MCPSim-Py: an open source python-based simulator of the performance of MCP photon-counting detectors

Prada, Iván, Yañez, Javier, Gómez de Castro, Ana I.

Iván Prada, Javier Yañez, Ana I. Gómez de Castro, "MCPSim-Py: an open source python-based simulator of the performance of MCP photon-counting detectors," Proc. SPIE 11444, Space Telescopes and Instrumentation 2020: Ultraviolet to Gamma Ray, 114449B (13 December 2020); doi: 10.1117/12.2576256

SPIE.

Event: SPIE Astronomical Telescopes + Instrumentation, 2020, Online Only

MCPSim-Py: an open source python-based simulator of the performance of MCP photon-counting detectors

Prada, Iván[a, b], **Yáñez, Javier**[a,b], **Gómez de Castro, Ana I.**[a, b *]
[a]Joint Center for Ultraviolet Astronomy, Universidad Complutense de Madrid, Edicio Fisac (Fac. EE Estadísticos), Av. Puerta de Hierro s/n, Madrid, Spain, E-28040
[b]AEGORA Research-Group, Facultad de CC. Matemáticas, Universidad Complutense de Madrid, Plaza de Ciencias 3, Madrid, Spain, E-28040

Abstract.
This contribution describes the open-source, python coded simulator **MCP-PySim** developed by our team to evaluate the performance of a given MCP architecture for a specific scientific purpose. The application is also designed to enable testing of user designed algorithms for event detection and centroiding for all types of sources and possible photon counting regimes. Unlike other tools, the MCP transfer function can be determined empirically from a set of images obtained from a real detector, ideally very similar to the one to be simulated. Also, it is feasible to characterize the MCP transfer function with a suit of parameters. Realistic simulations of the expected performance of space detectors can be carried in this manner to optimize the on-board software.

Keywords: MCP detectors, Instrument simulators, Python, ultraviolet.

*Gómez de Castro, Ana I. aig@ucm.es

1 Introduction

The simulator described in this work has been developed to assist in the design and the definition of the data processing algorithms for the MCP detector to be implemented in the far UV channel of the Field Camera Unit (FCU), the imaging instrument on board the space telescope Spektr-UF (WSO-UV).[1]

The architecture of the MCP detector is rather simple:[2] the detector is vacuum sealed and a photosensitive CsI substrate is deposited on the face of the MCP exposed to the radiation from astronomical sources. Photoelectrons are amplified within the MCP and them converted into optical photons at a phosphor (P46) interface. A bundle of optical fibres channel the optical photons into the CMOS detector used for the read-out. This architecture is similar to the implemented in the MCP detector of the Ultraviolet Imaging Telescope (UVIT), on board the ASTROSAT mission that developed a computer-based simulator of the detector performance for similar reasons.[3] However, UVIT simulator is based on a rather simple mathematical modelling of the detector that does not take into account the statistical nature of the photon counting process. Neither, it is feasible to characterize the detector performance with actual measurements obtained during the characterization tests. For these reasons, we developed the simulator, MCPSim-Py, described in this work.

The article is structured as follows: first, we describe the operation of the simulator with an example (Section 2). Section 3 includes a brief description of the GUI (graphical user interface). In section 5 instructions for the successfully installation of the simulator are provided. The details of the internal implementation are described in section 4. Finally, some closing remarks are provided in 6.

Space Telescopes and Instrumentation 2020: Ultraviolet to Gamma Ray, edited by Jan-Willem A. den Herder
Shouleh Nikzad, Kazuhiro Nakazawa, Proc. of SPIE Vol. 11444, 114449B · © 2020 SPIE
CCC code: 0277-786X/20/$21 · doi: 10.1117/12.2576256

