Multi-resolution photometry: TFIT and CONVPHOT

Emiliano Merlin
INAF – OAR

Astrodeep Meeting – Paris, Jun 24/25, 2013

MAIN IDEA

- Obtain detailed photometry from low resolution image (LRI) using high resolution image (HRI) information

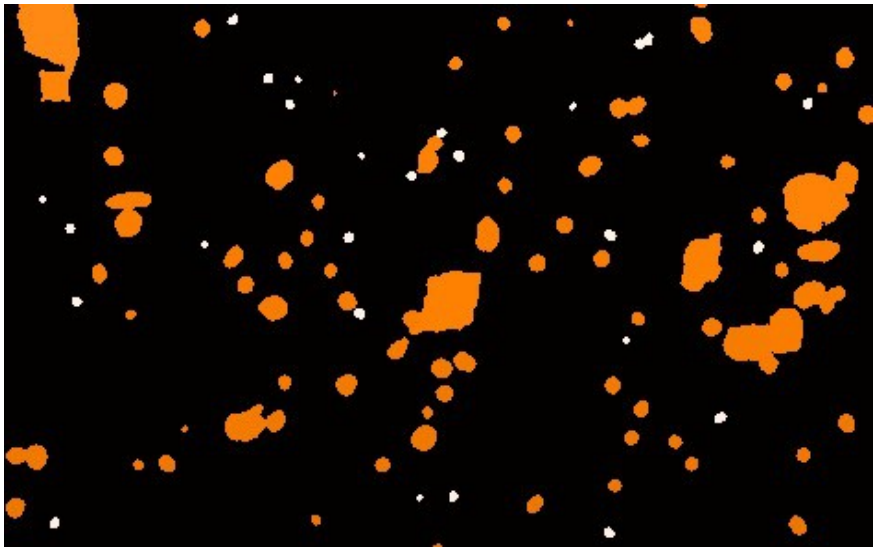
- Expecially desired for crowded fields, with overlapping and blending of sources in LRI

- Two similar codes: TFIT (Laidler+ 2007) and CONVPHOT (De Santis+ 2006)

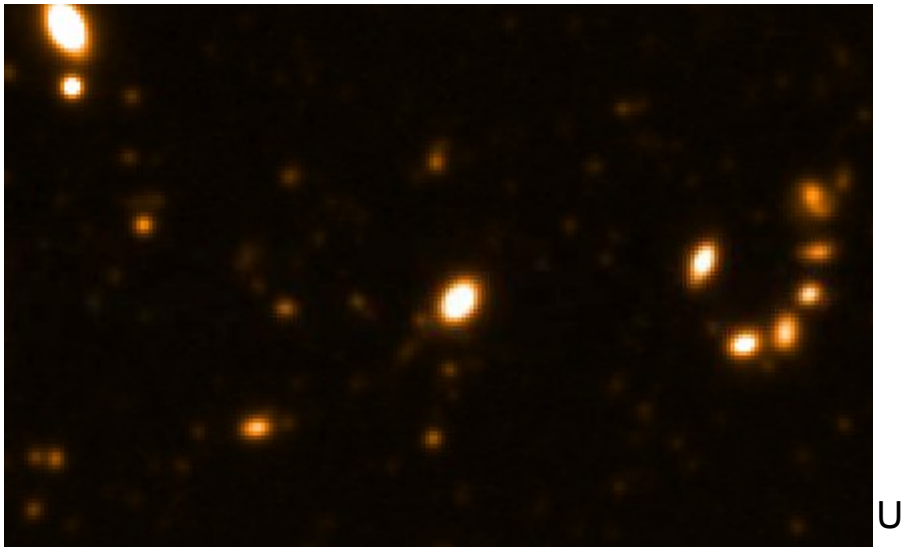
BASICS

1. Two images of the same field, at different resolution (wavelength):
- HRI + Seg (Sextractor) + cat
- LRI + RMS





UDS ACS



UDS Subaru I

BASICS

1. Two images of the same field, at different resolution (wavelength):
- HRI + Seg (Sextractor) + cat
- LRI + RMS

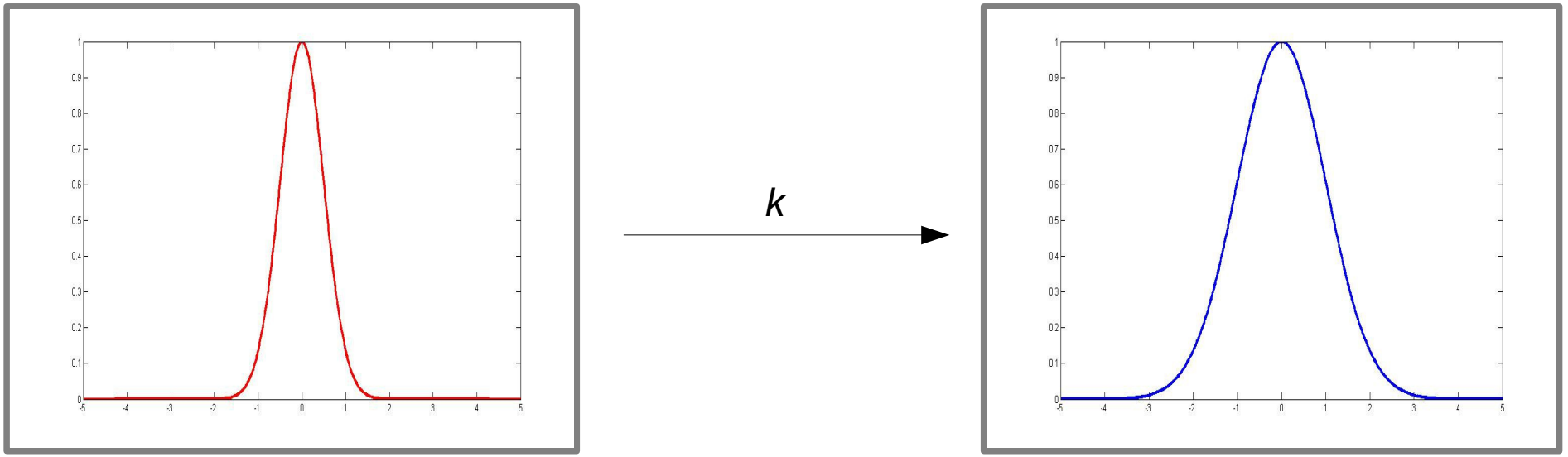
2. LRI images need to be properly prepared: background, re-binning...

