1 Introduction

Overhead Persistent InfraRed (OPIR) sensors are used in satellites to find interesting heat signatures. To better understand how filters will apply to data, test new filters, and testing applications we designed a simulation of OPIR data.

2 Simulation

The simulation of this data is intended to be as close to the real data as possible. In an attempt to achieve this, we must consider three important pieces of information: background noise, signal, and detector limits.

2.1 Background Noise

Thermal noise is easily modeled by a Rayleigh Distribution. Rayleigh Distributions are characterized by a large peak at a given value with a lingering tail to higher values, see figure **??**. The probability distribution function (PDF) of a Rayleigh Distribution is

$$f(x;\sigma) = \frac{x}{\sigma^2} e^{-\frac{x^2}{2\sigma^2}} \tag{1}$$

where x is the variable (greater than or equal to zero), and σ is the scale parameter. Setting σ is very simple, as the mean of the distribution is

$$\bar{x} = \sigma\sqrt{\frac{\pi}{2}} \tag{2}$$

By choosing a value for the mean of your distribution, you can find the σ for the distribution.

To use this distribution, we must calculate the Cumulative Distribution Function (CDF), which is merely the integral of the PDF. The CDF is normalized to 1 over the domain of [0,∞), as this means there is a 100% of getting a value.

$$F(x,\sigma) = \int_0^x \frac{x}{\sigma^2} e^{-\frac{x^2}{2\sigma^2}} dx = 1 - e^{-\frac{x^2}{2\sigma^2}} \tag{3}$$

We use this CDF to find a value for the thermal noise in this manner. We invert the CDF, solving for the value of x:

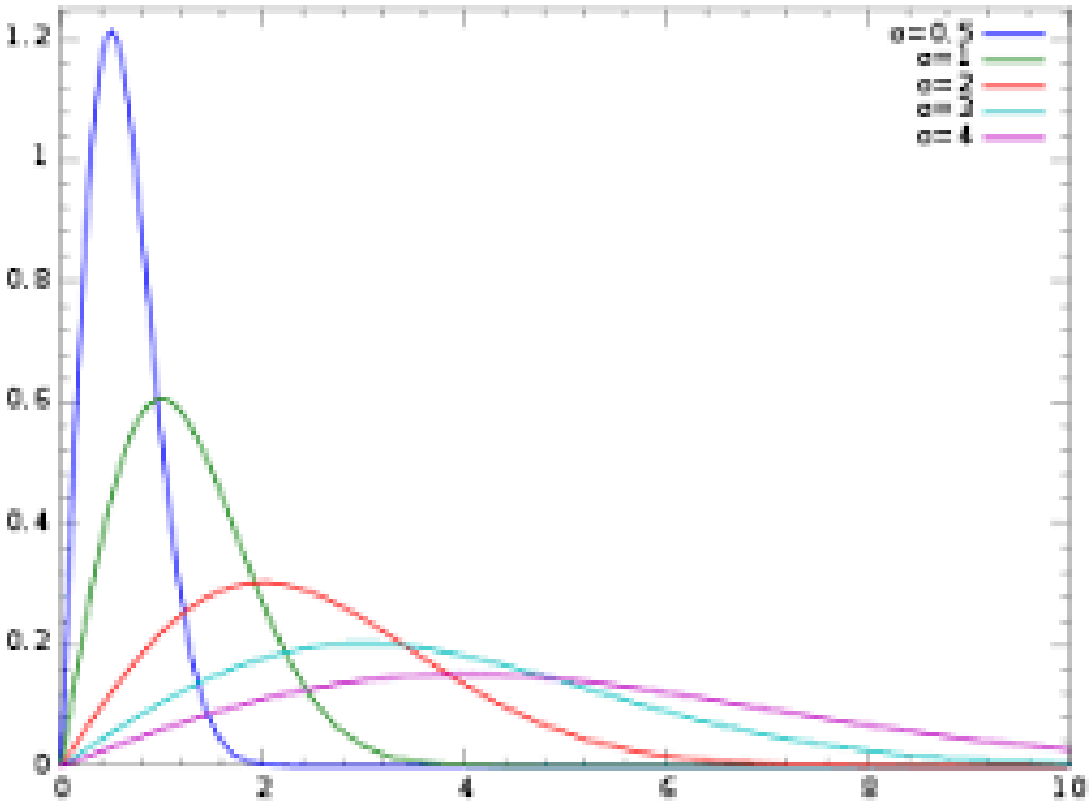


Figure 1: Example of Rayleigh Distribution from Wikipedia. Notice the only difference in distributions is due to the change in σ.

$$
\begin{aligned}
F(x,\sigma) = y &= 1 - e^{-\frac{x^2}{2\sigma^2}} \\
e^{-\frac{x^2}{2\sigma^2}} &= 1 - y \\
\frac{x^2}{2\sigma^2} &= -\ln(1-y) \\
x &= \sqrt{-2\sigma^2 \ln(1-y)}
\end{aligned}
$$

By choosing a random number between 0 and 1 for y, we now get a value of our thermal noise (x). This distribution follows the Rayleigh Distribution and is easily change to different variations due to the σ dependence.

