Tools for Archives: Image Processing

Richard L. White
Space Telescope Science Institute

HiPACC Summer School, July 2012

Overview

- Image processing
  - Fourier transforms
  - Wavelet transforms & multi-scale processing
  - Compression
  - Deconvolution
  - Etc. …

Fourier Transforms

- Key fact: Fourier transforms make convolutions fast

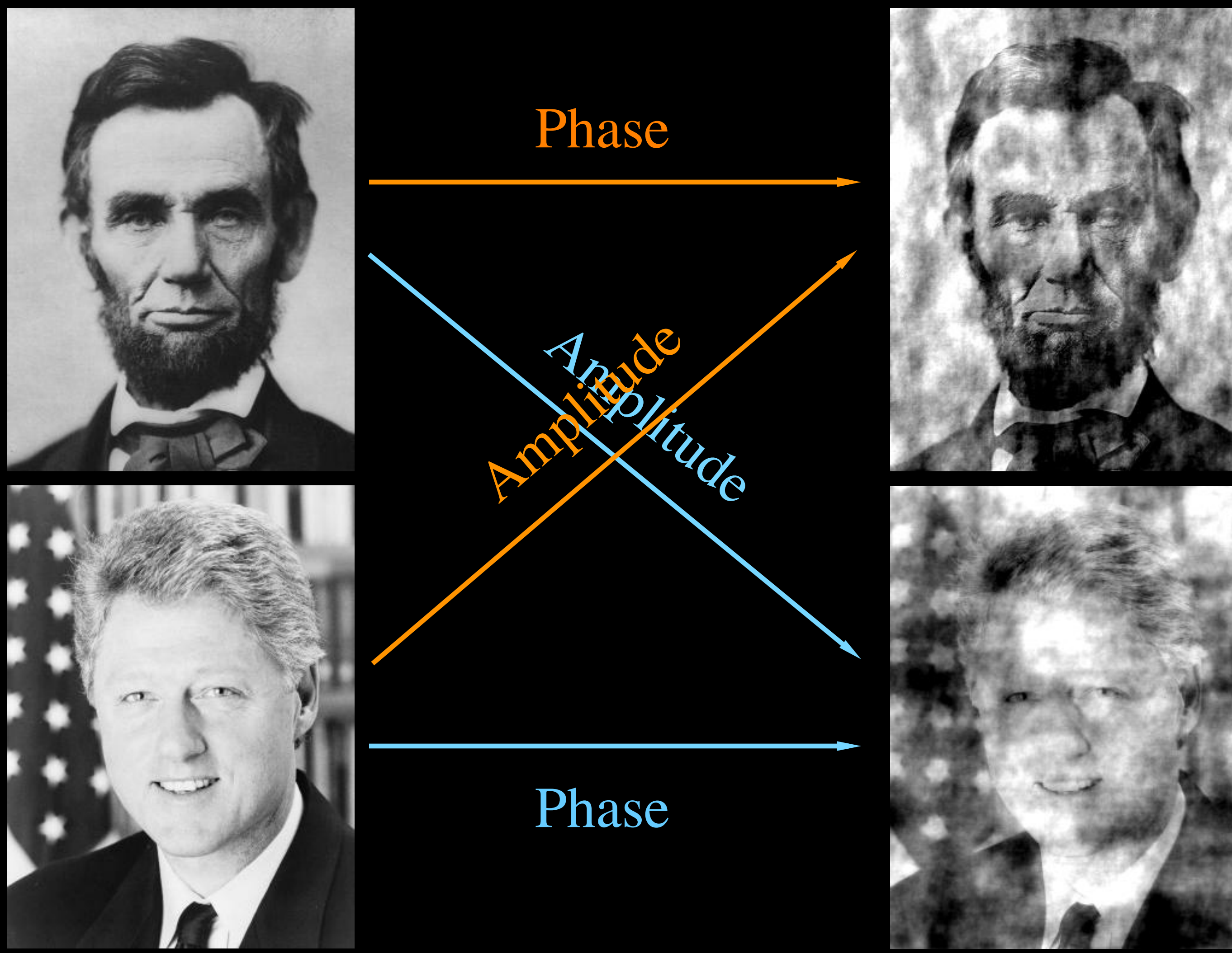
  $$F(X * Y) = F(X) \times F(Y)$$

  Fast FT (FFT) converts $O(N^2)$ operation to $O(N \log N)$
- FTs occur naturally in radio interferometry (which measures FT of image)
  - Look to radio astronomy for clever adaptations, e.g., FFT for unevenly spaced data

Fourier Transforms

- FTs are complex-valued with amplitude and phase
  - Perhaps surprising: phase is more important than amplitude in capturing image information

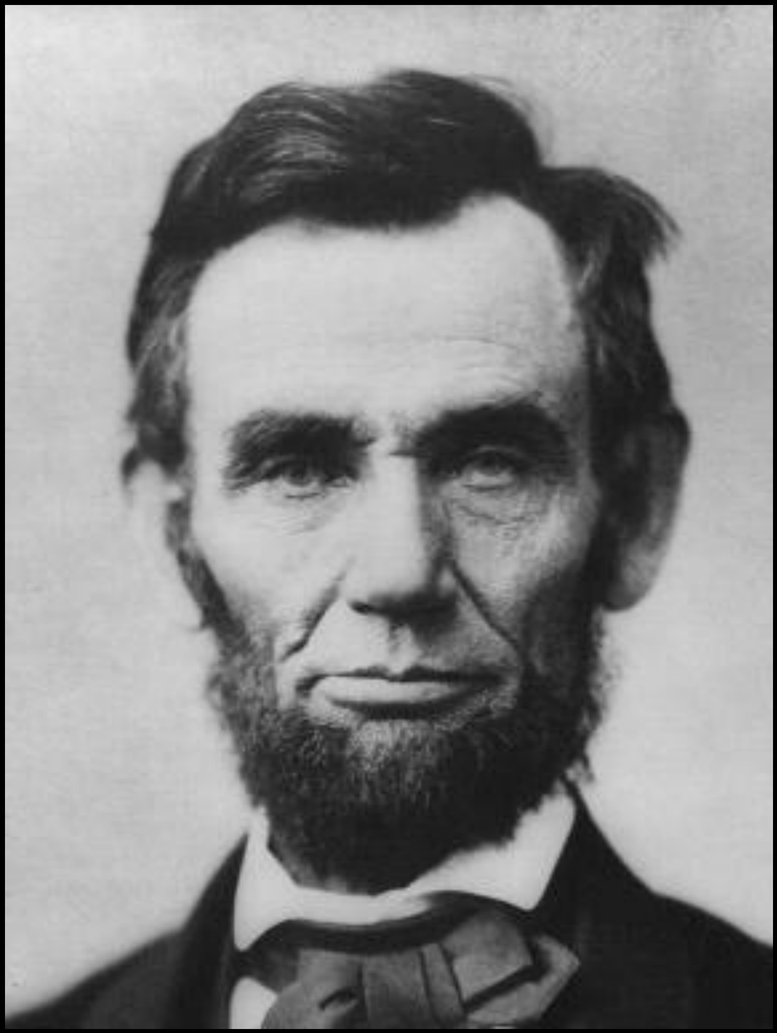
Phase versus amplitude?



Fourier Transforms

- FTs are complex-valued with amplitude and phase
  - Perhaps surprising: phase is more important than amplitude in capturing image information
- FT coefficients are global
  - Changing a single coefficient changes every pixel in the image

