Training set generator

Marcos Pérez Aviñoa

Beam-Trainer

A python program to generate pairs of intensity-phase values for neural network training or other purposes.

Layout

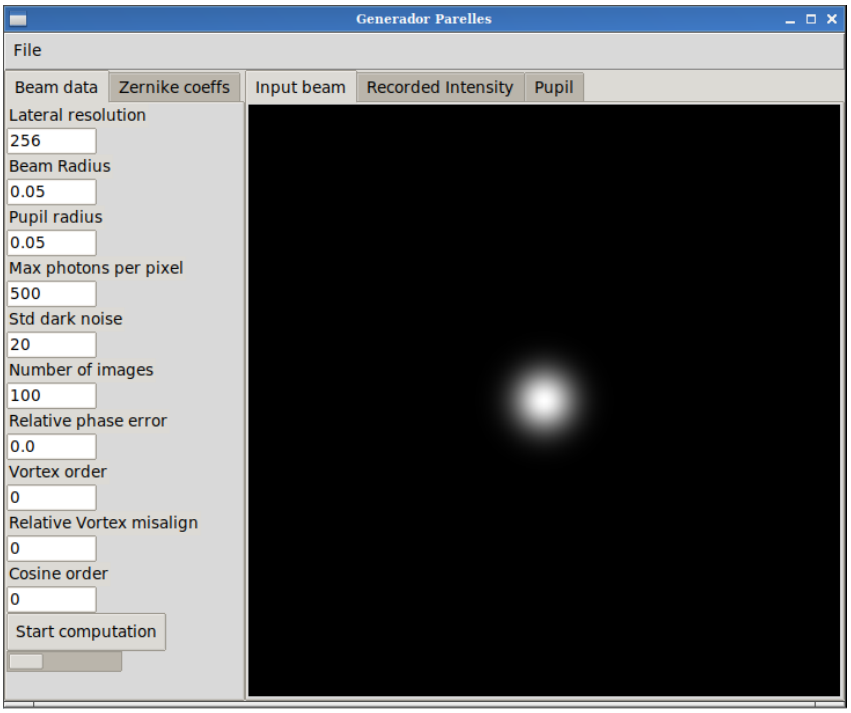

Figure 1: Default screen on opening `generador_parelles.py`.

The basic layout on opening the program can be seen in Fig. 1. On the left, there is a control panel containing all the information about the desired beam and on the left, som previsualizations on the final form of the images obtained. **These are not to be confused with the actual final rendering of the program**. These previsualizations are the shape of the input beam, a sample of the final intensity recorded and the real part of the pupil of the system.

All of the input properties may be saved and loaded again using the `File` dropdown in the menu bar.

Beam data layout

In order, the controls are as follows:

1. **Lateral resolution**: Lateral number of pixels of the final images. Without loss of generality, the final images are squared.
2. **Beam radius**: Maximum value of the radius of the beam as a fraction of the input window length. *Varies randomly when computing the set*.
3. **Pupil radius**: Radius of the pupil of the lens as a fraction of the pupil window. *Remains fixed when computing the set*.
4. **Max photons per pixel**: Number of photons corresponding to the maximum value of the output intensity. This number plays a role in photon detection, where it serves as the λ parameter for the Poisson distribution. *Remains fixed when computing the set*.
5. **Std dark noise**: Standard deviation of the intenity recorded by the camera pixels under dark conditions. *Remains fixed when computing the set*.
6. **Number of images**: Total number of intensity-phase pairs.
7. **Relative phase error**: Standard deviation of the noise added to the phase of the input beam. *Remains fixed when computing the set*.
8. **Vortex order**: Order m of the optional vortex phase on the input beam, $E_{in} = A\exp(im\phi)$. *Varies randomly when computing the set*.
9. **Relative misalign**: Relative misalignement of both vortex and cosine amplitude modulations (v.i.).
10. **Cosine order**: Order k of the cosine amplitude modulation on the input beam, $E_{in} = A\cos(l\phi)$. *Varies randomly when computing the set*.

All of the indicated variable values take the input value as the maximum of a random process, (e.g. `r_{iter} = r*random()`).

