aaa2016slides

September 22, 2016

#
Tratamiento formal de diferencia de imágenes con PSF espacialmente variable
###
Búsqueda de eventos transitorios
###
Bruno Sánchez, M. Domínguez, M. Lares
####
Reunión anual de Asociación Argentina de Astronomía
####
San Juan - Septiembre 2016
##
Astronomía de series temporales
|**Estallidos de rayos gamma**|**Ondas Gravitacionales**|** y más**| |:—:|:—:|:—:| ||||
##
Eventos transitorios

0.1 El desafío

Buscar objetos transitorios en el cielo requiere:

- Instrumento dedicado
- *Pipeline* veloz
- Análisis de diferencias de imágenes (DIA)

0.2 El instrumento: Proyecto TOROS

Un proyecto de telescopio sinoptico en el NOA Argentino ##
La Pipeline: TOROS Pipeline
###
El software CORRAL
<img src='logo_corral.jpg', style='float:center', width=350>
##
Estructura de la Pipeline

0.3 *Difference Image Analysis* (DIA)

- Alard & Lupton 1998: "A Method for Optimal Image Subtraction"

$$Ref(x,y) \otimes Kernel(u,v) = I(x,y)$$

$$Kernel(u,v) = \sum_n a_n N(\mu = 0, \sigma_u, \sigma_v) p_u(u) p_v(v)$$

0.4 *Difference Image Analysis* (DIA)

Es necesario igualar PSF entre ambas imágenes.

0.5 *Diference Image Analysis* (DIA)

- Bramich 2008: "A new algorithm for difference image analysis"

$$I_{ij} = \sum_{l,m} K_{lm} R_{i+l,j+m} + B_0$$

- Bramich 2013, 2015, 2016; Varias mejoras y detalles
- Zackay, Ofek & Gal-Yam 2016: "Proper image subtraction - optimal transient detection, photometry and hypothesis testing"

** En todas las técnicas es necesario medir PSF**
Algunas la asumen constante, o constante en regiones.

0.5.1 Zackay, Ofek, Gal-Yam (2016)

Referencia

$$R(x,y) = T(x,y) \otimes P_r(u,v,x,y) + \epsilon_r$$

Nueva Imagen

$$I(x,y) = T(x,y) \otimes P_n(u,v,x,y) + \epsilon_n$$

0.6 ProperImage

- *ProperImage* es un paquete en Python que implementa las técnicas de Zackay & Ofek (2016)
- Utiliza *Karhunen-Loeve* para estimar PSF variable en el espacio (Lauer 2002): es un método para extraer la máxima información posible de un conjunto de N observaciones, reduciendo la dimensionalidad:

$$P(u,v,x,y) = \sum_{i=1}^{\infty} a_i(u,v) p_i(x,y) \approx \sum_{i=1}^{K} a_i(u,v) p_i(x,y)$$

$$I(x,y) = \sum_{i=1}^{K} a_i(x,y)(T \otimes p_i)(x,y)$$

0.7 ProperImage

$$C_{i,j} =< P_i^*, P_j^* >$$

Autovalores, autovectores:

$$\lambda_j, \overrightarrow{\omega_j}$$

Al calcular los λ_j podemos despreciar aquellos que poseen auto-valores cercanos a cero.
Esto reduce la dimensión del problema a una dimension \$ K << N \$. Típicamente: \$ K = 1 \$.
Si la Psf varía intensamente es probable un orden mayor.

$$p_i = \sum_{j=1}^{N} (\overrightarrow{\omega_i})_j P_j^*$$

0.8 ProperImage

Los campos $ a_i(x, y) $ son determinados mediante proyecciones:

$$a_j(x_i, y_i) = \frac{< P_i^*, p_j >}{\|p_j\|^2}$$

Y luego se ajustan polinomios de orden 4, bidimensionales.
Nueva Imagen

$$I_{|H_0}(x, y) = T(x, y) \otimes P_n(u, v) + \epsilon_n$$

Transitorio

$$I_{|H_1(q,\alpha)}(x, y) = (T(x, y) + \alpha\delta_q(x, y)) \otimes P_n(u, v) + \epsilon_n$$

