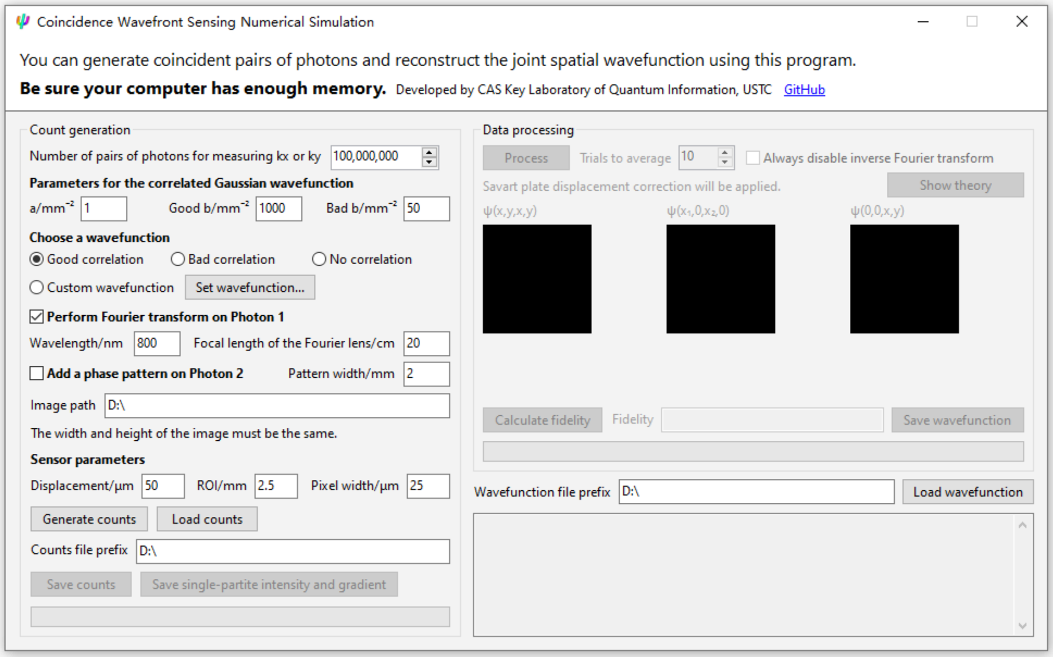
Manual of the Coincidence Wavefront Sensing Numerical Simulation Program

CAS Key Laboratory of Quantum Information, University of Science and Technology of China

This is a detailed manual of the coincidence wavefront sensing numerical simulation program. With this program, you can customize the wavefunction, generate coincidence count data, and reconstruct the wavefunction.

This program is written in C# with Microsoft Visual Studio. It can be directly run on Windows 8 or newer Windows editions. Microsoft .NET Framework 4.5 is required for Windows Vista and Windows 7. A 64-bit computer is recommended as it may take up a large amount of memory to store the generated and processed data.

The main window is shown in Fig. 1. Here are the descriptions of the configurations.



Coincidence Wavefront Sensing Numerical Simulation

You can generate coincident pairs of photons and reconstruct the joint spatial wavefunction using this program.

Be sure your computer has enough memory. Developed by CAS Key Laboratory of Quantum Information, USTC GitHub

Count generation

Number of pairs of photons for measuring kx or ky: 100,000,000

Parameters for the correlated Gaussian wavefunction

a/mm⁻²: 1 Good b/mm⁻²: 1000 Bad b/mm⁻²: 50

Choose a wavefunction

[✓ Good correlation] Bad correlation No correlation

Custom wavefunction [Set wavefunction...]

[✓] **Perform Fourier transform on Photon 1**

Wavelength/nm: 800 Focal length of the Fourier lens/cm: 20

[ ] **Add a phase pattern on Photon 2** Pattern width/mm: 2

Image path: D:\

The width and height of the image must be the same.

Sensor parameters

Displacement/μm: 50 ROI/mm: 2.5 Pixel width/μm: 25

Generate counts Load counts

Counts file prefix: D:\

Save counts Save single-partite intensity and gradient

Data processing

Process Trials to average: 10 [ ] Always disable inverse Fourier transform

Savart plate displacement correction will be applied. Show theory

ψ(x,y,x,y) ψ(x₁,0,x₂,0) ψ(0,0,x,y)

Calculate fidelity Fidelity Save wavefunction

Wavefunction file prefix: D:\ Load wavefunction

Fig. 1. The main window.

