Supplementary Information for

Redefinable Neural Network for Structured Light Array

Hengyang Li[1,†], Jiaming Xu[1,†], Huaizhi Zhang[1], Cong Hu[1], Zining Wan[2], Yu Xiao[1], Xiahui Tang[1], Chenhao Wan[1], Gang Xu[1,*], Yingxiong Qin[1,*]

1, School of Optical and Electronic Information & Wuhan National Laboratory for Optoelectronics, Huazhong University of Science and Technology, Optics Valley Laboratory, National Engineering Research Center for Laser Processing, Wuhan 430074, China
2, Neuroscience and Intelligent Media Institute, Communication University of China, Beijing 100024, China

Correspondence:
Gang Xu. gang_xu@hust.edu.cn
Yingxiong Qin. qyx@hust.edu.cn

The PDF file includes:

Supplementary Text
Figs. S1 to S6
Table. S1

1. Mathematical interpretation and implementation approach

1.1 Mathematical derivation of RediNet

In the section "The architecture of RediNet" in the main article, we gave some mathematical representations of the solutions for solving Fourier series primitive function from parameter space. Due to the limitation of writing length, these explanations mainly emphasize conclusive formulations, which leads to some skip-overs. Here, we present the relevant rigorous mathematical procedure in detail.

The essence of the problem is: a multi-dimensional parameter space P is given, which can be recognized as a real-valued and non-negative multi-dimensional matrix. The values $a_{l_1,\cdots,l_N}$ of this matrix are the moduli of the coefficients of a multi-dimensional Fourier series, where $l_1,\cdots,l_N$ are the matrix element order numbers and at the same time the Fourier series term order numbers. It is required to find a corresponding multi-dimensional primitive function, whose moduli are required to be all 1, and whose angle function S should fulfill the criterion that the moduli of the coefficients of this primitive function's Fourier series equal to the moduli of the $a_{l_1,\cdots,l_N}$.

Let us start from the simplest case. Suppose that we only need to modulate a single multi-dimensional structured light beam, and this structured light corresponds to N independent structured light properties, such as x, y, z shiftings, topological charge l, etc., as mentioned in the main article. Also, note that the CPFs $\alpha_i(x,y)$ corresponding to these structured light properties are linear functions of the modulation parameter, and independent with each other. At this point, the phase pattern of modulating this multi-dimensional structured light beam should be the superposition of the multiplication of the CPFs and their modulation parameters, i.e:

$$\exp\left(j\varphi(x,y)\right)=\exp\left(j\Phi(l_1,\ldots,l_N)\right)=\exp\left(j\sum_{i=1}^{N}\alpha_i(x,y)l_i\right), \tag{S1}$$

Where $\alpha_i(x,y)$ is the corresponding CPFs. Interpretively, the term on the far right $\sum_{i=1}^{N}\alpha_i(x,y)l_i$ can be regarded as a function $\Phi(l_1,\cdots,l_N)$ of the modulation parameter l_i, and can also be understood as a 2D complex amplitude distribution $\varphi(x,y)$ in terms of the underlying independent variables x and y.

Next, based on the idea of designing Dammann gratings, we consider the overall modulation phase for simultaneously generating multiple structured light beams as a weighted sum of multiple separated modulation phases corresponding to the individual structured light beams:

$$\begin{aligned}\exp[j\varphi(x,y)]&=\sum_{l_1=-\infty}^{\infty}\cdots\sum_{l_N=-\infty}^{\infty}a_{l_1,\ldots,l_N}\times\exp\left(j\Phi(l_1,\ldots,l_N)\right)\\&=\sum_{l_1=-\infty}^{\infty}\cdots\sum_{l_N=-\infty}^{\infty}a_{l_1,\ldots,l_N}\times\exp\left(j\sum_{i=1}^{N}\alpha_i(x,y)l_i\right)\end{aligned}, \tag{S2}$$

where $a_{l_1,\cdots,l_N}$ are the complex weights, whose squared moduli represent the intensity of each beam, and their angles are undetermined. These angles are crucial to $\exp[j\varphi(x,y)]$ because it is hard and ill-posed to realize our target that for every x and

y, the right part of Eq. S2 could offer a perfect value that has a modulus of 1, which is required by the left part of Eq. S2.