El *likelihood ratio* se define como:

$$\mathcal{L}(q, \alpha) = \frac{\mathcal{P}(I, R|\mathcal{H}_0)}{\mathcal{P}(I, R|\mathcal{H}_1(q, \alpha))}$$

El estadístico óptimo para detección de fuentes entonces es el siguiente:

$$\widehat{S} \equiv \widehat{\frac{1}{\alpha} \log \mathcal{L}}$$

El cual se calcula simultaneamente para todo valor de

$$\alpha$$

Y la diferencia de imágenes óptima es entonces:

$$\widehat{D} = \lambda(F_r\widehat{P_r}\widehat{N} - F_n\widehat{P_n}\widehat{R})$$

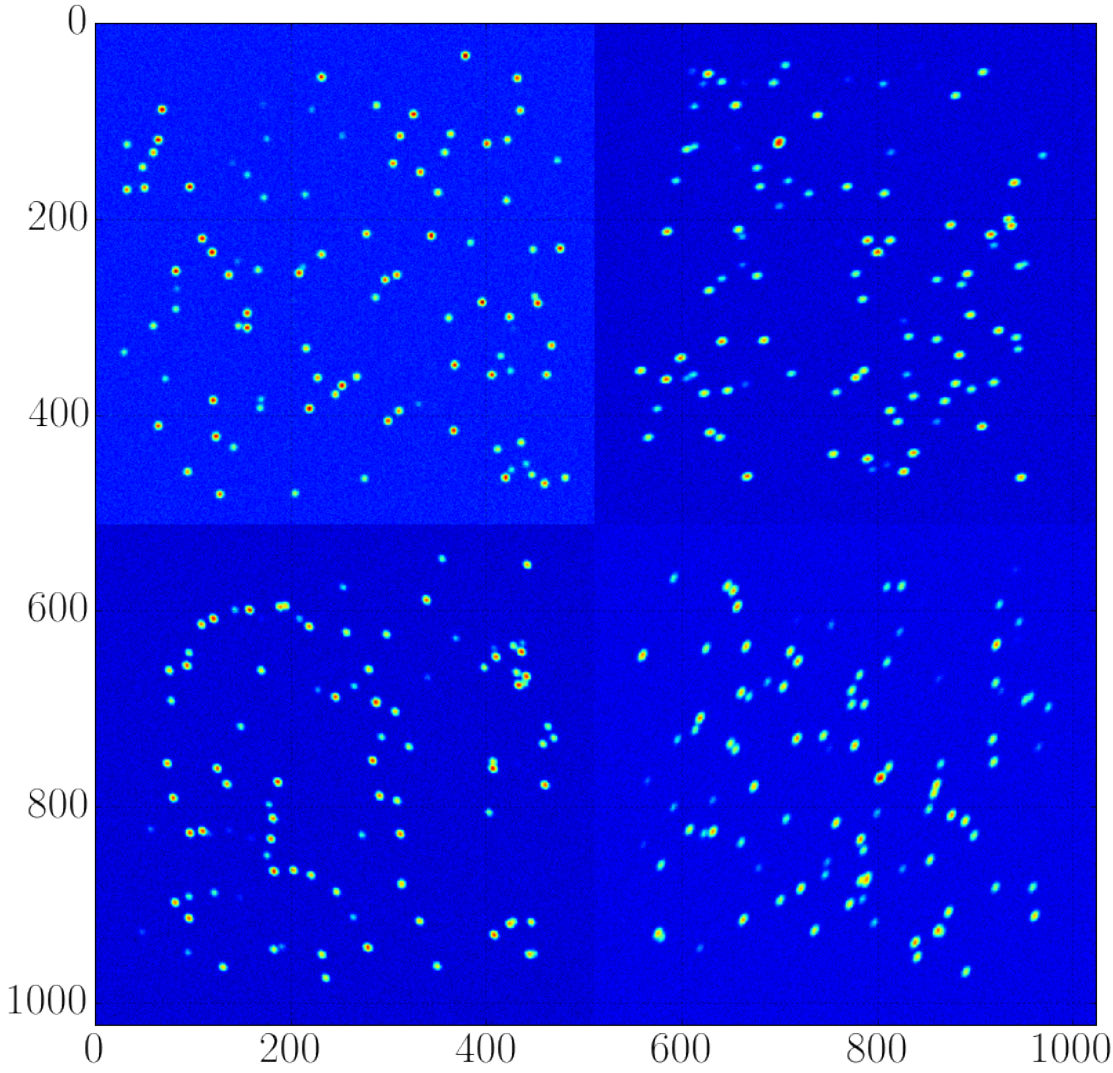
Y se relaciona con el valor de S:

$$\widehat{S} = \widehat{D}\widehat{\overline{P_D}}$$

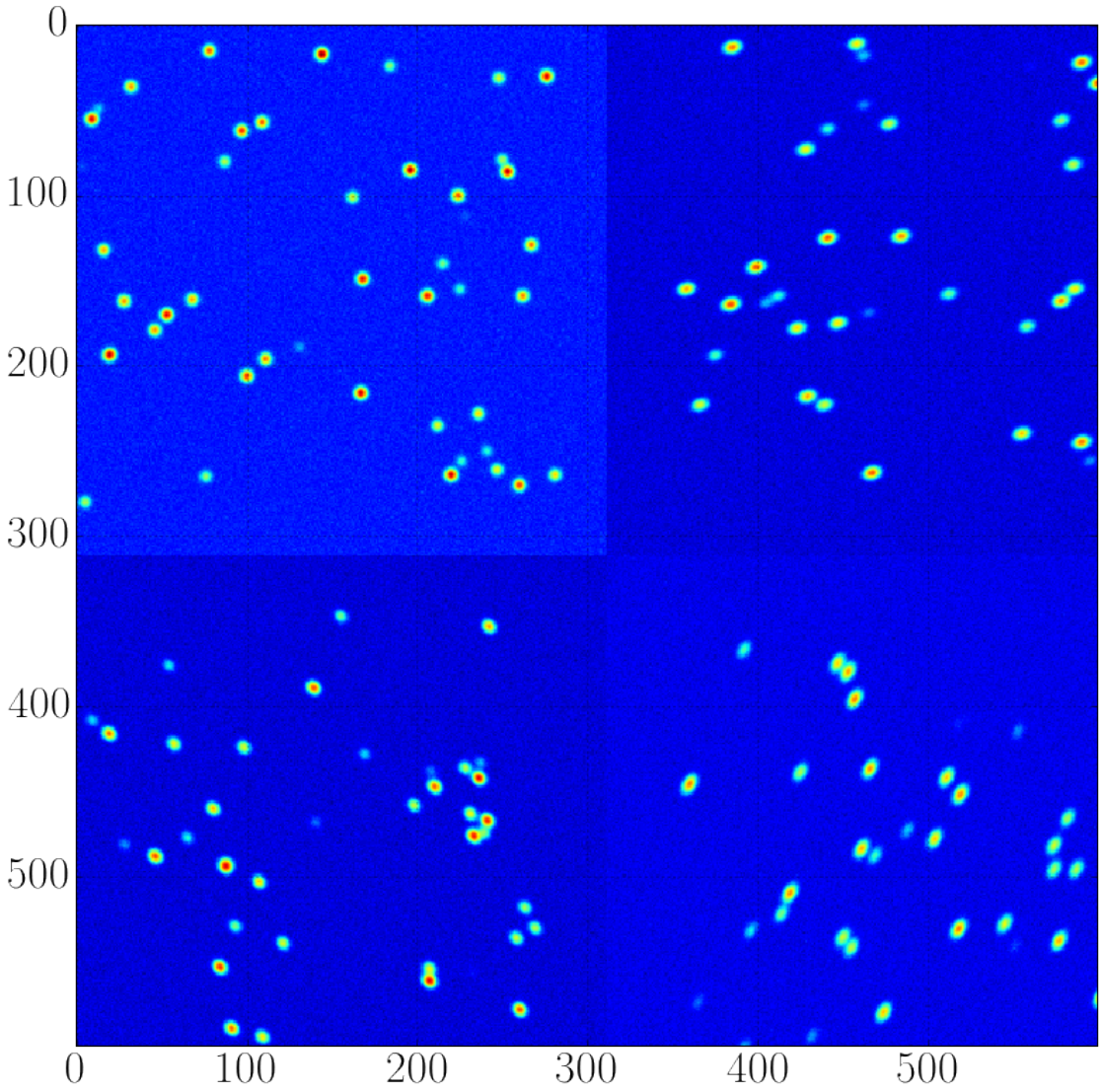
0.9 ProperImage

Un ejemplo:

```
In [1]: path = 'test_frame.fits'
        rebuildpath = 'frame_rebuild.fits'

In [2]: import numpy as np
        import matplotlib.pyplot as plt
        from astropy.io import fits
        from properimage import utils
        %matplotlib inline

