How to Implement the Gerchberg–Saxton Algorithm for 3D Printing of Diffractive Optical Element with Nanoscribe

Ye Pu

Introduction

In a wide range of imaging applications, spanning electron microscopy, astronomy, crystallography, three-dimensional imaging, and many other fields, it is often desired to retrieve the whole wave front, although only intensity can be measured in optics due to the extremely high frequency of light. Wave fronts are mathematical described with complex numbers as $A(x)=|A(x)|\exp[i\varphi(x)]$, where $A(x)$ is the complex amplitude representing the wave front, $|A(x)|$ is the non-negative modulus of $A(x)$ which is simply called the amplitude of the wave front, $\varphi(x)$ is the phase, and x is the spatial coordinates. The wave front amplitude $|A(x)|$ is linked to its intensity $I(x)$ through $I(x)=|A(x)|^2$ and can be obtained through measurements. The phase, on the other hand, is not directly measurable.

The determination of the phase generally requires interferometric methods, which is termed holography in imaging. However, interferometric approaches are usually complex in setup, require coherent light sources, are sensitive to numerous experimental conditions such as vibrations and misalignments, and are thus difficult to conduct. One is, therefore, interested in determining the two variables in a wave front, $|A(x)|$ and $\varphi(x)$, using intensity measurements made at two sufficiently separated locations. The problem of retrieving phase information from intensity measurements is generally termed the phase retrieval problem. It is a typical “ill-posed” problem, meaning that the solution is not unique and is highly sensitive to small changes or noise in the input.

The Gerchberg–Saxton algorithm [1] is a particularly successful approach to solving the phase retrieval problem, where one measurement is performed in the object domain and the other made in the Fourier domain. The algorithm relies on iterative Fourier-inverse Fourier transforms back and forth between the object and Fourier domains with the application of the measured data in each domain. It can be generalized to use measurements from any two planes related by diffraction other than a Fourier pair. Fienup [2] later made further extensions to the Gerchberg–Saxton algorithm to use incomplete data, such as known constraints rather than physical measurements. A great number of related methods have been developed to address a large variety of problems [3,4].

In the design of Diffractive Optical Elements (DOEs), usually the target image (the spatial intensity pattern that the DOE is required to produce) and the DOE complex transmittance or reflectance are a Fourier transform pair. Most often, the DOE is a pure phase mask with a unitary amplitude in its transmittance or reflectance. Therefore, the Gerchberg–Saxton algorithm can be effectively applied using these two known conditions as the measurements.

Algorithm Description

The principle of the Gerchberg–Saxton algorithm is shown in Figure 1, which operates alternately in the object-domain $A = |A|\exp(i\varphi)$, where the target image resides, and the Fourier-domain $F = |F|\exp(i\Phi)$, where the Fourier transform of the image lies. The algorithm first initializes with an initial guess using the target amplitude and an arbitrary initial phase, which can be either a zero or a random matrix. The system then performs the *k*th (*k* = 0, 1, 2, 3, ...) iteration in the following sequence: 1. Construct A_k from $|A_k|$ and φ_k. 2. Compute the Fourier transform of A_k to enter the Fourier domain F_k. 3. Construct F_k' using the measured Fourier amplitude $|F_k'|$, which in the case of DOE is a unitary matrix, and the phase $\Phi_k' = \Phi_k$. 4. Compute the inverse Fourier transform of F_k' to return to the object domain A_k'. 5. Let $|A_{k+1}| = |A_0|$ and $\varphi_{k+1} = \varphi_k'$, and repeat steps 1-5 until convergence criteria are met.

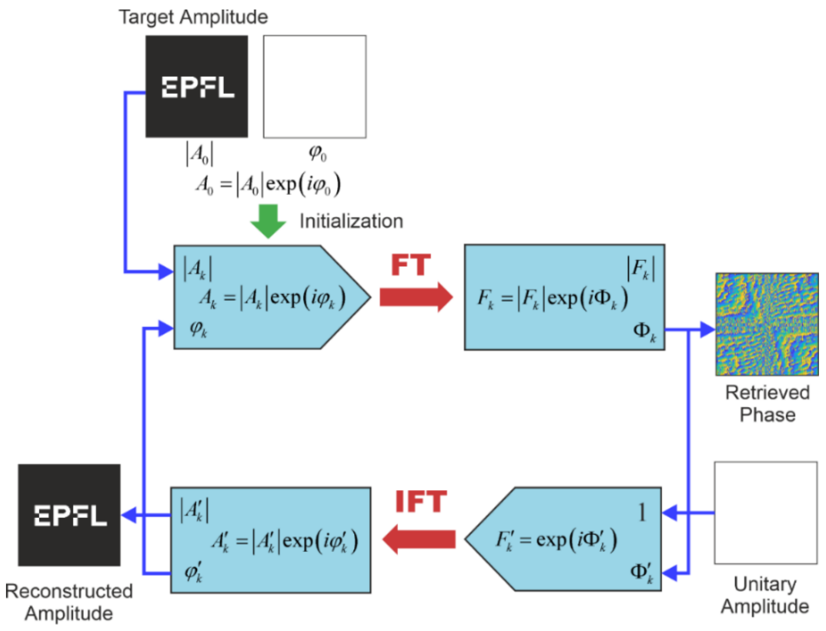

Figure 1. Diagram of the Gerchberg–Saxton Algorithm for the design of DOE. The algorithm initializes with an initial guessed phase and iterates over Fourier-inverse Fourier transforms

back and forth between the object and Fourier domains to gradually approach the required image output and the retrieved phase.