2.2 Signal

The signal strength is the most straight forward component. A 'signal' is defined as a signature that does not deviate in strength. For simulation purposes, that signal strength can be drawn from a flat distribution. There are minor modifications to the signal to help simulate realistic conditions. The signal could move around the image, it could start at a low signal strength and gain in power, or start strong and weaken in strength.

2.3 Detector limits

Every detector has unique capabilities and limits: resolution, field of view, saturation, frequency of capture to name a few. Those need to be handled with care in a simulation to make sure the concepts being tested work with that detector. These limitations work on a case by case basis, but the few encountered in simulation are the frequency of capture and the saturation of the cameras. Frequency of capture informs us how often the detector saves information. Saturation tells us the strongest signal the detector can save. For example, a detector may have a saturation of 50W. Signal one has a strength of 51W and signal two has a strength of 75 W. the detector would read both of those signals as having a strength of 50W due to saturation. You cannot make the saturation arbitrarily high, as there are other limits that need to be balanced against the saturation.

3 Implementation

The currently implementation of the simulation is written in python. The program asks for some items when starting up:

1. Size of X-dimension of array (pixels)
2. Size of Y-dimension of array (pixels)
3. Frequency of Capture (Hz), f
4. Length of record (sec), t
5. Include background noise

6. Fine noise or coarse [COARSE NOT FULLY IMPLEMENTED]

7. Filtering Algorithm

After that has been set, the algorithm will run according to those presets. The algorithm will calculate how many snapshots it must make, freq*time. It will choose a random number of signal events and a random time between (0,t) for each signal to appear in the image.

The procedure for the image is also very simple. For every pixel in the image, we get a thermal noise using the algorithm described in section 2.1. For that same pixel, we ask if a signal event is occupying that pixel at that time. If yes, we add the signal strength to that pixel. If no, we move on to the next pixel in the array. If the pixel value is above a saturation limit, we force the limit onto that pixel.

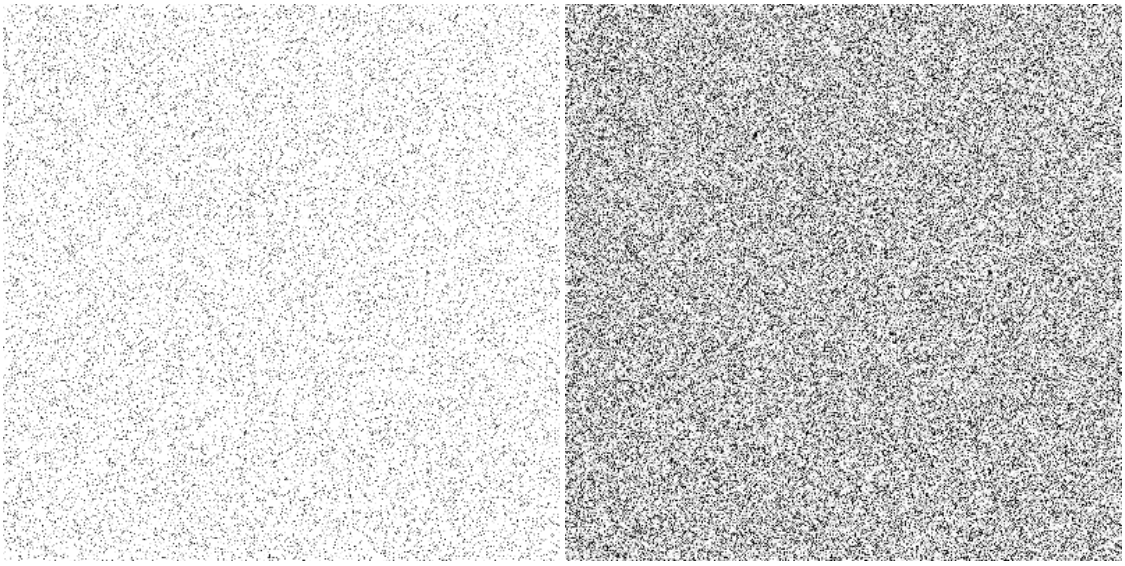
This method is done for every pixel in the array, for every snapshot of simulation. This can be quite time consuming depending on the size of the array and there may be a way to speed up the calculations to reduce time.

