3-D Source Finding Challenges in the Era of "Big Data"

Tobias Westmeier (ICRAR / UWA)
and the SoFiA collaboration

International Centre for Radio Astronomy Research

Curtin University

The University of Western Australia

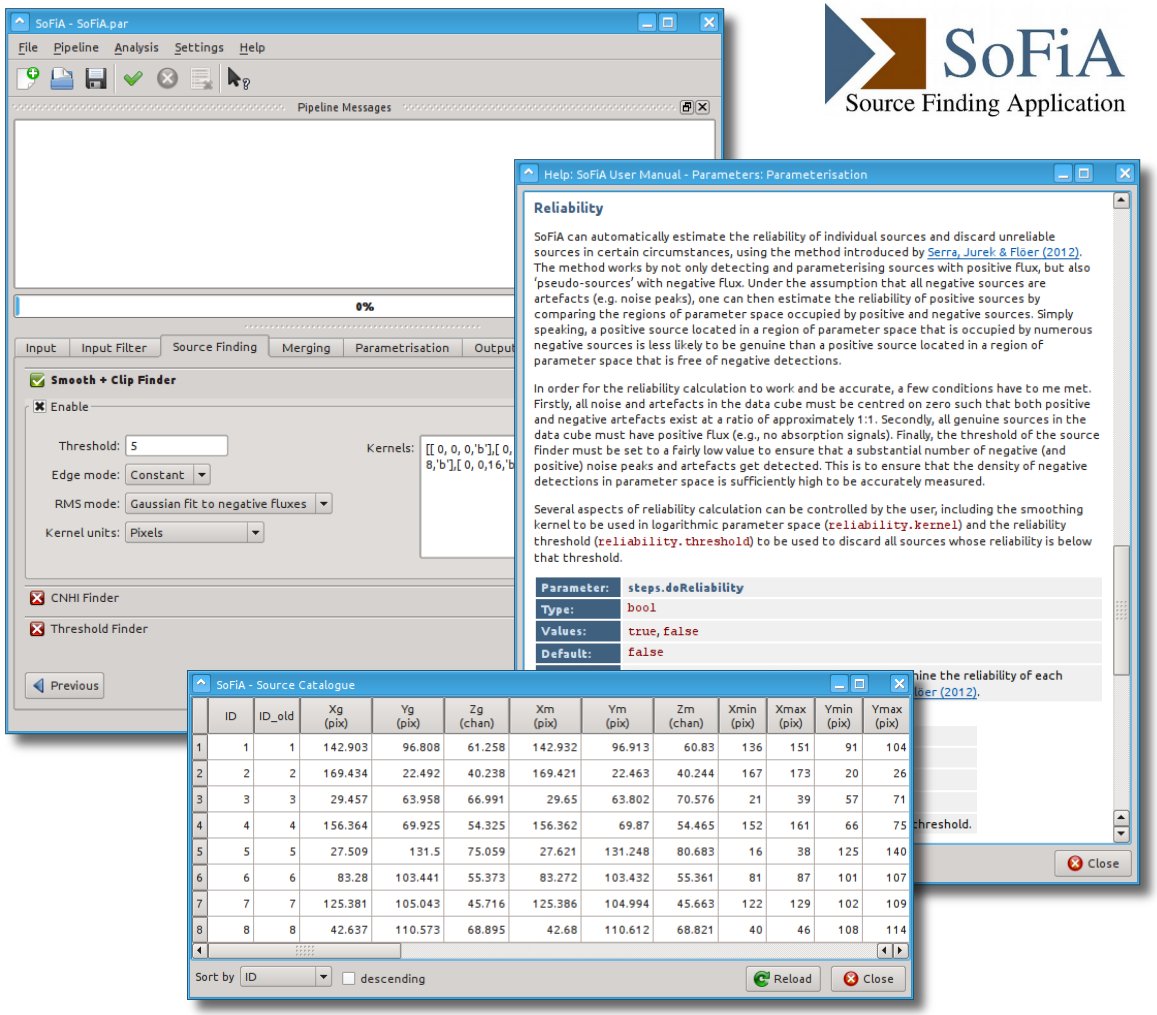
Government of Western Australia
Department of the Premier and Cabinet
Office of Science

SoFiA

Source Finding Application

- ★ Source Finding Application
  - Pipeline for source finding and parameterisation of H I data cubes
- ★ SoFiA website
  - https://github.com/SoFiA-Admin/SoFiA/
- ★ Programmers and contributors
  - Paolo Serra (CSIRO), Tobias Westmeier (ICRAR), Nadine Giese (Groningen), Russell Jurek (CSIRO), Lars Flöer (Bonn), Attila Popping (ICRAR), Benjamin Winkel (MPIfR), Thijs van der Hulst (Groningen), Martin Meyer (ICRAR), Bärbel Koribalski (CSIRO), Lister Staveley-Smith (ICRAR), Hélène Courtois (Lyon)
  - Thank you to SoFiA users for their feedback

- ★ Features of SoFiA
  - Modern graphical user interface



★ Features of SoFiA

- Modern graphical user interface
- Extensive user manual and tutorial

SoFiA Source Finding Application

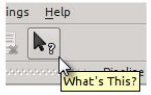
SoFiA Tutorial – 3 Basic Source Finding Run

SoFiA edited this page on 7 Dec 2015 · 24 revisions

3 Setting up a basic source finding run

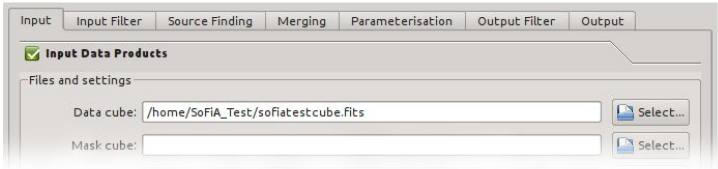
This section illustrates how to run SoFiA on the H I data cube provided for testing purposes on the SoFiA wiki at https://github.com/SoFiA-Admin/SoFiA/wiki/SoFiA-Test-Data-Set. The settings for this example are provided in the file `SoFiA_Tutorial_Section_3_S+C.par` which can be directly loaded into SoFiA by selecting "Open…" from the "File" entry in the menu. Alternatively, all parameters can be set manually by following the instructions below.

> **Note**
>
> In order to get more information about a particular parameter setting, you can first click on the "What's this?" icon in the tool bar (or the corresponding item in the help section of the menu bar) and then on the corresponding field or button. This should open a tool tip with some basic information about the parameter and its possible values.



3.1 Selecting the input data cube

Navigate to the first tab ("**Input**"). In the "Input Data Products" section click on the "Select…" button next to the "Data cube" field. This will open a file selection window in which you can select and open the input data cube named `sofiatestcube.fits`. The full path of the data cube should now appear in the text field, as shown below in Fig. 2. In addition, the small icon next to the section heading should have turned from red to green, indicating that an input data cube has been specified.

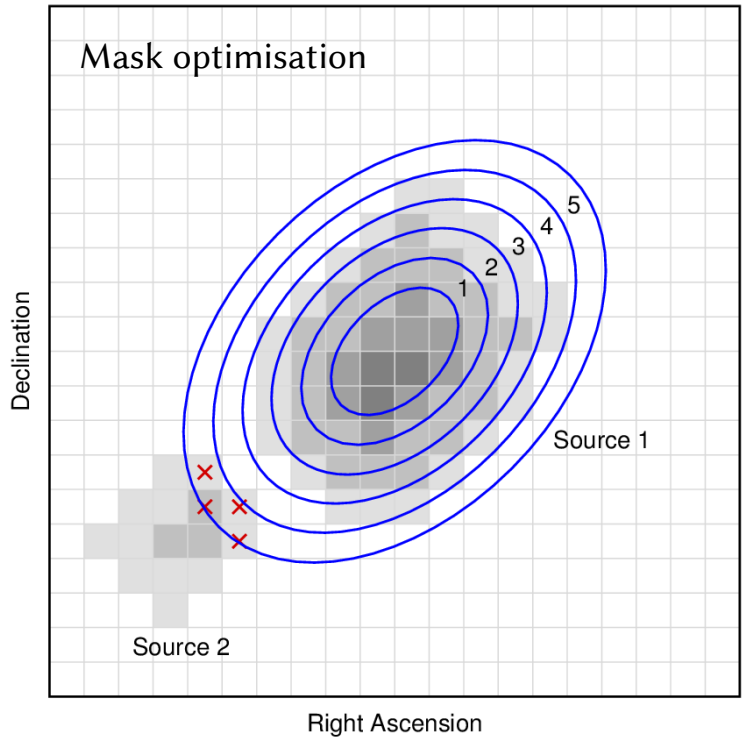


★ Features of SoFiA

- Modern graphical user interface
- Extensive user manual and tutorial
- Multiple source finding algorithms
  - Basic threshold finder
  - Smooth + Clip finder (Serra et al. 2012)
  - CNHI finder (Jurek 2012)
  - 2D–1D wavelet finder (Flöer & Winkel 2012)