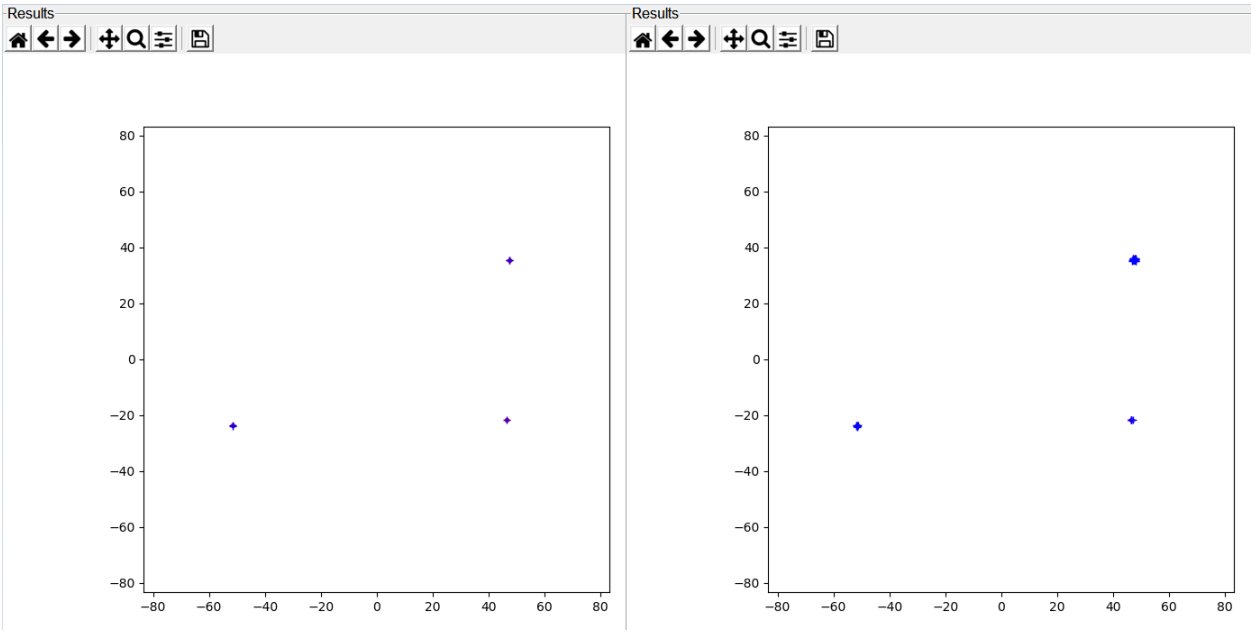

Fig 1 Example of MCPSim-PY usage to select the best centroiding algorithm (fast and reliable identification of events). Left image shows the result of 5-square algorithm. Right image shows the same example with the 3-cross algorithm. Blue crosses indicate events identified by the algorithm and red crosses the events fed into the MCP.

2 The simulator as a detector design tool

The simulator is designed to probe a range of possible architectures: diameter of the MCP pores, width and strength of the photoelectron in the amplification phase showers, response of the phosphor, size of the optical fibers and compression factor of the fiber tapping, CMOS detector pixel size and response. All these are set as free parameters, enabling instrument designers and scientists to search for the optimal set-up for a given purpose. To illustrate its potential, we have run some simulations.

2.1 Testing the optimal centroiding algorithm

Two centroid algorithms (5-square and 3-cross[4,5]) have been evaluated to show the number of events that each algorithm detect. After setting the basic parameters of the simulation is feasible to select the centroiding algorithm and compute and plot the centroids. The results are displayed in Figure 1. Left image shows the results using the 5-square algorithm, and right image shows the 3-cross algorithm. Clearly, the performance of the 3-cross algorithm is worse than the 5-square algorithm; the use of the 3-cross results in a significantly high and unrealistic number of event detections.

2.2 Testing the impact of pore pitch and optical fiber section

Let us compare the behaviour of two different detectors designed with pore pitch 12 μm and 14 μm, respectively all arranged in a hexagonal honeycomb structure. In both cases, the computation is made for 100 simultaneous UV photons impinging on the detector, distributed randomly on the surface and counted using the 5-square centroiding algorithm. The results from both simulations

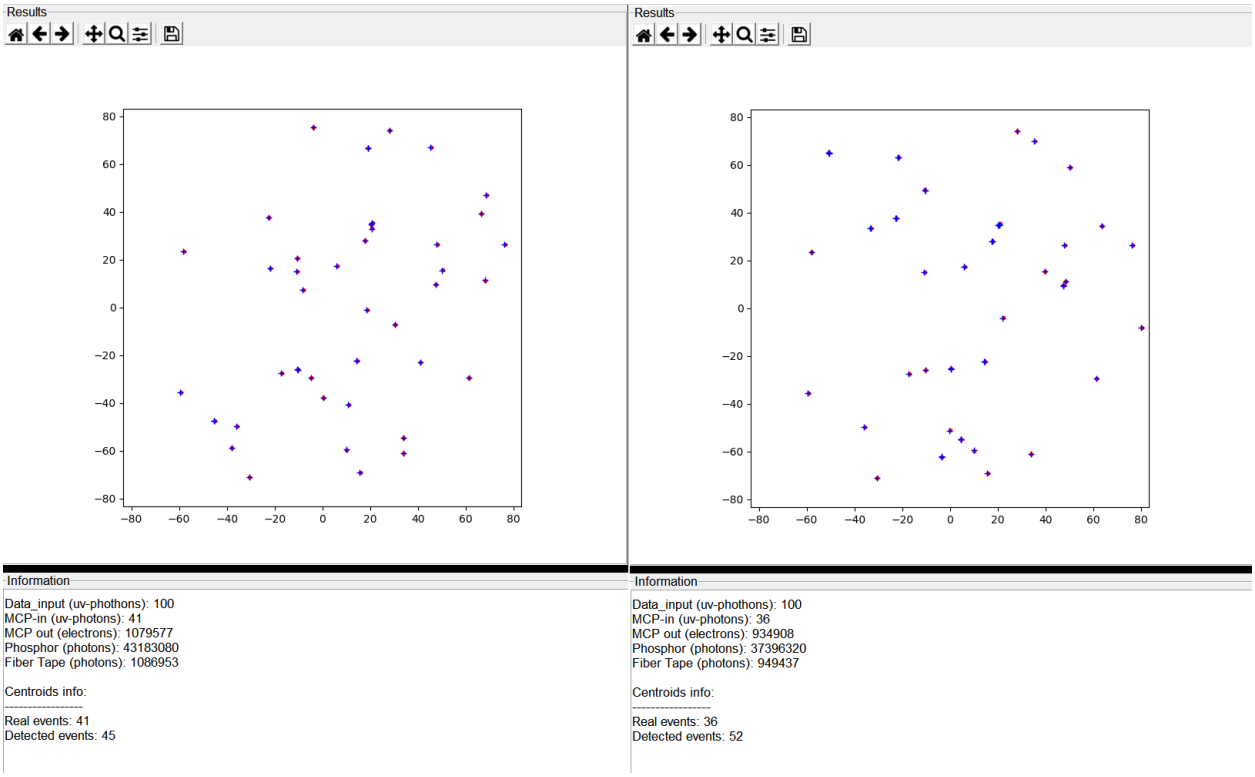

Fig 2 Left and right images show the results obtained for 12 μm and 14 μm pore pitchs.

are shown in Figure 2. We can observe that the second configuration is less favourable because of the number of undetected photons resulting from the efficiency of the geometry (a photon is detected if it fall in the interstice between pores).