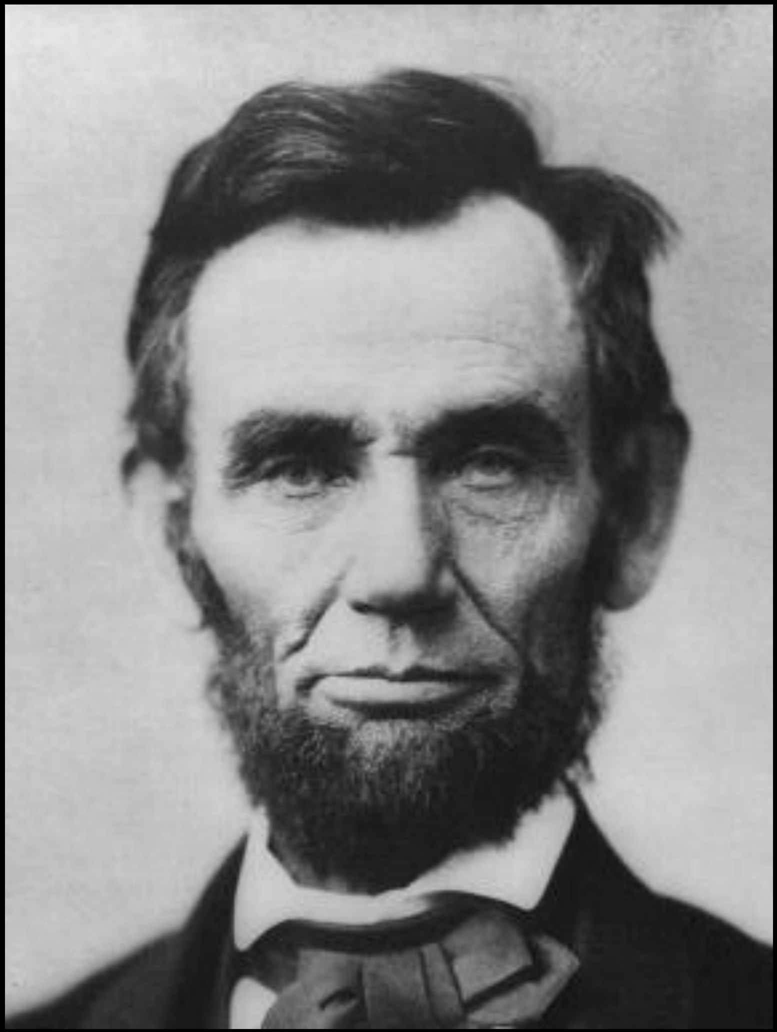
Local changes, global effects

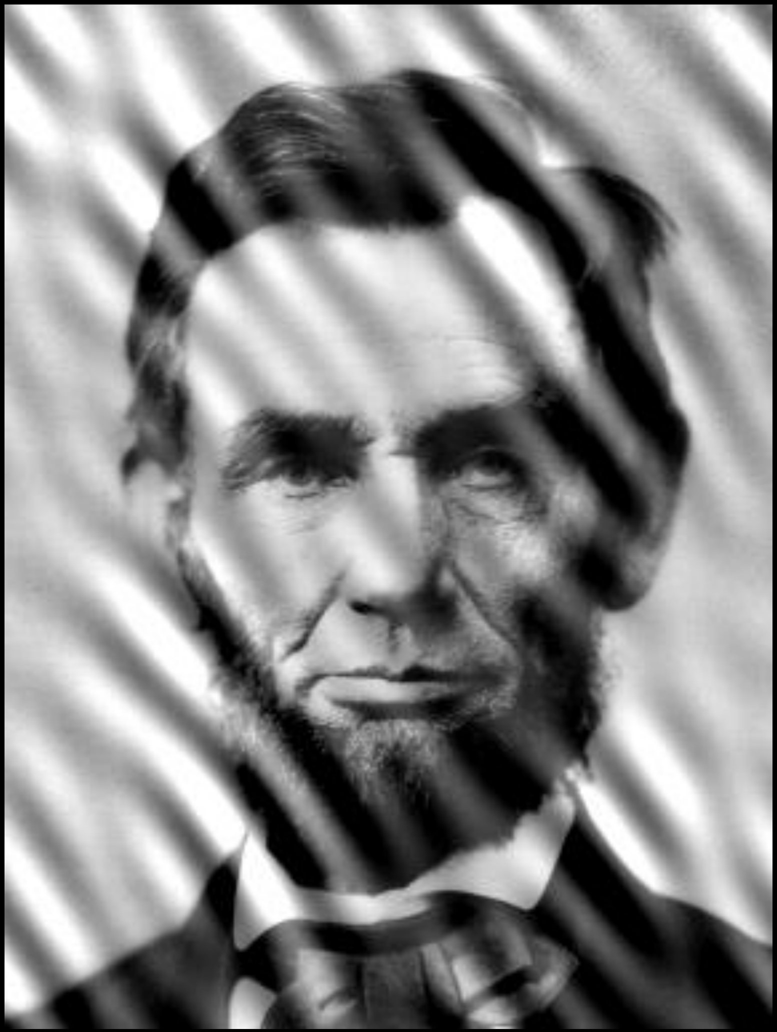


Changed single FT coefficient

Local changes, global effects

Changed block (0.1%) of coefficients





Fourier Transforms

- FTs are complex-valued with amplitude and phase
  - Perhaps surprising: phase is more important than amplitude in capturing image information
- FT coefficients are global
  - Changing a single coefficient changes every pixel in the image
  - Essential for convolution theorem, but awkward when using FTs for analysis

Wavelet Transforms

- Wavelet transforms decompose an image into a sum of localized functions with various spatial scales
  - Fast and easy to compute
- Because the functions are localized, changes in coefficients produce localized changes in the corresponding image
  - Very useful for image analysis

Haar transform

- The Haar transform (Haar 1916) is the simplest wavelet transform
- The algorithm: Given pixels $a_0, a_1, a_2, a_3, \ldots a_{N-1}$:

a_0	a_1	a_2	a_3	a_4	a_5	a_6

1. Compute sums & differences using pixel pairs:

   $s_0 = (a_1+a_0)/2 \qquad d_0 = (a_1-a_0)/2$

   $s_1 = (a_3+a_2)/2 \qquad d_1 = (a_3-a_2)/2$

   …

2. Repeat the paired sums/diffs using $s_0, s_1, s_2, \ldots \mathrm{s}_{N/2-1}$
3. Continue reductions until only one s value remains

Haar transform

- Haar is the simplest (lowest order) example of the class of orthonormal transforms
  - Transformed array is same size as original (# coefficients = # input pixels)
  - Not translation invariant
  - Can be made exactly reversible for integer computations using the lifting scheme
- Extension to 2-D is easy
  - Do one reduction step in X, then one in Y
  - That makes a half-size image; iterate on that

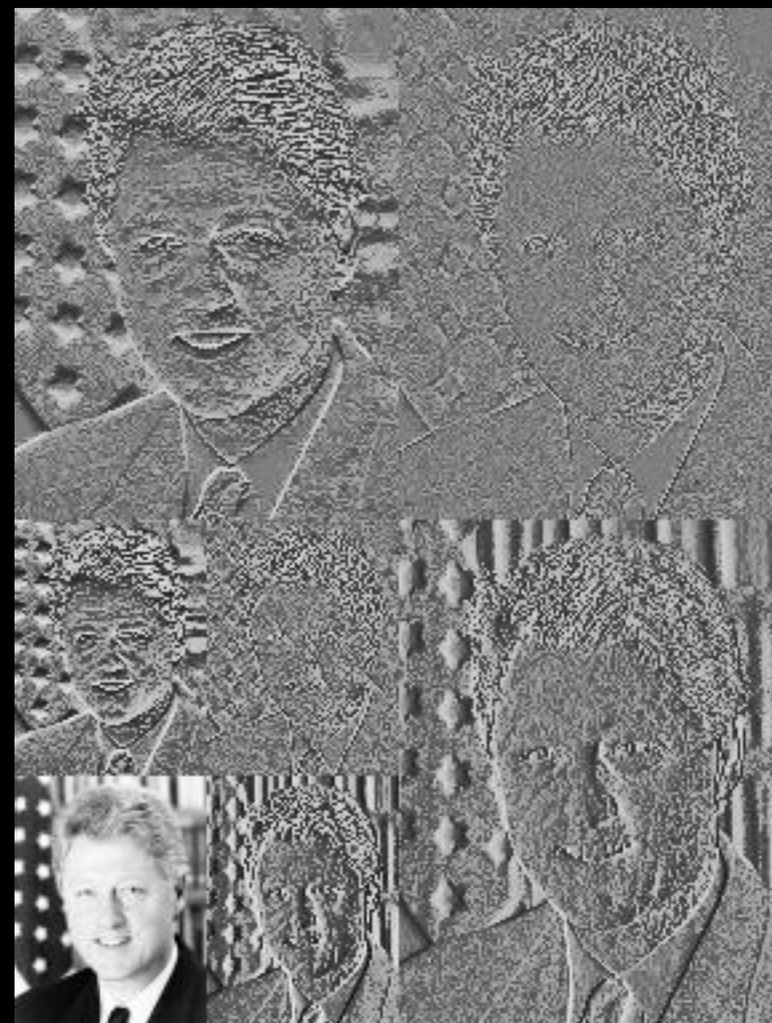
Wavelet example: Haar transform

Wavelet example: Haar transform

Step 1

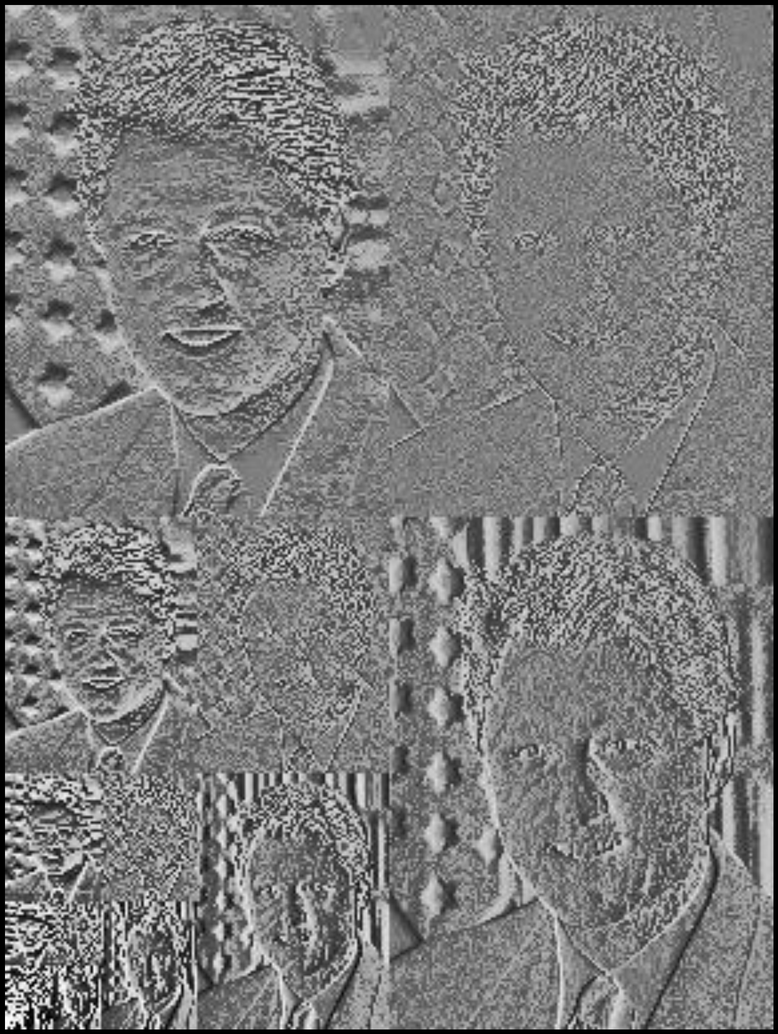
Wavelet example: Haar transform

Step 2



Wavelet example: Haar transform

Final



à trous transform

- The à trous ('with holes') transform (Bijaoui, Starck & Murtagh 1994) is an "undecimated" wavelet transform
  - Translation invariant (unlike orthonormal transforms)
  - Produces a stack of images the size of the original image
  - Simple and fast to compute

à trous algorithm

1. Start with data array and (small) kernel
   - Data

a_0	a_1	a_2	a_3	a_4	a_5	a_6

   - Kernel

k_0	k_1	k_2	=	¼	½	¼

2. Convolve kernel with data, $s^{(1)} = k * a$
   - Smoothed

s_0	s_1	s_2	s_3	s_4	s_5	s_6

3. Subtract smoothed from original, $d^{(1)} = a - s^{(1)}$
   - 1st Difference

d_0	d_1	d_2	d_3	d_4	d_5	d_6

à trous algorithm

4. Expand kernel by adding holes (zeroes):
   - 1st Kernel

k_0	k_1	k_2

   - 2nd Kernel

k_0	0	k_1	0	k_2

5. Convolve new kernel with data, $s^{(2)} = k * s^{(1)}$
   - Smoothed

s_0	s_1	s_2	s_3	s_4	s_5	s_6

6. Subtract smoothed from original, $d^{(2)} = s^{(1)} - s^{(2)}$
   - 2st Difference

d_0	d_1	d_2	d_3	d_4	d_5	d_6

à trous algorithm

- Repeat for *N* levels
  - *N* is arbitrary, typically ~4
  - Kernel doubles in size at each level
- Result is the stack of difference images plus the final smoothed image
  - Inverse is very simple: sum all the images
- Extension to 2-D is obvious: smooth in x, y
- Direct convolution is fast: number of non-zero coefficients is small, separable kernel