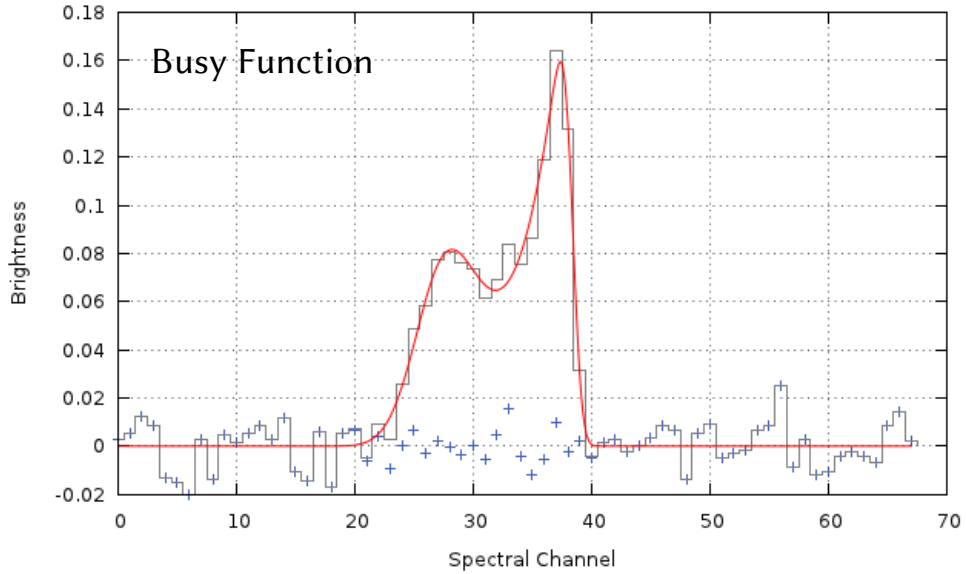
★ Features of SoFiA

- Modern graphical user interface
- Extensive user manual and tutorial
- Multiple source finding algorithms
  - Basic threshold finder
  - Smooth + Clip finder (Serra et al. 2012)
  - CNHI finder (Jurek 2012)
  - 2D–1D wavelet finder (Flöer & Winkel 2012)
- Different mask optimisation algorithms



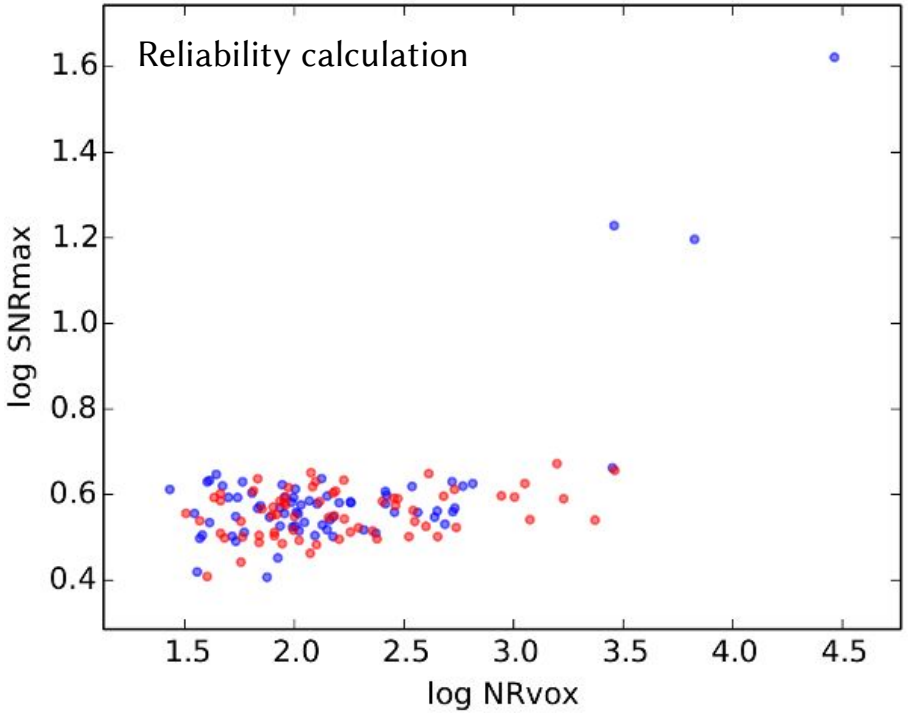
★ Features of SoFiA

- Modern graphical user interface
- Extensive user manual and tutorial
- Multiple source finding algorithms
  - Basic threshold finder
  - Smooth + Clip finder (Serra et al. 2012)
  - CNHI finder (Jurek 2012)
  - 2D–1D wavelet finder (Flöer & Winkel 2012)
- Different mask optimisation algorithms
- Busy Function fitting (Westmeier et al. 2014)



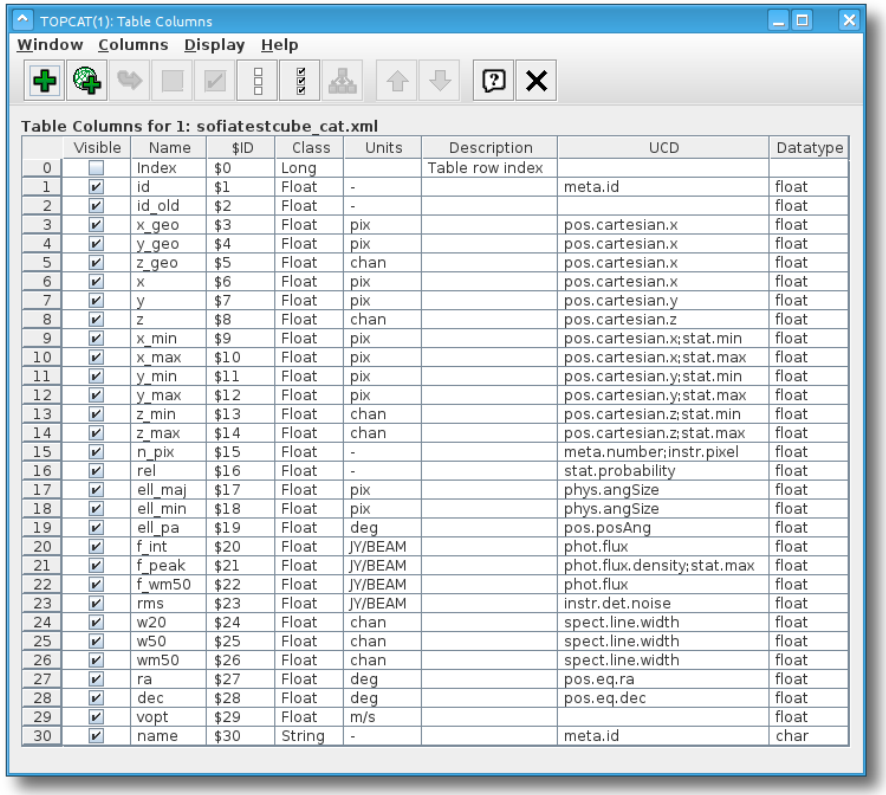
★ Features of SoFiA

- Modern graphical user interface
- Extensive user manual and tutorial
- Multiple source finding algorithms
  - Basic threshold finder
  - Smooth + Clip finder (Serra et al. 2012)
  - CNHI finder (Jurek 2012)
  - 2D–1D wavelet finder (Flöer & Winkel 2012)
- Different mask optimisation algorithms
- Busy Function fitting (Westmeier et al. 2014)
- Reliability calculation / filtering (Serra et al. 2012)