Number of pairs of photons for measuring k_x or k_y: In each measurement (k_x or k_y), a number of photon pairs are generated to count for the joint intensity distribution I_{LL}, I_{LR}, I_{RL} and I_{RR}. A larger number will let the intensity distribution be more smooth, and the errors will be reduced. The minimum and maximum are 10^5 and 10^{10} respectively. The joint intensities are stored as eight four-dimensioanl arrays of unsigned 16-bit integers (0–65535). Extremely large photon pair numbers are avoided as it may cause the intensity data to saturate, but this is very unlikely to happen.

Parameters for the correlated Gaussian wavefunction: The wavefunction without Fourier transform takes the form

$$\psi(\mathbf{r}_1,\mathbf{r}_2)=\exp\left[-a(|\mathbf{r}_1|^2+|\mathbf{r}_2|^2)-b|\mathbf{r}_1-\mathbf{r}_2|^2+i\phi_{\text{add}}(\mathbf{r}_2)\right],$$

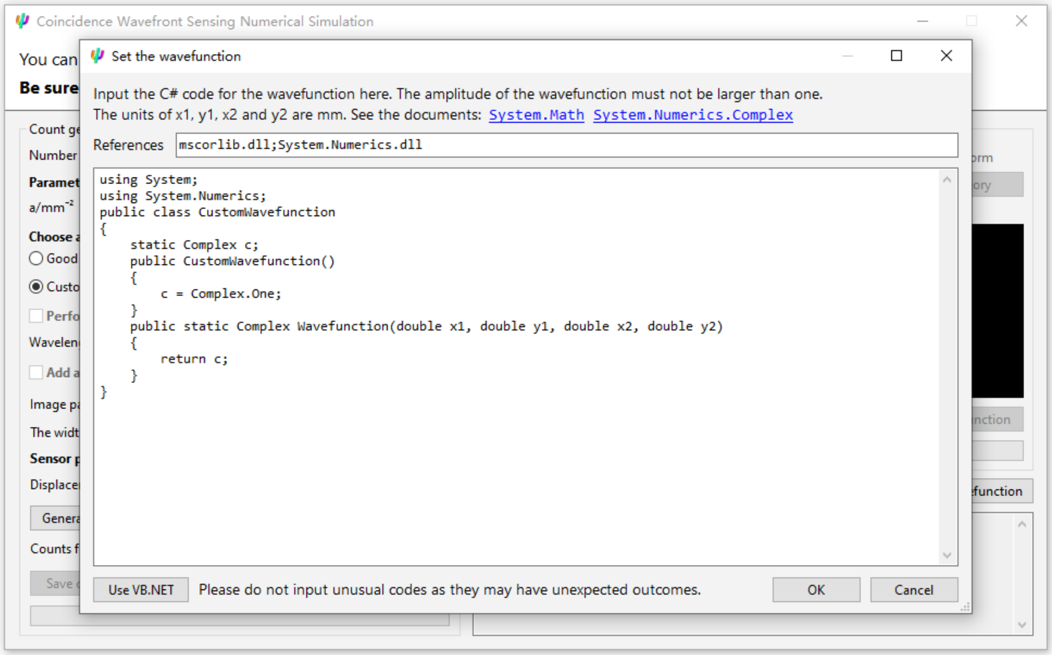
and that with the spatial mode of Photon 1 Fourier transformed is

$$\tilde{\psi}(\mathbf{K}_1,\mathbf{r}_2)=\exp\left[-\frac{|\mathbf{K}_1|^2+4ib\mathbf{K}_1\cdot\mathbf{r}_2+4a(a+2b)|\mathbf{r}_2|^2}{4(a+b)}+i\phi_{\text{add}}(\mathbf{r}_2)\right],$$

where $\mathbf{K}_1=(2\pi\mathbf{r}_1)/(\lambda f)$. Parameter a and b can be adjusted.

Choose a wavefunction: When you choose “Good correlation”, “Bad correlation” and “No correlation”, b takes its value from “Good b”, “Bad b” or zero. If you choose “Custom wavefunction”, you should click the “Set wavefunction...” button, and enter the dialog box shown in Fig. 2.

