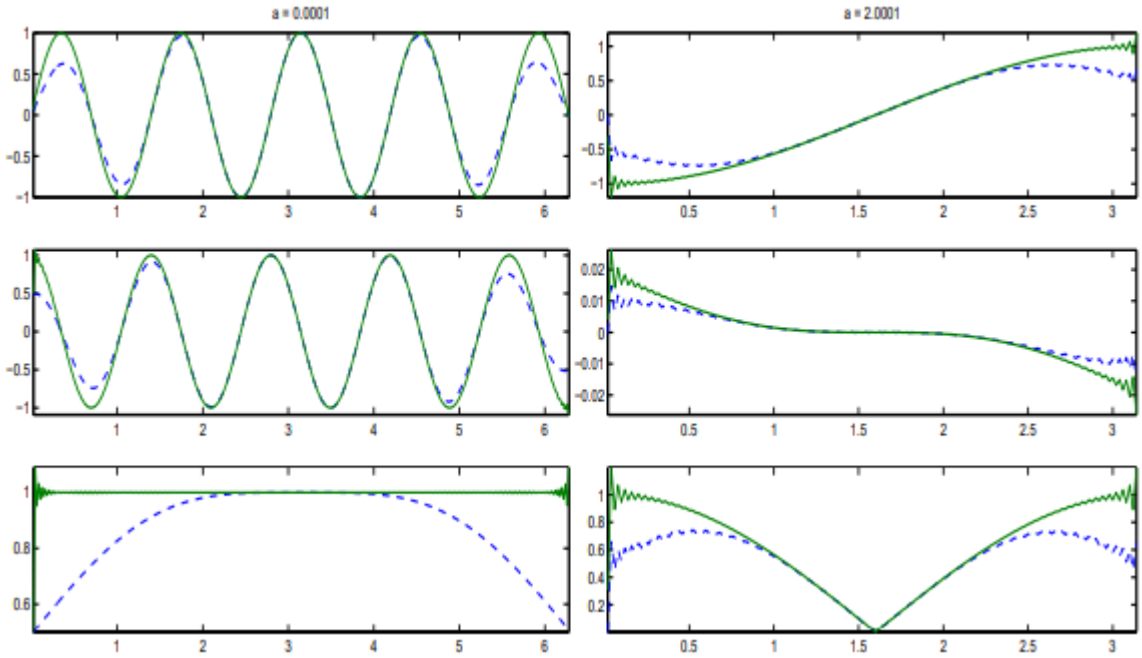

Figure 3: Real part, imaginary part and absolute value of $\mathcal{F}^{0.0001}(e^{-i(4.5x+\pi/2)})$, $x \in [0, 2\pi]$ (left) and of $\mathcal{F}^{2.0001}(\cos(x)), x \in [0, \pi]$ (right). Solid line for sinc interpolation, dashed line for Lagrange interpolation.

multiply their Fourier transforms and then, multiply with a chirp. If E_c represents the chirp function e^{icx^2} , then the approximation (1) can be written as

$$\mathcal{F}^a f \approx \frac{A_\alpha}{2\Delta} E_c \mathcal{F}^{-1}\{\mathcal{F}[E_d f] \cdot \mathcal{F}[E_c f]\}, \tag{2}$$

where $c = \cot\alpha - \csc\alpha = -\tan(\alpha/2)$ and $d = \csc\alpha$.

Once the approximation algorithm has been proposed, we use sinc interpolation and Lagrange interpolation. It's useful MATLAB because is a digital aid support, has the necessary integrated development functions, access to libraries that allow to compare the differences between the interpolations presented.