3 Description of the MCPSim-Py

The GUI aspect is shown in Figure 3. The simulation parameters are entered in the upper left panels and the plots and statistics are in the right panels.

The input parameters (and the tabs that contains these parameters) are:

- **Data input**: a set of UV photons are generated for the MCP detector.
  - Location of the centre of the simulation window (sets the centre in arcsec).
  - Size of the sky square edge (size in arcsec).
  - Type of input. There are three modes (parameter Entry distribution):
    - **Image**: this mode uses the information of a given image to generate events with the probability distribution of the brightness in each pixel in the image. The Image path parameter is the path of the image file.
    - **Uniform**: this option uses a randomly uniform distribution to generate a given number of photons.

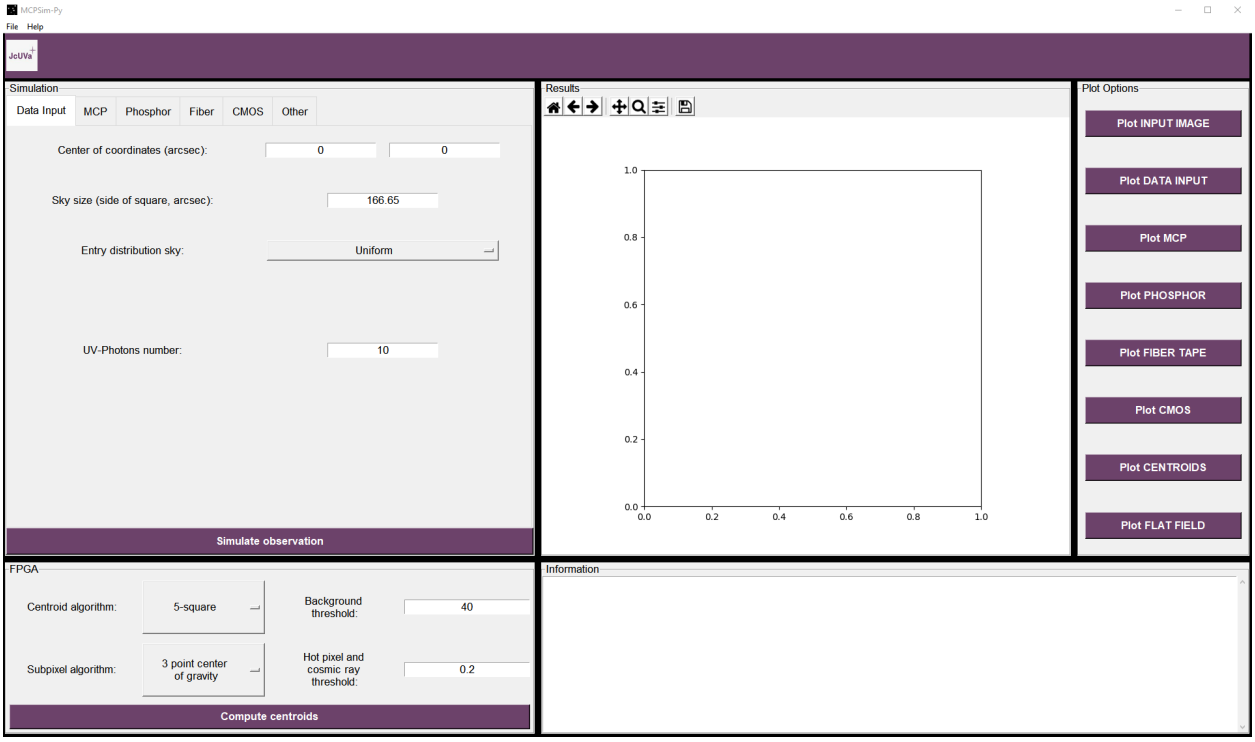

Fig 3 Layout of the GUI. The set-up is defined through a set of parameters in the left side of the screen. These parameters are organized in tabs that represent the different elements of the MCP. The showers of optical photons detected by the CMOS are displayed in the window at the right.

∗ **2-sources**: generate two point-like sources of UV photons to study the impact of the algorithms in resolving two nearby sources. These sources are modelled as a bidimensional uncorrelated normal distribution with standard deviation given by:

$$\sigma = \frac{FWHM}{\sqrt{8 \log 2}} \tag{1}$$

where FWHM is a simulation parameter. The distance between the sources can be set with Distance between sources, an additional parameter.

The input parameters for these two modes are displayed in Figure 4.

- **MCP**: simulates the effect of the microchannel plate in the transference/amplification of the signal (see Figure 5). MCP Gain defines the amplification factor of the photoelectrons and Pore diameter sets the effective diameter (in microns). In mcp-specific transfer function mode the user introduces an image of any MCP transfer function for simulating this effect (full details in 4). In the other case (simple transfer function), the electrons are randomly generated following a bidimensional uncorrelated normal distribution.

- **Phosphor**: anyone of these electron pulses is transformed into a photon pulse at the phosphor interface. The gain of this step is an input parameter (Phosphor Gain).