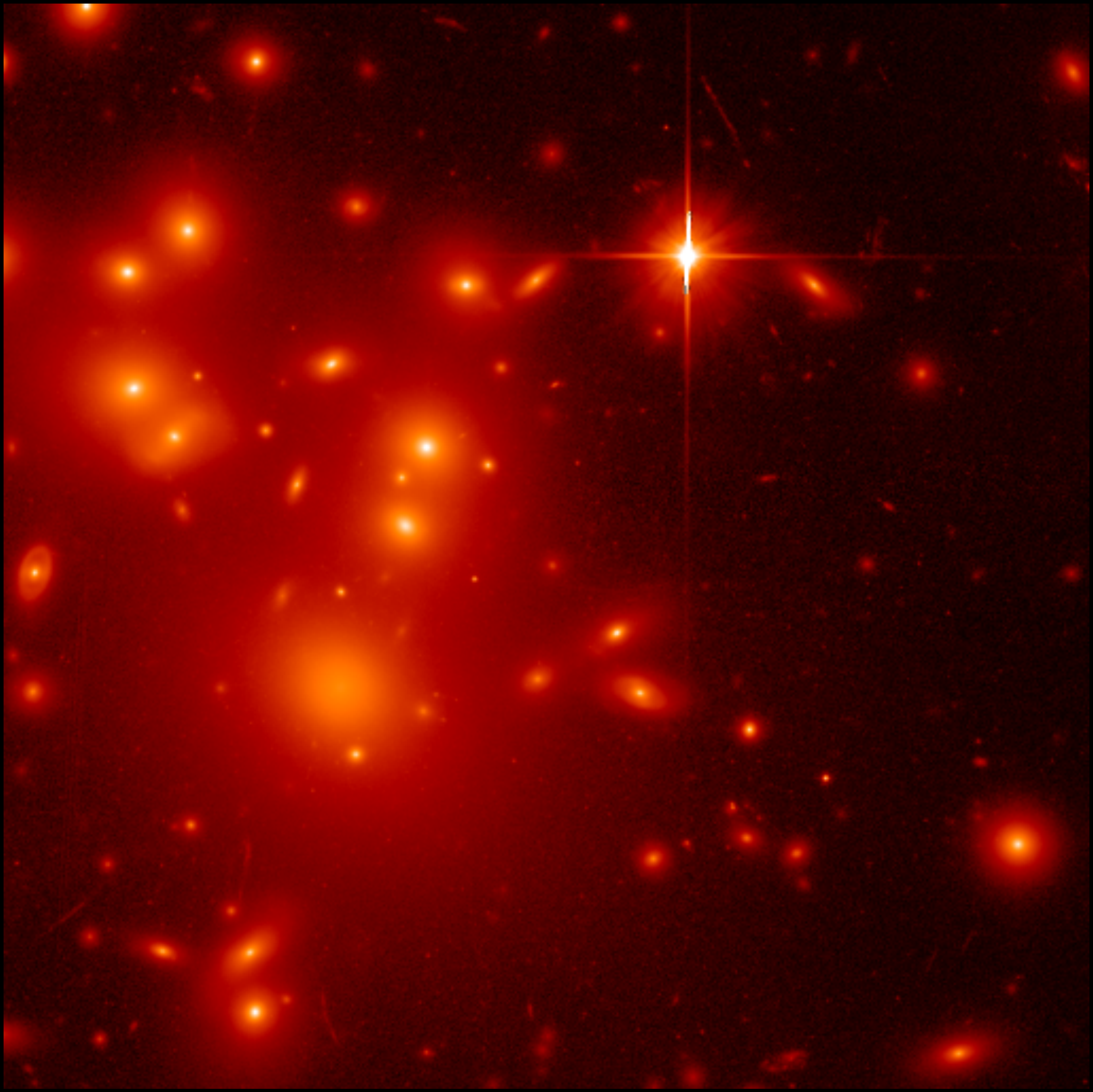
Wavelet example: à trous transform

4 levels

Multi-scale Data Processing

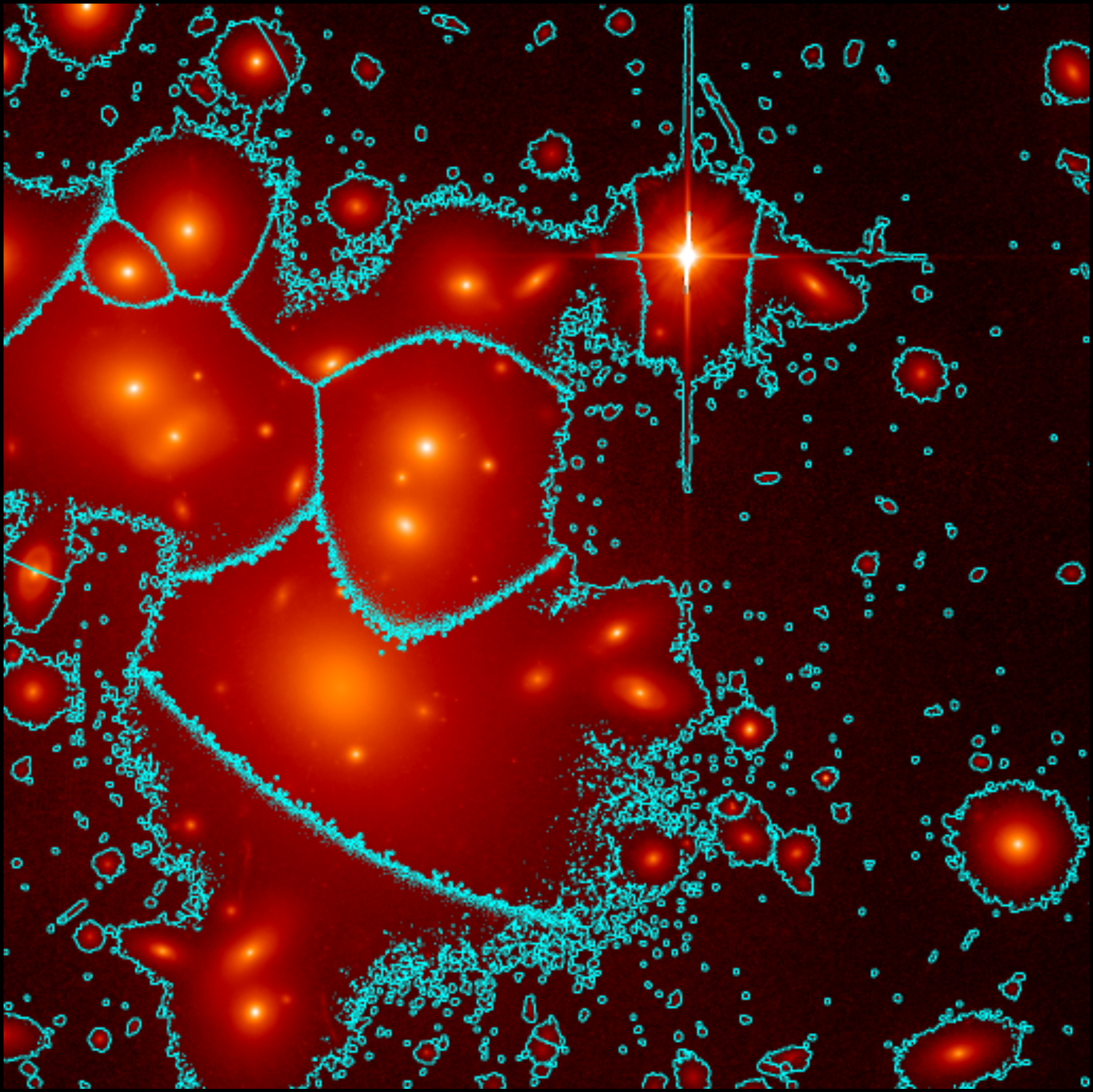
- Translation-invariant wavelet transforms are well-suited for image processing
- The conceptual simplicity of the à trous transform (smooth, subtract, repeat) makes it easily modified for custom applications
  - Example: multi-scale source detection