BASICS

1. Two images of the same field, at different resolution (wavelength):
- HRI + Seg (Sextractor) + cat
- LRI + RMS

2. LRI images need to be properly prepared

3. Convolution kernel *k* to transform HRI PSF into LRI PSF



$$PSF_LRI = k * PSF_HRI \rightarrow k = F^{-1}\,[F(PSF_LRI) / F(PSF_HRI)]$$

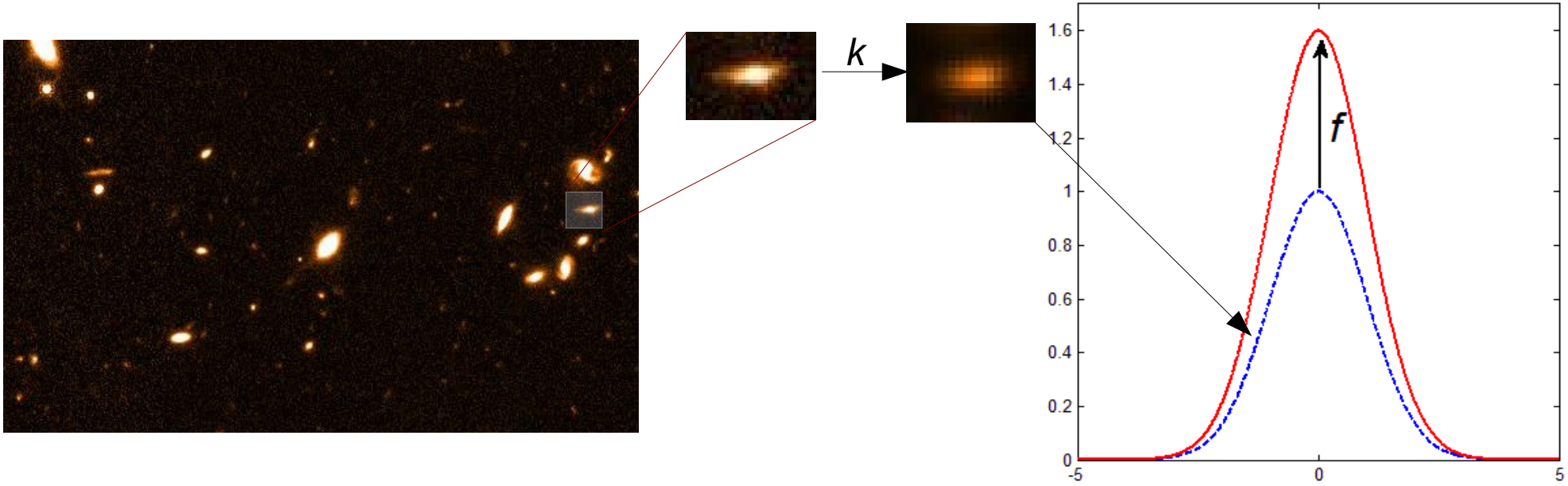
BASICS

1. Two images of the same field, at different resolution (wavelength):
- HRI + Seg (Sextractor) + cat
- LRI + RMS

2. LRI images need to be properly prepared

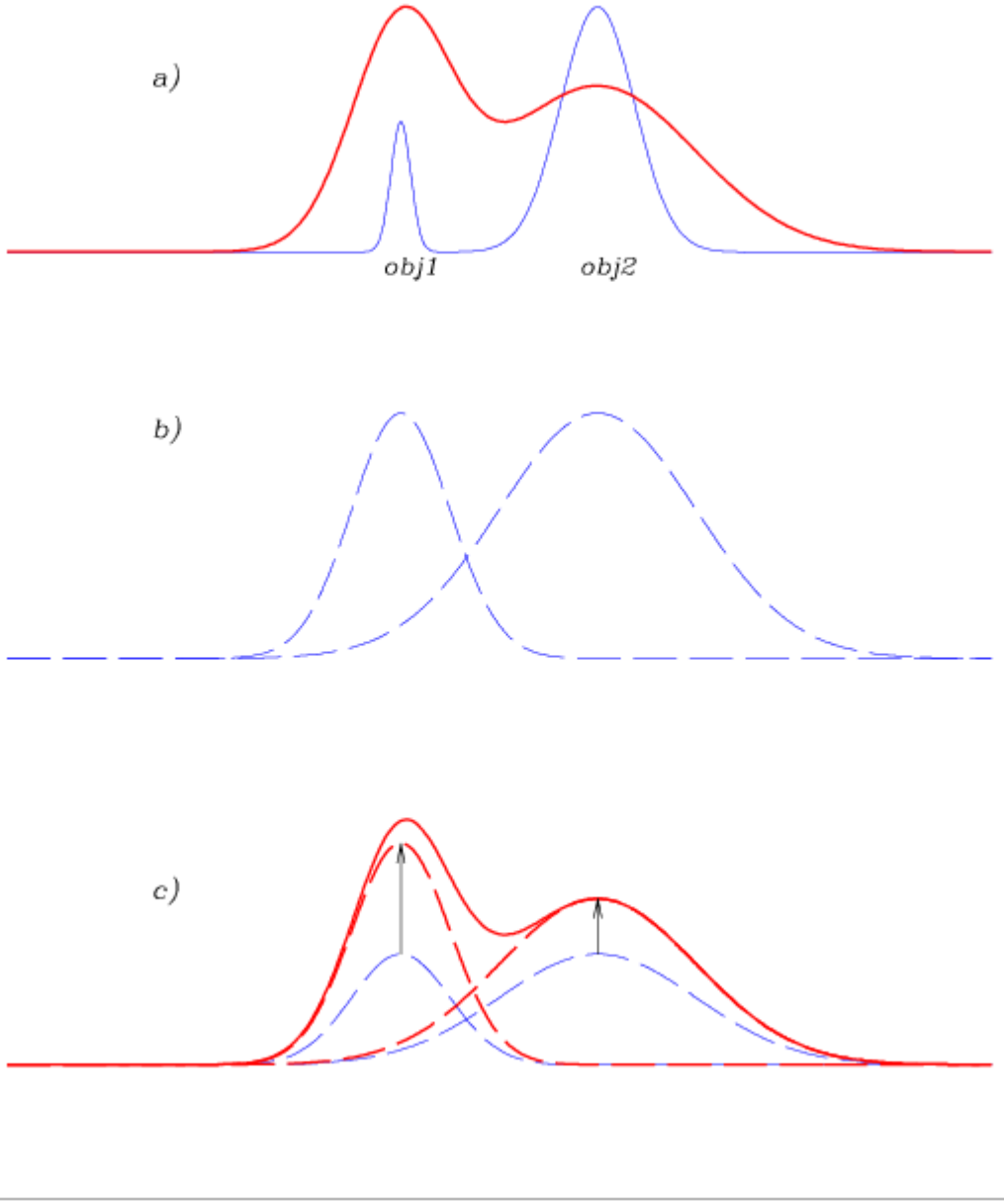
3. Smoothing kernel to transform HRI PSF to LRI PSF