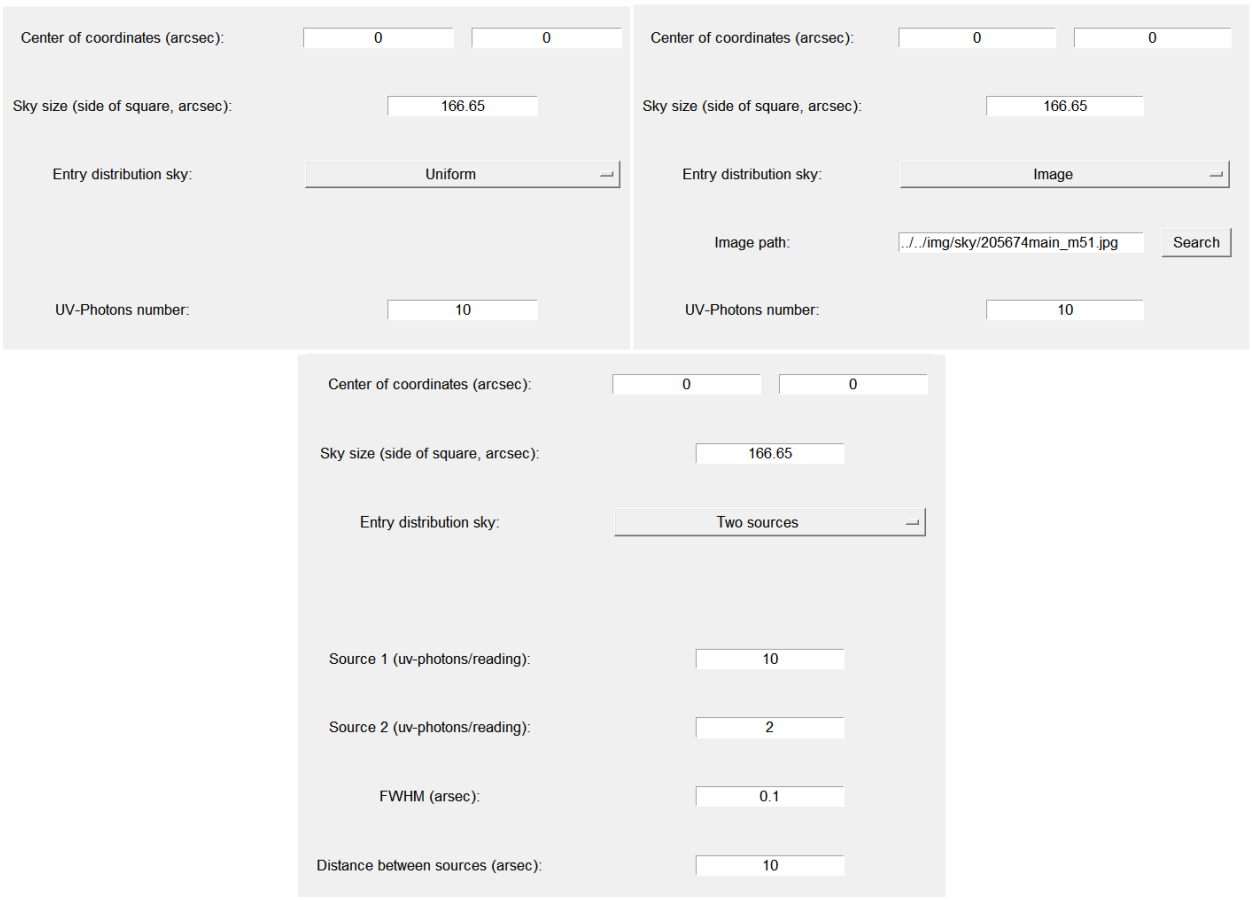

Fig 4 Data input parameters for the different values of the field *Entry distribution sky*.

- **Fiber**: the output photons shower is captured by the optical fibres of the fibre taper. The diameter of the inlet fibres at the phosphor converter is an input parameter of the Python simulator (Pore diameter on fiber_inlet parameter). This parameter sets the fraction and location of optical photons generated by the P46 that reach the detector. The fibre tape compresses the photons flux, with a default ratio of 3.55 into the CMOS (Factor parameter); this ratio is based on the FCU/FUV MCP detector architecture but can be modified for any other design. The photon flux transmission within the fibre is assumed to be uniform across the fibre section, and with fibre transmittance factor that represents the amount of decrement that the fibres induced in the final flux. Moreover, flux loss factor (geometry) take into account the photon losses by the fibres geometric covering factor of the P46 surface (fixed to $\pi/4$ for the current implementation). The flux decrease in the fiber tape factor is the multiplication of these last two values. The input window is shown in Figure 6.

- **CMOS**: the CMOS is a square grid, where the number of pixels depends on the window size and the pixel size. The size of the pixel is an input parameter (Pixel size parameter). There are parameters for establishing an area for the CMOS, the size of the pixel, hot pixels, flat field, readout noise and gain properties (see Figure 7 for the input window).

  To display the results from the simulation (intermediate results and final), four buttons are