Galaxy cluster
Abell 1689
HST/ACS



Galaxy cluster
Abell 1689
HST/ACS

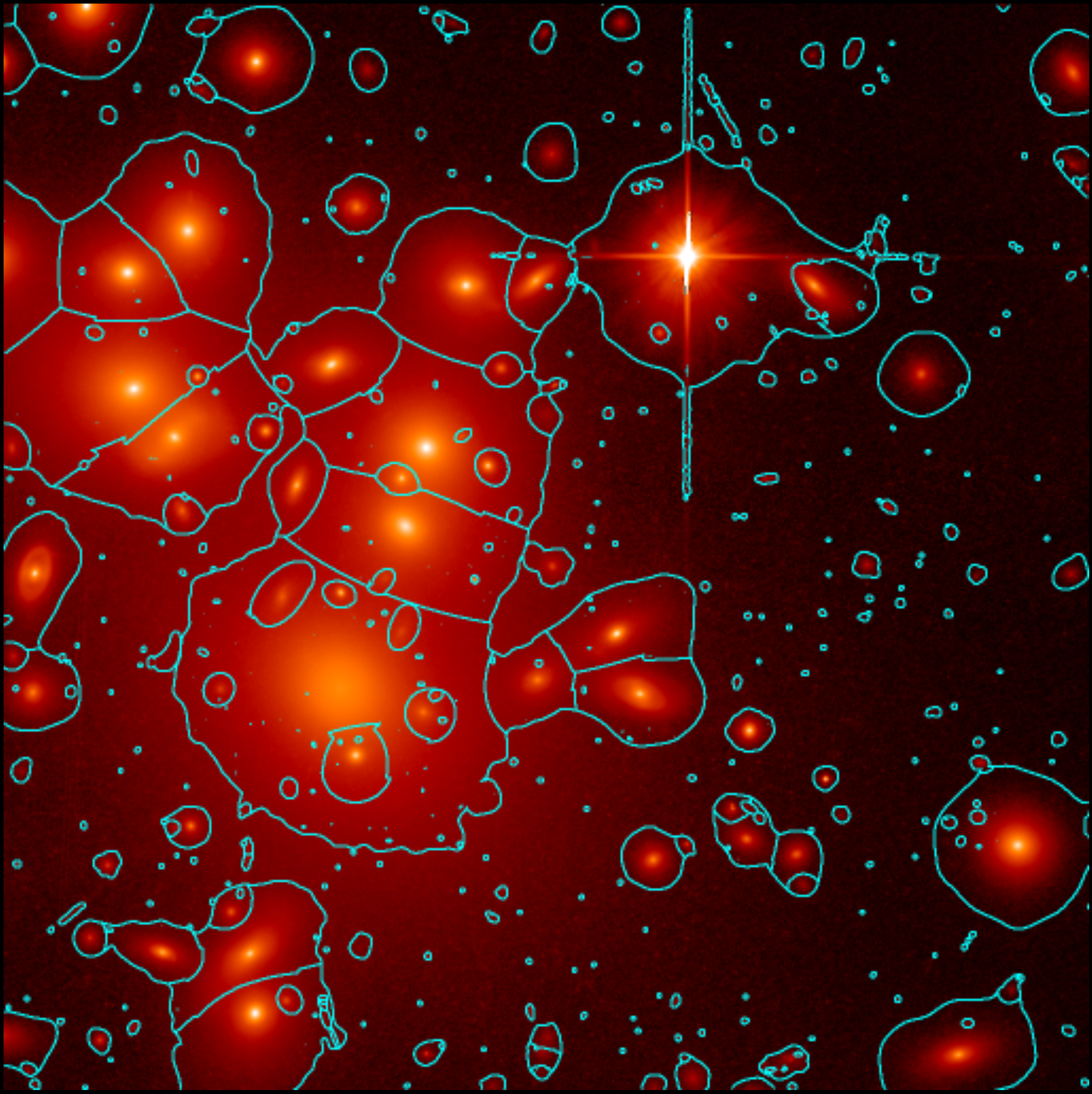
SExtractor
segmentation
map



7/2012

Galaxy cluster
Abell 1689
HST/ACS

Multi-scale
segmentation
using à trous
transform



Multi-scale source detection

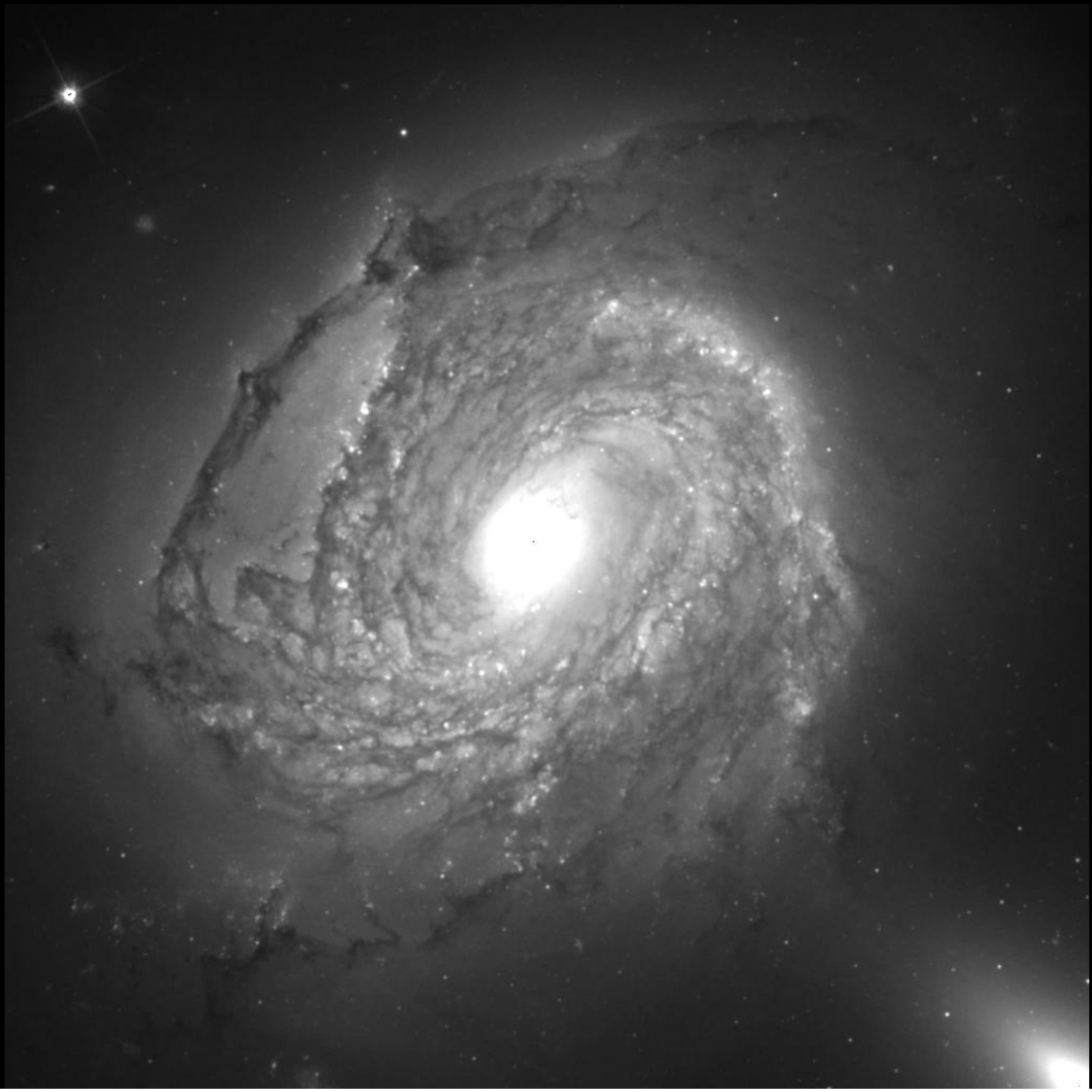
- Uses modified à trous transform that iterates to remove sources detected in difference image from the smoothed image
- Simple thresholding to define source islands from each difference image
- Islands at larger scales can create new sources or extend existing islands
  - Multiple overlaps: simple rule based on sizes and shapes of existing islands

Image Compression

- Orthonormal wavelet transforms keep minimal information needed to describe image
  - Ideal for image compression (lossless or lossy)
  - Approximating an image using wavelet coefficients is far more accurate than other schemes

NGC 4911: Hubble Legacy Archive
ACS/WFC F606W 1024x1024 (51”x51”)

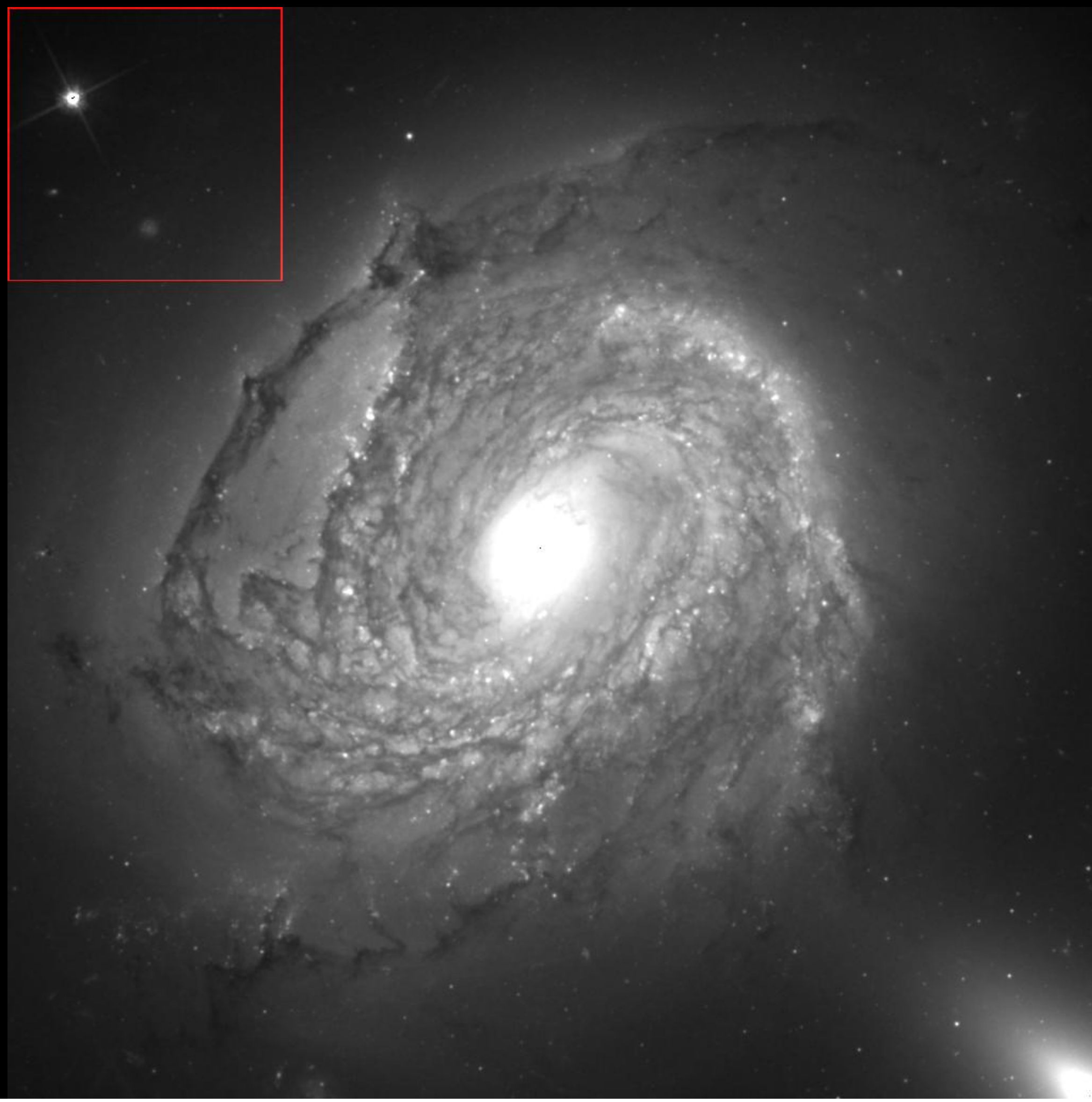
H-compress (Haar transform) 1.50 bits/pixel