4. Cores of the task:
- Convolve HRI *cutouts* (obtained from the *segmentation map*) with kernel to produce LR normalized models (*templates*)
- Fit with *Chi squared* technique to obtain fluxes in LRI



BASICS

Blending sources:

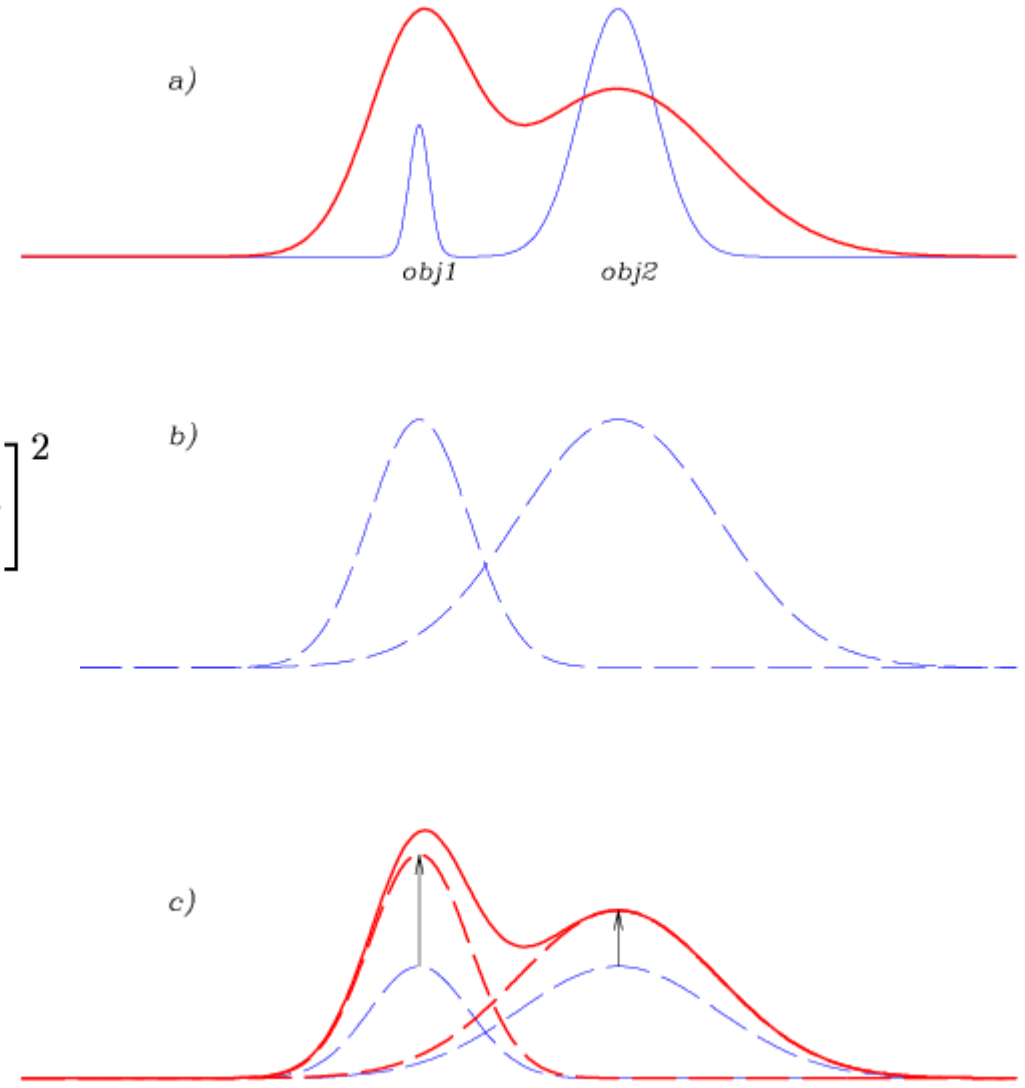


BASICS

Blending sources:

$$\chi^2 = \sum_{m,n} \left[\frac{I(m,n) - B_i - M(m,n)}{\sigma(m,n)} \right]^2$$

$$M(m,n) = \sum_{i=1}^{N_{\text{obj}}} F_i P_i(m,n)$$



$$\chi^2 = \sum_{m,n} \left[\frac{I(m,n) - \Sigma_i F_i P_i(m,n) - B(m,n)}{\sigma(m,n)} \right]^2$$

$$\frac{\partial \chi^2}{\partial F_i} = 0, \quad i = 1, \ldots, N_{sources}$$

$$\frac{\partial \chi^2}{\partial F_i} = -2 \sum_{m,n} \frac{I P_i}{\sigma^2} + 2 F_i \sum_{m,n} \frac{P_i^2}{\sigma^2} + 2 \sum_{j,\, j \neq i} F_j \sum_{m,n} \frac{P_i P_j}{\sigma^2}$$

$$AF = B$$

$$A = \begin{bmatrix} A_{11} & A_{12} & \ldots & A_{1i} \\ A_{21} & A_{22} & \ldots & \ldots \\ \ldots & \ldots & \ldots & \ldots \\ A_{il} & \ldots & \ldots & A_{ii} \end{bmatrix}, \quad A_{ij} = \sum_{m,n} \frac{P_i P_j}{\sigma^2} \qquad B = \begin{bmatrix} B_1 \\ B_2 \\ \ldots \\ B_i \end{bmatrix}, \quad B_i = \sum_{m,n} \frac{I P_i}{\sigma^2}$$