As mentioned in the main text, the primitive function $S(t_1,\cdots,t_N)$, strictly speaking is the angle part of the primitive function taking the values in parameter space as the Fourier series coefficients' moduli. We give its Fourier series expansion from the direct mathematical definition:

$$\exp[iS(t_1,...,t_N)] = \sum_{l_1=-\infty}^{\infty}\cdots\sum_{l_N=-\infty}^{\infty} a_{l_1,...,l_N} \times \exp(j\sum_{i=1}^{N} t_i l_i)\,, \tag{S3}$$

Formally, this equation is very similar to the Eq. S2. If there is $\alpha_i(x,y) = t_i$, then

$$\exp[j\varphi(x,y)] = \exp[jS(\alpha_1(x,y),...,\alpha_N(x,y))]\,. \tag{S4}$$

This means that once we have the CPF $\alpha_i(x,y)$ and the solved $S(t_1,\cdots,t_N)$ through the neural network, we can evaluate $\alpha_i(x,y)$, and further use them as coordinates t_i to evaluate $S(t_1,\cdots,t_N)$. This process is similar to composite function evaluation.

1.2 Acquisition of the dataset

In order to train a neural network, datasets, or say the ground truth of the input and output of the network are needed. The inputs and outputs of RediNet's neural network are the 3D parameter space (with a resolution of 8^3) and the 3D primitive function (the angle of the Fourier series' primitive function) as mentioned in the main text. In fact, the process from the given 3D parameter space to the primitive function is a difficult problem, whereas its inverse problem is simple because given an arbitrary 3D function, one can directly obtain its parameter space by Fourier series expansion. After tests, we found the results obtained by this method were unsatisfactory.

For this issue, here we use the following process: firstly, in a parameter space whose elements are all equal to zero, a part of the elements (at least 2, and at most 32) are randomly chosen and assigned to 1. Then, we use an iterative method to compute the primitive function, which can guarantee two things that the primitive function S is real-valued are all 1 and the Fourier series coefficients of the primitive function is very close to the distribution of the parameter space. Finally the angle of the iteratively computed complex primitive function is extracted and used as a data set with the corresponding target parameter space.

The flow chart of this iterative algorithm is shown in Fig. S1:

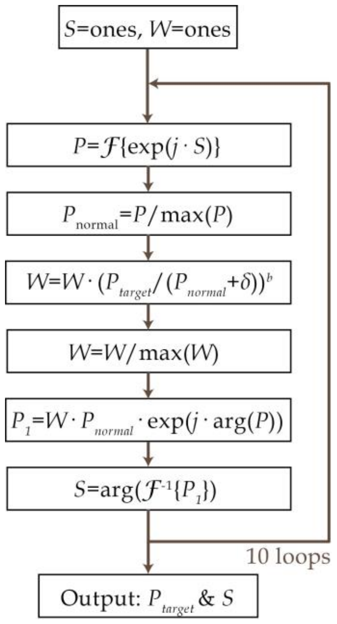

S=ones, W=ones
$P=\mathcal{F}\{\exp(j\cdot S)\}$
$P_{\text{normal}}=P/\max(P)$
$W=W\cdot(P_{target}/(P_{normal}+\delta))^{b}$
$W=W/\max(W)$
$P_1=W\cdot P_{normal}\cdot\exp(j\cdot\arg(P))$
$S=\arg(\mathcal{F}^{-1}\{P_1\})$
10 loops
Output: P_{target} & S

Figure S1. Iteration flow chart for generating the primitive function S in preparing dataset

Here, P_{target} is the target parameter space, S is the primitive function solved iteratively, and W is a momentum-like variable used to store the results of the iterations. δ is a very small value to avoid a zero denominator. b controls the speed of convergence of the iterations; in our practice, b is taken to be 1, and values between 0.7 and 1 could ensure the iteration fast and effective. The final outputs of this iterative algorithm, P_{target} and S, are saved as .mat files by the commercial software Matlab.