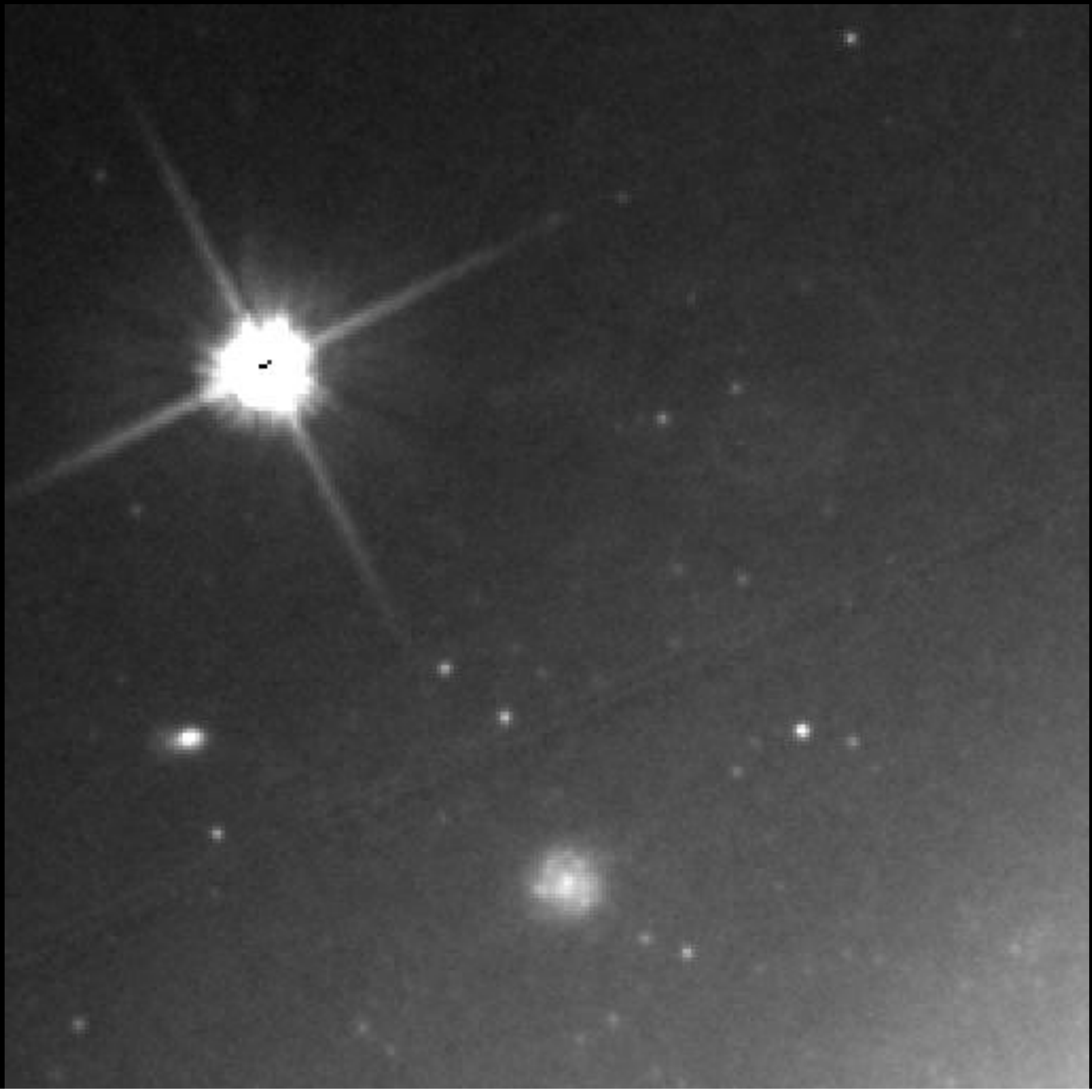
H-compress 0.899 bits/pixel

H-compress 0.515 bits/pixel

NGC 4911: Hubble Legacy Archive
ACS/WFC F606W 1024x1024 (51”x51”)



NGC 4911: Hubble Legacy Archive
ACS/WFC F606W 256x256 (12.8”x12.8”)

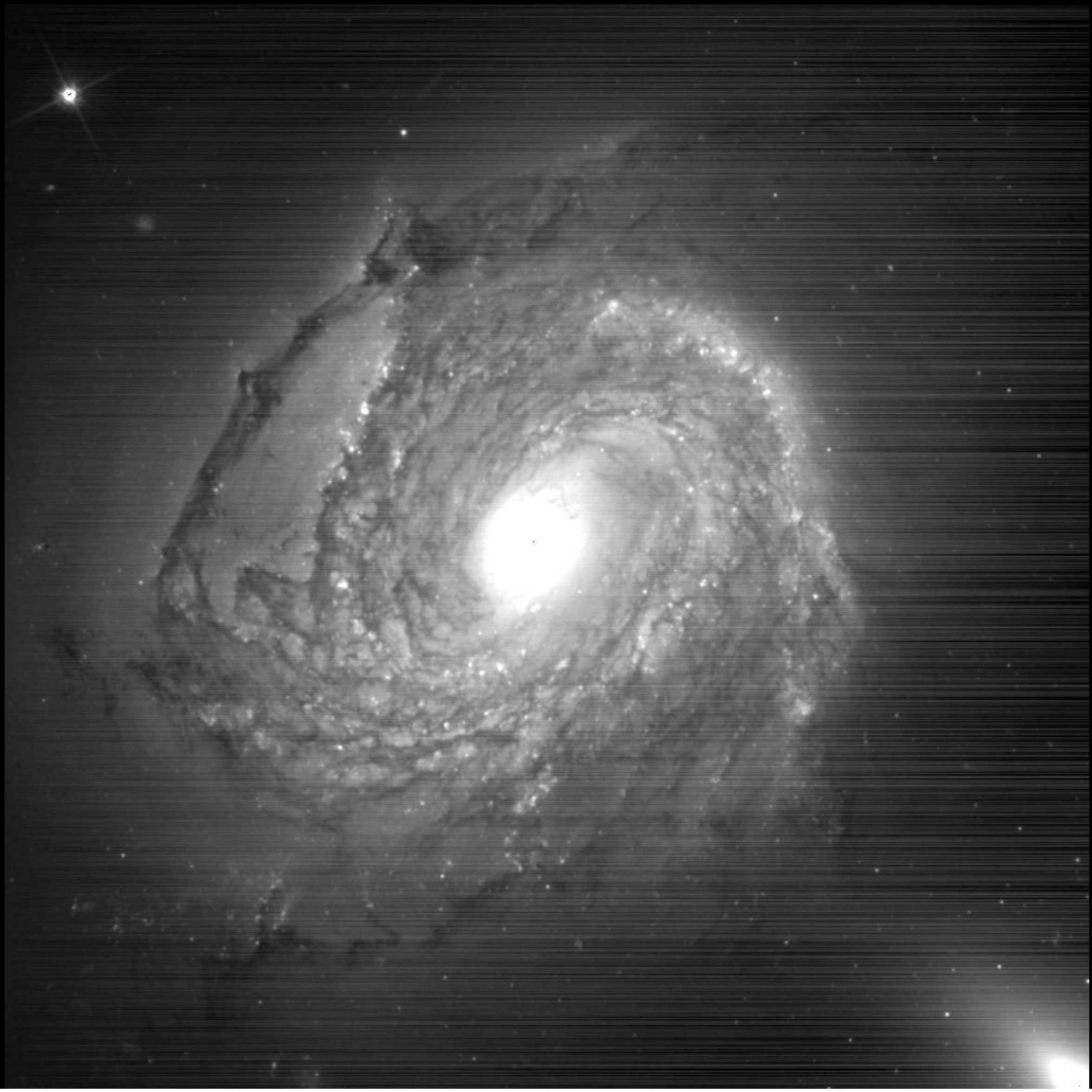


H-compress 1.50 bits/pixel

H-compress 0.899 bits/pixel

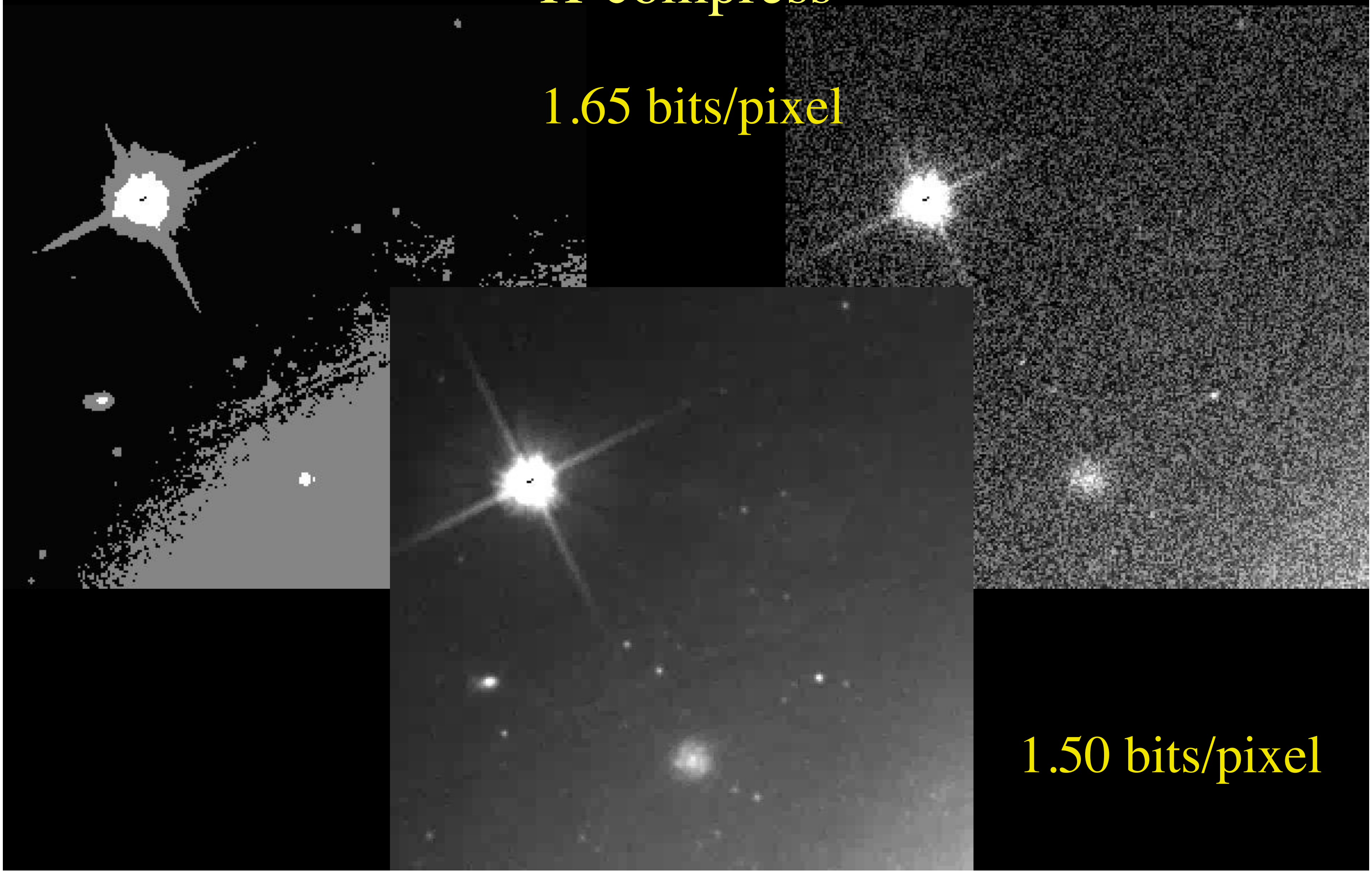
H-compress 0.515 bits/pixel

Don't try this without the wavelet transform!
Results of row-by-row difference compression