Then, you can use the C# or Visual Basic .NET language to input the code to be compiled. When you click the “OK” button, the program will compile the code. If it fails or it cannot find the `CustomWavefunction` class or the static `Wavefunction`


Coincidence Wavefront Sensing Numerical Simulation

Set the wavefunction

Input the C# code for the wavefunction here. The amplitude of the wavefunction must not be larger than one.
The units of x1, y1, x2 and y2 are mm. See the documents: System.Math System.Numerics.Complex

References mscorlib.dll;System.Numerics.dll

```
using System;
using System.Numerics;
public class CustomWavefunction
{
    static Complex c;
    public CustomWavefunction()
    {
        c = Complex.One;
    }
    public static Complex Wavefunction(double x1, double y1, double x2, double y2)
    {
        return c;
    }
}
```

Use VB.NET Please do not input unusual codes as they may have unexpected outcomes. OK Cancel

Fig. 2. The custom wavefunction dialog box.

function with the correct signature (four `double` inputs; `System.Numerics.Complex` output), it will show a messagebox about the reason. Otherwise, the dialog box closes, and the custom wavefunction can be used. Note that the amplitude of the wavefunction must not be larger than one, which is limited by the Monte Carlo method. Here is the code we used for the second case in the Article, where two phase patterns are added on (x_1, x_2) and (y_1, y_2) respectively. The references are `mscorlib.dll;System.Numerics.dll;System.Drawing.dll`.

```
using System;
using System.Numerics;
using System.Drawing;
public class CustomWavefunction
{
    static double[,] bmphasx, bmphasy;
    static double bmwidth;
    static int bmpixs;
    static double BMPhase(double x, double y, double[,] arr)
    {
        x = (x / bmwidth + 0.5) * bmpixs;
        y = (y / bmwidth + 0.5) * bmpixs;
        if (x < 0 || x >= bmpixs || y < 0 || y >= bmpixs) return 0.0;
        return arr[(int)x, (int)y];
    }
    internal static void LoadBitmap(string src, double width, ref double[,] arr)
    {
        using (Bitmap bm = new Bitmap(src))
        {
            bmpixs = bm.Width;
            bmwidth = width;
            arr = new double[bmpixs, bmpixs];
            for (int i = 0; i < bmpixs; i++) for (int j = 0; j < bmpixs; j++)
                    arr[i, j] = bm.GetPixel(i, bmpixs - 1 - j).GetHue() / 180.0 * Math.PI;
        };
    }
    public CustomWavefunction()
    {
        LoadBitmap(@"D:\hidex.png", 2.0, ref bmphasx);
        LoadBitmap(@"D:\hidey.png", 2.0, ref bmphasy);
    }
    public static Complex Wavefunction(double x1, double y1, double x2, double y2)
    {
        double p = BMPhase(x1, x2, bmphasx) + BMPhase(y1, y2, bmphasy);
        double a = Math.Exp(-2.0 * (x1 * x1 + y1 * y1 + x2 * x2 + y2 * y2));
        return new Complex(Math.Cos(p) * a, Math.Sin(p) * a);
    }
}
```

The method `public CustomWavefunction()` is the constructor of the class, which is not necessary if the wavefunction is a simple formula.

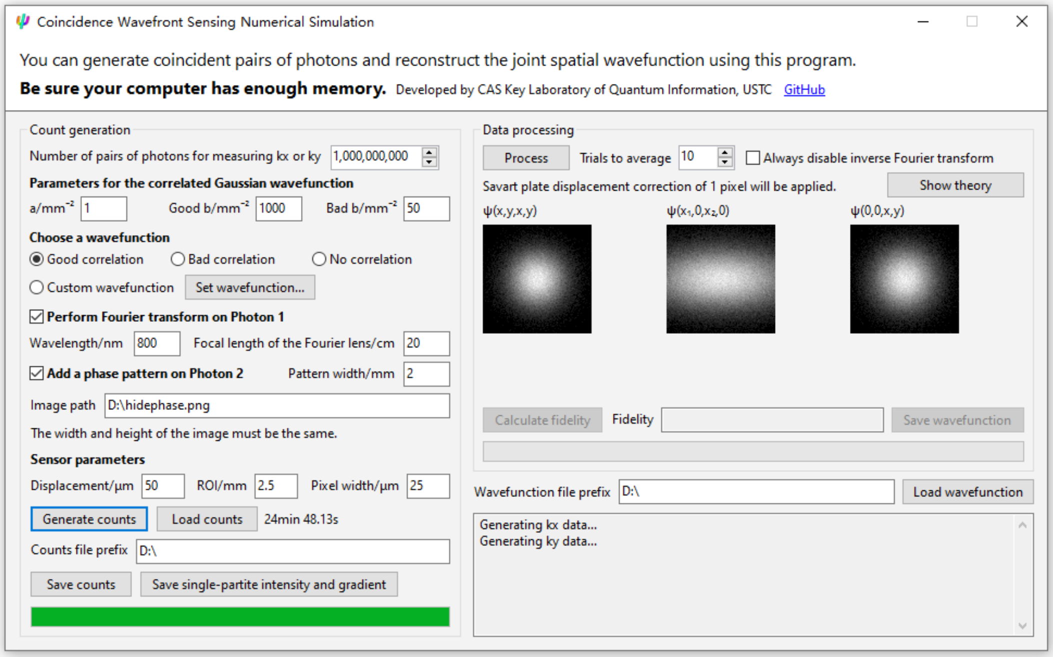
Perform Fourier transform on Photon 1: If it is checked, the program will use the Fourier transformed wavefunctions, and inverse Fourier transform will be performed in the data processing section. The light wavelength and the focal length of the Fourier lens can be adjusted. This check box is disabled when the user uses the custom wavefunction.