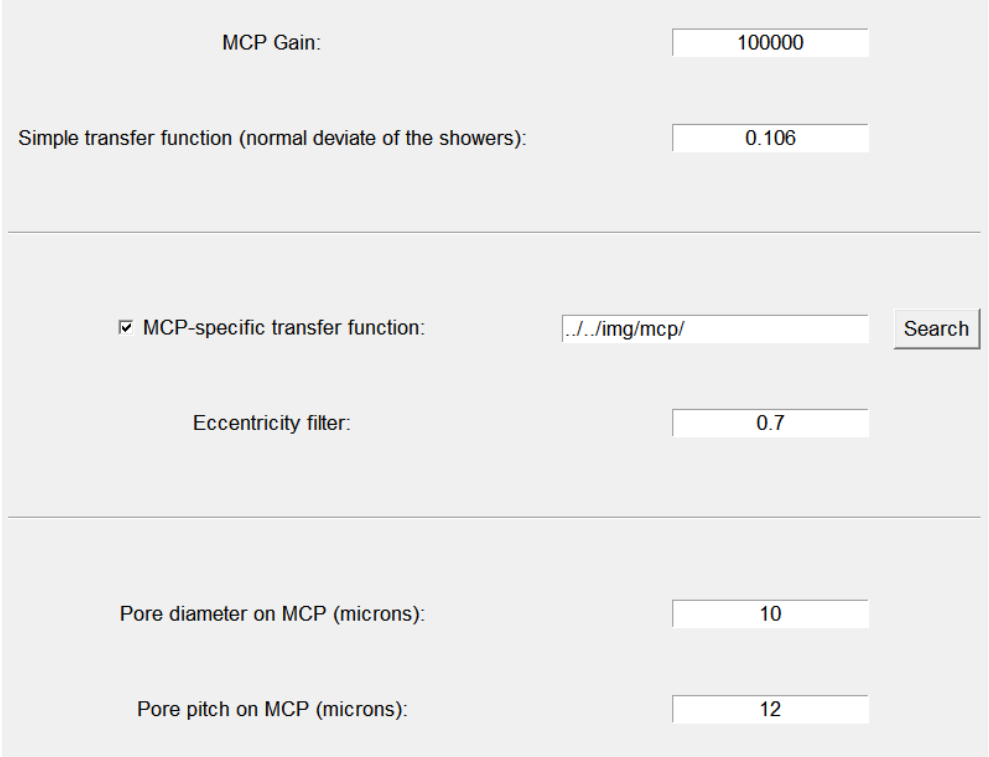

Fig 5 Input window for MCP parameters.

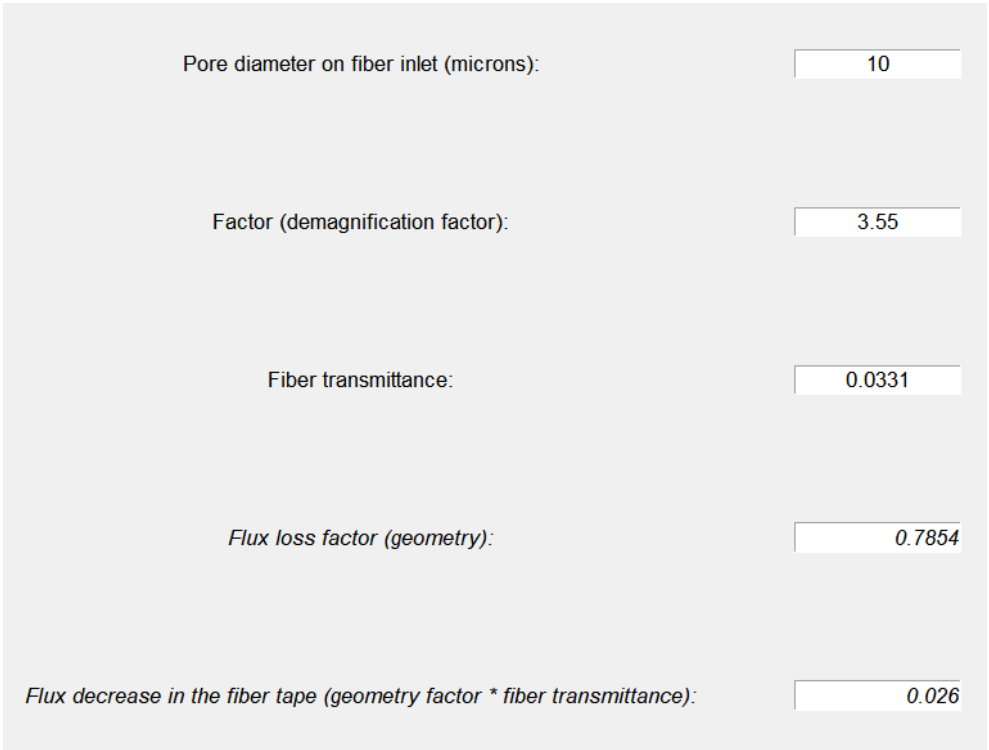

Fig 6 Input window for fiber parameters.

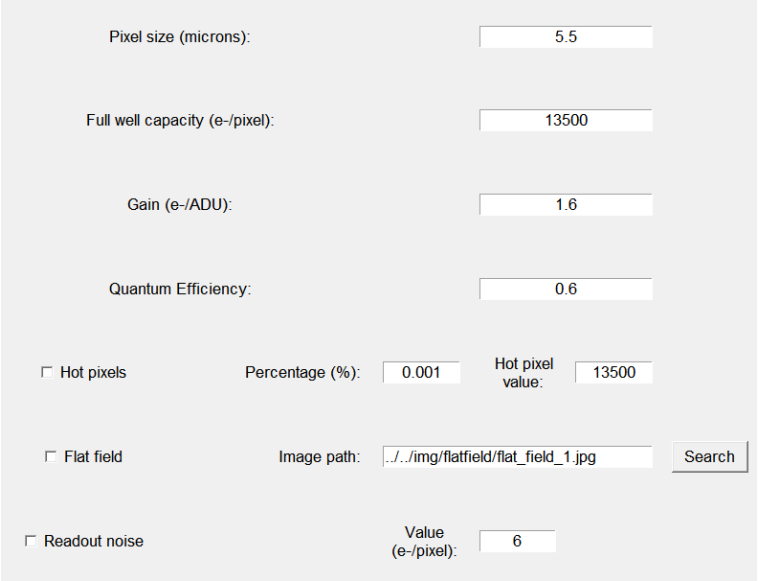

Fig 7 CMOS parameters.

included in the GUI:

- Plot INPUT IMAGE: shows the input image in the case that it is enabled.
- Plot DATA INPUT: shows the input window.
- Plot MCP: shows the clouds of photoelectrons generated in the MCP.
- Plot PHOSPHOR: shows the distribution of secondary electrons at the $P46$ layer.
- Plot FIBER TAPER: shows the distribution of electrons as captured by the Fiber taper.
- Plot CMOS: shows the final distribution of the photons on the square grid of the CMOS.
- Plot CENTROIDS: the calculated centroids are displayed for analysis purposes.
- Plot FLAT FIELD: shows the flat field image in the case that it is enabled.