★ Features of SoFiA

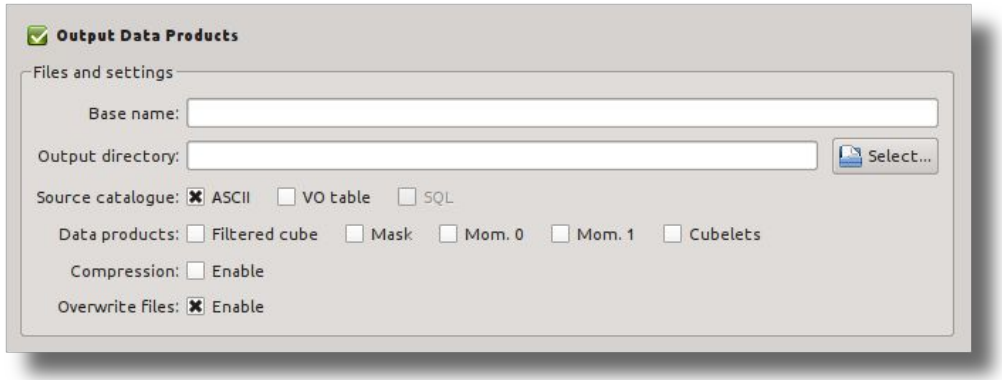
- Modern graphical user interface
- Extensive user manual and tutorial
- Multiple source finding algorithms
  - Basic threshold finder
  - Smooth + Clip finder (Serra et al. 2012)
  - CNHI finder (Jurek 2012)
  - 2D–1D wavelet finder (Flöer & Winkel 2012)
- Different mask optimisation algorithms
- Busy Function fitting (Westmeier et al. 2014)
- Reliability calculation / filtering (Serra et al. 2012)
- ASCII and VOTable output



	Visible	Name	$ID	Class	Units	Description	UCD	Datatype
0		Index	$0	Long		Table row index		
1	✔	id	$1	Float	-		meta.id	float
2	✔	id_old	$2	Float	-			float
3	✔	x_geo	$3	Float	pix		pos.cartesian.x	float
4	✔	y_geo	$4	Float	pix		pos.cartesian.x	float
5	✔	z_geo	$5	Float	chan		pos.cartesian.x	float
6	✔	x	$6	Float	pix		pos.cartesian.x	float
7	✔	y	$7	Float	pix		pos.cartesian.y	float
8	✔	z	$8	Float	chan		pos.cartesian.z	float
9	✔	x_min	$9	Float	pix		pos.cartesian.x;stat.min	float
10	✔	x_max	$10	Float	pix		pos.cartesian.x;stat.max	float
11	✔	y_min	$11	Float	pix		pos.cartesian.y;stat.min	float
12	✔	y_max	$12	Float	pix		pos.cartesian.y;stat.max	float
13	✔	z_min	$13	Float	chan		pos.cartesian.z;stat.min	float
14	✔	z_max	$14	Float	chan		pos.cartesian.z;stat.max	float
15	✔	n_pix	$15	Float	-		meta.number;instr.pixel	float
16	✔	rel	$16	Float	-		stat.probability	float
17	✔	ell_maj	$17	Float	pix		phys.angSize	float
18	✔	ell_min	$18	Float	pix		phys.angSize	float
19	✔	ell_pa	$19	Float	deg		pos.posAng	float
20	✔	f_int	$20	Float	JY/BEAM		phot.flux	float
21	✔	f_peak	$21	Float	JY/BEAM		phot.flux.density;stat.max	float
22	✔	f_wm50	$22	Float	JY/BEAM		phot.flux	float
23	✔	rms	$23	Float	JY/BEAM		instr.det.noise	float
24	✔	w20	$24	Float	chan		spect.line.width	float
25	✔	w50	$25	Float	chan		spect.line.width	float
26	✔	wm50	$26	Float	chan		spect.line.width	float
27	✔	ra	$27	Float	deg		pos.eq.ra	float
28	✔	dec	$28	Float	deg		pos.eq.dec	float
29	✔	vopt	$29	Float	m/s			float
30	✔	name	$30	String	-		meta.id	char

★ Features of SoFiA

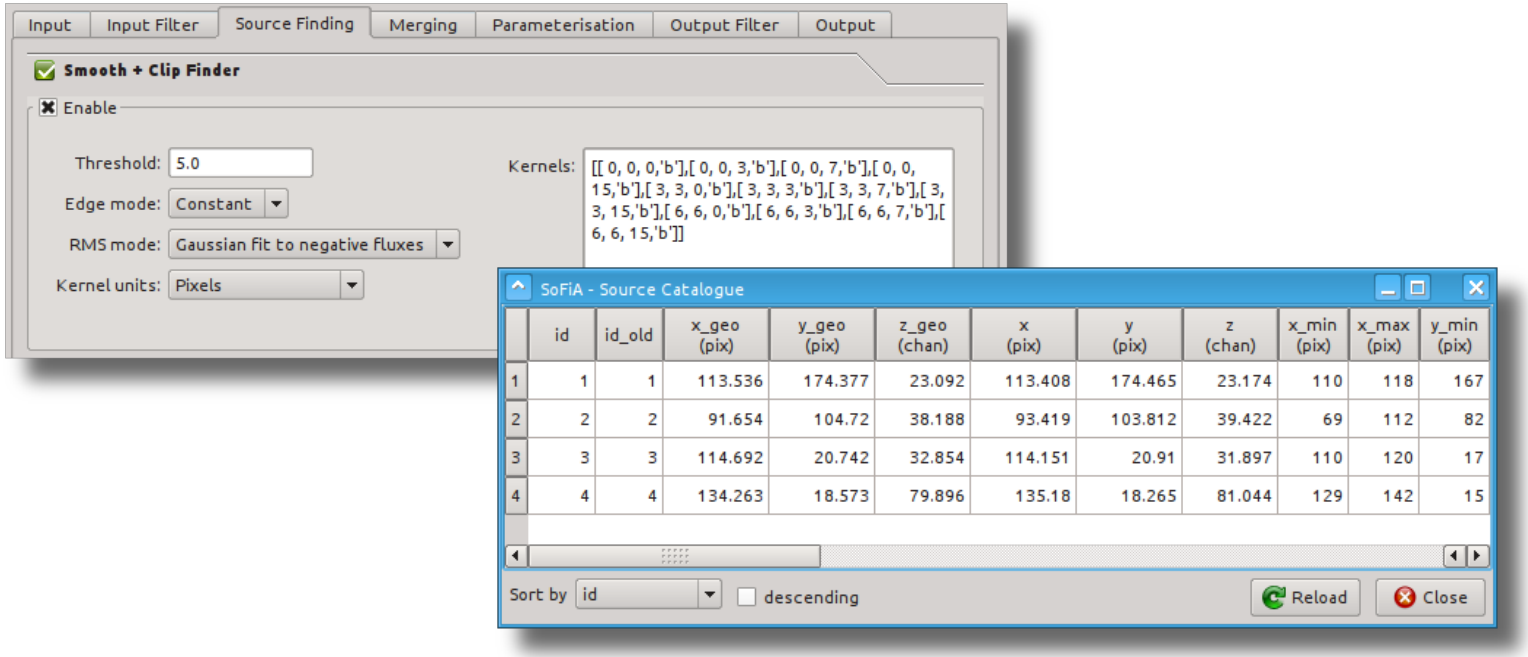
- Modern graphical user interface
- Extensive user manual and tutorial
- Multiple source finding algorithms
  - Basic threshold finder
  - Smooth + Clip finder (Serra et al. 2012)
  - CNHI finder (Jurek 2012)
  - 2D–1D wavelet finder (Flöer & Winkel 2012)
- Different mask optimisation algorithms
- Busy Function fitting (Westmeier et al. 2014)
- Reliability calculation / filtering (Serra et al. 2012)
- ASCII and VOTable output
- Wide range of output data products



