Applications of the fractional Fourier transform

Cary Yang

Carnegie Mellon University
caryy@andrew.cmu.edu

Abstract

Images are widely used to convey information in an easy to understand format; however, the conventional representation of an image as 3 channels per pixel per image creates difficulty in implementing certain widely used classes of algorithms. In this paper, we summarize two different applications of the fractional Fourier transform, which transforms the representation of the data in the image and therefore allows us to perform novel techniques in this new domain. This allows for effective and efficient solutions to problems that were difficult or inefficient to solve previously.

1 Introduction

Effective and efficient image processing algorithms for certain domains of problems are difficult due to the traditional representation of image as intensity at a position, which fails to capture that pixel's relationship with other pixels and colors in the image. In this paper, we investigate the application of the fractional Fourier transform to 2 different areas of image processing and highlight the benefits of these applications.

1.1 Image Encryption

Compared to plain text, images have significantly more information, have more local similarities, and are less sensitive to mutations in the data. For example, if we replaced every other letter in a sentence with a random character, it would be extremely difficult to piece back together the original meaning, as in the following:

```
Tsenqjigklbeomnyfkxtjdmbet yvvrftke
llzy yoo.
The quick brown fox jumped over the
lazy dog.
```

However, if we changed every other pixel in an image to a random color, the image would still be easily discernible, as in Figure 1.

(a) *Modified image.* **(b)** *Original image.*

Figure 1: *Comparing a picture with every other pixel changed to a random color with the original image.*

This same idea holds for encryption algorithms as well.

Additionally, because images contain a significant amount of data, conventional encryption techniques that work well on plain-text can take significant amounts of time on images.

1.2 Noise Removal

Another difficult image processing problem to efficiently solve given the traditional representation of an image is noise removal, since noise only makes sense in relation to neighboring pixels. One way to deal with noise that has a constant frequency throughout the image is the Fourier transform; however, for noise that varies in frequency in the image, the fractional Fourier transform must be used instead.

In this paper, we will present summaries to solutions for both of these problems:

- We will give intuitive and in-depth explanations of the more esoteric mathematical concepts that are key to understanding the algorithm proposed (Section 2).
- Give a brief intuitive explanation of the algorithm before diving into the actual implementation of the algorithm and showing that it solves the problems outlined above (Sections 3.1 and 3.2 for image encryption, Sections 4.1 and 4.2 for noise removal).
- Analyze the cryptographic effectiveness of the algorithm being proposed and show that it is reasonably difficult to decrypt the image without the proper keys (Section 3.3).
- Show the effectiveness of the noise removal algorithm, especially in comparison to the noise removal applied with the ordinary Fourier transform (Section 4.3).

2 Overview of Mathematical Concepts

Many mathematical concepts that this paper relies on will be described in detail in the following sections. Only working knowledge of concepts up to calculus is assumed.

2.1 Fourier Transform

The Fourier transform is a mathematical transformation that converts between two representations of a signal:

- The time-domain representation, or the value of the signal as a function of time.
- The frequency-domain representation, or the amplitude and phase of the signal as a function of frequency.

Note that phase has the same definition here as it does for waves – it is the angle offset at which we begin drawing a wave.

We can express the (continuous) Fourier transform $\mathcal{F}$ of a function $f : \mathbb{R} \to \mathbb{C}$, where the units of ξ are hertz (Hz), or cycles per second, as:

$$F(\xi) = \int_{-\infty}^{\infty} f(t) e^{-i2\pi\xi t} dt \tag{1}$$

And its sibling, the (continuous) Inverse Fourier transform:

$$f(t) = \int_{-\infty}^{\infty} F(\xi) e^{i2\pi\xi t} dt \tag{2}$$

These equations are rather dense, so we will give an intuitive explanation of the Fourier transform before diving into the actual mathematics surrounding the equation.

Consider the function $f(t) = \sin(3t) + \sin(7t) + \sin(12t)$, graphed in Figure 2:

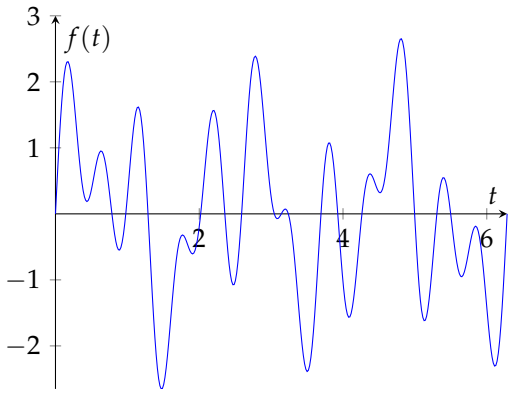

Figure 2: *Graph of* $\sin(3t) + \sin(7t) + \sin(12t)$.