1.3 Structure and training of neural network:

The architecture of the neural network in RediNet is relatively common. It is shown in the fig. S2 below:

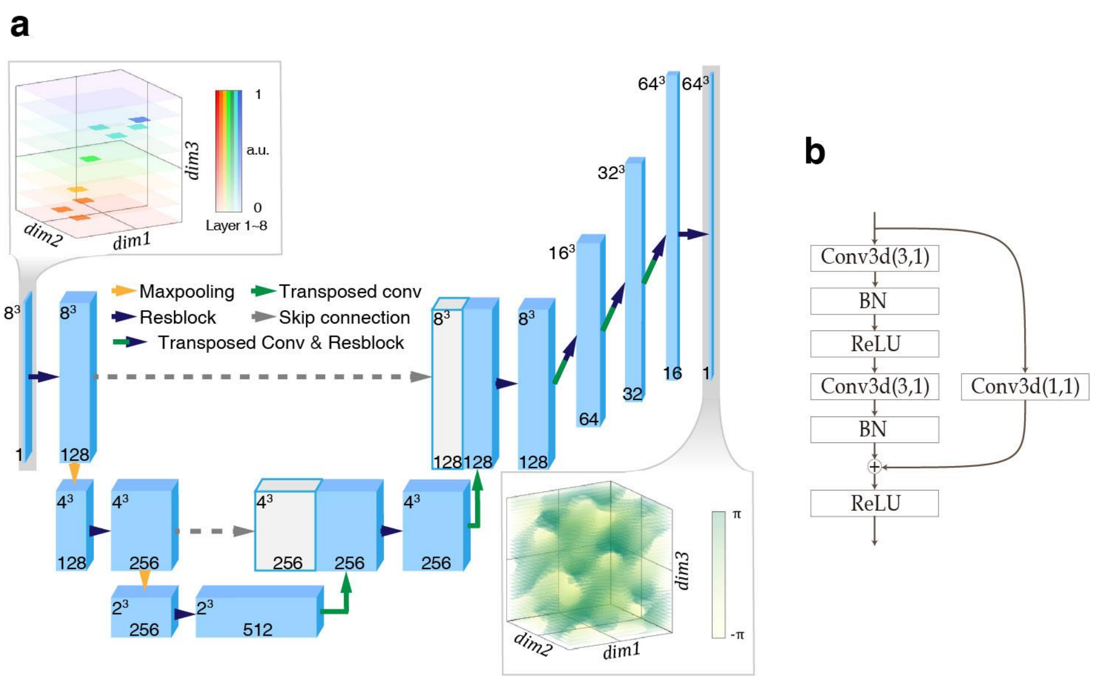


Figure. S2 Architecture of the neural network of RediNet. **a**, The architecture of asymmetric 3DUnet. The size of each network block and the process between them are labeled. An example of input and output data are illustrated. **b**, The res-block architecture. Conv3d, 3D convolution. BN, batch normalization. ReLU, rectified linear unit. Numbers in the brackets are the kernel sizes and strides.

In Fig. S2a, it can be seen that the 8^3 input are downsampled 2 times to 2^3, while the channel increases to 512. Subsequently, the opposite operation is performed, upsampling to 64^3 while compressing the number of channels, finally outputting a single channel data. The structure of this network is not completely symmetrical due to the different sizes of the input and output data. In the network, downsampling is processed using the Maxpool3d function with a kernel size of 2 and a stride of 2. Upsampling is processed using the ConvTranspose3d function with a kernel size of 2 and a stride of 2. Res-block can make the training of deep neural networks feasible, so it is applied to our network in a structure shown in Fig. S2b. Skip connections are added between the downsampling and the corresponding upsampling, allowing the original details and overall characteristics to be considered simultaneously.

In the training of the neural network, we divided the training into 4 sectors. The descending curve of the loss function during training is shown in Fig. S3:

(1) The loss function used is $\sum 0.5(x-\hat{x})^2/N$, classical L2 Loss. The training set is generated using the above method, comprising 50000 pairs. Training is performed for 79 epochs.

(2) The same loss function is used. The training set is regenerated using the above

method with 300000 pairs. Training is performed for 31 epochs.

(3) The loss function used is $\sum 0.5\left(1-\cos(x-\hat{x})\right)/N$. This loss function adequately reflects the property of the output of the neural network as an angle value, i.e., 0 and 2π are the same value, while 0.01π and 1.99π are very close. The dataset used is a 50000-pair set regenerated using the method described above. 20 rounds of training were performed.

(4) The same loss function is used. The training set is regenerated using the above method with 300000 pairs. Training is performed for 43 epochs.