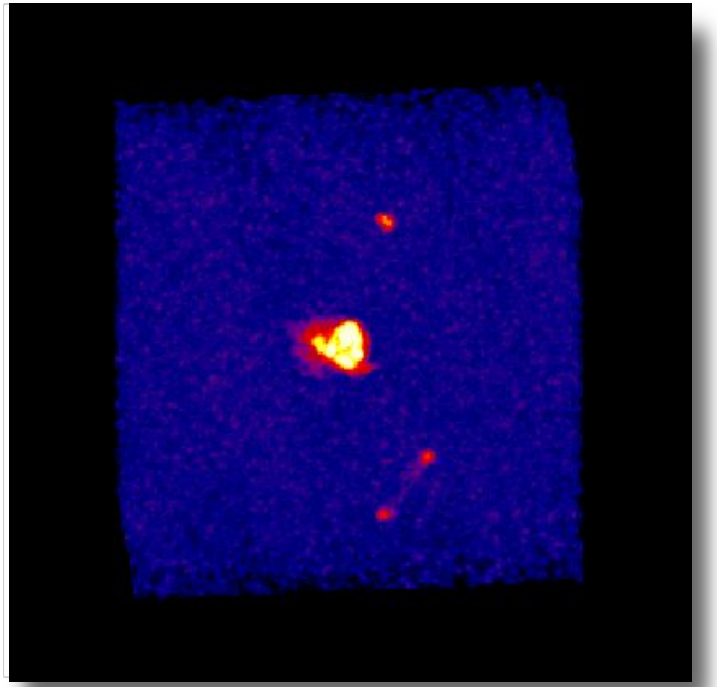
Source Finding Example

Source Finding Example

- ★ SoFiA test cube
  - WSRT cube from ATLAS^{3D} survey (Serra et al. 2012)
  - Using S + C finder with 5σ threshold



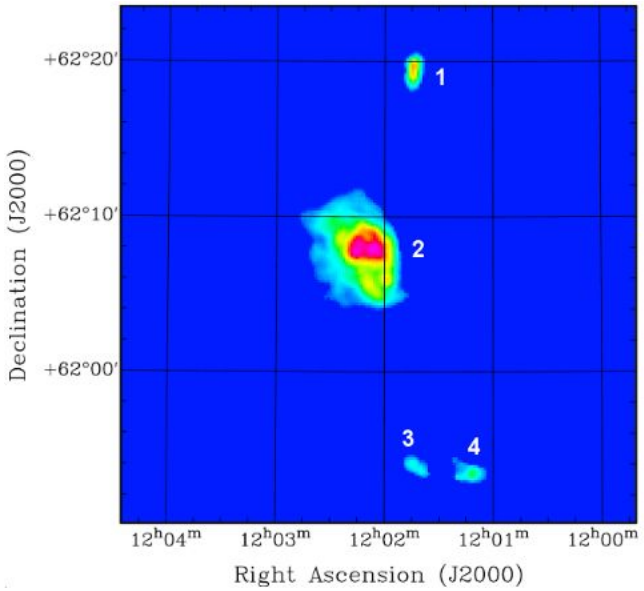
	id	id_old	x_geo (pix)	y_geo (pix)	z_geo (chan)	x (pix)	y (pix)	z (chan)	x_min (pix)	x_max (pix)	y_min (pix)
1	1	1	113.536	174.377	23.092	113.408	174.465	23.174	110	118	167
2	2	2	91.654	104.72	38.188	93.419	103.812	39.422	69	112	82
3	3	3	114.692	20.742	32.854	114.151	20.91	31.897	110	120	17
4	4	4	134.263	18.573	79.896	135.18	18.265	81.044	129	142	15

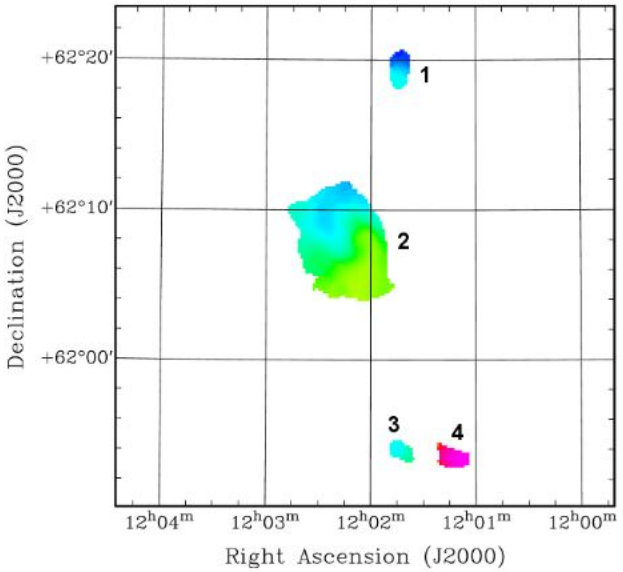


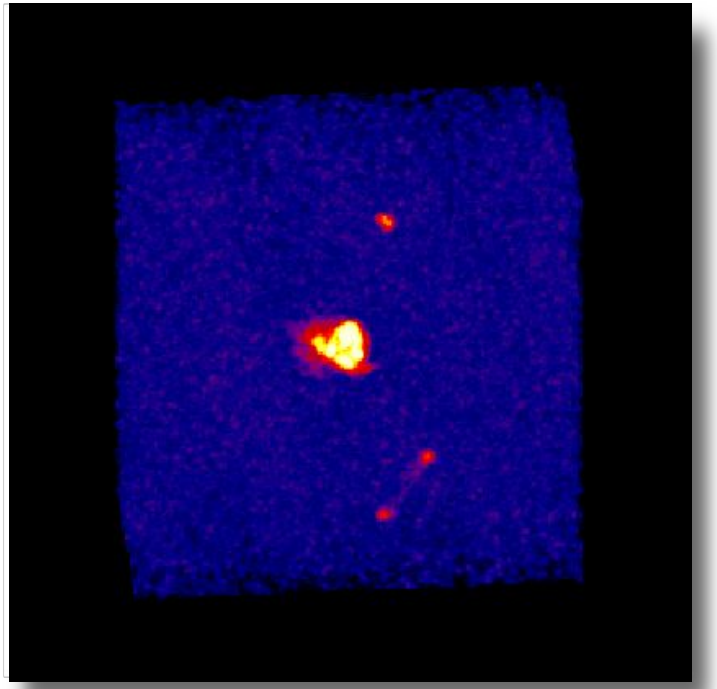
SoFiA test data cube

Source Finding Example

- SoFiA test cube
  - Problem: edge-on galaxy split into two sources





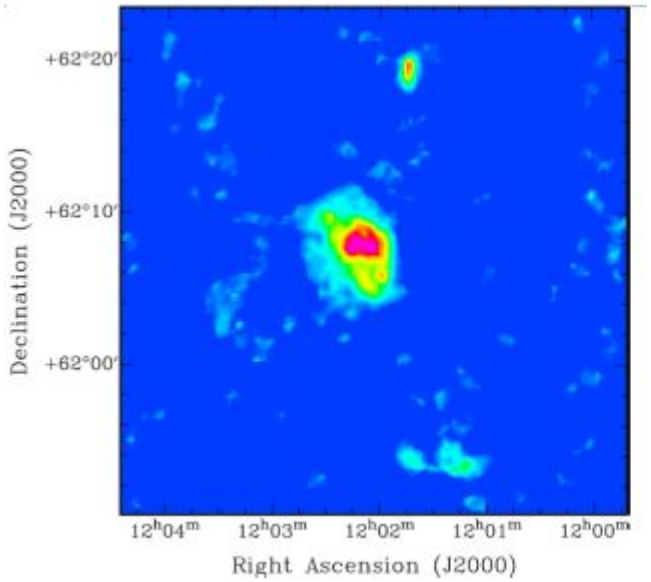


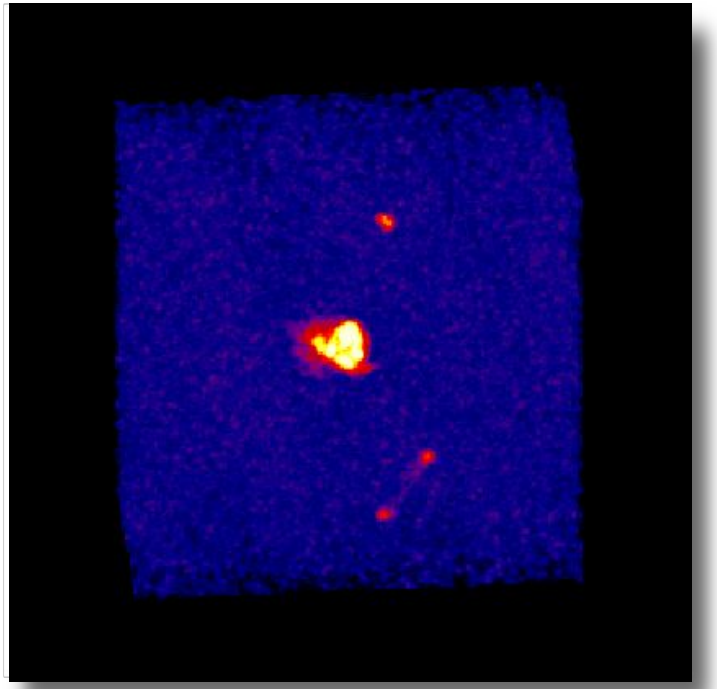
SoFiA test data cube

  - Let's use a lower threshold of 3σ...