Errors

$$\chi^2 = \chi^2_{best} + \sum_i \frac{\partial \chi^2}{\partial F_i} \Delta F_i + \sum_i \frac{\partial^2 \chi^2}{\partial F_i^2} \Delta F_i^2 + \sum_{i,j} \frac{\partial^2 \chi^2}{\partial F_i \partial F_j} \Delta F_i \Delta F_j + \ldots$$

$$\frac{\partial^2 \chi^2}{\partial F_i^2} = \sum \frac{P_i^2}{\sigma^2} = A_{ii} \qquad \frac{\partial^2 \chi^2}{\partial F_i \partial F_j} = \sum \frac{P_i P_j}{\sigma^2} = A_{ij}$$

$$\Delta\chi^2 = \chi^2 - \chi^2_{best} \quad = \quad 1 \quad = \quad \sum_i A_{ii} \Delta F_i^2 + \sum_{i,j} A_{ij} F_i F_j$$

Errors

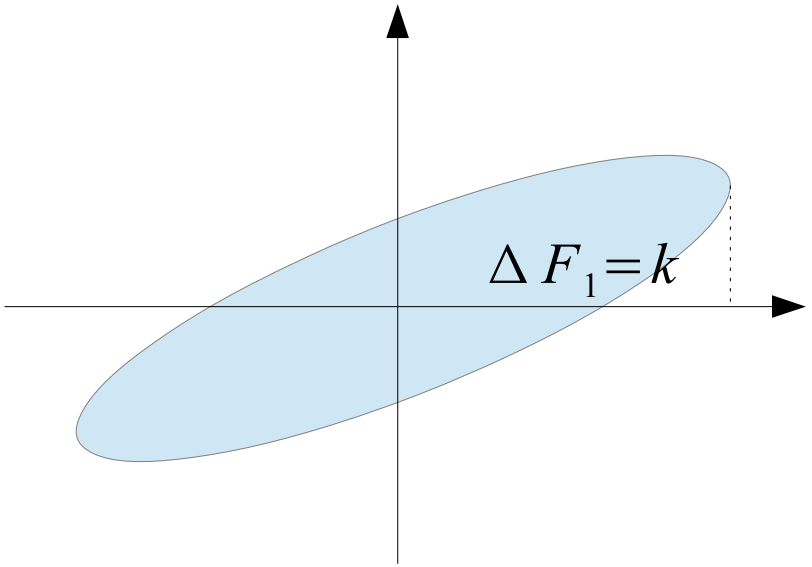
$$\chi^2=\chi^2_{best}+\sum_i \frac{\partial \chi^2}{\partial F_i}\Delta F_i+\sum_i \frac{\partial^2 \chi^2}{\partial F_i^2}\Delta F_i^2+\sum_{i,j} \frac{\partial^2 \chi^2}{\partial F_i \partial F_j}\Delta F_i \Delta F_j+\ldots$$

$$\frac{\partial^2 \chi^2}{\partial F_i^2}=\sum \frac{P_i^2}{\sigma^2}=A_{ii} \qquad \frac{\partial^2 \chi^2}{\partial F_i \partial F_j}=\sum \frac{P_i P_j}{\sigma^2}=A_{ij}$$

$$\Delta\chi^2=\chi^2-\chi^2_{best} \quad = \quad 1 \quad = \quad \sum_i A_{ii}\Delta F_i^2+\sum_{i,j} A_{ij}F_i F_j$$

For two sources:

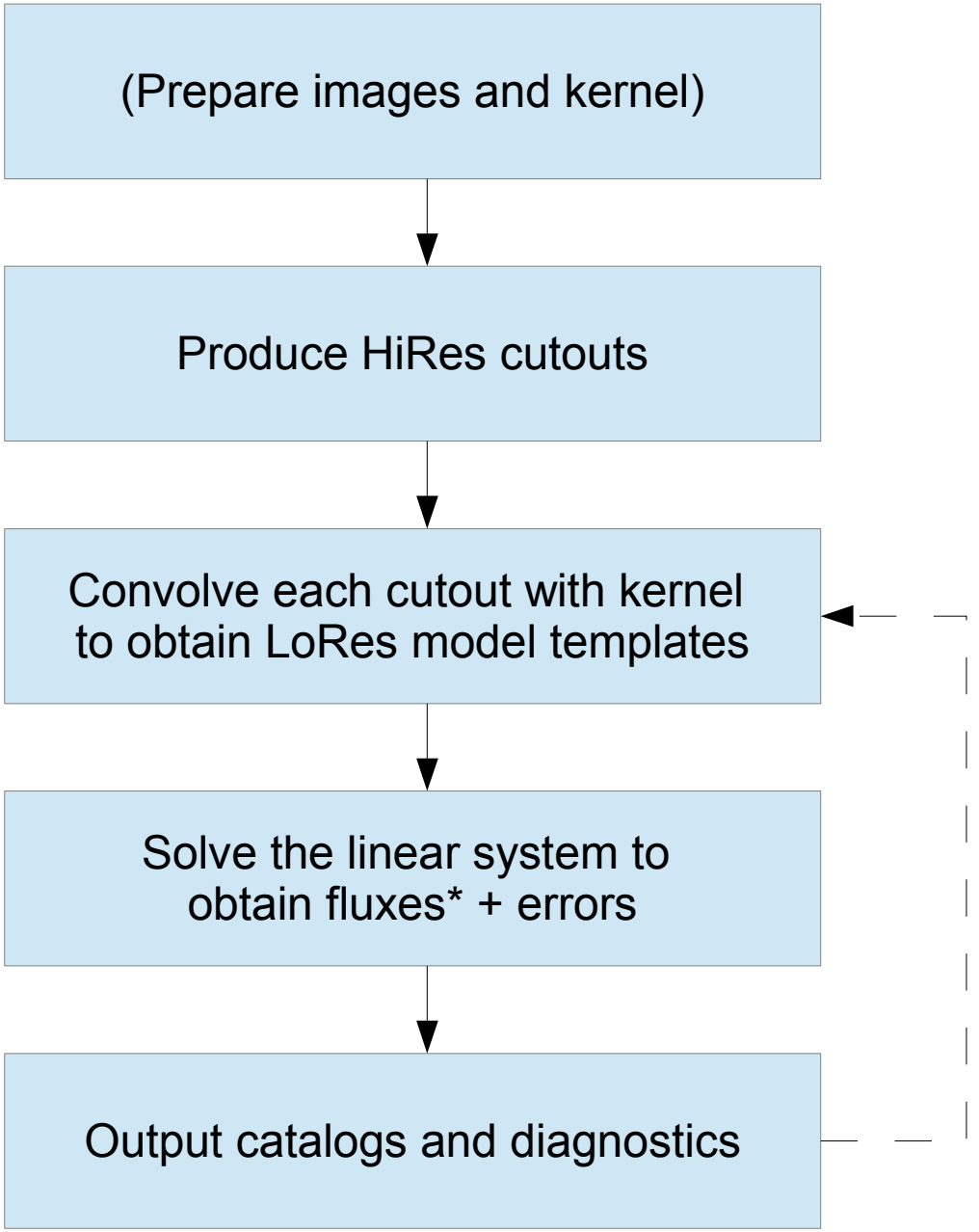
$$\begin{cases} A_{11}\Delta F_1^2+A_{22}\Delta F_2^2+2A_{12}F_1F_2=1 \\ \Delta F_1=k \end{cases}$$