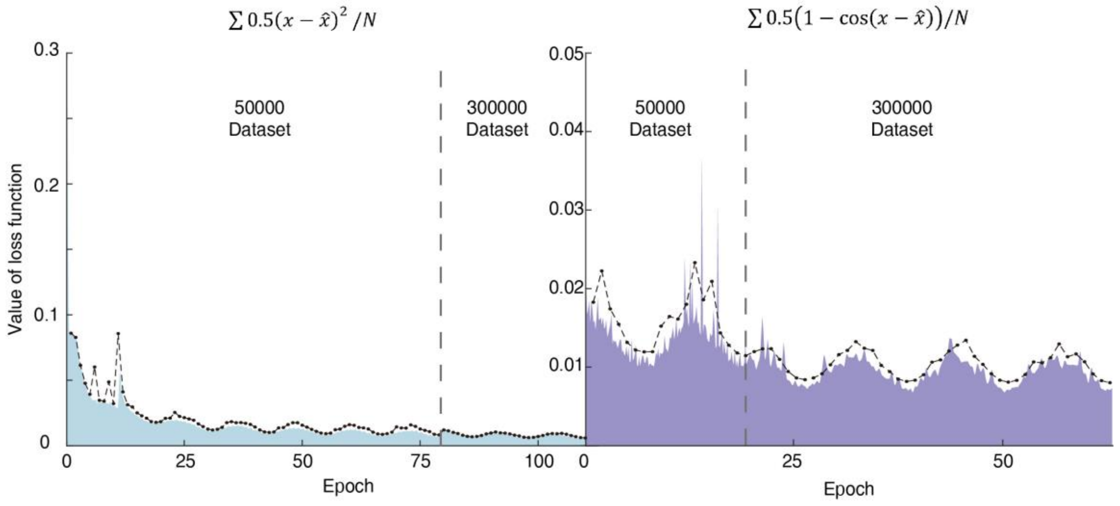


Figure. S3 Value of the loss function with the epochs. The second sector training starts with the trained network parameters of the first sector.

The performance improvement brought by the latter loss function is obvious, especially beneficial for the diffractive efficiency. Taking the planar equal-energy four foci as an example, it can be found that the noise point vanishes and the contrast ratio becomes higher. Besides, the phase skips around 0 and 2π becomes precise and clear.

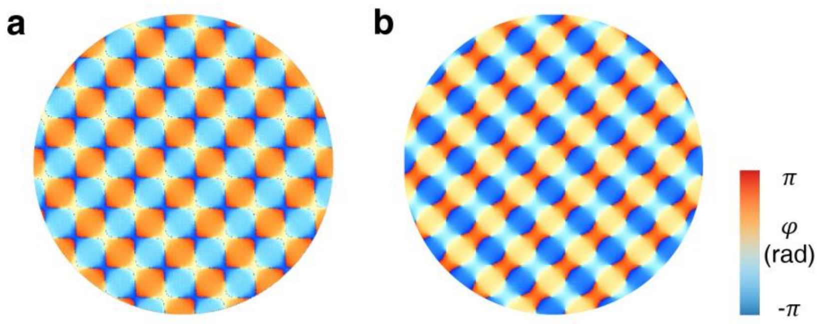


Figure. S4 Phase CGH using different loss functions. **a**, Result of $\sum 0.5(x-\hat{x})^2/N$. **b**, Result of $\sum 0.5\left(1-\cos(x-\hat{x})\right)/N$

The number of network parameters is 22.4 million. The parameter file of the neural network, codes for training, codes used to generate the dataset, and some of the dataset examples have been uploaded to https://github.com/LiHengyang1/RediNet.

1.4 Characteristic phase functions (CPFs) of structure light

Theoretically, any structured lights that can be generated by a single pure phase CGH corresponds to a CPF. A CPF can be inserted into RediNet if it can be written as a linear function of the modulation parameters. Some examples of CPFs have been given in the main text of the article, and here we list more formulations of commonly used CPFs.