This function is composed of 3 sin waves with frequencies of 3 Hz, 7 Hz, and 12 Hz and a 0° phase angle.

Now, consider the same equation graphed in the polar coordinate system instead, using t as the angular coordinate (Figure 3).

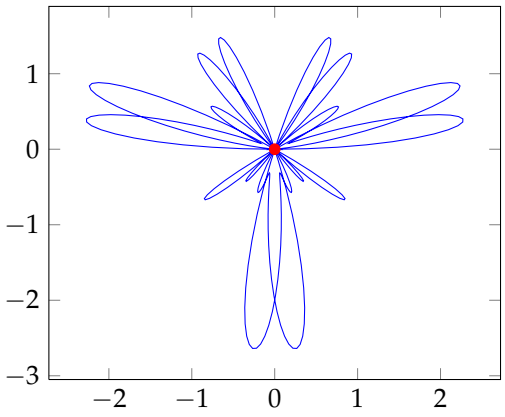

Figure 3: *Graph of* $\theta = t$, $r = \sin(3t) + \sin(7t) + \sin(12t)$.

Note that if we were to take the average of the location of all the points on that line, we would end up with the red point, which is squarely at the origin.

Next, consider the previous graph, but drawn 3 times as fast. That is, given a point $(f(t), t)$ from the previous graph, graph it now at $(f(t), 3t)$, effectively making 3 complete cycles as we graph the same equation (Figure 4).

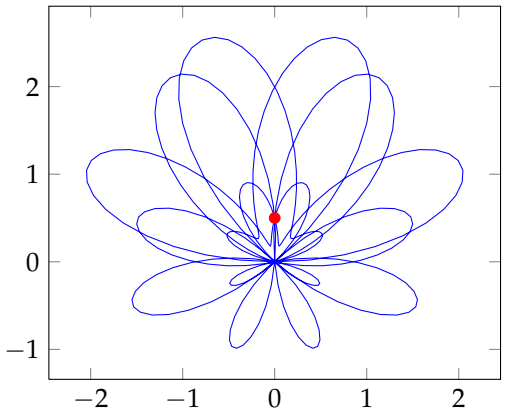

Figure 4: *Graph of* $\theta = 3t$, $r = \sin(3t) + \sin(7t) + \sin(12t)$

Note that this time, the average location of all the points ends up a distance of 0.5 from the origin and 0° angle from the vertical. Coincidentally, the equation $f(t)$ that we're graphing has a sin component with a frequency of 3 Hz that has unit amplitude and a 0° phase angle!

Why isn't the distance to the origin of this point exactly the amplitude? The mathematical reason for this is a bit complicated, but it turns out that the Fourier transform splits the amplitude of a wave of some frequency evenly between the transformed function's value at some frequency, and its value at that same negative frequency. That is, if we graphed all points of the form $(f(t), -3t)$ in the polar coordinate system, we would get the same graph reflected over the y-axis and find the average point again at 0.5 distance from the origin. Summing these two values gives us the original amplitude of 1.

Thus, we have derived the general procedure for the Fourier transform of a function $f(t)$: given a frequency in hertz ξ, it essentially finds the average location of all the points along $f(t)$ when it is graphed on the polar axis ξ times as fast as normal. The distance of this average location from the origin is half the amplitude, and its angle from the vertical is the phase angle.

Now, we'll go deeper into the mathematics surrounding the equation.

First, we know by Euler's Theorem that we can represent any complex number in polar coordinates as:

$$z = re^{i\theta} \tag{3}$$

Using the analogy of complex numbers being coordinates on a plane, we can see that to graph a function $f(t)$ in the polar coordinate system on the complex plane with angular coordinate t, we would have the function:

$$g(t) = f(t)e^{it} \tag{4}$$

Now, if we wanted to find average of the location of all the points on this curve, we would just take the integral over all time values with respect to time. However, this would only give us the amplitude and phase angle of the signal if we drew it at normal speed. To get the amplitude and phase angle at any frequency, we would have to add a $2\pi\xi$ term in the exponent, where ξ would be the frequency at which we

sample the signal. Thus, we arrive at:

$$F(\xi) = \int_{-\infty}^{\infty} f(t) e^{i2\pi\xi t} dt \tag{5}$$

Note that Equation 5 is almost exactly the Fourier transformation from Equation 2, and it follows the intuition discussed before exactly! The only difference occurs in the exponent, which is negative in the original presentation of the Fourier transform.

