CTA Data Processing

Image cleaning algorithms

Jérémie Decock

CEA Saclay - Irfu/SAp

July 28, 2016

Introduction

Subject

First step:

- Detect, locate and characterize events in images
- $\Rightarrow$ Use *Hilas parametrization*

Problem:

- Images are noised
- $\Rightarrow$ They have to be "cleaned" first!

Clarifications

Different kind of "noise" in telescope images (*to be completed...*)

1. Instrumental noise (Photomultiplier Tubes, ...)
   - Thermionic emission
   - Radiations
   - Electric noise
2. Background noise (*Night Sky Background* or NSB)
   - Parasite light (moon, stars, planes, light pollution, ...)
3. Distortions (not actually "noise" but alter the signal too)
   - Atmospheric distortion of the air (density, ...)
   - Magnetic fields
4. Undesired cosmic rays
   - Atomic nuclei (make nearly similar showers than γ photons)
   - Electrons (make similar showers than γ photons)

Objectives here

Preprocessing for Hillas parametrization

Algorithms to remove:

- Instrumental noise
- Background noise

Distortions: ignored so far

Undesired cosmic rays: the next step!

Related presentations

CTA data pipeline - Introduction:

- PDF: `http://www.jdhp.org/dl/cta-data-pipeline-introduction.pdf`
- Source code: `https://github.com/jdhp-sap-docs/cta-data-pipeline-introduction`

Cleaning preprocessing for Hillas parametrization: current method (HESS)

The "Tailcuts clean" algorithm

Remove the noise in images with a very simple filter:

- keep pixels above a given threshold
- keep some neighbors of these selected pixels: those above a second (lower) threshold

The "Tailcuts clean" listing (Python)

```python
def tailcuts_clean(geom, image, pedvars,
    picture_thresh=4.25, boundary_thresh=2.25):

    clean_mask = image >= picture_thresh * pedvars
    boundary_mask = image >= boundary_thresh * pedvars

    boundary_ids = []
    for pix_id in geom.pix_id[boundary_mask]:
        if clean_mask[geom.neighbors[pix_id]].any():
            boundary_ids.append(pix_id)

    clean_mask[boundary_ids] = True

    return clean_mask
```

listings/tailcuts_clean.py

Remarks and analyse

- Fast and simple
- Sufficient for bright events
- But surely we can do better!

Discrete Fourier Transform method

First alternative

Discrete Fourier Transform method

- Previous filter: threshold in the main space
- Better idea: threshold in a different space where signal and noise can be easily separated

Fourier transform

$$f(t) = \frac{a_0}{2} + \sum_{n=1}^{\infty}(a_n \cos(nt) + b_n \sin(nt))$$

$$a_n = \frac{1}{\pi}\int_{-\pi}^{\pi} f(t)\cos(nt)dt$$

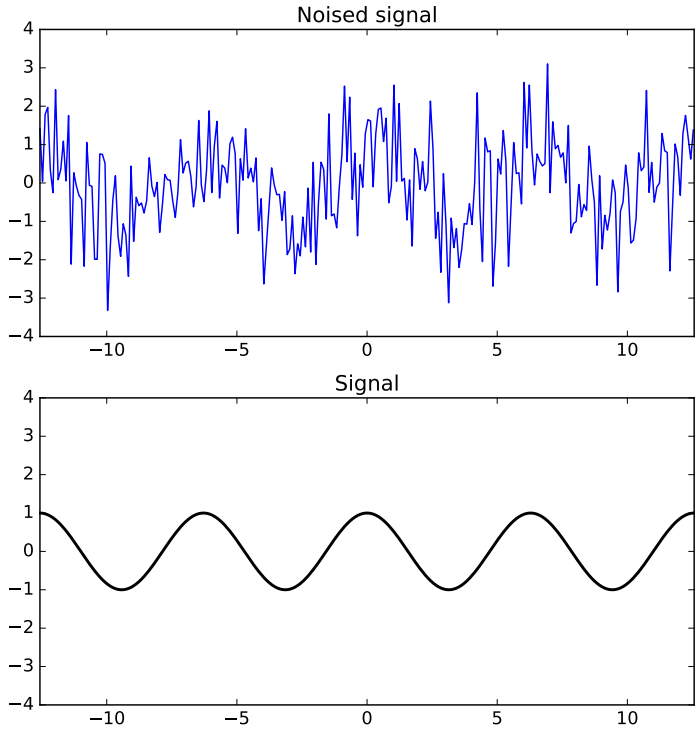
$$b_n = \frac{1}{\pi}\int_{-\pi}^{\pi} f(t)\sin(nt)dt$$

Discrete Fourier Transform (DFT)

Fourier Transform for discrete signals (digital pictures, ...)