Zernike coefficients layout

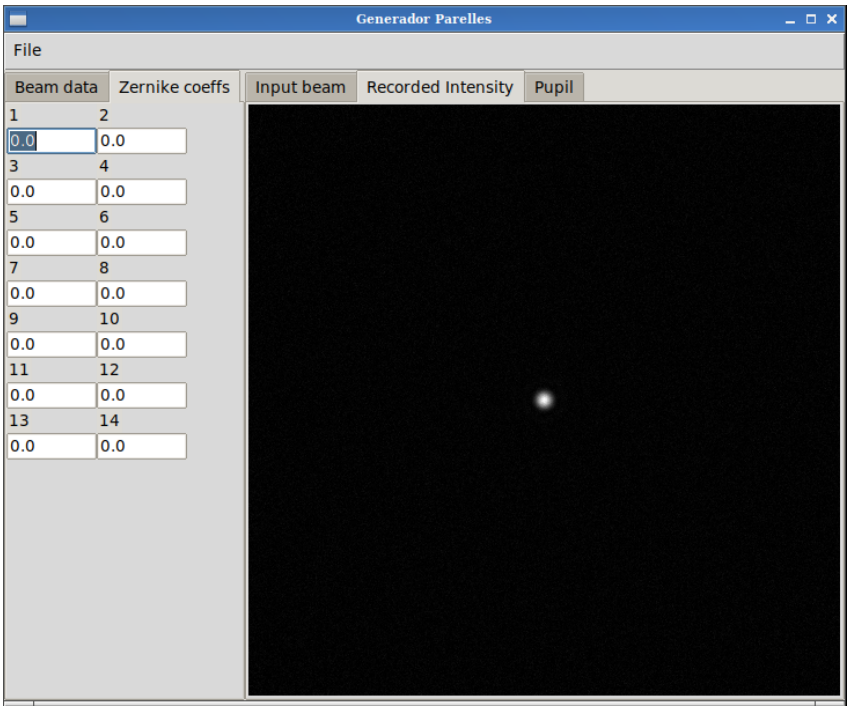

Figure 2: View of the Zernike coefficients tab.

In this tab we can select the number of non zero Zernike polynomial coefficients from number 1 to number 14. The numbering criterion corresponds to the OSA criterion. The zeroth order is not present as it represents a constant phase shift, irrelevant to our purposes.

To note some important numbers:

- Coefficients **1 and 2** correspond to vertical and horizontal tilts respectively.
- Coefficient **4** corresponds to a focusing error.

- Coefficients **3 and 5** correspond to oblique and vertical astigmatism.
- Coefficient **12** corresponds to spherical aberration.

Visualization layout

Three tabs are present in the right part of the GUI:

1. Input beam, the input intensity of the beam
2. Intensity, the simulated recorded intensity of the beam
3. Pupil, the real part of the pupil function of the system.

Changing any kind of parameter in the beam data layout or Zernike coefficient changes the properties of the beam. To update the views, just press `Return` on your keyboard.

Theoretical background

The program caclulates the paraxial intensity of one linear component of a focalized beam. The input beam has a Gaussian intensity profile, modulated maybe by a cosine function and with an optional vortex phase. Its most general form is, therefore,

$$U(r,\phi) = A\exp\left(-\frac{r^2}{2\sigma^2} + im\phi\right)\cos l\phi$$

where A is a constant, σ is the width of the Gaussian, m is the order of the optical vortex and l, the order of the cosine amplitude mosulation factor.

The optical system is modeled in its entirety as a single pupil function. Its most general form is given by

$$P(r,\phi) = \mathrm{circ}\left(\frac{r}{R}\right)\exp\left[ikW(r,\phi)\right]$$

where $\mathrm{circ}(\cdot)$ is the circle function, R is the radius of the entrance pupil of the optical system and $W(r,\phi)$ is the wavefront error or aberration of the optical system. It is represented as a truncated series of Zernike polynomials, having the general form

$$W(r,\phi) = \sum_{n=1}^{N_max}\sum_{m=1}^{n} A_n^m R_n^m(r)\cos(m\phi)$$

where $R_n^m(r)$ is the radial polynomial of order n, m. The coefficients n, m are synthetised into a single one, j, following OSA criterion. Finally, the focal field is given by

$$U_f(r,\phi) = C\iint U(r',\phi')P(r',\phi')'\exp\left[-\frac{ik}{f}rr'\cos(\phi'-\phi)\right]r'dr'd\phi$$

which is just the Fourier Transform of the input field multiplied by the pupil function.

To determine the recorded intensity by a photodetector, we make use of the probability of photodetection for laser light,

$$P(n,T) = \frac{(\eta TI)^n}{n!}\exp(\eta TI)$$

where n is the number of photodetections, T is the time interval considered, eta is the quantum efficiency and I is the intensity falling onto the detector. We generate a sample of random numbers with this probability function using `numpy.random.poisson`, with a parameter $\lambda = \eta TI$. This represents a single realization of the photodetections by the camera sensor after the time interval T. For simplicity's sake, the `Max number of photons` parameter in the program corresponds to the λ parameter for the brightest pixel of the camera.

Output data

The resulting pairs are saved as `*.npz` files inside a user selected folder. The file dialog asks for a folder name, which **need not exist**. At the same time, a JSON file containing the values used to generate the set is saved inside the directory, with the key `conf_yyyymmddhhmm.json`. The data is saved with the keys `intensity` and `phase`. To access the generated data, the function `load` from `numpy` must be used, which returns a dictionary with the keys defined earlier. To retrieve the data, an example script is provided below

```
data = np.load("0.npz")
intensity = data["intensity"]
phase = data["phase"]
```