In [3]: font = {'size' : 32}
        plt.rc('font', **font)

In [4]: plt.figure(figsize=(13,13))
        plt.imshow(np.log(fits.getdata(path)), interpolation='none')
        plt.grid()
```



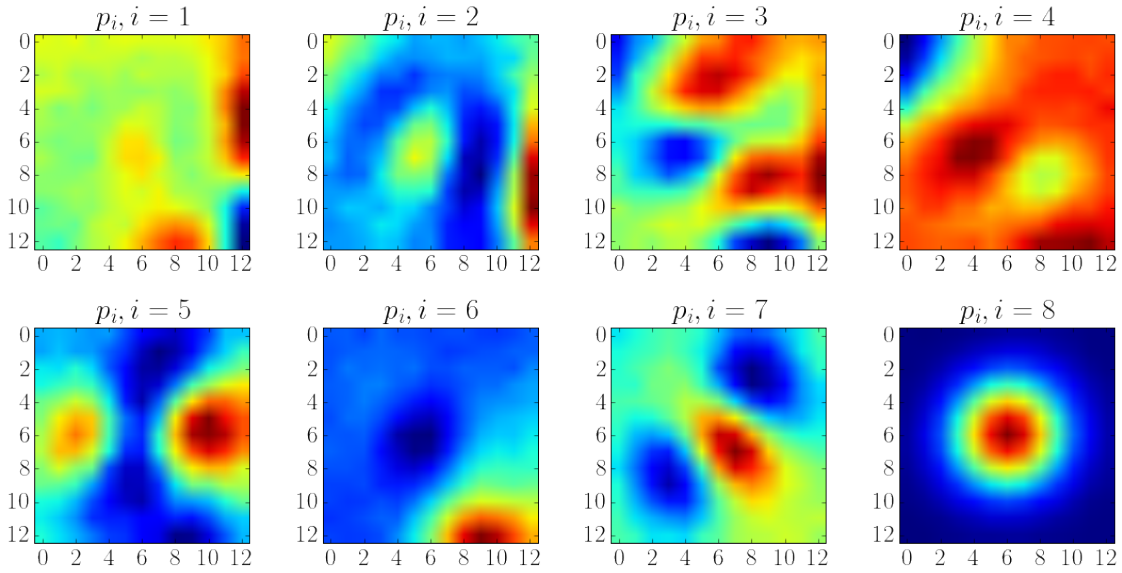

```
In [5]: plt.figure(figsize=(13,13))
        plt.imshow(np.log(fits.getdata(path)[200:800, 200:800]), interpolation='none')
        plt.grid()
```