Table S1 The CPFs of structured lights

Structured light, parameter	CPF expression
x-shifting, low NA, Δx	$kx/z(\Delta x)$
y-shifting, low NA, Δy	$ky/z(\Delta y)$
z-defocusing, low NA, Δz	$kr^2/(2z^2)(\Delta z)$
x-shifting, high NA, Δx	$k\cos\varphi\sin\theta(\Delta x)$
y-shifting, high NA, Δy	$k\sin\varphi\sin\theta(\Delta y)$
z-defocusing, high NA, Δz	$k\cos\theta(\Delta z)$
Bessel beam, ϕ	$kr(\phi)$
Airy beam, b	$(x^3+y^3)(b)$
OAM of vortex beam, l	$\theta(l)$
Radius of ring focus, Δr	$kr/f(\Delta r)$
Snowflake CPF1, u	$\left[6\left(\text{sgn}(\sin(8r)-0.1)\right)+1.5r\right](u)$
Snowflake CPF2, v	$[0.8r\sin(6\theta)](v)$

1.5 Evaluation on $RMSE$ and diffractive efficiency

$RMSE$ in the article has two implementation forms. One is experimental-oriented, like the evaluation of the intensity of the 2D focus array in Fig. 3a in the main text. Here $RMSE$ is

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \bar{y})^2}, \tag{S5}$$

where N is the total number of foci, y_i is the measured value and $\bar{y}$ is the average of all y_i. The other is used to directly evaluate the performance of the network like that in the Fig. 6d in the main text. Considering the output of the network is the primitive function, we expanded it into Fourier series in order to evaluate the variation between the Fourier series coefficients and the values in the given parameter space. The $RMSE$ here is

$$RMSE = \sqrt{\frac{1}{N}\sum_{l_i \in \Omega}\left(\left|\alpha_{l_1,l_2,l_3}\right|^2 - \left|A_{l_1,l_2,l_3}\right|^2\right)^2}, \tag{S6}$$

where α_{l_1,l_2,l_3} are the Fourier series coefficients, A_{l_1,l_2,l_2} is the corresponding target value in the given parameter space as ground truth, and Ω is the set of all coordinates that take nonzero values in the given parameter space.

Likely, the diffraction efficiency is also based on the coefficients expanded from the primitive function, defined as

$$\mathrm{e} = \sum_{l_i \in \Omega}\left|\alpha_{l_1,l_2,l_3}\right|^2 / \sum_{l_i}\left|\alpha_{l_1,l_2,l_3}\right|^2 \tag{S7}$$

In view of the analysis on Eq. 1 and Eq. 2 in the main text, it is precise to use $\left|\alpha_{l_1,l_2,l_3}\right|^2$ as the energy proportion of every sub-beam in an array.

2. Experiment and data processing

2.1. Experimental setup

We built the optical path shown in Fig. S5. We used a collimated fundamental mode Gaussian beam with a wavelength of 1064 nm. It is an Nd:YAG laser and coupled into a single-mode fiber, and the M2 factor of the output laser is smaller than 1.2. The beam was expanded to about 4.4 mm Gaussian radius and only one linear polarization component was retained. The main body of the optical path was a Mach–Zehnder interferometer, where one arm (pink) was modulated by the spatial light modulator (SLM) and attenuated using a set of half wave plate and polarized beam splitter. The lens ($f = 400\ mm$) had the SLM and the camera in the front and back focal plane. The other arm (green) was used as a reference beam, so the interference pattern can be captured. If the phase distribution at the Fourier plane is needed, it can be solved using the interference fringe pattern.