Add a phase pattern on Photon 2: If checked, the phase function $\phi_{\text{add}}(\mathbf{r}_2)$ will be loaded from the hue of the image (0 when red, $\pi/3$ when yellow, π when cyan), with the pattern width adjustable. The width and height of the pattern image must be the same. This check box is disabled when the user uses the custom wavefunction as well.

Sensor parameters: The overall displacement of the Savart plate ($2l$, the distance between the diagonally- and antidiagonally-polarized light field), the width of the region of interest (ROI) and the pixel width of the camera can be set.

With these parameters set, you can begin to generate count data. If the number of photon pairs is large or the amplitude of the wavefunction is small in many areas, the process will be slow. After generation, the amplitude distributions will be given as shown in Fig. 3. Clicking the second and third picture will switch the displayed slice ($(x_1, 0, x_2, 0) \leftrightarrow (0, y_1, 0, y_2)$ and $(0, 0, x_2, y_2) \leftrightarrow (x_1, y_1, 0, 0)$). The data can be saved as files after clicking the "Save counts" button to be loaded in the future.

Setting the reconstruction times to average, and you can start the data processing. If Photon 1 is performed Fourier transform and the check box "Always disable inverse Fourier transform" is unchecked, the time-consuming inverse Fourier transform


Coincidence Wavefront Sensing Numerical Simulation

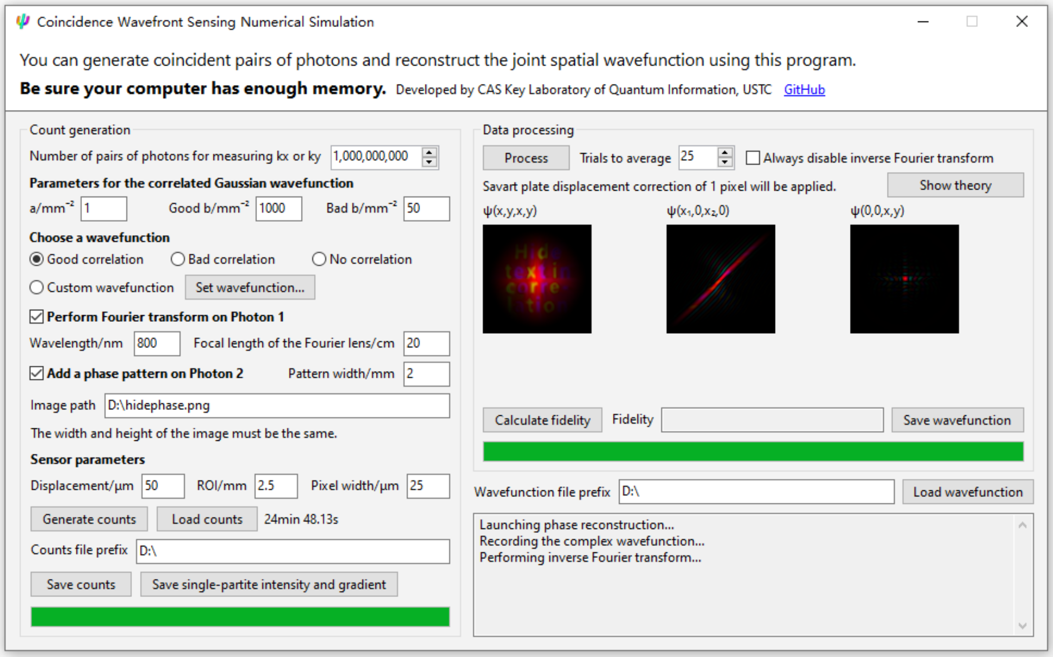
You can generate coincident pairs of photons and reconstruct the joint spatial wavefunction using this program.
Be sure your computer has enough memory. Developed by CAS Key Laboratory of Quantum Information, USTC GitHub