4 Internal implementation

4.1 Data Input

The simulator has three modes:

- Uniform generates events uniformly distributed on the detector surface.
- 2-sources mode generates two sources with a Gaussian flux separated by a fixed distance. This mode is enabled to study the impact of the centroding algorithm in the resolution of two very close sources.
- Image mode reproduces the flux distribution provided by an image with *N* photons reaching the detector in the time scale relevant for the simulation. The image is flux normalized to convert the photon counts into probabilities. If a false colour image is provided, the image is converted into an array with elements the accumulated counts per pixel for the 3 RGB subframes.

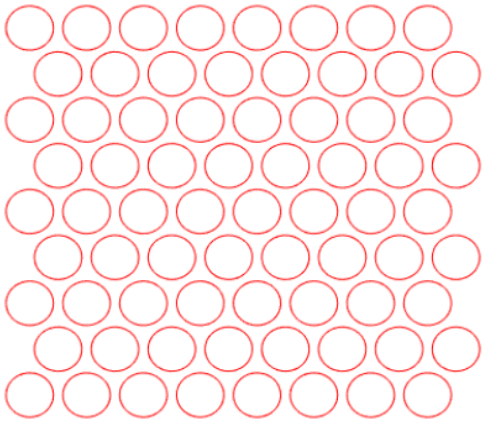

Fig 8 Hexagonal patron of MCP.

In all three modes, photons are not generated simultaneously instead, they are fed sequentially into the simulator and the coordinates of each event are determined from the probability matrix as follows:

- A random uniform number is generated in [0, 1].
- The first index in the accumulation array that is equal or greater than this number is found. This index corresponds to a two-dimensional location in the array that can be converted into a pair of coordinates, if required.

4.2 MCP

The function with name drop_hex filters the input UV photons by a hexagonal patron of pores as shown in Figure 8.. The MCP pores patron works like a geometrical filter on the input image/data. Computationally, this filtering is implemented as follows:

- Firstly, the location of the centres of the pores is defined. The right offset in the X-axis between rows is $\frac{size_{pore+pitch}}{2}$ and the distance between rows (Y-axis) is:

$$h = \cos\frac{\pi}{6} * size_{pore+pitch} \tag{2}$$

- For each incoming photon, the nearest pore is identified (more details on this step are in the code).
- Finally, it is checked that the photon impact within the pore and not in the interface between two pores.

In the subsequent step, the electron showers induced by the UV photons are simulated. We have defined the *MCP transfer function* as the mathematical operator that transforms the photon coming into a pore in an electron shower with known parameters (intensity, dispersion in the X and Y axis). Given the statistical nature of the MCP response, the transfer function also operates statistically. In works in two modes: simple and MCP-specific. In the simple transfer function mode, a bidimensional uncorrelated normal is generated for each UV photon, as follows:

- Generate electron showers with distribution $Z = [Z_1, Z_2]$ where Z_i are normal uncorrelated distribution with mean μ_i and variance σ^2 (input data entered by the user that represents the dispersion of the shower). This distribution Z is a bidimensional normal with mean $\mu = [\mu_1, \mu_2]$ (the location of the UV photon that induces the electron shower) and covariance matrix:

$$\Sigma = \begin{pmatrix} \sigma^2 & 0 \\ 0 & \sigma^2 \end{pmatrix} \tag{3}$$

  The simulator takes pairs of standard normal random numbers in $[0, 1]$ that are denoted with X. The dimension $dim(X) = (n, 2)$, with n the number of electrons which form the electron shower.

- Calculate the Cholescky decomposition $\Sigma = A * A^T$

- Generate the resultant electrons Y vector (the electron showers) as follow:

$$Y = M + A * X = M + \sigma * X \tag{4}$$

Where M is a vector with μ row repeated n times, $dim(M) = (n, 2)$. This mode is implemented to work with generic or poorly characterized MCP detectors.

In the MCP-specific transfer function mode the steps are different. The user introduces an image (or set of images) that describes well the statistical properties of the MCP. This image is processed by the simulator to obtain the statistical data that will be used to define the MCP transfer function. A sample image to illustrate the process is shown in Figure 9. After loading the image into the simulator, the next steps are:

1. To remove the noise (bwareaopen).

2. To fill gaps if needed (strel, imclose, imfill).

3. To detect the events (bwboundaries, regionprops) and define their boundaries. The boundary is measured at the level where the event energy distribution meets a user provided threshold (normally based on the image noise). The Python function regionprops is used to define the boundaries. It provides: area, bounding box, eccentricity, major and minor axis of the ellipse regions, max intensity of the region...

4. To filter the events by the eccentricity of the boundary. The boundary is fitted to an ellipse and those events with eccentricity larger than eccentricity filter value are removed to avoid polluting the characterization of the MCP by inclusion of very close events. We can see the results in Figure 9, with 0.7 eccentricity filter value.

The flux distribution of a given event is assumed to be bi-dimensional Gaussian. For each event the parameters of the Gaussian (dispersion in X and Y, and intensity or total flux) are computed. This is not done by Gaussian fitting of each event because it is too much time consuming instead the regionprops function is used.