If a filter is applied, before saving the information the pixel value is run through the filter. The 2 working filters are First Order Filter and Weightd Average Filter. The first order filter uses a simple equation to help filter the image

$$\mathrm{P[i,j,t]} = \mathrm{decay} * \mathrm{X[i,j,t]} - (1 - \mathrm{decay}) * \mathrm{P[i,j,t-1]} \tag{4}$$

where P is the output value for the filtered pixel, decay is a constant, and X is the non-filtered pixel value at that time, and P[i,j,t- 1]] is the pixel value from the previous time frame. This filter has been used in other programs, and is easily adjustable by changing the value of the decay constant. The difference between filtered and non-filtered can be seen in figure 2.

The Weighted Average filter is seeking to take advantage of the fluctuations in the thermal noise. When averaging, signal has a constant strength, so the average will be approximately the same as the signal strength. But thermal noise fluctuations will vary wildly either up or down. By averaging the noise, we can attempt to suppress it to a lower value. We average over 3 samples from the past: $\mathrm{t}_0, \mathrm{t}_0 - 1, \mathrm{t}_0 - 2$. We give weights to each value, with the current time having the largest weight and the latest time having the smallest. The equation for this filter is

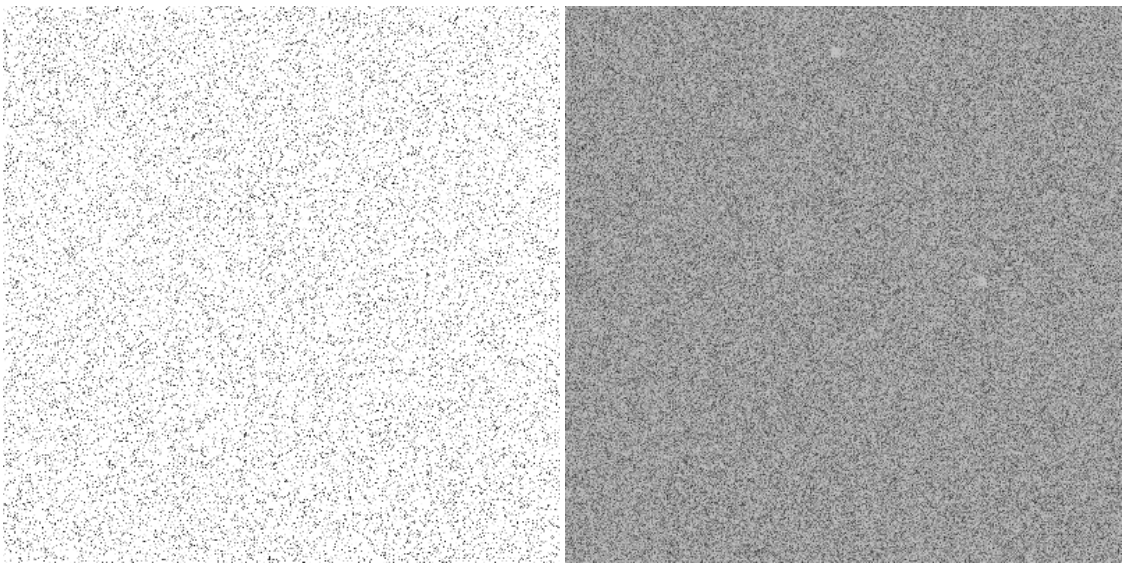


(a) Non-filtered Simulation example. (b) First Order Filter Applied to Simulation.

Figure 2: An example of the first order filter at work. The left image is the simulation without the filter, while the right figure has the First Order Filter applied

$$\mathrm{P[i,j,t]} = \sum_{\mathrm{k=0}}^{2} \mathrm{Weight[k]} * \mathrm{X[i,j,t-k]} \tag{5}$$

where P[i] is the filtered value of the pixel, X[i-k] is the read in value at that pixel, and Weight[k] is the weight given to that pixel. The difference between filtered and non-filtered can be seen in figure 3



(a) Non-filtered Simulation example.

(b) Weighted Average Filter Applied to Simulation.

Figure 3: An example of the weighted average filter at work. The left image is the simulation without the filter, while the right figure has the Weight Average Filter applied