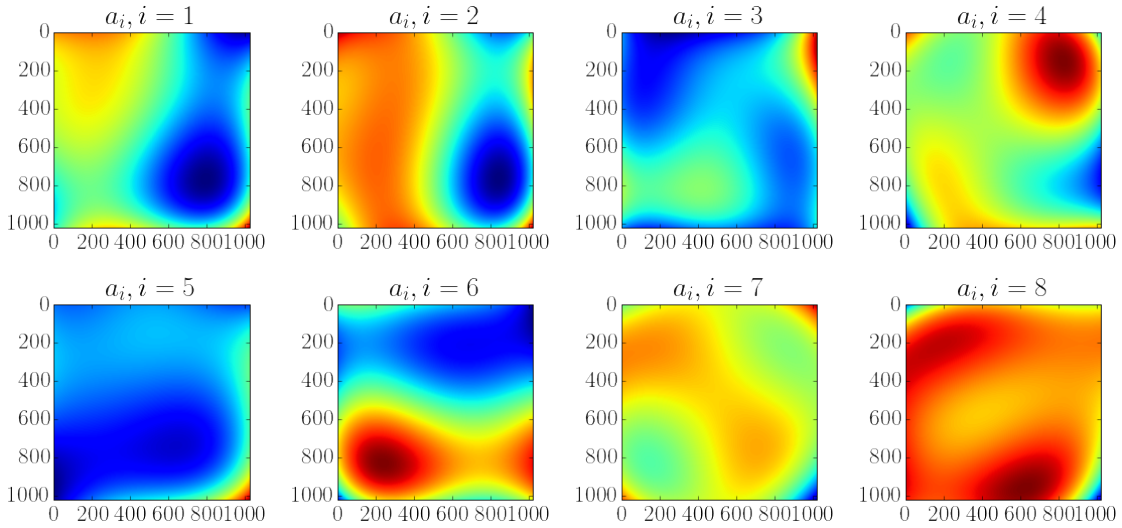

```
In [6]: import properimage.propercoadd as pc

        with pc.SingleImage(path) as img:
            afields, psf = img.get_variable_psf(pow_th=0.0008)
```

```
background subtracted image obtained
raw sources = 361
Sources good to calculate = 181
returning best sources
Fitshape = (13, 13)
returning Covariance Matrix
obtainig KL basis, using k = 8
obtaining a fields
returning variable psf
cleaning...
```

In [12]:
```
reload(utils)
font = {'size' : 23}; plt.rc('font', **font)
utils.plot_psfbasis(psf, nbook=True)
```