Brightness level equal to 4 has been set at the threshold to measure the boundary of the events. Given the Gaussian like shape of the flux distribution, the boundary is elliptical. To calculate the

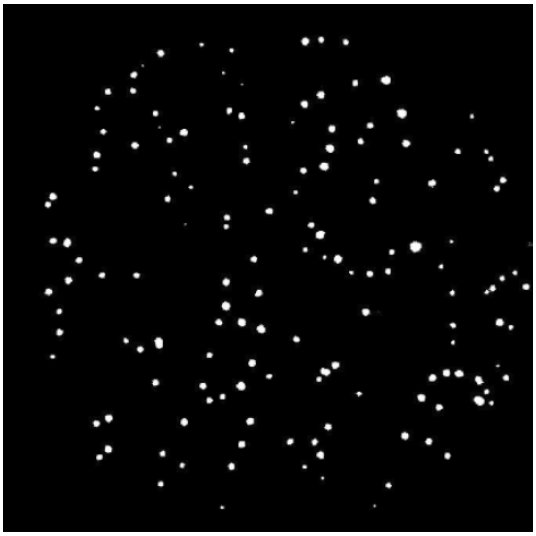

Fig 9 Real MCP example image with 0.7 eccentricity filter value.

dispersions (σ_x and σ_y) of the Gaussian from measuring the semimajor and semiminor axis of the ellipse, the properties of the Gaussian function have been taken into account.

The Gaussian function is:

$$G(x,y) = I_{max} e^{-\frac{x^2}{2\sigma_x^2}+\frac{y^2}{2\sigma_y^2}} \tag{5}$$

At height 4, the semi major and semi minor axes of the ellipse (a and b respectively) using the regionsprop function that also provides the maximum intensity within the region, I_{max} . Hence along the x-axis,

$$4 = G(a) \tag{6}$$

$$4 = I_{max} e^{-\frac{a^2}{2\sigma_x^2}} \longrightarrow \sigma_x = \frac{a}{\sqrt{2 * \ln \frac{I_max}{4}}} \tag{7}$$

and similarly for σ_y, hence:

$$(\sigma_x, \sigma_y) = (\frac{a}{2\sqrt{2 * \ln \frac{I_{max}}{4}}}, \frac{b}{2\sqrt{2 * \ln \frac{I_{max}}{4}}}) \tag{8}$$

both given in pixels.

Also, the total flux of the event (the so-called intensity, I) can be determined since,

$$I = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} G(x,y)dxdy = 2 * \pi * I_{max} * \sigma_x * \sigma_y \tag{9}$$

Three vectors are created containing σ_x, σ_y and I for all the events in the image. From them, functions I-σ_x and I-σ_y are built (approximated by a linear regression with 3^{rd} degree polynomial, see Figure 10). For the intensity values, the simulator builds a histogram (Figure 11). With this statistical information, the program generates new showers with the next procedure (for each uv-photon):

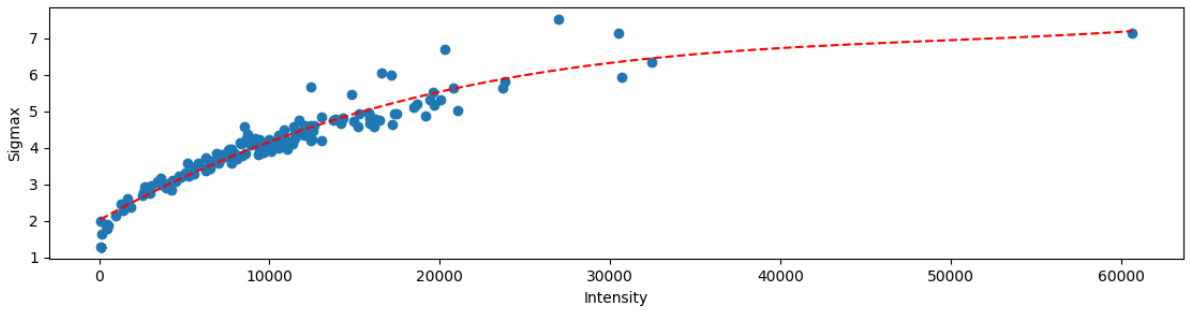

Fig 10 I-σ_x figure with regression line (red), with 0.7 eccentricity value for eccentricity filter.

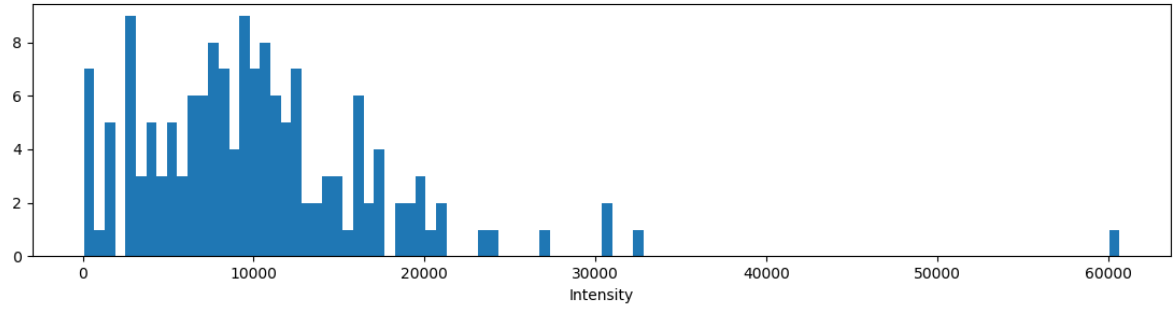

Fig 11 Histogram of Intensity with 0.7 eccentricity value for eccentricity filter.

1. Random value is generated for the intensity (with the distribution give by the previous histogram).

2. σ_x, σ_y values are generated with the previous regressions.