As it turns out, this negative sign is a matter of convention, and can go on either the forward or inverse Fourier transform as long as it is kept consistent [10]. Intuitively, we can think of it as following the circles in one direction in the forward transform and the opposite direction in the inverse transform, reversing the operation. Because most texts express the Fourier transform with the negative sign on the forward direction, we will do the same.

One detail of importance is that because of the $e^{-i2\pi\xi t}$ component in the Fourier transform, the result $F(\xi) = |F(\xi)|e^{i\theta(\xi)}$ is a complex number, where the magnitude $|F(\xi)|$ is the amplitude of the signal at the given frequency and $\theta(\xi)$ is the phase angle.

Note that above, we only go into the mathematics of the Fourier transform. Computationally, evaluating the full integral is somewhat expensive, but can be estimated using the Fast Fourier Transform algorithm in $O(n \log n)$ asymptotic time. However, continuous Fourier transforms can also be physically computed *instantly* with spectrometers or prisms, which has interesting implications for the runtime of applications that use the Fourier transform if special purpose hardware is designed that takes advantage of this fact.

2.2 Fractional Fourier Transform

A generalization of the Fourier transform is the fractional Fourier transform (FrFT). Recall that the continuous Fourier transform converts between the time- or spatial-domain representation of a signal and the frequency-domain representation of a signal. If we consider these two domains as orthogonal, then the continuous Fourier transform can be thought of as rotating a signal $\pi/2$ radians from its time- or spatial-domain representation to its frequency-domain representation. Generalizing this to any angle of rotation, the fractional Fourier transform can transform a function to intermediate domains *between* the time- or spatial-domain and frequency domain.

To put it more formally, let the a-th order FrFT of a function $f(x)$ be denoted as $\mathcal{F}_a\{f(x)\}(x_a)$. Note that the a-th order FrFT corresponds to an $\phi = a\pi/2$ radian rotation in the time-frequency plane. From our above discussion, we can see that $\mathcal{F}_1$ is simply the continuous Fourier transform. Going further, we can see that $\mathcal{F}_2$ of a function $f(t)$ would rotate the function into the frequency domain and then back into the time domain, mapping the function $f(t) \mapsto f(-t)$. $\mathcal{F}_3$ would simply be the Fourier transform of this inverse time domain, and $\mathcal{F}_4$ would rotate a full circle and result in the original function.

The mathematical definition is conventionally expressed as:

$$\mathcal{F}_a\{f(x)\}(x_a) = \int_{-\infty}^{\infty} K_a(x, x_a) f(x)\, \mathrm{d}x \tag{6}$$

Where $K_a(x, x_a)$, the fractional kernel, is defined as:

$$K_a(x, x_a) = \begin{cases} A_\phi \exp[i\pi(x^2 \cot\phi \\ \quad -2xx_a + x_a^2 \cot\phi)] & 0 < |a| < 2 \\ \delta(x - x_a) & a = 0 \\ \delta(x + x_a) & a = \pm 2 \end{cases},$$
$$A_\phi = \exp[-i\pi \mathrm{sgn}(\sin\phi)/4 + i\phi/2],$$
$$\phi = a\pi/2, \text{and}$$
$$\delta \text{ is the Dirac delta function.} \tag{7}$$

While the above equation is included for completeness, it is not particularly intuitive. Fortunately, this paper relies only on certain properties of the FrFT.

One important property to note is the addi-

tive property of the FrFT. Formally,

$$\mathcal{F}_b\{\mathcal{F}_a\{f\}\} = \mathcal{F}_{a+b}\{f\} = \mathcal{F}_a\{\mathcal{F}_b\{f\}\} \quad (8)$$

Intuitively, we can explain the above as one rotation of angle $\alpha = a\pi/2$ in the time-frequency plane followed by another rotation of angle $\beta = b\pi/2$. By the basic properties of rotations, these can be combined into a single rotation of angle $\alpha + \beta$, or applied in the reverse order, and still describe the same transformation.