Don't try this without the wavelet transform!
Results of row-by-row difference compression

Simple quantization, subtractive dither, H-compress



1.65 bits/pixel

1.50 bits/pixel

Image Compression

- Image and data compression have obvious applications
  - Reduce storage
  - Reduce transmission bandwidth (esp. for space missions)
- There are less obvious applications too
  - Speed I/O bound processes (it can be faster to read and uncompress)
  - Reduce memory bandwidth (e.g., for GPUs)

Deconvolution & Denoising

- A classic use of wavelets is denoising images: filter out noise while leaving significant structures
  - This is what happens in image compression too: noise is incompressible, so discard it
- Denoising is a helpful addition in deconvolution algorithms too

Deconvolution in a Nutshell

- Images are blurred by a point-spread function
  - Spatially invariant PSF -> blurring is convolution with PSF (but PSF may vary too)
- Many algorithms exist that attempt to deconvolve data & recover unblurred image
  - E.g., Richardson-Lucy iteration for data with Poisson noise

Iterative Deconvolution Algorithm

1. Start with initial guess for image (e.g., flat)
2. Convolve model image with PSF to create simulated data
3. Compute difference between simulated data and observed data
4. Use differences to adjust model image
   – This step is specific to the algorithm being used
5. Repeat steps 2–4 until desired convergence

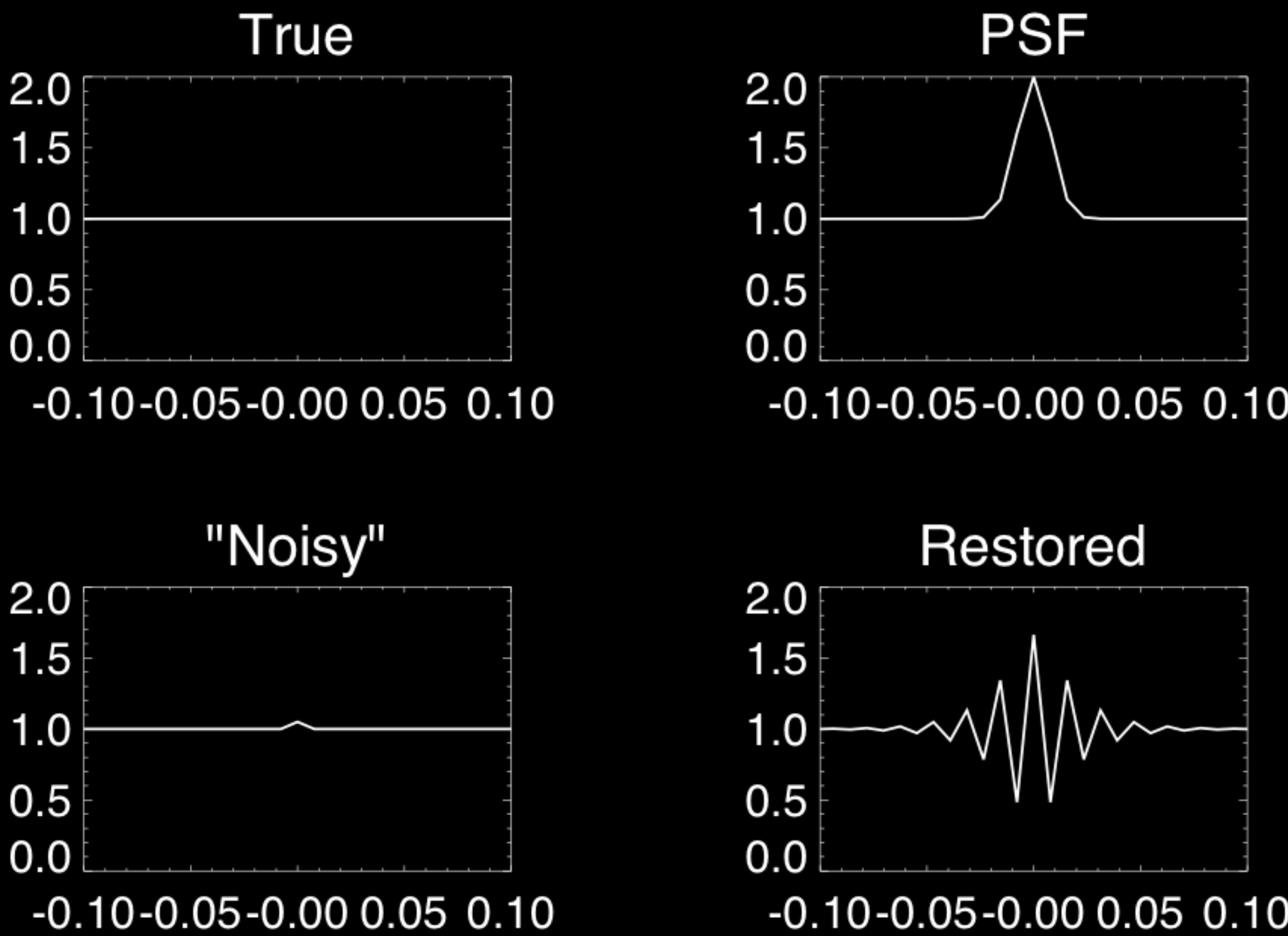
Iterative Deconvolution Algorithm with Denoising

1. Start with initial guess for image (e.g., flat)
2. Convolve model image with PSF to create simulated data
3. Compute difference between simulated data and observed data

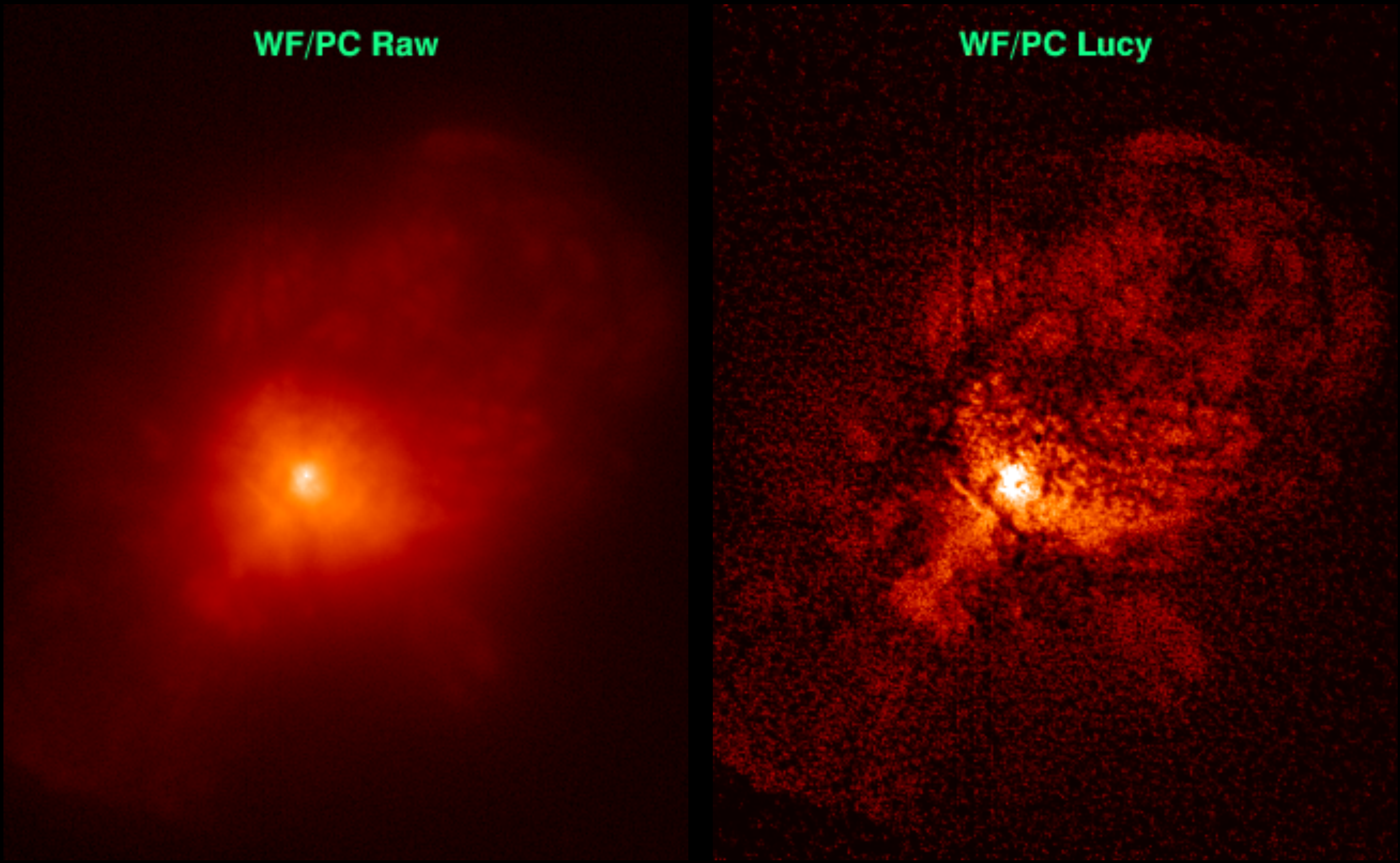
3.5 Denoise differences with wavelet filter (Starck & Murtagh 1994)