$$\Delta F_1=\sqrt{\frac{A_{22}}{[A]}}=A_{11}^{-1} \qquad \Delta F_2=\sqrt{\frac{A_{11}}{[A]}}=A_{22}^{-1}$$

$$cov(F_1,F_2)=\sqrt{\frac{-A_{12}}{[A]}} \qquad [A]=A_{11}A_{22}-A_{12}^2$$

TFIT and CONVPHOT: FLOWCHART



TFIT vs. CONVPHOT

TFIT:
- C++ core (fitting) + Python envelop
- 10,500 lines .py + .cc (without externals)
- Cell fitting + Dithering
- LU decomposition (direct method) to solve the LS
- Implemented FFT convolution

CONVPHOT:
- C
- 4,200 lines .c (without externals)

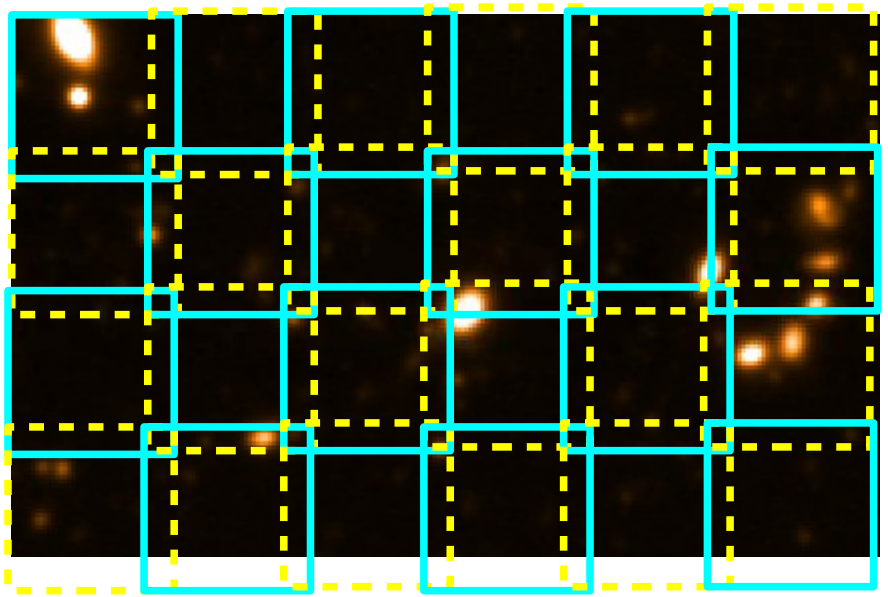
- Single fit on whole image
- Iterative biconjugate gradient method to solve the LS
- No FFT convolution (…)

TFIT vs. CONVPHOT

TFIT:
- C++ core (fitting) + Python envelop
- 10,500 lines .py + .cc (without externals)
- Cell fitting + Dithering
- LU decomposition (direct method) to solve the LS
- Implemented FFT convolution

CONVPHOT:
- C
- 4,200 lines .c (without externals)

- Single fit on whole image
- Iterative biconjugate gradient method to solve the LS
- No FFT convolution (…)

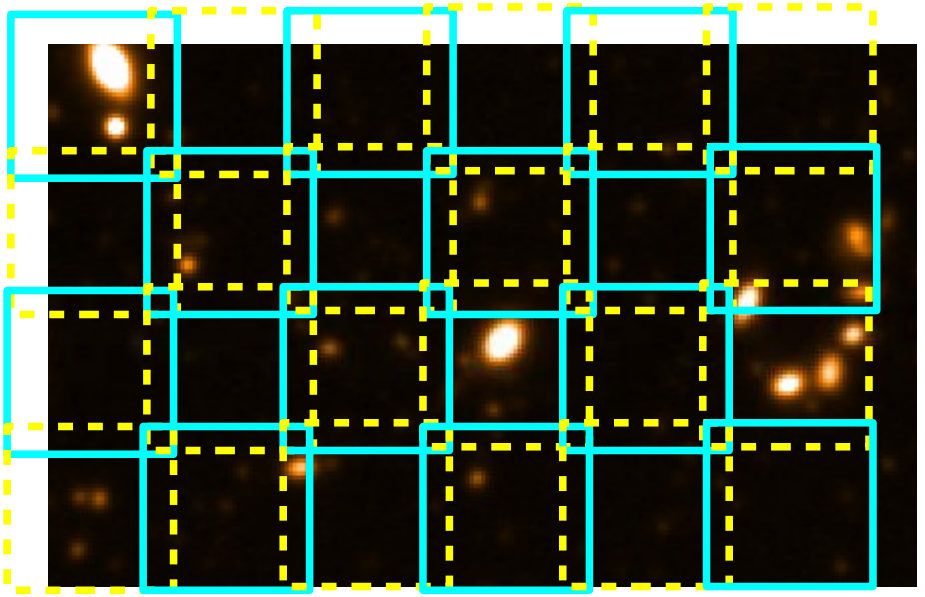


TFIT vs. CONVPHOT

TFIT:
- C++ core (fitting) + Python envelop
- 10,500 lines .py + .cc (without externals)
- Cell fitting + Dithering
- LU decomposition (direct method) to solve the LS
- Implemented FFT convolution

CONVPHOT:
- C
- 4,200 lines .c (without externals)

- Single fit on whole image
- Iterative biconjugate gradient method to solve the LS
- No FFT convolution (…)



TFIT vs. CONVPHOT

TFIT:
- C++ core (fitting) + Python envelop
- 10,500 lines .py + .cc (without externals)
- Cell fitting + Dithering
- LU decomposition (direct method) to solve the LS
- Implemented FFT convolution

CONVPHOT:
- C
- 4,200 lines .c (without externals)