Despite the ill-posed nature of the phase retrieval problem, this simple iterative method usually provides a reasonable solution.

Implementation

Follow the steps outlined below for the implementation of the Gerchberg–Saxton algorithm. You can choose either MATLAB or Python as your programming language in your implementation. Note that all variables are implicitly matrices, reflecting their nature of a function of spatial coordinates.

1. Calculate the target image amplitude from the target image intensity.
2. Pre-transform of target image amplitude using ifftshift for the initial amplitude. See explanation below.
3. Generate a random or uniform matrix as the initial phase.
4. Construct the initial complex amplitude from the initial amplitude and phase.
5. Perform Fourier transform over the constructed complex amplitude.
6. Separate the resulting complex Fourier matrix into amplitude and phase.
7. Construct a new complex Fourier matrix with the resulting phase and a unitary amplitude.
8. Perform inverse Fourier transform over the new complex Fourier matrix.
9. Separate the resulting complex image matrix into amplitude (which is also the output matrix in each iteration) and phase.
10. Construct a new complex image matrix with the resulting phase and the target image amplitude.
11. Repeat steps 5-11 until a predetermined number of iterations is reached or the error is smaller than a predetermined level.

Figure 2 shows an example of the program output in a particular MATLAB implementation of the Gerchberg–Saxton algorithm.

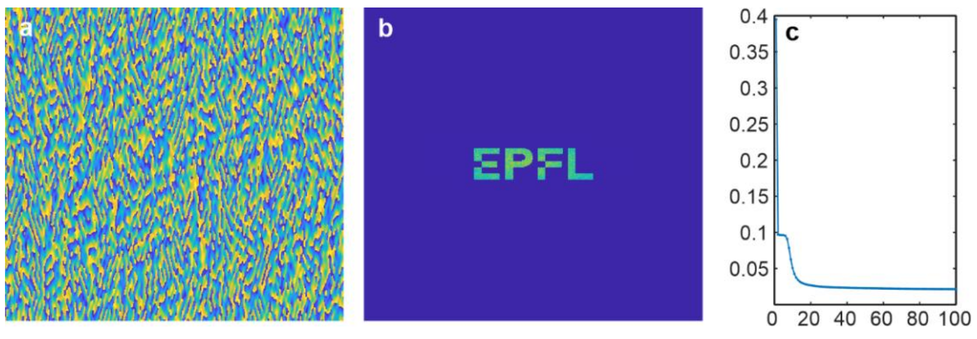

Figure 2. Example output from a particular MATLAB implementation of the Gerchberg–Saxton algorithm. (a) Retrieved phase. (b) Output image. (c) Relative error as a function of the number of iterations.

Usually running the algorithm for a certain number of iterations (for example, 100) is sufficient to obtain a good result. A relative error can be formulated by comparing the output image with the target, which can be used as a criterion of convergence. The calculation of the relative error must consider careful normalizations to keep the variables at the same scale.

Special attention should be paid to the subtle difference between a numerical Fourier transform and an optical Fourier transform. In an optical Fourier transform, the zero frequency is located at the center of an image, while in numerical Fourier transform, such as the fast Fourier transform routine provided in MATLAB or Python, the zero frequency is located at the corner. Therefore, an invocation of the ifftshift routine should be performed on the target image to shift the zero position to the corner. This will ensure that the optically reconstructed image from the DOE is one copy rather than four copies.

Once the phase is retrieved, it is ready to be saved into an image file of proper format for printing. An additional implementation detail for printing with Nanoscribe is that the practical printable area is limited to approximately 1 mm^2 (512×512 pixels at 2 μm pixel size) because of the printing time. Since the laser beam is larger than this size, some of the beam will directly pass, forming a bright spot at the center of the reconstructed image. This can be addressed by placing the pattern to be printed at an off-center position.

References

1. R. W. Gerchberg and W. O. Saxton, "A Practical Algorithm for the Determination of Phase from Image and Diffraction Plane Pictures", Optik **35**, 237-246 (1972).
2. J. R. Fienup, "Phase Retrieval Algorithms: A Comparison", Appl. Opt. **21**, 2758-2769 (1982).
3. J. R. Fienup, "Reconstruction and synthesis applications of an iterative algorithm", in *Transformations in Optical Signal Processing*, 23-25 Feb. 1981 Seattle, WA, USA, Proc SPIE 373, 147-160 (1984).
4. K. Jaganathan, Y. C. Eldar, and B. Hassibi, "Phase retrieval: An overview of recent developments", in *Optical Compressive Imaging*, pp. 279-312 (2016).