Source Finding Example

- ★ SoFiA test cube
  - Problem: too many false detections



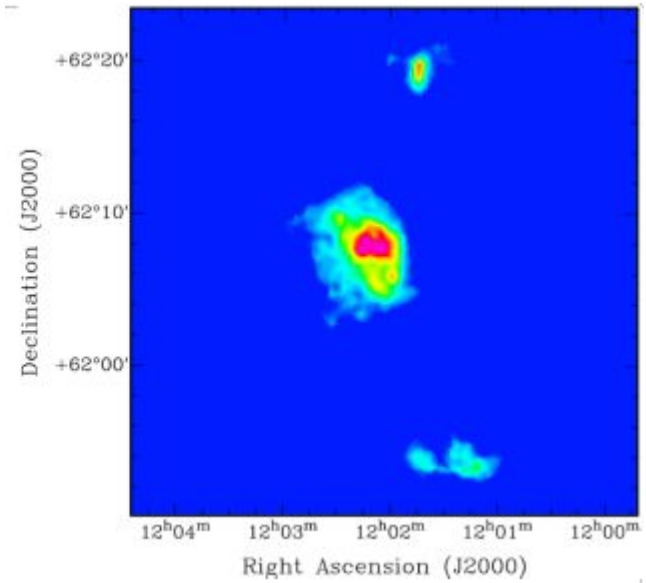


SoFiA test data cube

  - Solution: reliability filtering

Source Finding Example

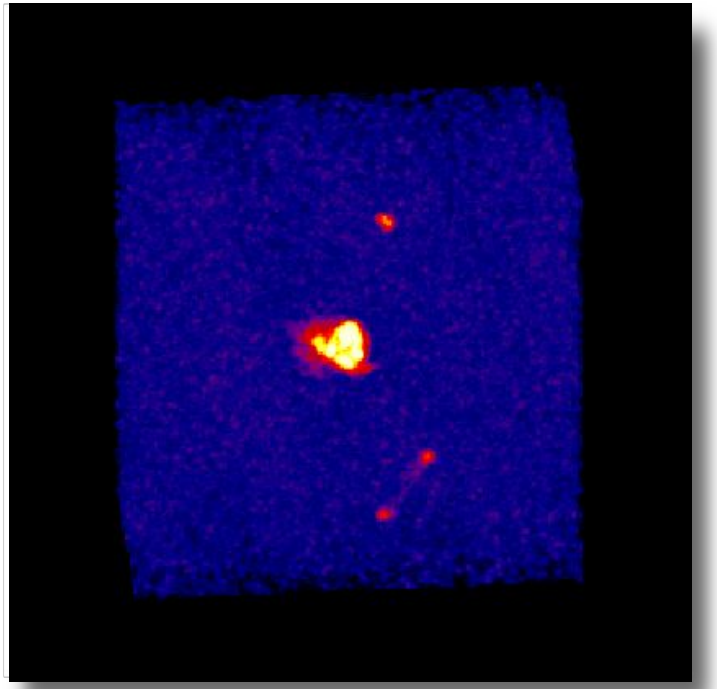
- ★ SoFiA test cube
  - All three galaxies detected above 90% reliability threshold



SoFiA - Source Catalogue

	z_max (chan)	n_pix	rel	ell_maj (pix)	ell_min (pix)	ell_pa (deg)
1	96	668	1	18.8552	6.3744	84.3352
2	63	2906	1	21.3729	15.8734	31.6328
3	38	2873	1	7.5632	5.5275	165.136
4	57	75	0.882118	2.9334	2.0033	47.9229
5	94	44	[illegible]	3.8123	1.3081	56.3058
6	8	41	0.846014	3.0517	1.3232	129.816
7	53	40	0.830608	3.7634	1.7965	58.026
8	86	528	0.8173	5.2782	2.4251	36.3286
9	17	27	0.774734	4.4023	1.4041	115.779
10	11	36	0.760476	2.3143	1.9623	118.461
11	97	539	0.759669	4.5809	2.7959	160.906
12	98	448	0.678576	6.2555	2.4472	169.203
13	18	101	0.634916	2.4558	2.0636	146.367
14	78	47	0.560401	2.5135	1.6798	118.946
15	82	105	0.554371	2.45	2.1176	74.0788