Additionally, fractional Fourier Transforms are unitary, which means, among other things, that the inner product of two values is preserved.

Also, note that because the FrFT is a generalization of the Fourier transform, it too can be computed in real-time with special purpose optics hardware. Therefore, any algorithm that uses this transformation has the potential for massive speedups if this hardware is used.

2.3 Two-Dimensional Fractional Fourier Transform

Because this paper deals with images, the functions we deal with will be two-dimensional functions. Fortunately, the above definition of the FrFT naturally extends into multiple dimensions. Namely, for the two-dimensional case, we have:

$$\mathcal{F}_{a_x,a_y}\{f(x,y)\}(x_{a_x}, y_{a_y}) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} K_{a_x}(x, x_{a_x})K_{a_y}(y, y_{a_y})f(x,y)\ \mathrm{d}x\ \mathrm{d}y \quad (9)$$

Note that this is simply the multiplication of two fractional kernels with possibly different fractional orders, one in each dimension.

3 Image Encryption

3.1 The Idea

Given the FrFT, we can perform image transformations that cannot be undone unless the correct fractional order is known.

Therefore, one idea to encrypt an image might be to apply the FrFT on it. However, because the FrFT is periodic over the range of fractional orders 0 to 4, it might be possible to brute force the correct fractional order in a short amount of time. A logical follow-up to this would be to iteratively apply FrFT's on the image. However, this cannot be done directly since FrFT's have an additive property, as discussed previously, and therefore, multiple FrFT's applied on an image one after another would be the same as a single FrFT.

Therefore, for our above idea to work, we should perform some transformation in each fractional plane before applying the next FrFT so that the fractional orders cannot be simply added. This paper will use the Jigsaw transformation, which takes an image, divides it into tiles of the same size, and then randomly permutes these tiles. For example, in Figure 6a, the original 64x64 image was broken into 8x8 tiles, and then these tiles were permuted randomly to produce the arrangement seen. Because a total ordering can be put on all the different possible permutations, we can encode the Jigsaw transformation as a single number, which would be the index into the list of all possible permutations in their total ordering.

We also need to multiply the original image by a random phase mask before the repeated Jigsaw transformations and FrFT's are applied. The math for this will be discussed in the next section, but what is important to note is that multiplying by this random phase mask essentially turns the image into white noise, removing the correlation between nearby pixels that is common in images, without changing the amplitude of a function.

Given the above definitions, the procedure to encrypt the image proceeds as follows [3]:

1. Multiply the image to be encrypted by a random phase mask.
2. For $i = 1...3$
3. Apply the jigsaw transformation with index b_i.

4. Then, apply a FrFT of order a_i. Note that this is in reality two FrFT, one in the x direction and one in the y direction, and each with their own fractional orders a_{ix} and a_{iy}.

In reality, we could continue this loop further and further, but the diminishing returns in security and increasing computation time make this option unattractive.

The decryption process would be essentially the same procedure in the opposite direction, that is:

1. For $i = 3...1$
2. Apply a FrFT of order $-a_i$.
3. Reverse the jigsaw transformation with index b_i.
4. Use the amplitude at each location as the color of the pixel in the decrypted image.

Note that we do not need to know the random phase mask used during the encryption phase to decrypt the image since the amplitude of the image at a pixel is the color of that pixel, and the phase mask does not change the amplitude of the pixel. However, the phase mask is still necessary to whiten the image to be encrypted and remove the effect that the sharp color differences between jigsaw pieces would have on the FrFT result. Without this phase mask, the encryption result would look like Figure 6f, which has undesirable patterns in the noise.

Therefore, the decryption process requires the knowledge of 9 keys in total: 3 indices for the 3 jigsaw transformations, and 6 fractional orders, 1 in x and 1 in y for each FrFT.

3.2 Detailed Description

We will go over the algorithm outlined above in more detail and firmly develop a mathematical basis for our claims.

First, to make this section more concise, denote the image with a function of one input $f(x)$. Note that all of the below results would still apply given the conventional representation of an image as a function of two inputs since the FrFT naturally extends to two dimensions, as discussed previously.