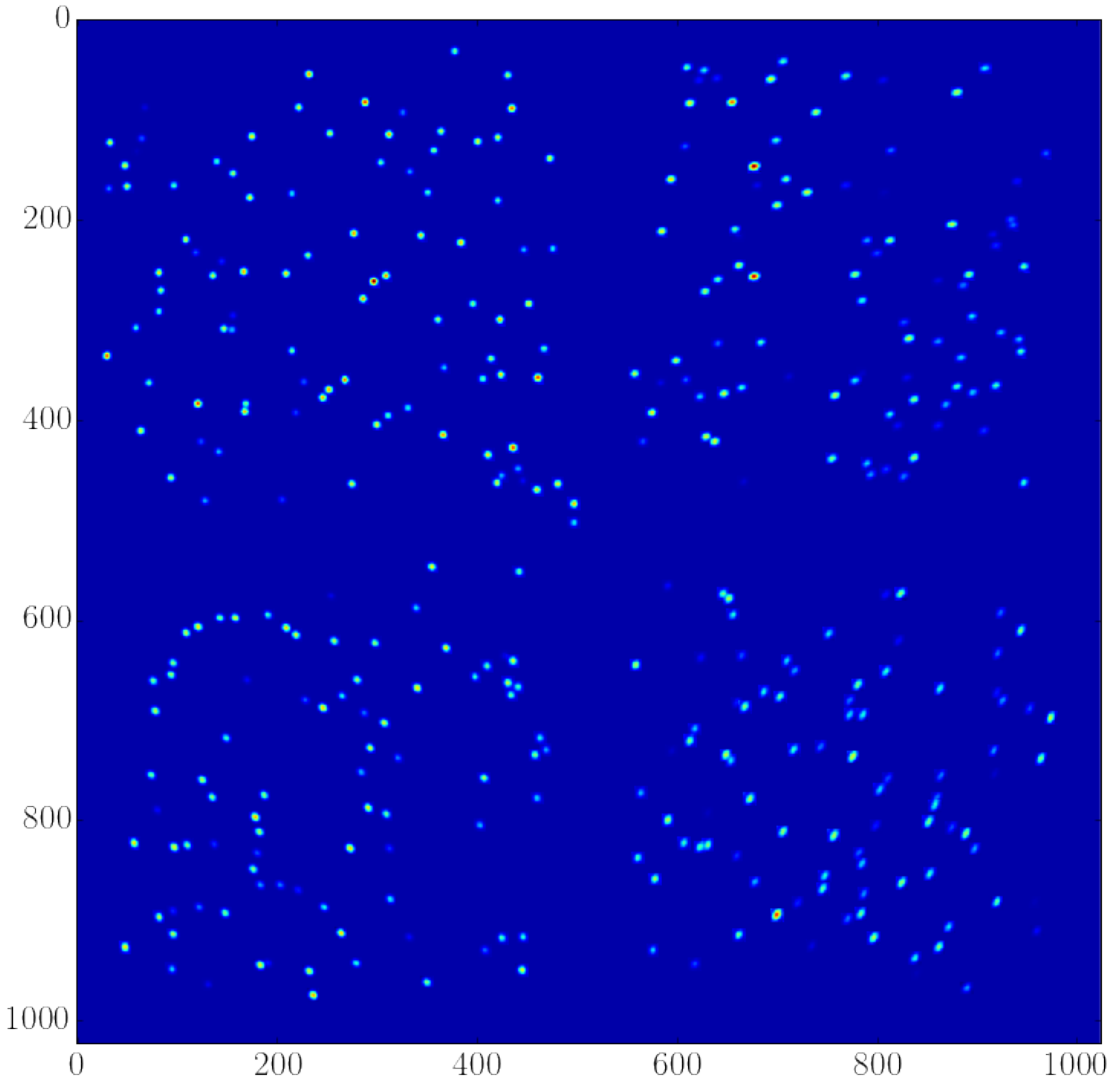


In [13]:
```
utils.plot_afields(afields, fits.getdata(path).shape, nbook=True)
```