- Single fit on whole image
- Iterative biconjugate gradient method to solve the LS
- No FFT convolution (…)

TFIT vs. CONVPHOT

TFIT:
- C++ core (fitting) + Python envelop
- 10,500 lines .py + .cc (without externals)
- Cell fitting + Dithering
- LU decomposition (direct method) to solve the LS
- Implemented FFT convolution

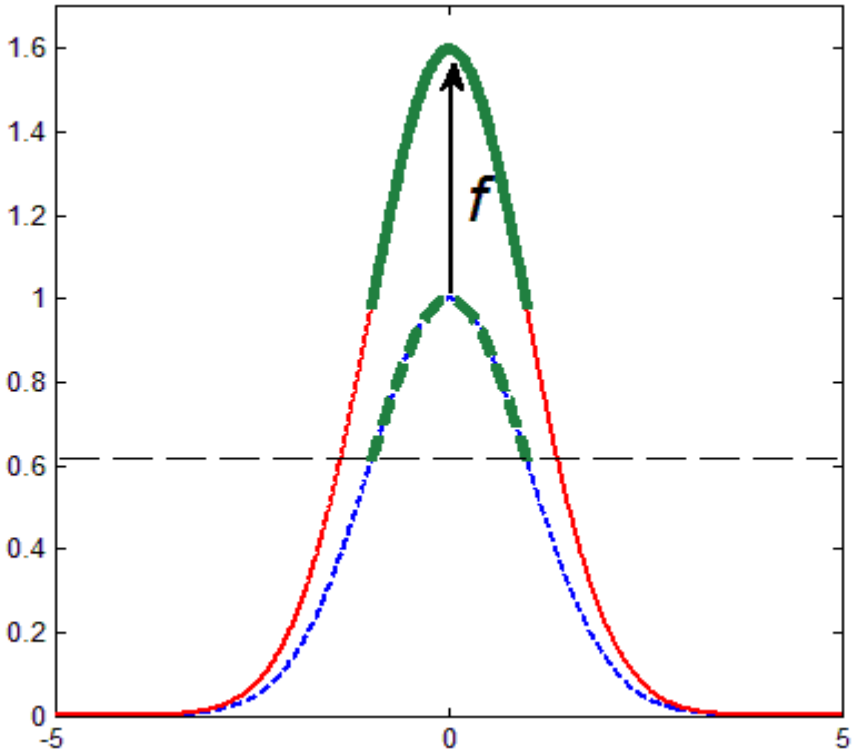
- “Cull”
- Best flux chosen geometrically
- “Dance” recentering + double pass with multi-kernel

CONVPHOT:
- C
- 4,200 lines .c (without externals)

- Single fit on whole image
- Iterative biconjugate gradient method to solve the LS
- No FFT convolution (…)

TFIT vs. CONVPHOT

TFIT:
- C++ core (fitting) + Python envelop
- 10,500 lines .py + .cc (without externals)
- Cell fitting + Dithering
- LU decomposition (direct method) to solve the LS
- Implemented FFT convolution



CONVPHOT:
- C
- 4,200 lines .c (without externals)

- Single fit on whole image
- Iterative biconjugate gradient method to solve the LS
- No FFT convolution (…)

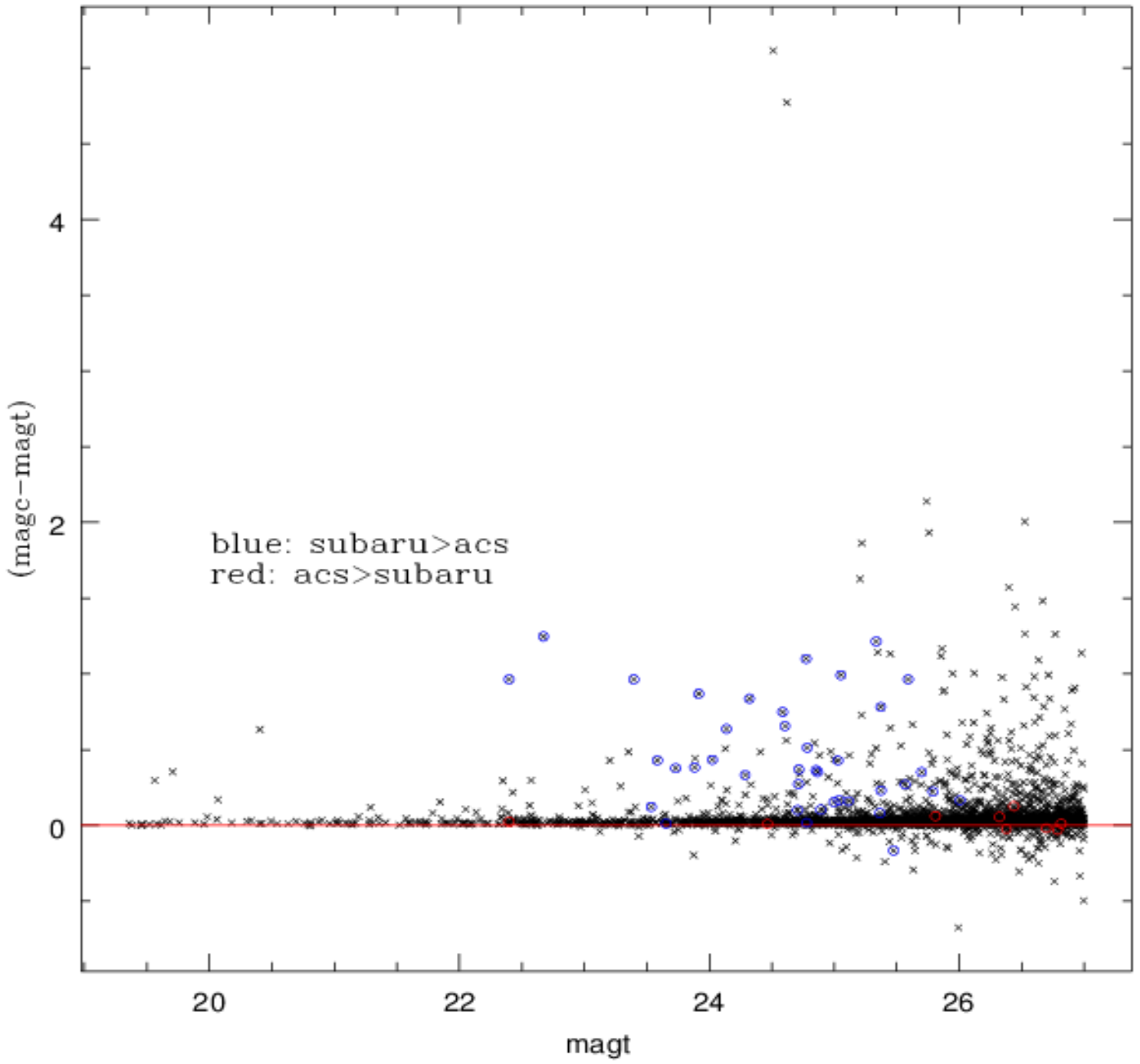
- Allows for threshold for fitting
- Excludes negative sources
- Residual fluxes manually added
- Needs LRI-HRI offset in input