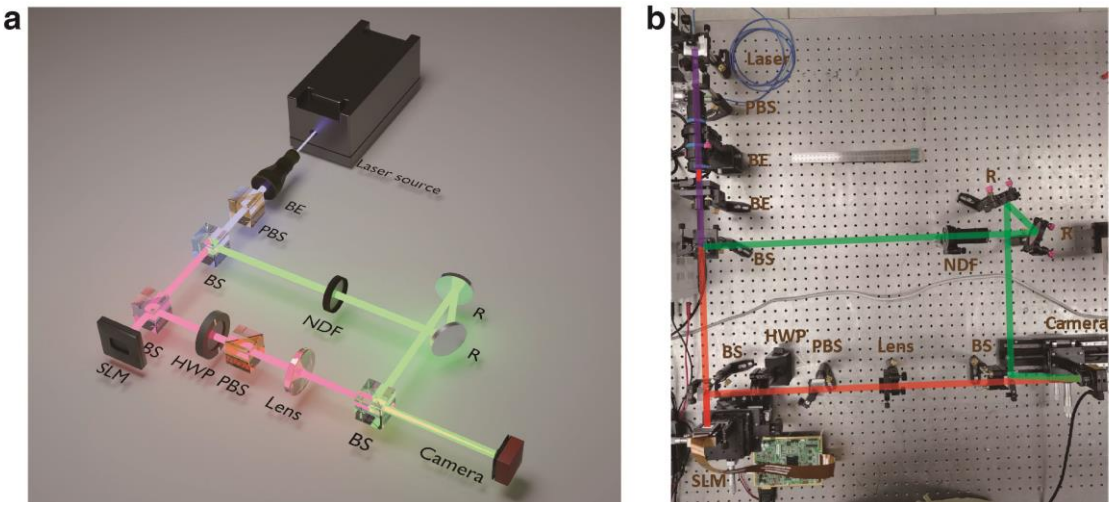


Figure. S5 Experimental setup. **a**, Schematic picture. **b**, Real setup picture. BE: beam expander, PBS: polarized beam splitter, BS: beam splitter, SLM: spatial light modulator, HWP: half wave plate, NDF: neutral density filter, R: reflective mirror.

2.2 Phase pattern obtained with single interferometric fringe picture

In our Mach-Zehnder interferometer, the reference beam is a collimated Gaussian beam that can be roughly considered as an inclined plane wave. The interference picture is captured by a camera, and using this single intensity distribution, an accurate complex amplitude distribution of the modulated light is obtained. The principle and procedure are illustrated as follows:

Suppose that the complex amplitude of the signal light is $U = A\exp(j\varphi)$, and in addition there is a reference light with complex amplitude $V = B\exp\left(j(k_x x + k_y y)\right)$, a plane wave with wavevector direction (k_x, k_y). The complex amplitude of the light field on the camera is then:

$$\begin{aligned} W &= U + V \\ &= A\exp(j\varphi) + B\exp\left(j(k_x x + k_y y)\right) \\ &= B\exp\left(j(k_x x + k_y y)\right)\cdot\left(1+\frac{A}{B}\exp\left(j\left(\varphi-(k_x x+k_y y)\right)\right)\right) \\ &= B\exp\left(j(k_x x + k_y y)\right)\cdot\left(1+\left(\frac{A}{B}\right)^2-2\frac{A}{B}\cos\left(\varphi-(k_x x+k_y y)\right)\right)\exp\left(j\phi\right) \end{aligned} \quad , \qquad \text{(S8)}$$

where ϕ is the phase carried by $\left(1 + A/B\exp\left(j\left(\varphi - \left(k_x x + k_y y\right)\right)\right)\right)$, not used in followed content. The amplitude, i.e., the square root of the intensity captured by the camera, is:

$$\begin{aligned} |W| &= B\cdot\left(1+\left(\frac{A}{B}\right)^2-2\frac{A}{B}\cos\left(\varphi-(k_x x+k_y y)\right)\right) \\ &= \left[B+\frac{A^2}{B}\right]-\left[2A\cos\left(\varphi-(k_x x+k_y y)\right)\right] \\ &= \left[B+\frac{A^2}{B}\right]-\left[A\exp\left(j\left(\varphi-(k_x x+k_y y)\right)\right)+A\exp\left(-j\left(\varphi-(k_x x+k_y y)\right)\right)\right] \end{aligned} \quad . \qquad \text{(S9)}$$

Perform a two-dimensional Fourier transform on it, and assume $\Im\{U\} = u$.

$$\begin{aligned} \mathcal{F}\{|W|\} &= \mathcal{F}\left\{B+\frac{A^2}{B}\right\}-\mathcal{F}\left\{A\exp\left(j\left(\varphi-(k_x x+k_y y)\right)\right)\right\}-\mathcal{F}\left\{A\exp\left(-j\left(\varphi-(k_x x+k_y y)\right)\right)\right\} \\ &= \mathcal{F}\left\{B+\frac{A^2}{B}\right\}-u\left(\xi-k_x,\eta-k_y\right)-u^*\left(-\xi-k_x,-\eta-k_y\right) \end{aligned}$$

(S10)