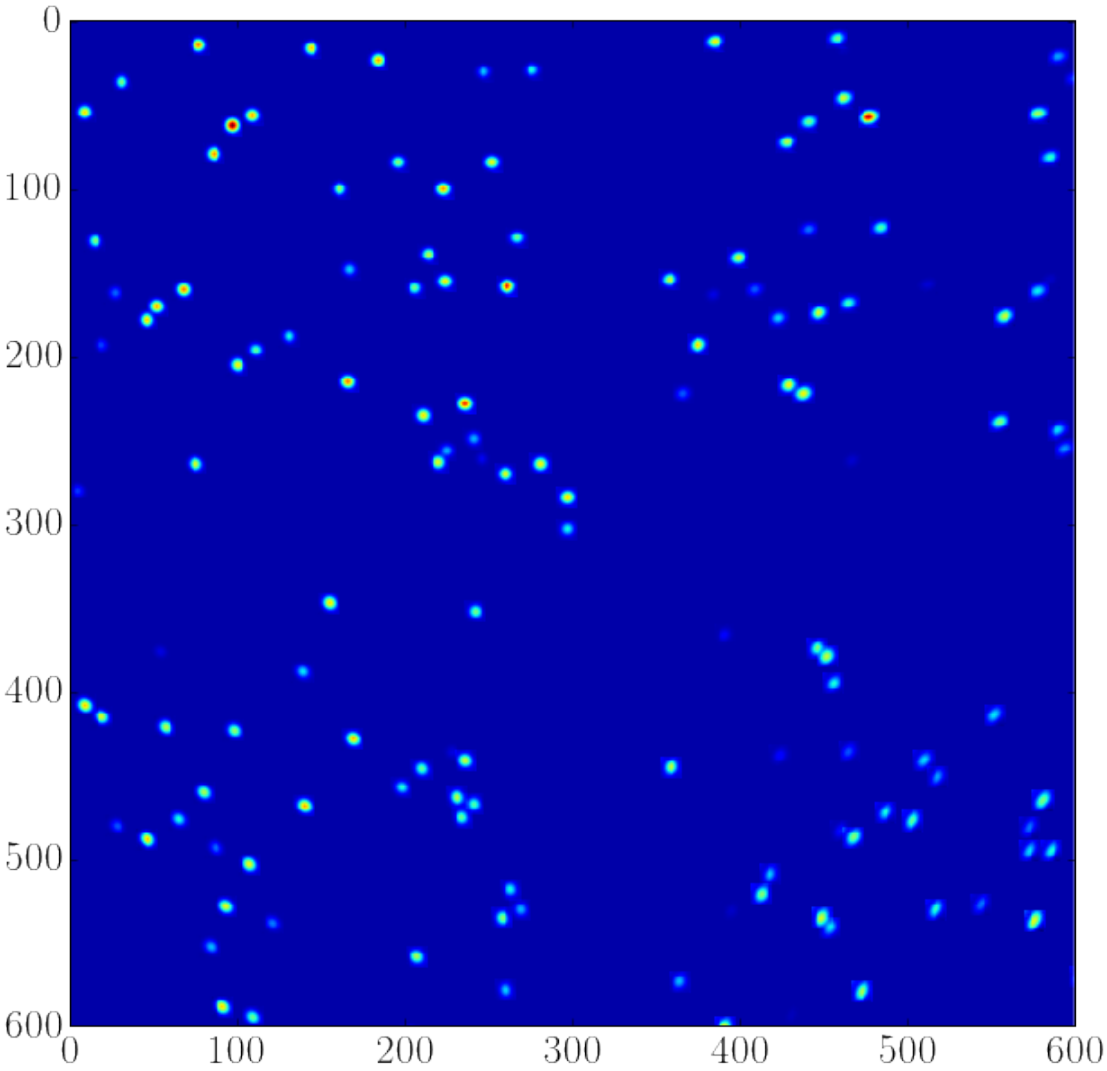
In [14]:
```
plt.figure(figsize=(12,12))
plt.imshow(np.log(fits.getdata(rebuildpath)), interpolation='none')
```

Out[14]:
```
<matplotlib.image.AxesImage at 0x7f8aa5810890>
```




```
In [15]: utils.plt.figure(figsize=(10,10))
         utils.plt.imshow(np.log(fits.getdata(rebuildpath)[200:800, 200:800]), interpolation='none')
```

```
Out[15]: <matplotlib.image.AxesImage at 0x7f8aadac6f50>
```



$$I(x,y) = \sum_{j=1}^{K} a_j(x,y) \left[\sum_{x=1}^{N} \delta_{(x_i,y_i)} \otimes p_j \right]$$

0.10 Aplicaciones:

- El método de medición de PSF está aplicado en ***ProperImage*** para
  - Coadición de imágenes
  - Substracción de imágenes **(es necesario *Real-Bogus*?)**
  - Fotometría -apertura y *PSF*- y detección de fuentes (*matched filter*)
  - Mediciones de forma de objetos extendidos

1 Preguntas??