KNOWN ISSUES and/ot ToDos

- Negative fluxes (TFIT / both)
- Morphology and color gradients (both)
- Different fluxes depending on cell size and external sources contributions? Why use cells? (TFIT)
- Which method is best suited to solve the sparse linear system?
- Strong sensitivity on PSF and kernel (both)
- Memory requirement (storing of thumbnails, large matrixes) (CONVPHOT, both)
- HRI blending (both)
- Computing time loss in “non fundamental” issues (TFIT)
- Use WCS info to locate sources (instead of pixel position)? (both)
- Iteratively determine the registration by comparing model collages to data, source by source (both)
- …?

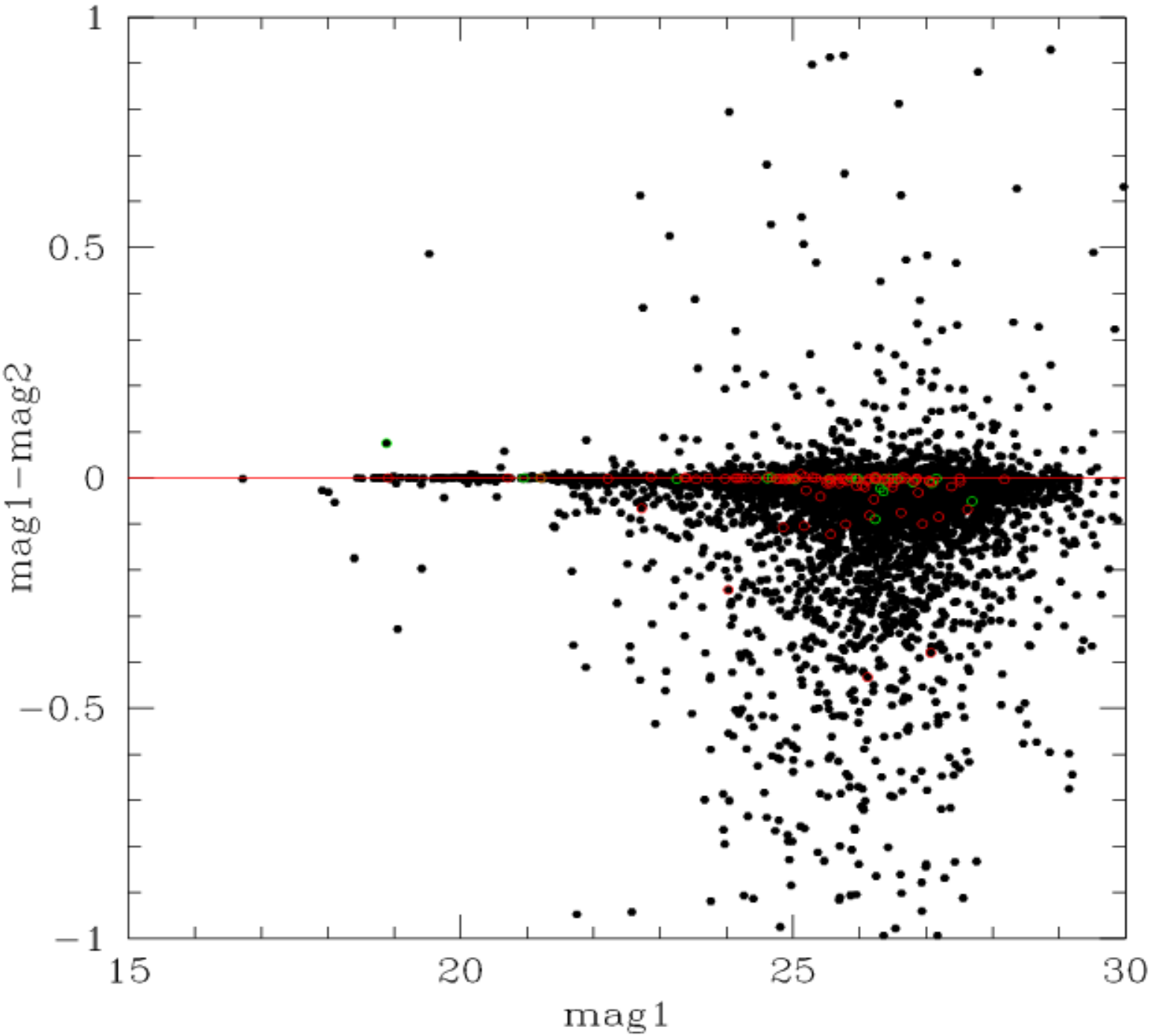
KNOWN ISSUE

- Negative flux
- Morphology a
- Different flux contributions
- Which metho
- Strong sensi
- Memory requ (CONVPHOT,
- HRI blending
- Computing ti
- Use WCS info
- Iteratively de to data, sour
- …?



KNOWN ISS

- Negative f
- Morpholog
- Different f
  contributic
- Which me
- Strong se
- Memory r
  (CONVPHC
- HRI blend
- Computin
- Use WCS
- Iteratively
  to data, sc
- …?