Next, we present the mathematical definition of the random phase mask described earlier, which is simply Equation 10, for some function $n(x)$ that generates random numbers uniformly distributed in $[0, 1)$ based on its input parameter.

$$e^{i2\pi n(x)} \tag{10}$$

We can see that if we multiply $f(x)$ by this phase mask, we get the polar representation of a complex number in the complex plane, with an amplitude of $f(x)$ and a phase angle of $2\pi n(x)$. Because the phase angle is random, any correlations between adjacent pixels is erased, and the image can be considered white noise after this step.

Now, denote the Jigsaw transform with index b as $J_b\{\}$ and its inverse as $J_{-b}\{\}$. If we apply it to the image multiplied by the random phase mask, we have Equation 11.

$$J_{b1}\{f(x)e^{i2\pi n(x)}\} \tag{11}$$

Now, apply the FrFT of order a_1. Recall that this is actually two transformations, one in the x direction and one in the y direction with two different fractional orders a_{1x} and a_{1y}. This results in Equation 12.

$$\mathcal{F}_{a1}\{J_{b1}\{f(x)e^{i2\pi n(x)}\}\} \tag{12}$$

Now, repeating this procedure twice more, we get the final encrypted image $g(x)$:

$$g(x) = \mathcal{F}_{a3}\{J_{b3}\{\mathcal{F}_{a2}\{J_{b2}\{\mathcal{F}_{a1}\{J_{b1}\{f(x)e^{i2\pi n(x)}\}\}\}\}\}\} \tag{13}$$

We can see the result of this equation in Figure 6c when $a_{1x} = a_{1y} = a_{2x} = a_{2y} = a_{3x} = a_{3y} = 0.5$.

Decryption following our intuition is given by Equation 14.

$$f(x) = J_{-b1}\{\mathcal{F}_{-a1}\{J_{-b2}\{\mathcal{F}_{-a2}\{J_{-b3}\{\mathcal{F}_{-a3}\{g(x)\}\}\}\}\}\} \tag{14}$$

Note that as before, we can leave out the random phase mask from our decryption step since we can just take the amplitude of the function as the color of each pixel.

3.3 Results

Performance of this encryption algorithm is analyzed in terms of Mean Squared Error (MSE) between the original image and the encrypted image. Mathematically [4],

$$\begin{aligned} \text{MSE} &= ||\text{in} - \text{out}||^2 \\ &= \frac{1}{N^2} \sum_{i=1}^{n} \sum_{j=1}^{n} |\text{out}(i,j) - \text{in}(i,j)|^2 \end{aligned} \tag{15}$$

To visualize the sensitivity of the fractional order parameters a_{1x}, a_{2x}, and a_{3x} (or, equivalently, a_{x1}, a_{x2}, and a_{x3}), we have included in Figure 5 a plot of the MSE of decryption attempts as a function of that parameter's difference from the correct value, where, in each case, all other parameters used are correct. Changes in a_{1x} are graphed as the thickest line, changes in a_{2x} are graphed as the next thickest line, and changes in a_{3x} are graphed as the thinnest line. While not depicted, changes in a_{1y}, a_{2y}, and a_{3y} exhibit the same behavior as the changes in a_{x1}, a_{x2}, and a_{x3} respectively in the graph.

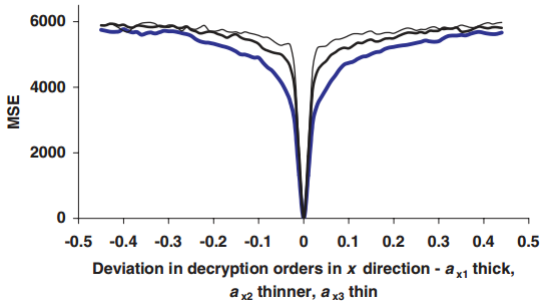

Figure 5

As we can see from the attempted decryption in Figure 6e and from the graph above, an error of 0.05 in a fractional order still results in a completely encrypted image. An individual attempting to brute force the keys would have to go at a resolution of at least 0.05 through the range of fractional orders $[0, 4)$, which means they would have to test $4/0.05 = 80$ values for each of the 6 fractional orders, resulting in $80^6 = 262,144,000,000$ combination of values to test, which would take a significant amount of time.

On top of this, there are still the permutation indices to get correct. With just 64 tiles, there are $64! = 1.27 \times 10^{89}$ possible permutations, a search space that is far outside the reach of any brute force program. In fact, the jigsaw problem has been show to be NP-complete [1]. If just one of the three applied permutations is incorrect, as in Figure 6d, the resulting image is still completely encrypted. Thus, given the above two calculations, it is likely that brute-forcing an image encrypted with the technique outlined in this paper is entirely unfeasible.