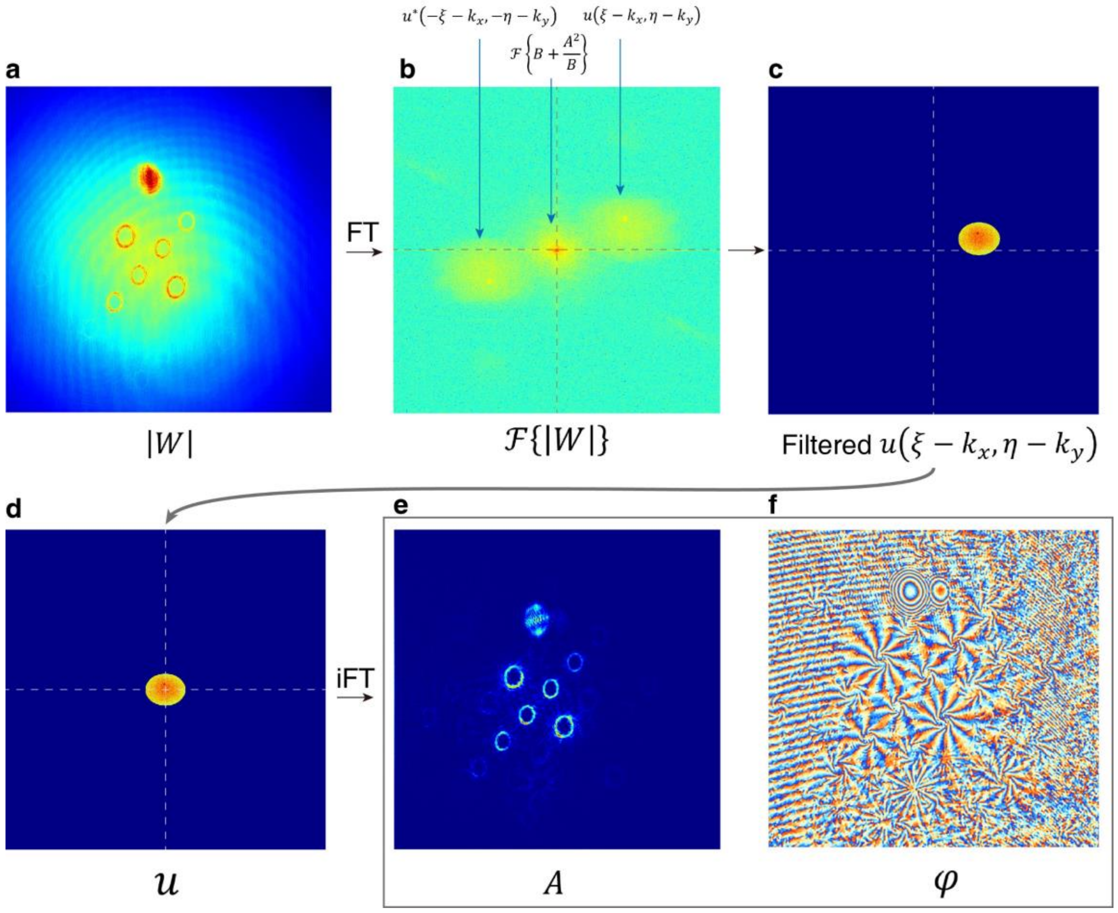
It is obvious that the first term is centered on the frequency plane, while the last two terms are symmetric with respect to the center point about the frequency plane. Accordingly, when k_x and k_y are relatively large, we can filter out $u(\xi - k_x, \eta -$

$k_y)$, shift it to the center of the frequency plane, and then do the inverse Fourier transform to get the $U = A\exp(j\varphi)$, i.e.,

$$U = \mathcal{F}^{-1}\left\{u\left(\xi - k_x + k_x, \eta - k_y + k_y\right)\right\}. \tag{S11}$$

In this way, the complex amplitude of U is recovered (in which the phase part is what we are mainly interested). The entire process uses only one interference fringe map $|W|$.

A flow chart is used to represent the process:



$u^*(-\xi - k_x, -\eta - k_y)$
$u(\xi - k_x, \eta - k_y)$
$\mathcal{F}\left\{B + \frac{A^2}{B}\right\}$
a
b
c
FT
$|W|$
$\mathcal{F}\{|W|\}$
Filtered $u(\xi - k_x, \eta - k_y)$
d
e
f
iFT
u
A
φ

Figure. S6 Flow chart of obtaining complex amplitude from a single fringe pattern. **a**, Fringe pattern captured with camera. **b**, Pattern of spatial frequency. **c**, Filtered spatial frequency that only $u(\xi - k_x, \eta - k_y)$ is left. **d**, Filtered spatial frequency that is shifted to the center. **e** and **f**, the amplitude and phase of the signal beam.