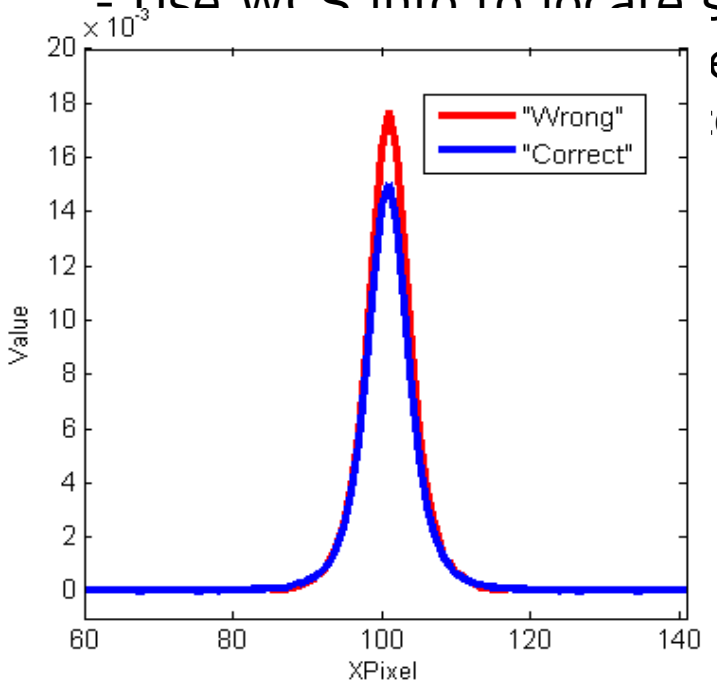


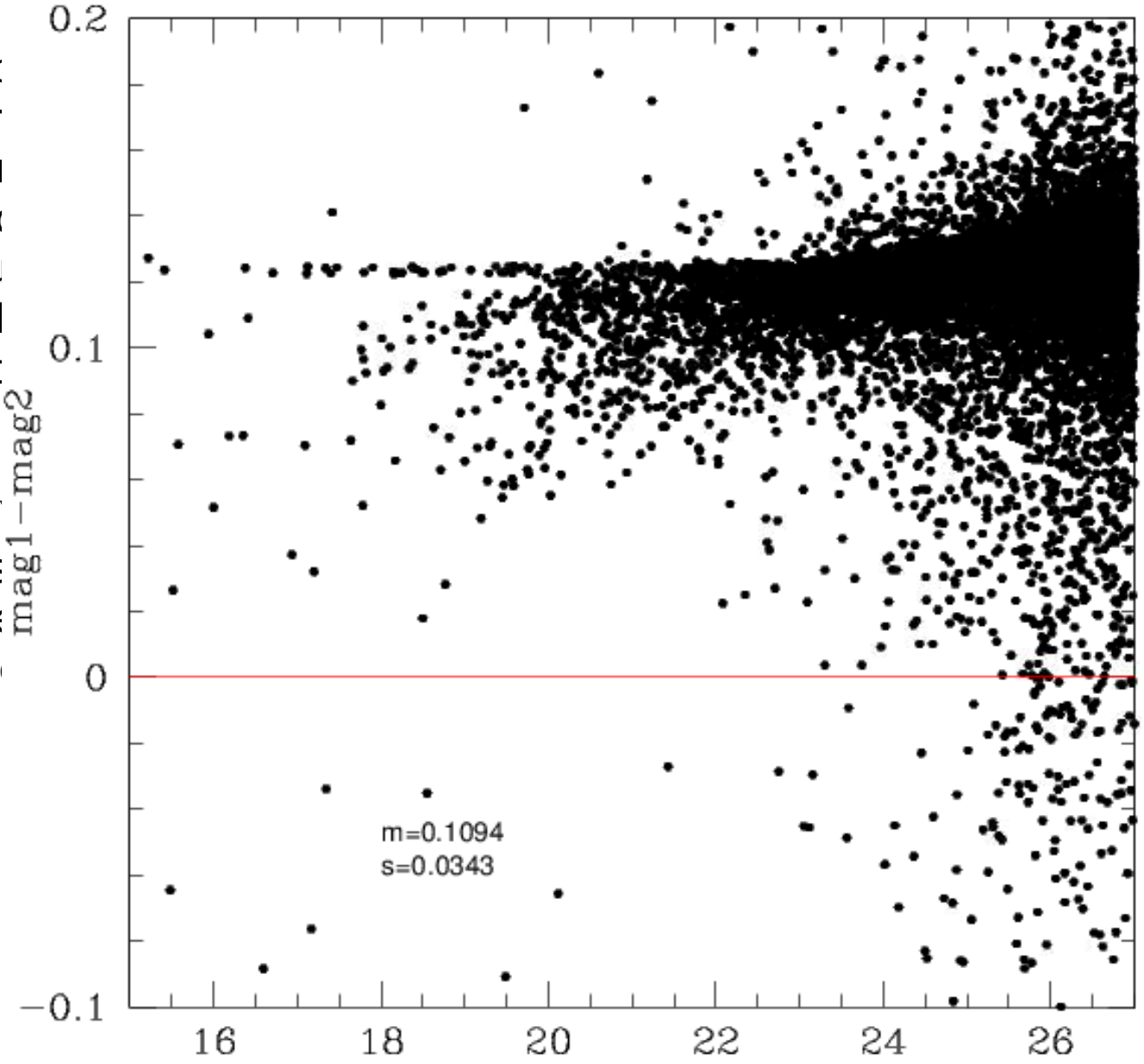
KNOWN ISSUES and/ot ToDos

- Negative fluxes (TFIT / both)
- Morphology and color gradients (both)
- Different fluxes depending on cell size and external sources contributions? Why use cells? (TFIT)
- Which method is best suited to solve the sparse linear system?
- Strong sensitivity on PSF and kernel (both)
- Memory requirement (storing of thumbnails, large matrixes) (CONVPHOT, both)
- HRI blending (both)
- Computing time loss in “non fundamental” issues (TFIT)
- Use WCS info to locate sources (instead of pixel position)? (both)
- Iteratively determine the registration by comparing model collages to data, source by source (both)
- …?

KNOWN ISSUES and/ot ToDos

- Negative fluxes (TFIT / b
- Morphology and color gr
- Different fluxes dependi
  contributions? Why use
- Which method is best su
- Strong sensitivity on PS
- Memory requirement (st
  (CONVPHOT, both)
- HRI blending (both)
- Computing time loss in
- Use WCS info to locate s





KNOWN ISSUES and/ot ToDos

- Negative fluxes (TFIT / both)
- Morphology and color gradients (both)
- Different fluxes depending on cell size and external sources contributions? Why use cells? (TFIT)
- Which method is best suited to solve the sparse linear system?
- Strong sensitivity on PSF and kernel (both)
- Memory requirement (storing of thumbnails, large matrixes) (CONVPHOT, both)
- HRI blending (both)
- Computing time loss in “non fundamental” issues (TFIT)
- Use WCS info to locate sources (instead of pixel position)? (both)
- Iteratively determine the registration by comparing model collages to data, source by source (both)
- …?