4.3 Phosphor

This element only multiplies the incoming electrons by a factor set by the user. In the actual implementation, the phosphor effect generates factor optical photons for each electron (in the same position). For other different behaviour, this specific implementation must be changed.

Finally, note that the statistical properties of the events are derived from an image that incorporates the statistics of the MCP electron showers but also the geometry, amplification, thresholding of the various components of the MCP. This affects both the dispersion and the intensity. In particular, concerning the intensity, the total number of electrons from each micro-tube is scaled additionally from the image to derive the expected amplification in the micro-tube as,

$$\frac{Total\ flux\ of\ the\ electron\ shower\ (e^-)}{Total\ flux\ of\ the\ event\ (CMOS\ image)} = \frac{Gain_{CMOS}}{QE_{CMOS} * FluxDecrement_{FiberTape} * Gain_{Phosphor}} \tag{10}$$

Within the simulator, the total flux,**I**, is multiplied by the Gain of the CMOS to obtain the intensity as number of optical photons and is divided by the quantum efficiency of the CMOS, the flux decrement in the fiber tape and the phosphor gain to revert the effect in the flux.

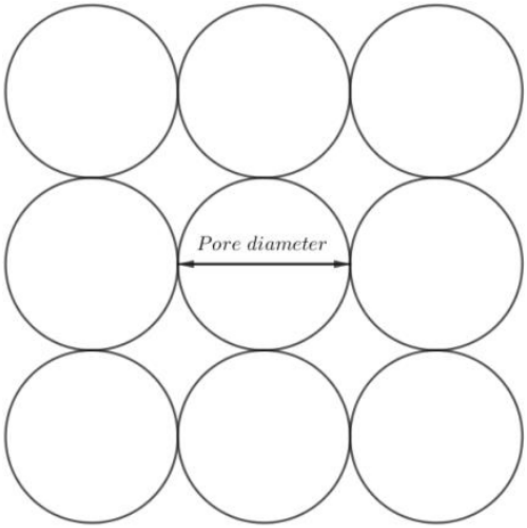

Fig 12 Fiber bundle implemented pattern.

4.4 Fiber Tape

Fibre tape process the optical photons that enter in the pores and distribute them uniformly inside the tubes (this step reduces the shower in a factor of $\pi/4$ because of the geometry pattern of the fibres, Figure 12.

To implement this feature, we calculate the centre of the pores and the number of electrons that stay inside each pore, and the algorithm distributes uniformly the number of photons that come in inside each fibre. The next steps are apply the fibre transmittance (that decrements the flux that cross the fibres) and the fibre tape scaled process (which will be applied in the CMOS element when the pixel is scaled for the mapping) .

4.5 CMOS

The CMOS element accumulates the photons from the end of the fibres into pixels and store them in a matrix. If the hot pixel effect is enabled, the program applies to the image a salt and pepper effect (for simulate the hot pixels). Additionally, if flat field effect is enabled, the program adds this effect to the previous matrix. If the user wants to select an area, the option Selection Area should be enabled.

5 Installation

To be able to execute the program, you must have installed the following Python packages (available in pip or conda package installers):

It leaves the choice of newer versions of these packages to the user but the safe functioning of the program cannot be guaranteed. The next step is unpack the MCPSim-Py file in the working directory. The following files or directories will be created:

- app: folder that contains all source code related with the GUI (*view* folder), the instrumentation (*model* folder) and the interface code between them (*controller* folder).

Table 1 Python packages and links.

Package name	Version	Link
Numpy	1.18.1	https://numpy.org/
Numba	0.48.0	https://numba.pydata.org/
Scikit-image	0.16.2	https://scikit-image.org/
Joblib	0.14.1	https://joblib.readthedocs.io/en/latest/
Scikit-learn	0.22.1	https://scikit-learn.org/stable/
Pandas	1.0.1	https://pandas.pydata.org/
Tk	8.6.8	https://docs.python.org/3/library/tk.html

- img: folder that contains some sample images of the sky, the CMOS read out flat field and the MCP behaviour.

To execute the program, go to *app/controller/* folder, and launch *Controller.py* with your Python interpreter. The program will open a new window to interact with the simulator.

6 Closing remarks

The simulator as defined is very versatile and fully operational. On going developments include the addition of cosmic rays and the simulation of a photon counting mode from the expected spatial distribution of the count rate at the entrance window of the detector.

Acknowledgments

This work has been funded by the Ministry of Science and Innovation of Spain under grant ESP 87813-R.

References

[1] M. Sachkov, B. Shustov and A. I. Gómez de Castro, "The World Space Observatory - Ultraviolet mission: science program and status report", *Proc. SPIE*, in press, (2020).

[2] A.I. Gómez de Castro, L. Diez, J. Yañez et al., "The detector for the far ultraviolet channel of the imaging instrument on board the Spektr-UF (WSO-UV) space telescope", *Proc. SPIE*, in press (2020).

[3] Dipanjan Mukherjee and Shyam Tandon, "Report for topical project on UVIT simulator", "urlhttp://astrosat.iucaa.in/˜astrosat/uvit_simulator/Uvit_simulator_guide.pdf" (2010).

[4] K. Suhling, R. Airey and B. Morgan, "Optimisation of centroiding algorithms for photon event counting imaging", *Nuclear Instruments and Methods in Physics Research, Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 437, no. 2-3, pp. 393-418, (1999).

[5] J. Hutchings, J. Postma, D. Asquin and D. Leahy, "Photon event centroiding with UV photon-counting detectors", *Publications of the Astronomical Society of the Pacific*, vol. 119, no. 860, pp. 1152-1162, (2007).

List of Figures

List of Tables