One final point to note here is that this algorithm can be implemented very efficiently in hardware, and in fact, the original paper proposes a schematic for a possible optical implementation [3]. Because the Jigsaw transformation portions of this algorithm are very easy to compute in software, and the difficult FrFT can be computed with optical hardware at essentially instantaneous speeds, this algorithm has the potential to perform extremely well, even on large images.

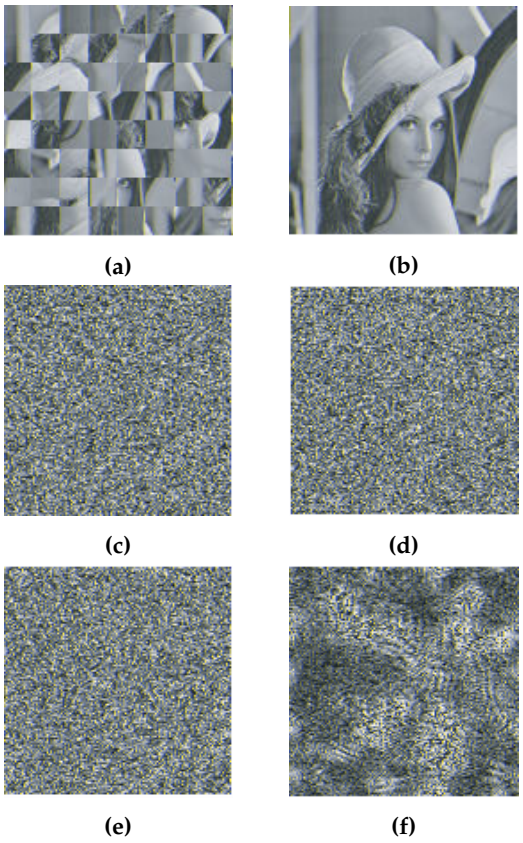

Figure 6: ***a)*** *An example of the Jigsaw transformation applied on the original image.* ***b)*** *Correctly decrypted image, MSE = 0.00.* ***c)*** *Fully encrypted image,* $a_{1x} = a_{1y} = a_{2x} = a_{2y} = a_{3x} = a_{3y} = 0.5$*, MSE = 3943.14.* ***d)*** *Incorrectly decrypted with randomly incorrect permutation for* b_3*, MSE = 5441.89.* ***e)*** *Incorrectly decrypted with* $a_{1x} = a_{1y} = a_{2x} = a_{2y} = a_{3y} = 0.5$ *and* $a_{3x} = 0.55$*, MSE = 5690.98.* ***f)*** *Encrypted without the initial random phase mask multiplication step, MSE = 5937.99.*

4 Noise Removal

4.1 The Idea

Because the FrFT can transform a function into domains between the time- or spatial-domain and the frequency-domain, it naturally lends itself to simplifying applications where the signal's frequency varies over time or space. Of particular interest is the reduction of noise in images that is not constant throughout.

Note that the ordinary Fourier transform is excellent at removing noise that is constant through an image, simply by transforming the image into the frequency domain, removing the culprit frequencies, and then transforming back. A similar idea can be applied with the FrFT, but without the limitation that the noise is constant. As we can see in Figure **??**, if we rotate the function in the time-frequency plane by angle a, then we can perform a simple multiplicative filter to remove the noise as in the ordinary Fourier transform case and then transform the image back.

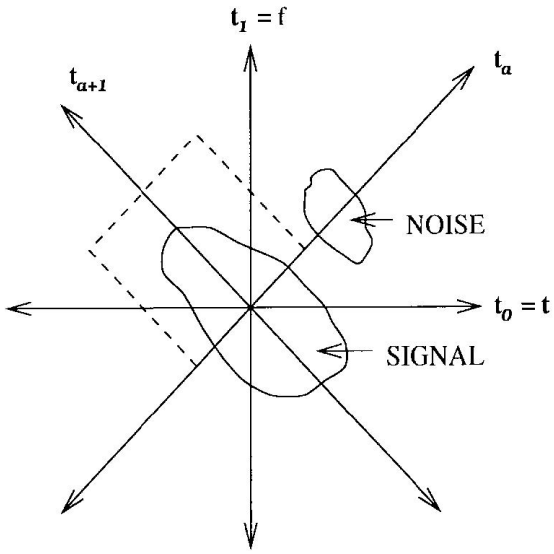

Figure 7: *Removing noise from an image with the FrFT.*

Thus, we have a general procedure for our noise removal algorithm.