4. Use differences to adjust model image
   - This step is specific to the algorithm being used
5. Repeat steps 2–4 until desired convergence

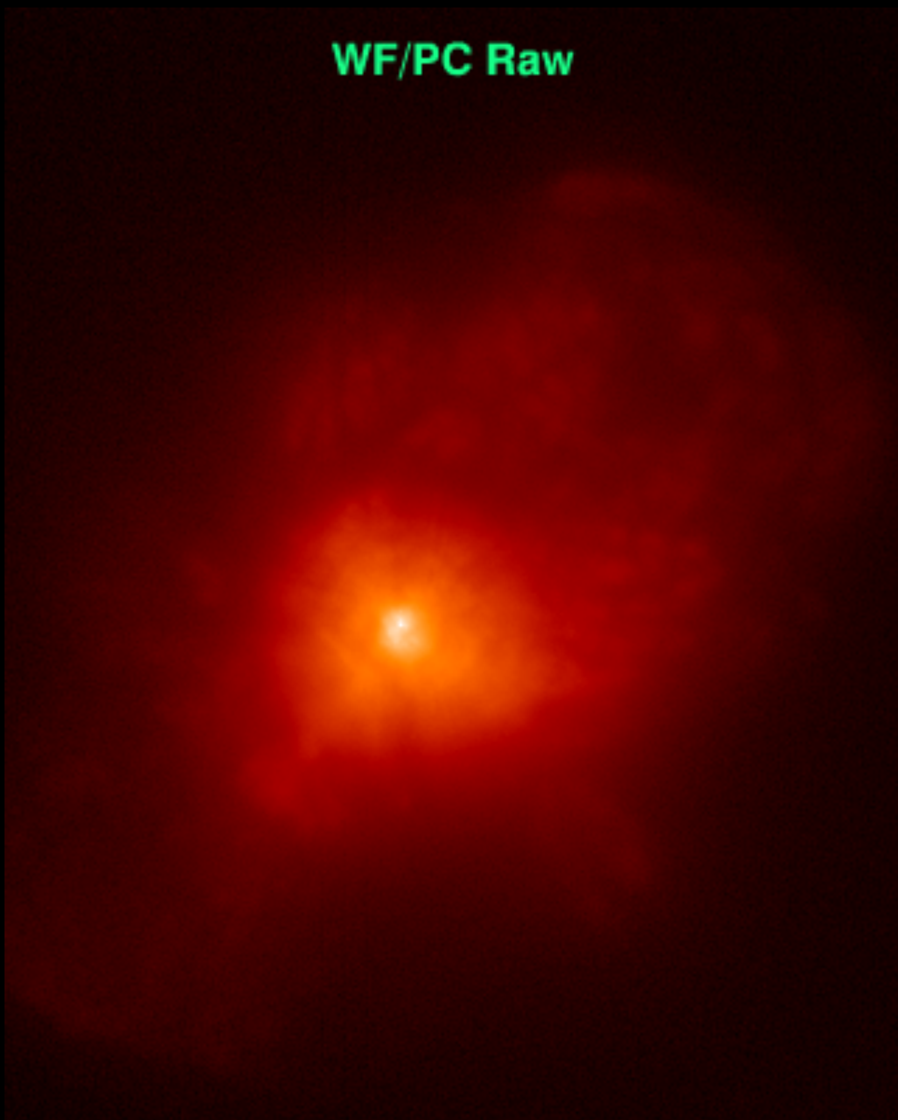
Deconvolution Amplifies Noise

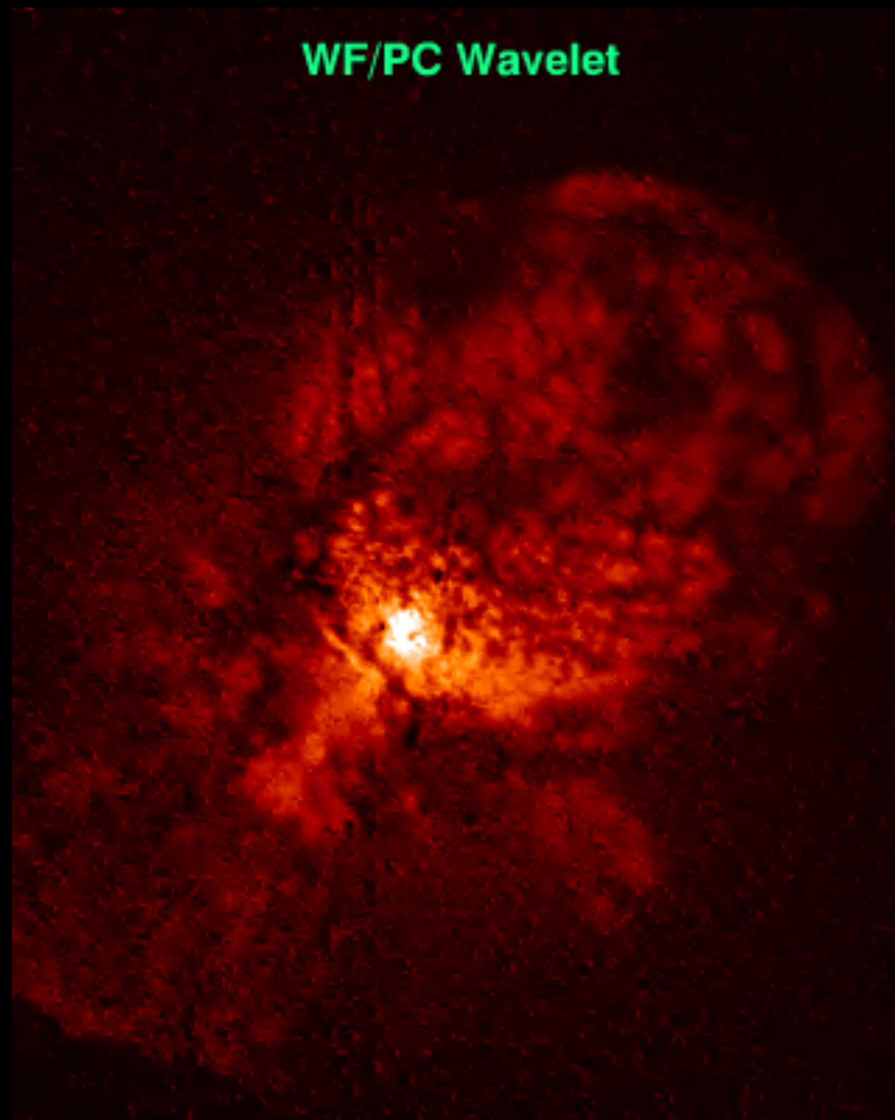


Deconvolution Eta Carina

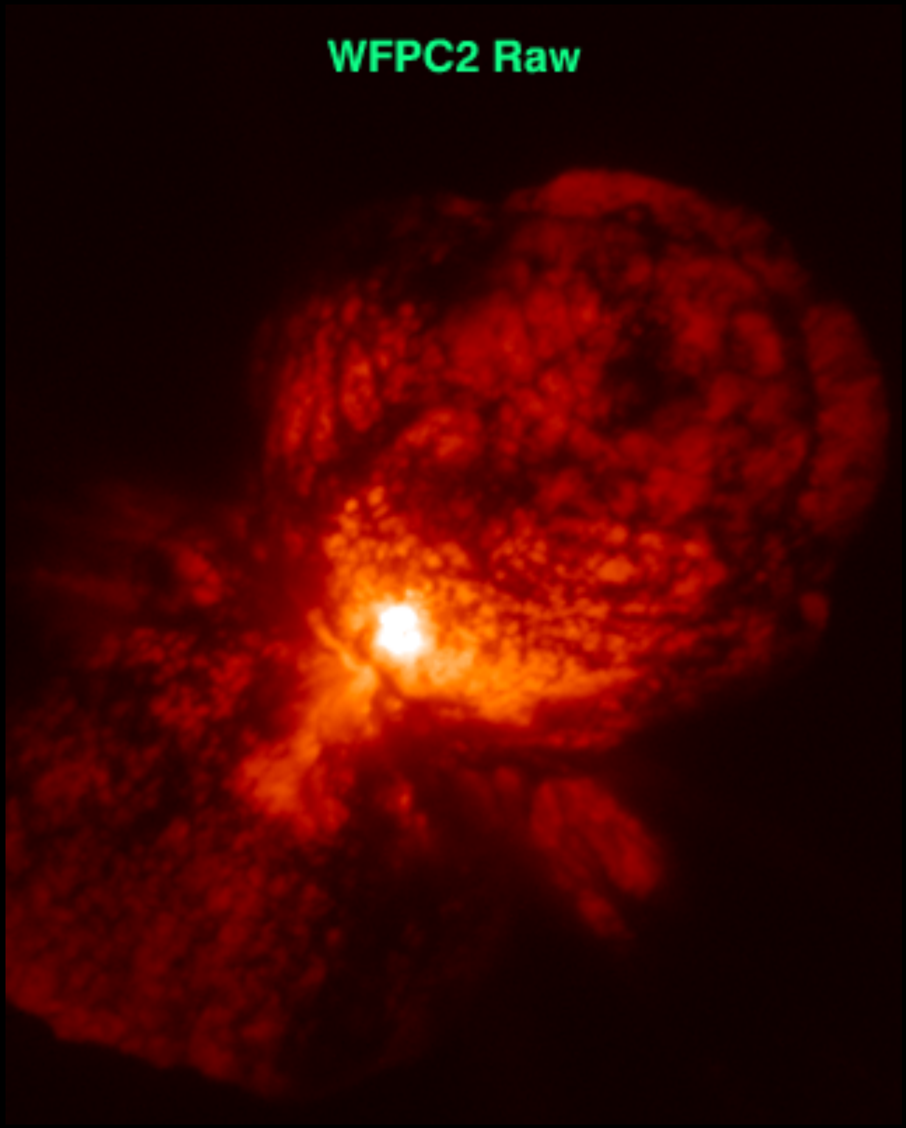


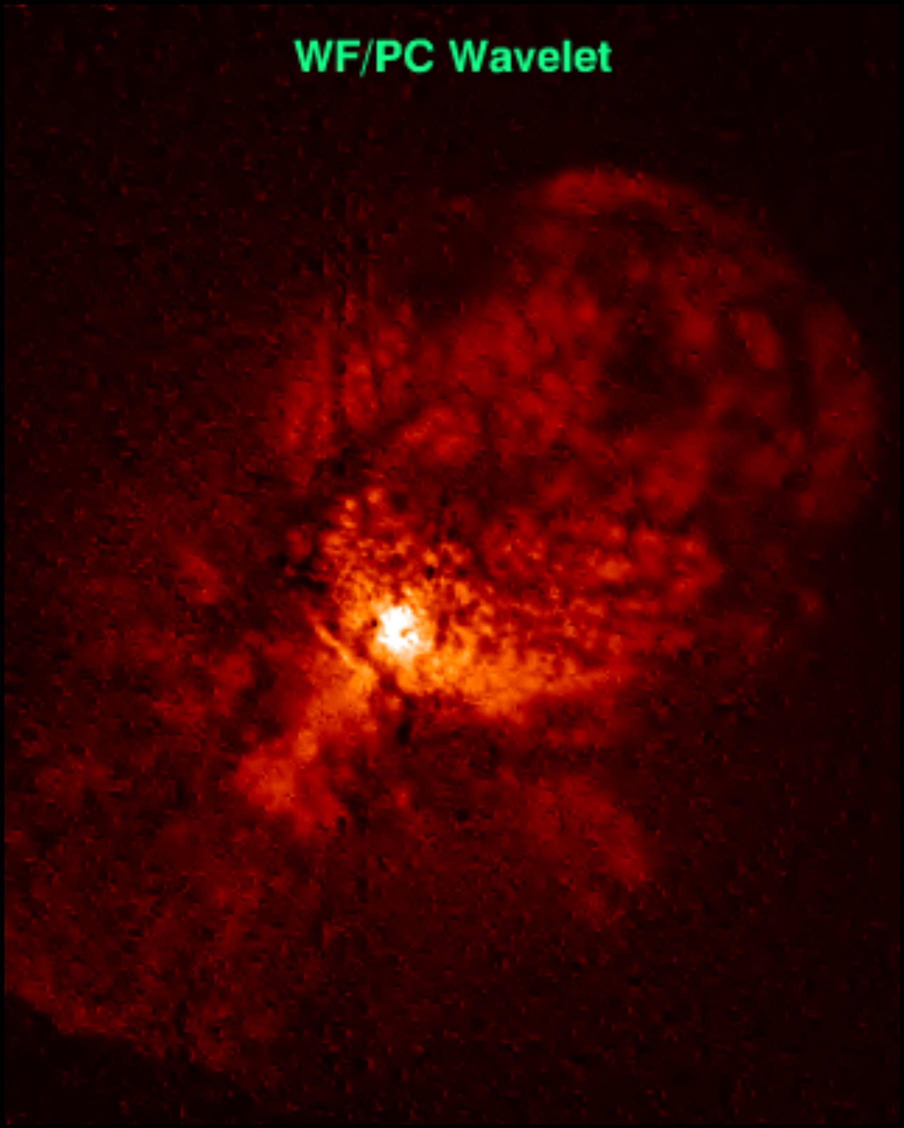
Deconvolution Eta Carina