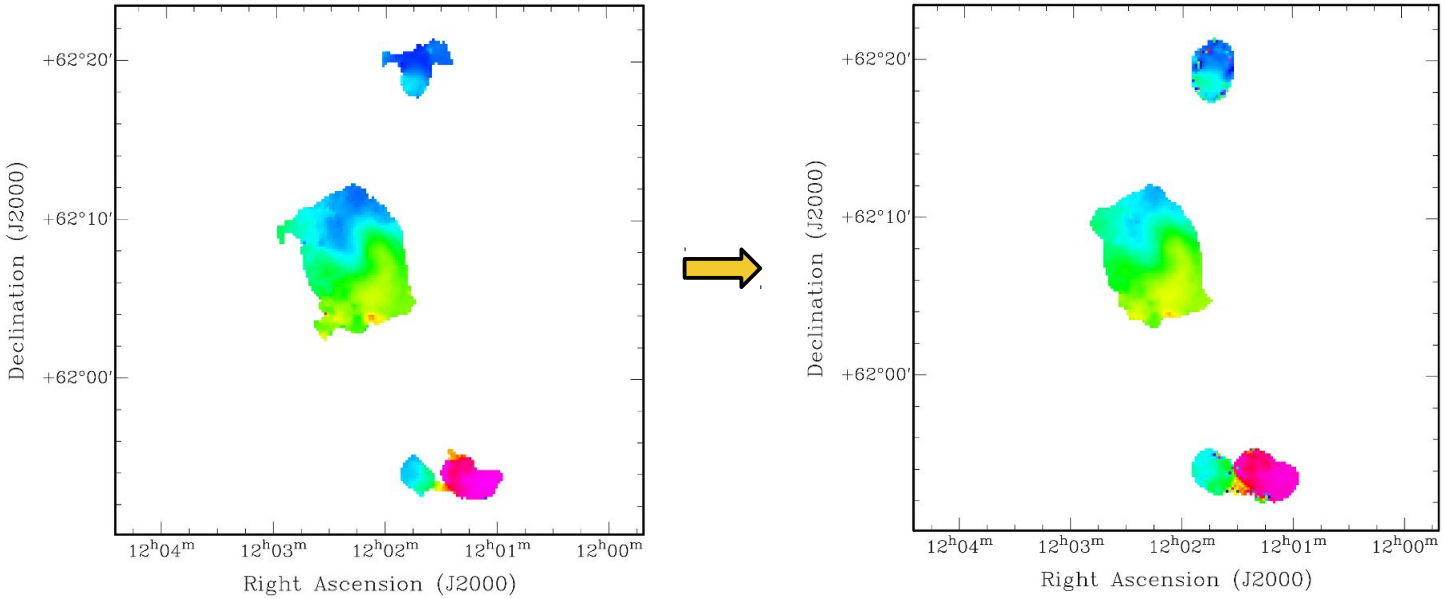
Sort by rel ☒ descending Reload Close

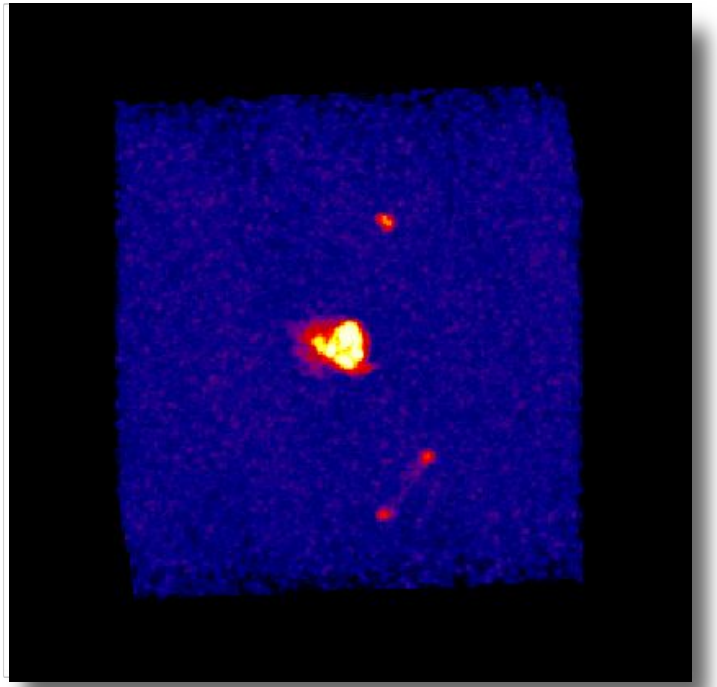


SoFiA test data cube

Source Finding Example

- SoFiA test cube
  - Mask optimisation for accurate parameterisation

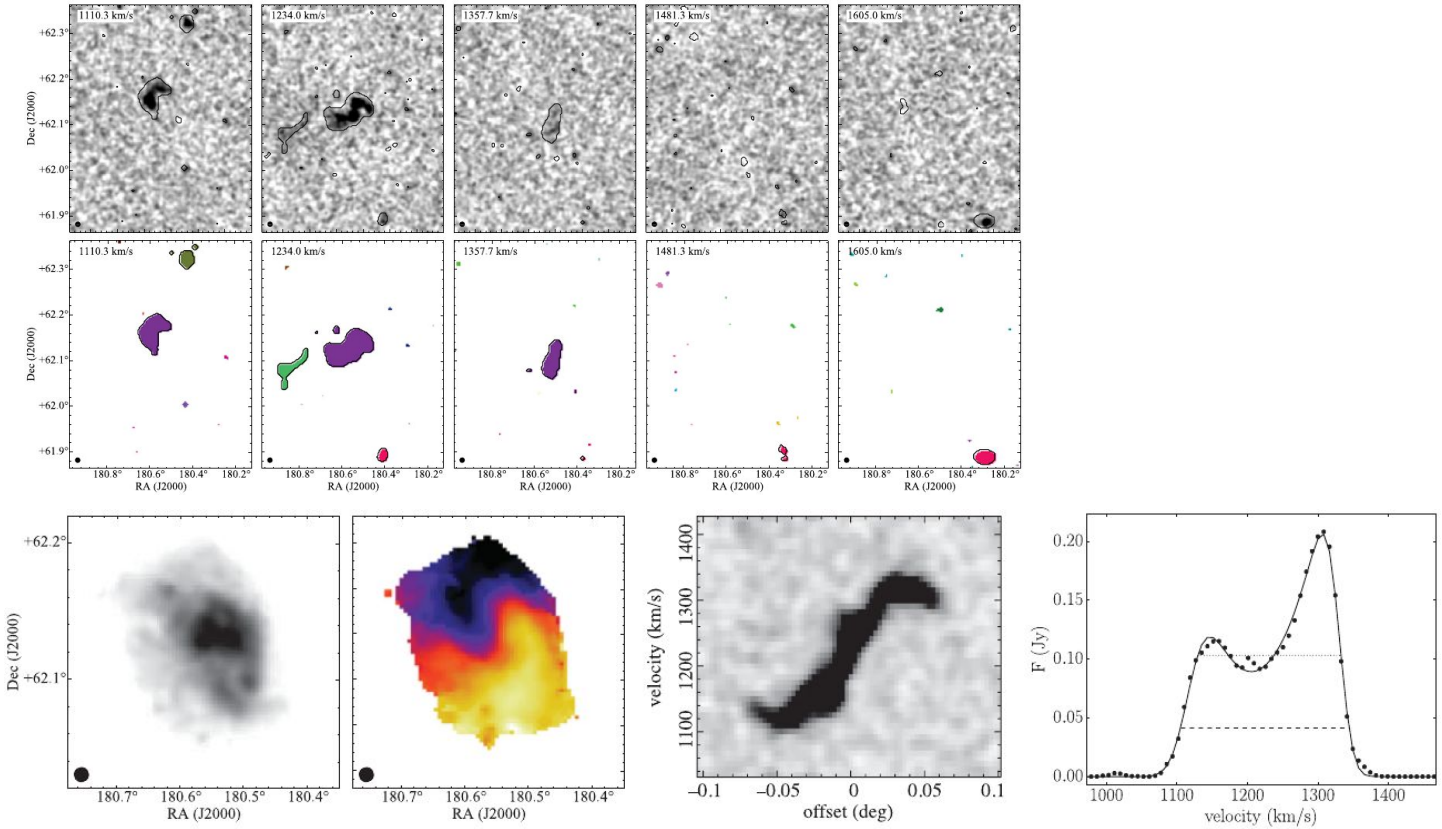


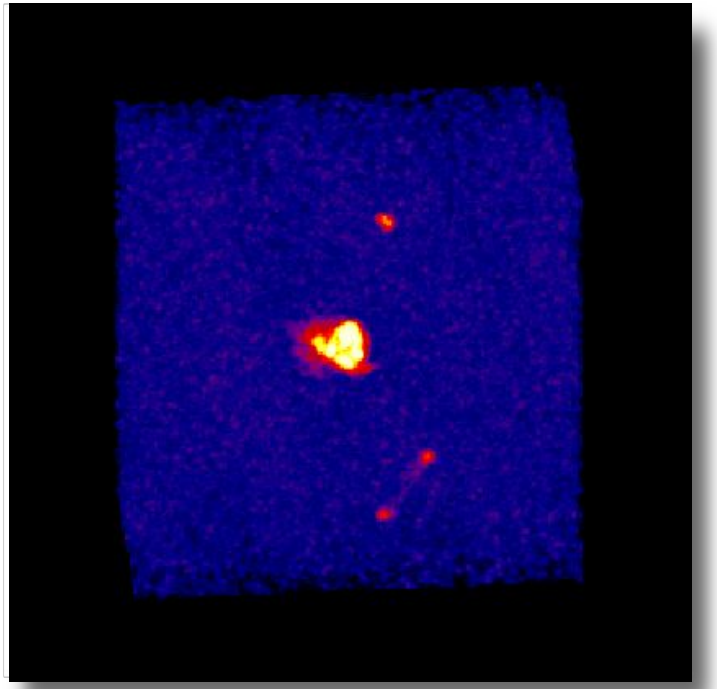


SoFiA test data cube

Source Finding Example

- ★ SoFiA test cube
  - Examples of SoFiA data products





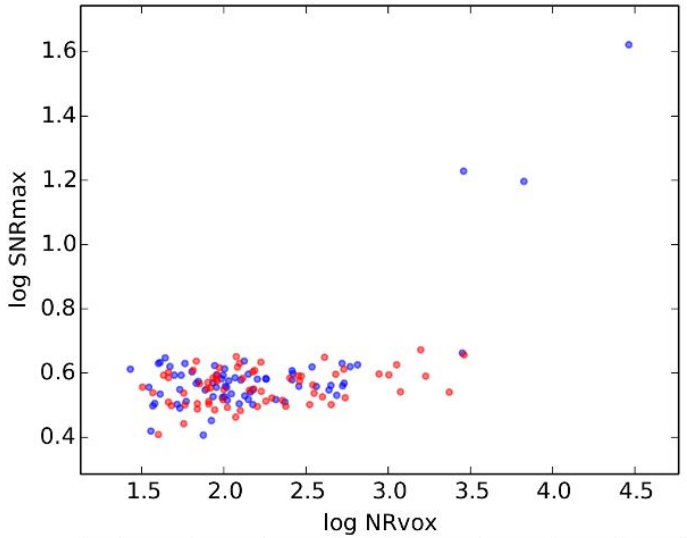
SoFiA test data cube

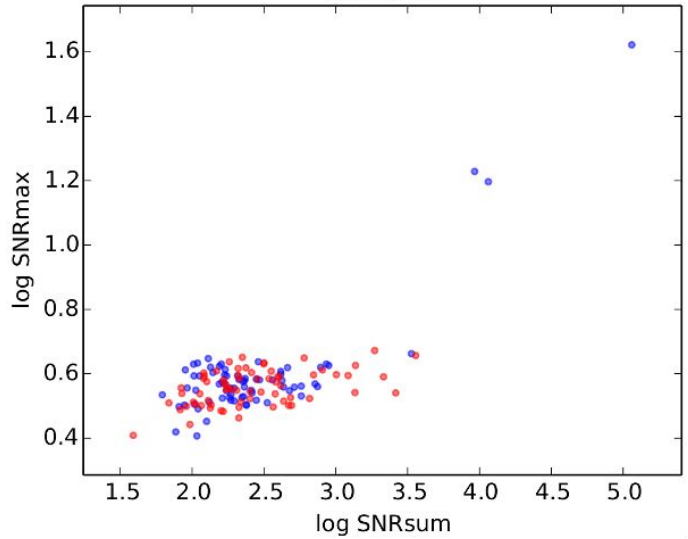
Era of “Big Data”