4.2 Detailed Description

Now, for a more formal representation of the algorithm, suppose we have a signal represented by $f_{obs}(u)$ that we want to remove noise from. We estimate the original signal $f(u)$ with a filtering operation in a fractional Fourier domain, represented by the following equation:

$$f_{est}(u) = (\mathcal{F}_{-a} \circ \Lambda_g \circ \mathcal{F}_a) f_{obs}(u) \qquad (16)$$

Where Λ_g represents the operator corresponding to multiplication by a filter function $g(u)$, and a is the optimal fractional Fourier domain for this filtering. Thus, to optimally estimate the original signal, we must find the optimal filtering function $g_{opt}(u)$ and the optimal fractional Fourier domain for the filtering.

Solving for $g_{opt}(u)$ analytically, we find that [6],

$$g_{opt}(u) = \frac{\int\int K_a(u,u_a)K_{-a}(u',u_a)R_{ff_{obs}}(u,u')\,\mathrm{d}u'\,\mathrm{d}u}{\int\int K_a(u,u_a)K_{-a}(u',u_a)R_{f_{obs}f_{obs}}(u,u')\,\mathrm{d}u'\,\mathrm{d}u} \tag{17}$$

Where the correlation functions $R_{ff_{obs}}(u,u')$ and $R_{f_{obs}f_{obs}}(u,u')$ can be computed from the auto-correlation functions $R_{ff}(u,u')$ and $R_{nn}(u,u')$ of the signal and the noise, which are assumed to be known.

The derivation of this optimal function is a bit complex, but it relies on a couple of fairly simple concepts.

Recall from statistics that the mean-squared error of an estimation f_{est} of a signal f is defined as:

$$\begin{aligned}\mathrm{MSE}(f_{est}) &= E[||f - f_{est}||^2] \\ &= E[\langle f - f_{est}, f - f_{est}\rangle]\end{aligned} \tag{18}$$

Where $\langle\cdot\rangle$ represents the inner product and $||\cdot||$ represents the norm.

Because we are trying to remove the noise from our observed signal, it makes sense that our objective would be to minimize the value of this function.

Additionally, since the fractional Fourier transform is unitary, as discussed earlier, the MSE applies directly in the fractional Fourier domain a:

$$\begin{aligned}\mathrm{MSE}(f_{est}) &= E[||f_a - f_{est_a}||^2] \\ \mathrm{MSE}(g_{opt}) &= E[||f_a - g_{opt}f_{obs_a}||^2]\end{aligned} \tag{19}$$

Where f_a, f_{est_a}, and f_{obs_a} represent the functions f, f_{est}, and f_{obs} respectively in the fractional Fourier domain a, and the second definitions follows by our definition of f_{est}.

Expanding this definition by distributing and linearity of expectations, we have:

$$\begin{aligned}\mathrm{MSE}(g_{opt}) = E[(f_a)^2] - 2g_{opt}E[f_a f_{obs_a}] \\ + (g_{opt})^2 E[(f_{obs_a})^2]\end{aligned} \tag{20}$$

Taking the derivative of this with respect to g_{opt} and setting it to 0 to find critical points, we have:

$$0 = -2E[f_a f_{obs_a}] + 2g_{opt}E[(f_{obs_a})^2] \tag{21}$$

And solving for g_{opt}:

$$g_{opt} = \frac{E[f_a f_{obs_a}]}{E[(f_{obs_a})^2]} = \frac{R_{f_a f_{obs_a}}}{R_{f_{obs_a} f_{obs_a}}} \tag{22}$$

However, we only have $R_{ff_{obs}}$ and $R_{f_{obs}f_{obs}}$ and not $R_{f_a f_{obs_a}}$ and $R_{f_{obs_a} f_{obs_a}}$, so transform it to the spatial domain, then back to the fractional domain a, giving us exactly Equation 17!

Also note that the second derivative is $2E[(f_{obs_a})^2]$, which will always be positive, and therefore, we have indeed found a local minimum. Because it is the only critical point, it is also the absolute minimum of the MSE function.