Clean signals with DFT

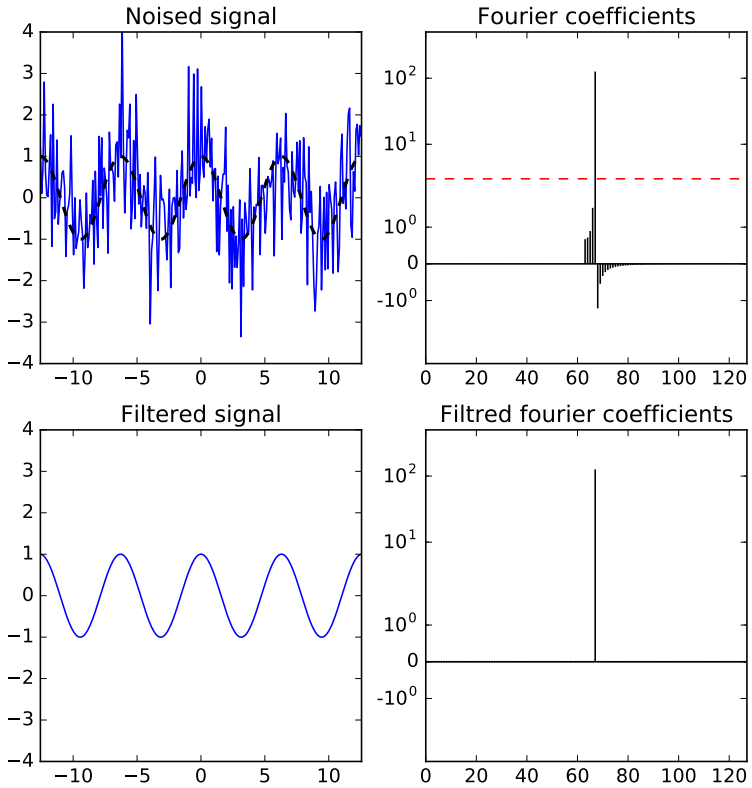
Remove noise in direct space: difficult (here)



Clean signals with DFT

Remove noise in transformed space: easy (here)

- Apply DFT
- Apply a threshold
- Apply invert DFT



Remarks

FFT can be applied to any T-periodic function f verifying the *Dirichlet conditions*:

- f must be continuous
- *and* monotonic
- on a finite number of sub-intervals (of T)

Signals defined on bounded intervals (e.g. images) can be considered as periodic functions (applying infinite repetitions)

Analyse

Works well:

- when the Fourier coefficients for the signal and the noise can easily be separated in the Fourier space (obviously...)
- e.g. when either the signal *or* the noise can be defined with few big Fourier coefficients (i.e. signal or noise have a few number of significant harmonics)

Wavelet Transform method

Second alternative

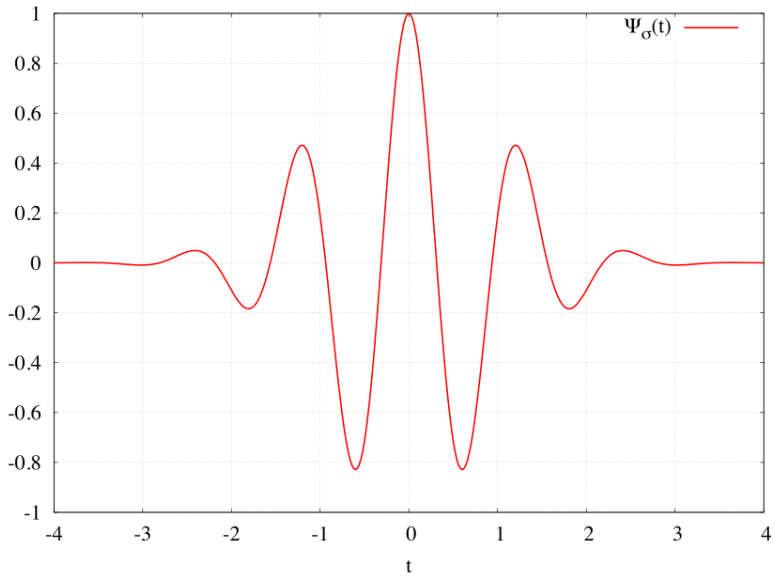
Wavelet Transform method

Filter images with wavelets: roughly the same idea

- Use wavelets instead sin and cos functions as new bases in the transformed space
- The transformed space now contains spatial information: better for non-periodic signals

Overview

A wavelet looks like this (Morlet):



"A wave-like oscillation with an amplitude that begins at zero, increases, and then decreases back to zero"

Clean signals with Wavelets

The same procedure than for DFT:

- Apply Wavelet Transform
- Apply a threshold in the transformed space
- Invert the Wavelet Transform

Conclusion

Tools and Documents

CTA pipe is available here:
`https://github.com/cta-observatory/ctapipe`

Thanks to CosmoStat we have some tools to apply wavelet transforms ("mr_transform")

Tools and Documents

Some tools too:

- Some scripts to get pictures from CTA pipe: `https://github.com/jdhp-sap/snippets`
- A Jupyter notebook about DFT: `https://github.com/jdhp-docs/fft_notebook`
- Some DFT tests with Python/Numpy: `https://github.com/jeremiedecock/snippets/tree/master/python/numpy/fft_transform`

What's next ?

Test Wavelet filters (CosmoStat)

Make a benchmark using the MC simulator:

- Test Wavelet and DFT filters on a more realistic dataset
- Compare filtered images with the actual signal
- Exhaustive test (many different events from different sources at different energy)

References I


S. Mallat, *A wavelet tour of signal processing: The sparse way*, Elsevier Science, 2008.