The library `FTFAT` (Large Time-Frequency Analysis Toolbox) is available for computation. the routines `fracF` and `fracft` can to compute the interpolation algorithm by either a convolution of the signal with a sinc function, if the number of partitions N for interval of a is large, it might seem to be faster to use Lagrange interpolation. For instance, if we want to compute $\mathcal{F}^{0.0001}(f)$, where f is the vector containing the samples of e^{ix} computed in the interval $x \in [0, 2\pi]$ with steps 0.01. Recall that this is computed as $\mathcal{F}^{1.0001}(\mathcal{F}^{-1}(f))$. Thus, the first step is to compute $f_{-1} = \mathcal{F}^{-1}(f)$ which should be a vector containing the samples of a delta-function. Next, the general routine is applied to compute the transform $f_{0.0001} = F^{1.0001}(f_{-1})$. This is approximately the inverse transform, and one would expect the absolute value of $f_{0.0001}$ to be approximately equal to the constant function 1. In this process, f_{-1} is upsampled by interpolation. When Lagrange interpolation is used, the fractional transform will concentrate most of its energy in the middle and deteriorate near the boundary of the interval. On the other hand, if $a \in \{0, 1, 2, 3\}$ are treated before the interpolation, the FFT and inverse FFT is computed using the general routine for the fractional Fourier transform which results in a transform $F^2(\cos(x))$, that looks much like the transform using sinc interpolation, as is clearly seen in Fig. 3.

Two-dimensional transform

To compute the discrete fractional Fourier transform, we have to compute the transformation matrix, and then multiply the given vector with this matrix. The computational complexity is $O(n^2)$, which is higher than for the fast approximate fractional Fourier transform. We

Figure 4:

superposed a chirp noise on an image as can be seen in the left of Figure 5. After the appropriate one-dimensional discrete fractional Fourier transforms on the rows and the columns, the chirp is transformed to a delta-function. This can be seen in Figure 4, where we have plotted a mesh for the rows and columns in the neighborhood of the delta-function.

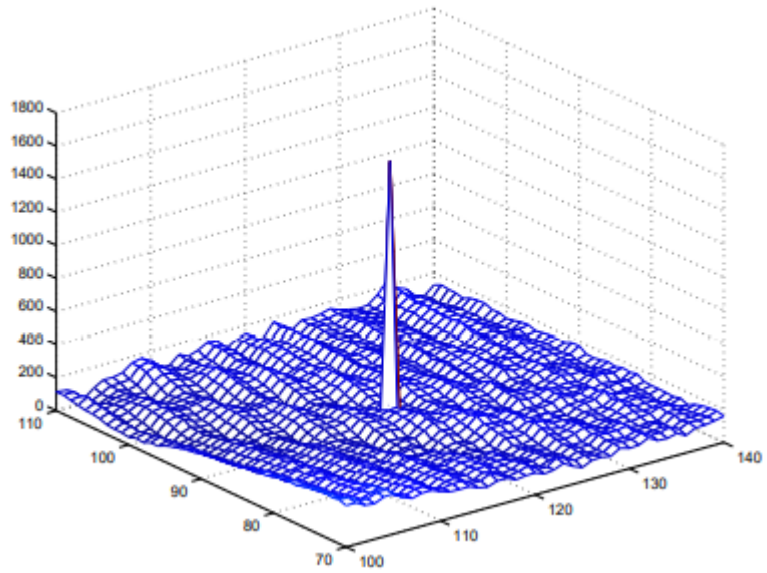

Figure 5:

Even though the noise is hardly seen in the image, the peak clearly stands out and this identifies that the appropriate transformation has been made. After removal by replacing the peak by the average on the neighboring pixels, the image is back transformed and a clean image is found. The library for this algorithm can to be found in the link [2], along with library and code `dFRT`.

4 Conclusion

We can consider the fractional Fourier transform as an option for information security, as the two statistically different random phase images act as keys to keep the image safe. Implementing other resources that complement the work makes possible a better understanding of the fractional Fourier transform. It differs under which conditions of FFRT of order a interval sinc interpolation or Lagrange is most suitable. Also, as a graph viewer, it helps to detect the concentration of chirp on an image and thus get a cleaner result.

References

[1] Adhemar Bultheel and Héctor E Martınez Sulbaran. "Computation of the fractional Fourier transform". In: *Applied and Computational Harmonic Analysis* 16.3 (2004), pp. 182–202.

[2] Cagatay Candan, M Alper Kutay, and Haldun M Ozaktas. "A Matlab program". In: (1996). URL: `https://nalag.cs.kuleuven.be/research/software/FRFT/`.

[3] Haldun M Ozaktas and M Alper Kutay. "The fractional Fourier transform". In: *2001 European Control Conference (ECC)*. IEEE. 2001, pp. 1477–1483.