Era of "Big Data"

- ★ WALLABY Stokes-I data cubes
  - ≈ 800 GB per field
  - ≈ 1 PB for the survey
  - ≈ 500,000 galaxies
- ★ Issues
  - Time and memory limitations ⟶ parallelisation required
  - Visual quality checks virtually impossible

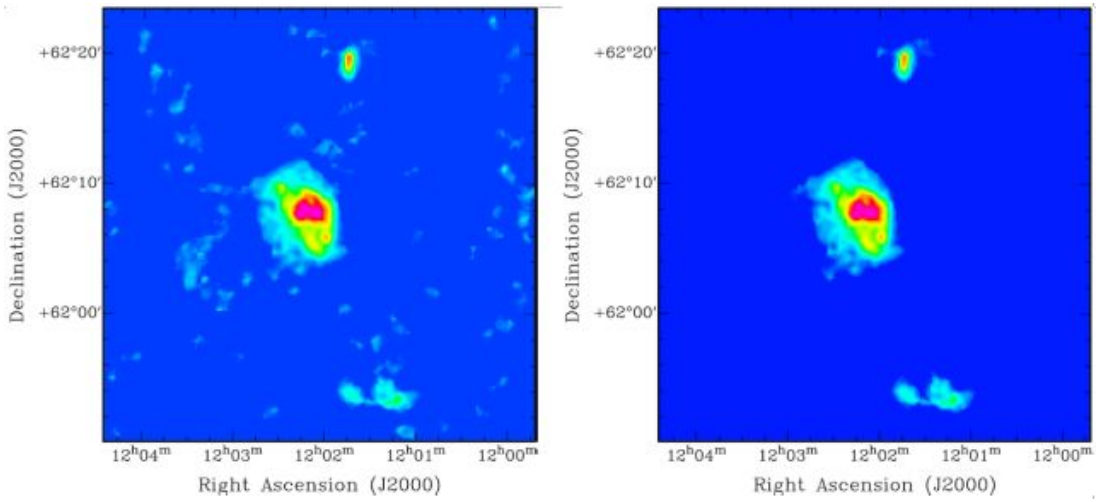
- ★ Improving reliability
  - Assumptions:
    - Genuine sources have positive flux
    - Flux distribution of noise is symmetric about zero
  - Compare density of pos. and neg. detections in parameter space ⟶ reliability of detections
  - Serra et al. 2012, PASA, 29, 296