To find a, the optimal fractional Fourier domain to apply this filter, we simply test all fractional Fourier domains in $[0,4)$ at some fixed resolution and pick the one that results in the greatest reduction of noise.

4.3 Results

Similar to the evaluation of our results from image encryption, we will also evaluate the effectiveness of the noise removal by measure of MSE. However, in this case, the MSE values will be normalized by $||\text{in}||^2$, resulting in the equation:

$$\mathrm{MSE} = \frac{||\text{in} - \text{out}||^2}{||\text{in}||^2} \tag{23}$$

Our first case study will be Figure 8, the image of a plane that is intentionally corrupted, and then recovered using the algorithm discussed above. As we can see, the method described is significantly more effective than the

ordinary Fourier transform, resulting in an almost 10-fold improvement.

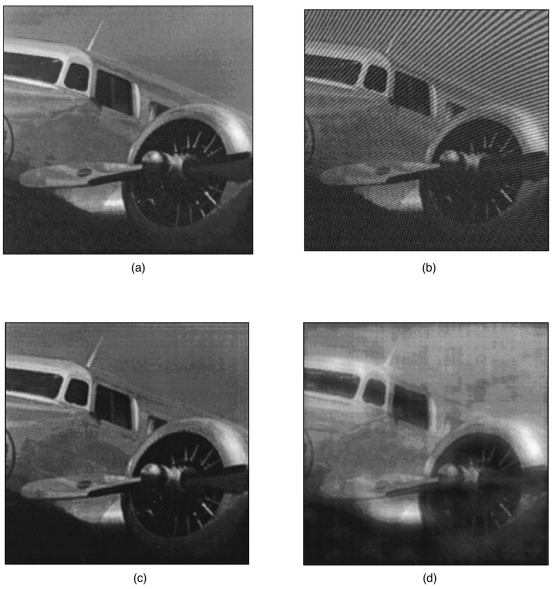

Figure 8: ***a)*** *Original image.* ***b)*** *Image with added noise, SNR (Signal to Noise Ratio)* ≈ 1. ***c)*** *Filter in the optimum fractional Fourier domain* ($a_x = 0.4, a_y = -0.6$)*, MSE = 0.003.* ***d)*** *Filtered in the ordinary Fourier domain, MSE = 0.035.*

Thus, we can see that filtering with the fractional Fourier transform produces significant benefits in image quality over filtering with the standard Fourier transform for many types of noise with no additional computational cost. While not depicted, the original paper tested a couple more of image degradation, all with superior results for the FrFT method over the ordinary Fourier transform method [6].

5 Further Work

The application of fractional Fourier transforms to image encryption has been further extended to color image encryption [5] and the simultaneous encryption of two images [9], along with the usage of different transformations in the fractional Fourier domain [4].

Additionally, the fractional Fourier transform has been adapted to a variety of other fields, such as digital watermarking, image compression, and more [8].

References

[1] Erik D Demaine and Martin L Demaine. Jigsaw puzzles, edge matching, and polyomino packing: Connections and complexity. *Graphs and Combinatorics*, 23(1):195–208, 2007.

[2] B Hennelly and John T Sheridan. Optical image encryption by random shifting in fractional fourier domains. *Optics Letters*, 28(4):269–271, 2003.

[3] Bryan M Hennelly and John T Sheridan. Image encryption and the fractional fourier transform. *Optik-International Journal for Light and Electron Optics*, 114(6):251–265, 2003.

[4] Madhusudan Joshi, Kehar Singh, et al. Color image encryption and decryption using fractional fourier transform. *Optics communications*, 279(1):35–42, 2007.

[5] M Alper Kutay and Haldun M Ozaktas. Optimal image restoration with the fractional fourier transform. *JOSA A*, 15(4):825–833, 1998.

[6] Ervin Sejdić, Igor Djurović, and LJubiša Stanković. Fractional fourier transform as a signal processing tool: An overview of recent developments. *Signal Processing*, 91(6):1351–1369, 2011.

[7] Ran Tao, Yi Xin, and Yue Wang. Double image encryption based on random phase encoding in the fractional fourier domain. *Optics Express*, 15(24):16067–16079, 2007.

[8] T Theul. Sampling and reconstruction in volume visualization. *See at: http://www.cg.tuwien.ac.at/˜theussl/DA/ thesis.html*, page 28, 2000.