Count generation
Number of pairs of photons for measuring kx or ky: 1,000,000,000
Parameters for the correlated Gaussian wavefunction
a/mm^{-2}: 1 Good b/mm^{-2}: 1000 Bad b/mm^{-2}: 50
Choose a wavefunction
[✓ Good correlation] Bad correlation No correlation
Custom wavefunction Set wavefunction...
[✓ **Perform Fourier transform on Photon 1**]
Wavelength/nm: 800 Focal length of the Fourier lens/cm: 20
[✓ **Add a phase pattern on Photon 2**] Pattern width/mm: 2
Image path: D:\hidephase.png
The width and height of the image must be the same.
Sensor parameters
Displacement/μm: 50 ROI/mm: 2.5 Pixel width/μm: 25
Generate counts Load counts 24min 48.13s
Counts file prefix: D:\
Save counts Save single-partite intensity and gradient

Data processing
Process Trials to average: 10 Always disable inverse Fourier transform
Savart plate displacement correction of 1 pixel will be applied. Show theory
ψ(x,y,x,y) ψ(x₁,0,x₂,0) ψ(0,0,x,y)
Calculate fidelity Fidelity Save wavefunction
Wavefunction file prefix: D:\ Load wavefunction
Generating kx data...
Generating ky data...

Fig. 3. The window after photon count generation.

will be performed after the phase reconstruction completes. The fidelity compared with the original wavefunction (not Fourier transformed unless both the “Perform Fourier transform on Photon 1” and “Always disable inverse Fourier transform” boxes are checked) $|\langle\psi|\psi_{\mathrm{rec}}\rangle|^2/(\langle\psi|\psi\rangle\langle\psi_{\mathrm{rec}}|\psi_{\mathrm{rec}}\rangle)$ can be calculated. **The calculated fidelity is affected by many factors, such as the finite ROI size when performing inverse Fourier transform, the pixelization, and the phase reconstruction algorithm. The value is for reference only and may not be high enough.** The reconstruction result is shown in Fig. 4. The wavefunction is recorded as a four-dimensional array of single-precision floating-point number pairs, and can be saved into or loaded from files.



Coincidence Wavefront Sensing Numerical Simulation

You can generate coincident pairs of photons and reconstruct the joint spatial wavefunction using this program.

Be sure your computer has enough memory. Developed by CAS Key Laboratory of Quantum Information, USTC GitHub

Count generation

Number of pairs of photons for measuring kx or ky: 1,000,000,000

Parameters for the correlated Gaussian wavefunction

a/mm⁻²: 1   Good b/mm⁻²: 1000   Bad b/mm⁻²: 50

Choose a wavefunction

[✓ Good correlation]   Bad correlation   No correlation

Custom wavefunction   Set wavefunction...

[✓] **Perform Fourier transform on Photon 1**

Wavelength/nm: 800   Focal length of the Fourier lens/cm: 20

[✓] **Add a phase pattern on Photon 2**   Pattern width/mm: 2

Image path: D:\hidephase.png

The width and height of the image must be the same.

Sensor parameters

Displacement/μm: 50   ROI/mm: 2.5   Pixel width/μm: 25

Generate counts   Load counts   24min 48.13s

Counts file prefix: D:\

Save counts   Save single-partite intensity and gradient

Data processing

Process   Trials to average: 25   [ ] Always disable inverse Fourier transform

Savart plate displacement correction of 1 pixel will be applied.   Show theory

ψ(x,y,x,y)   ψ(x₁,0,x₂,0)   ψ(0,0,x,y)

Calculate fidelity   Fidelity   Save wavefunction

Wavefunction file prefix: D:\   Load wavefunction

```
Launching phase reconstruction...
Recording the complex wavefunction...
Performing inverse Fourier transform...
```

Fig. 4. The window after data processing and fidelity calculation.

The program will check the estimated memory usage and will ask the user whether to continue if it is larger than 4 GB. When loading the count data or the wavefunction, the program will check whether the file size matches the number of pixels, so the user should set the parameters accordingly before loading. If there is no enough memory for inverse Fourier transform, this process will be skipped.