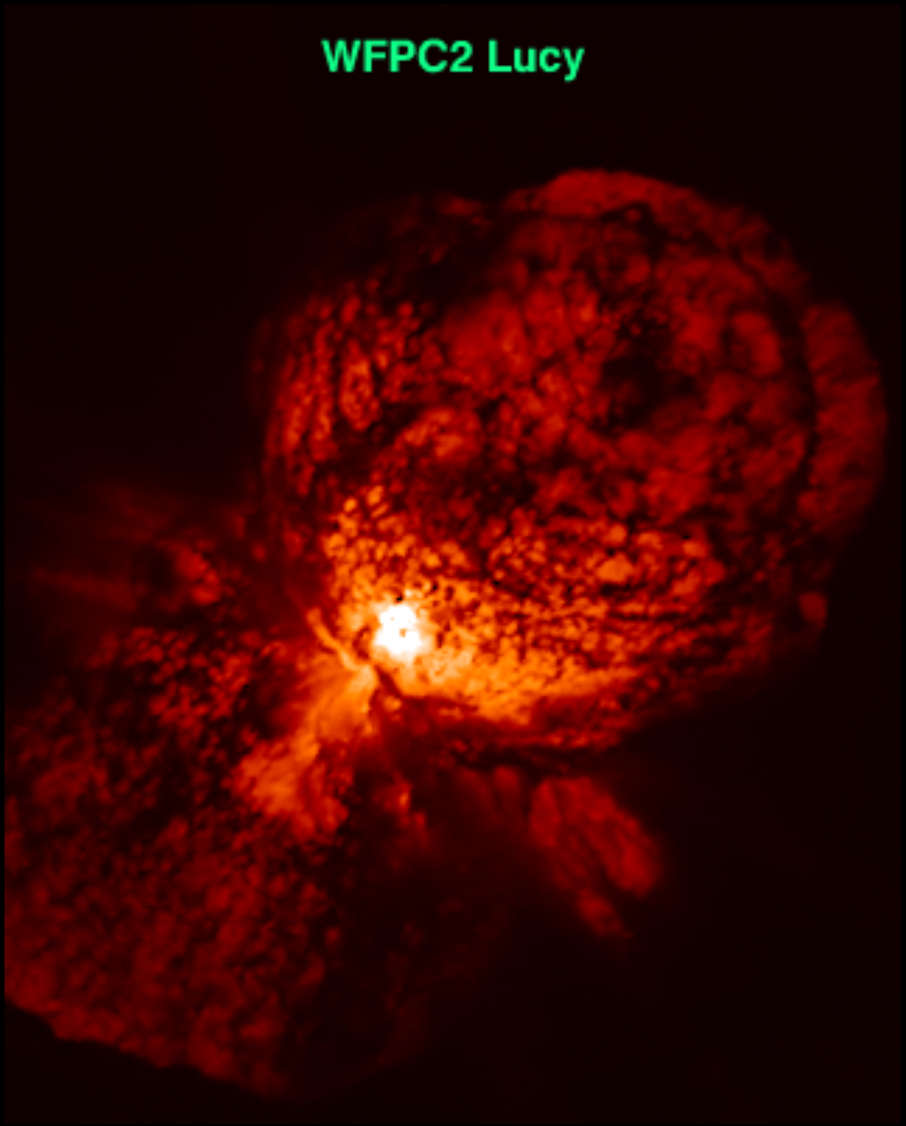


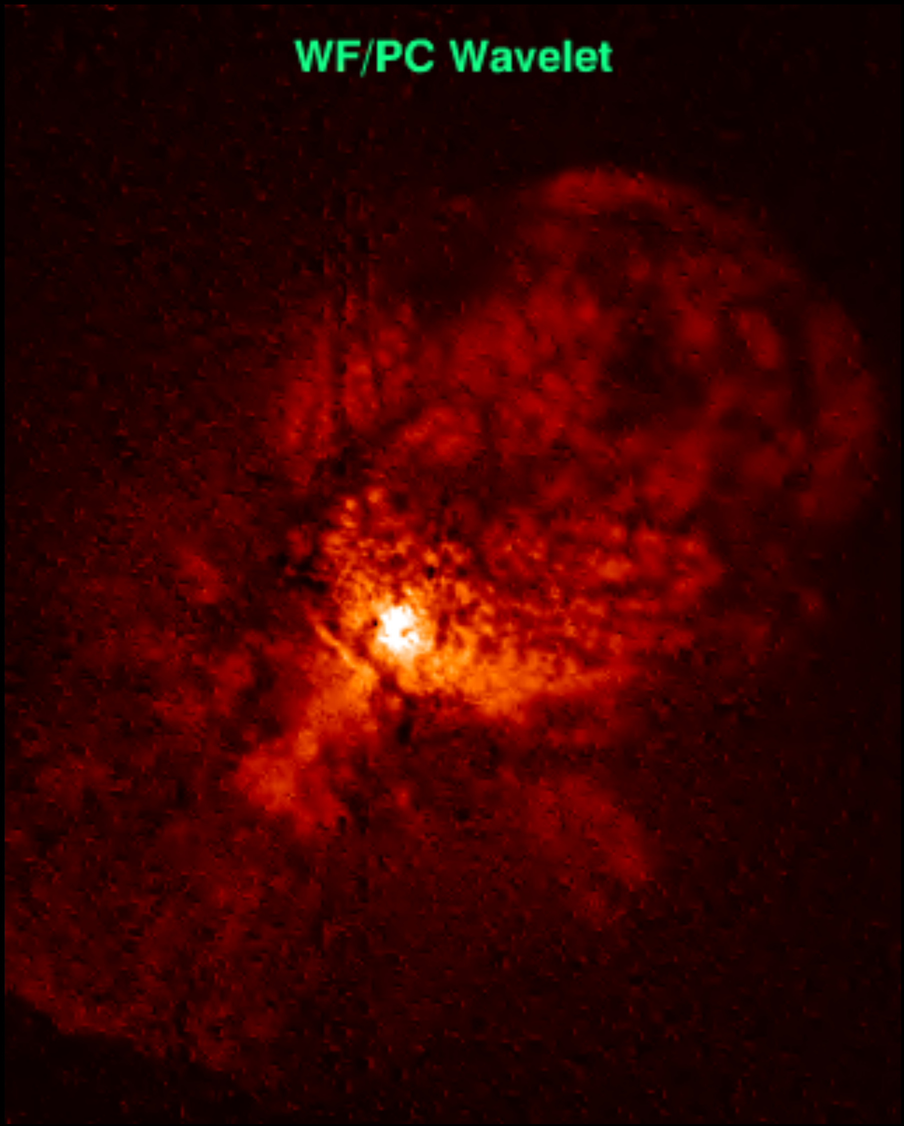
Deconvolution Eta Carina





Deconvolution Eta Carina





Summary: Image Processing

- Use Fourier transforms for convolutions
  - Very fast when they are the right tool
  - Also key for period finding, interferometry

- Use wavelet transforms for almost everything else …
  - Denoising, compression, multi-scale processing, …