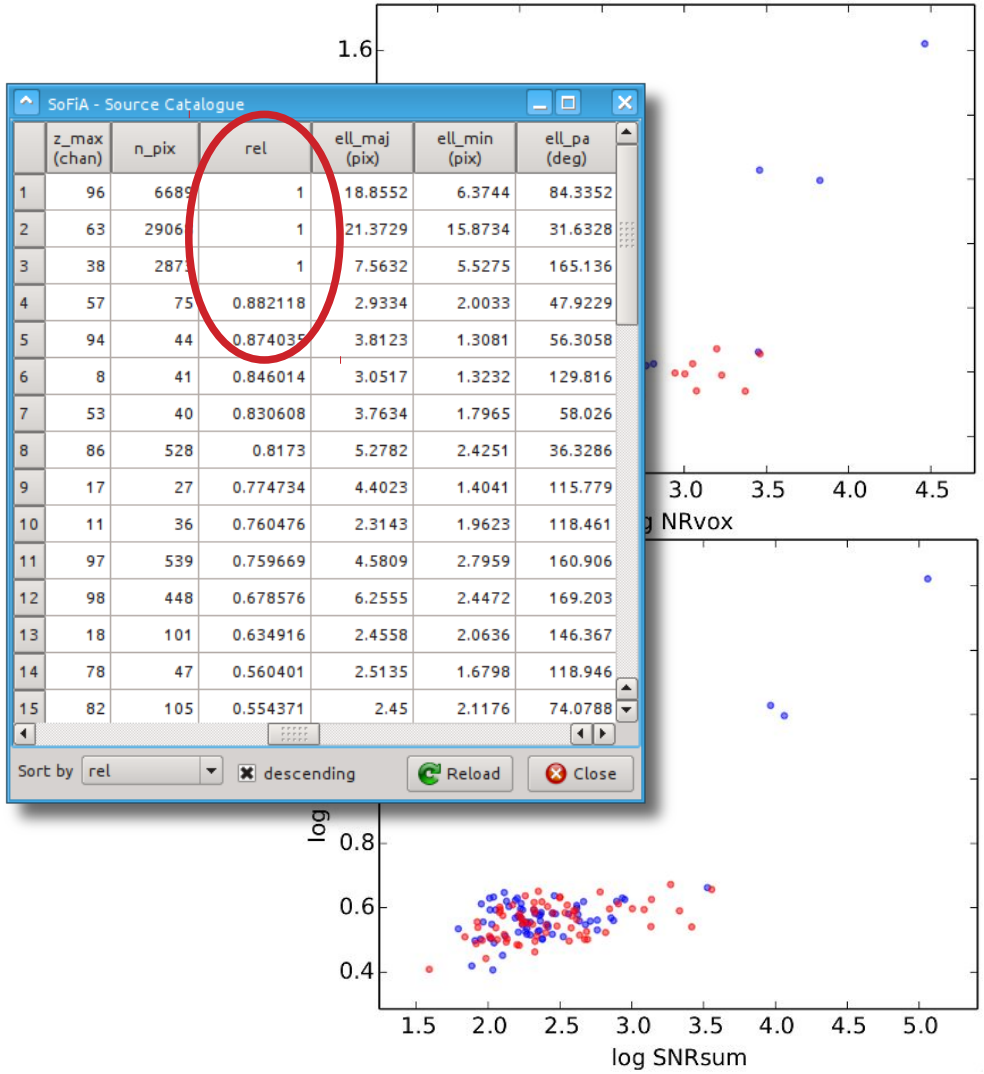




★ Improving reliability

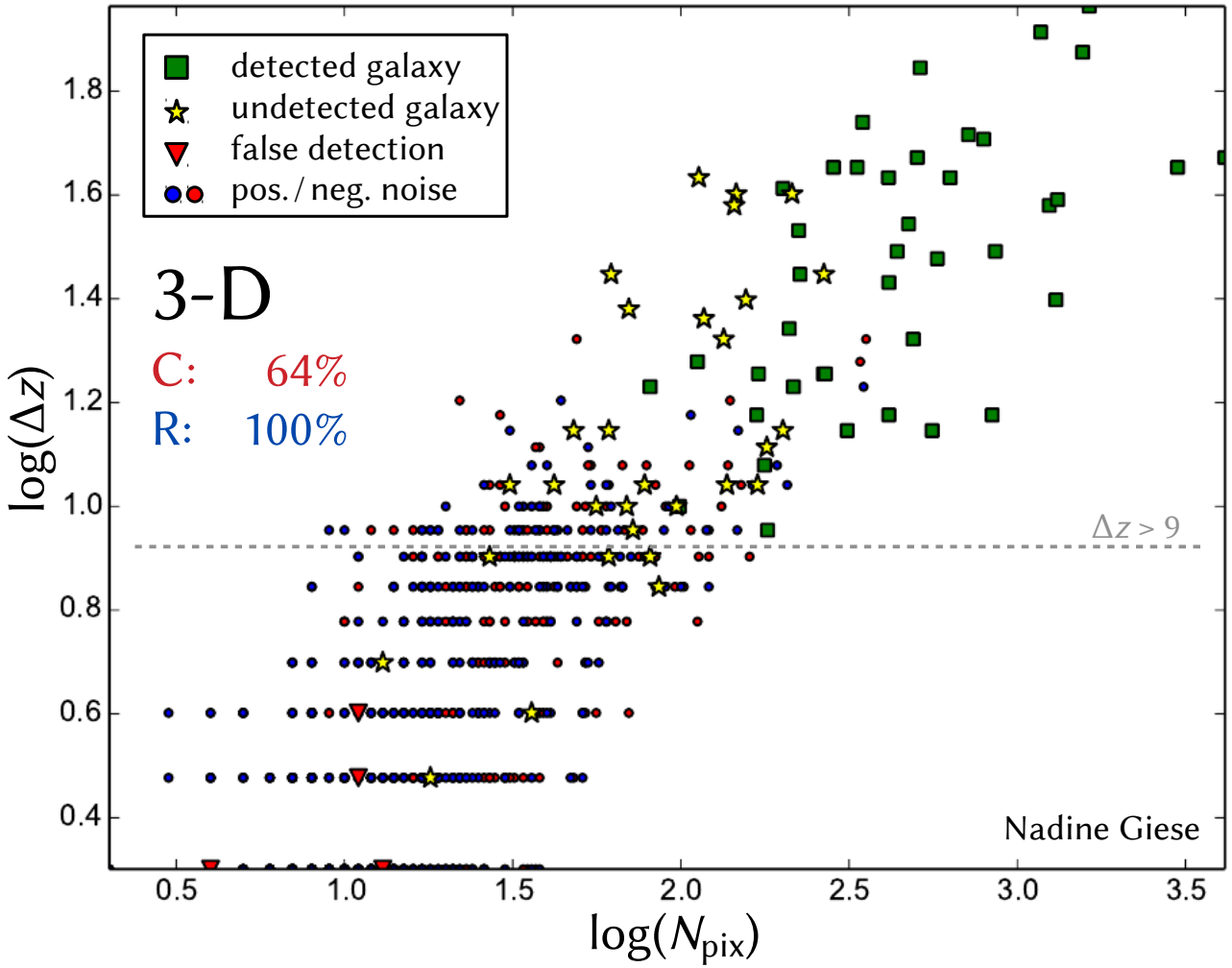
- Assumptions:
  - Genuine sources have positive flux
  - Flux distribution of noise is symmetric about zero
- Compare density of pos. and neg. detections in parameter space ⟶ reliability of detections
- Serra et al. 2012, PASA, 29, 296



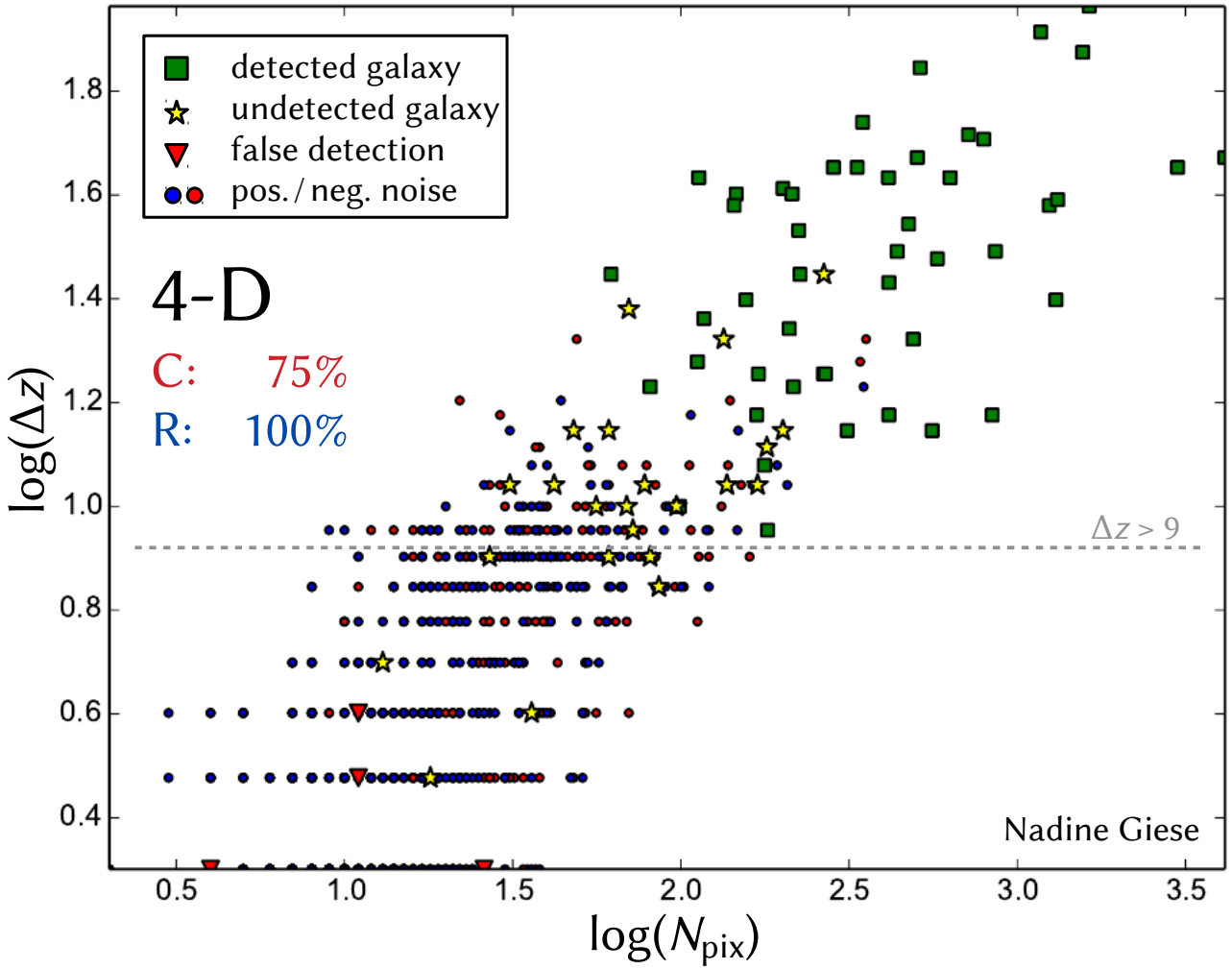


	z_max (chan)	n_pix	rel	ell_maj (pix)	ell_min (pix)	ell_pa (deg)
1	96	6689	1	18.8552	6.3744	84.3352
2	63	2906	1	21.3729	15.8734	31.6328
3	38	287	1	7.5632	5.5275	165.136
4	57	75	0.882118	2.9334	2.0033	47.9229
5	94	44	0.874035	3.8123	1.3081	56.3058
6	8	41	0.846014	3.0517	1.3232	129.816
7	53	40	0.830608	3.7634	1.7965	58.026
8	86	528	0.8173	5.2782	2.4251	36.3286
9	17	27	0.774734	4.4023	1.4041	115.779
10	11	36	0.760476	2.3143	1.9623	118.461
11	97	539	0.759669	4.5809	2.7959	160.906
12	98	448	0.678576	6.2555	2.4472	169.203
13	18	101	0.634916	2.4558	2.0636	146.367
14	78	47	0.560401	2.5135	1.6798	118.946
15	82	105	0.554371	2.45	2.1176	74.0788

- ★ Improving reliability
  - ▸ 3-D ⟶ 4-D parameter space
    - $N_{pix} - \Delta z - SNR_{int} - SNR_{peak}$

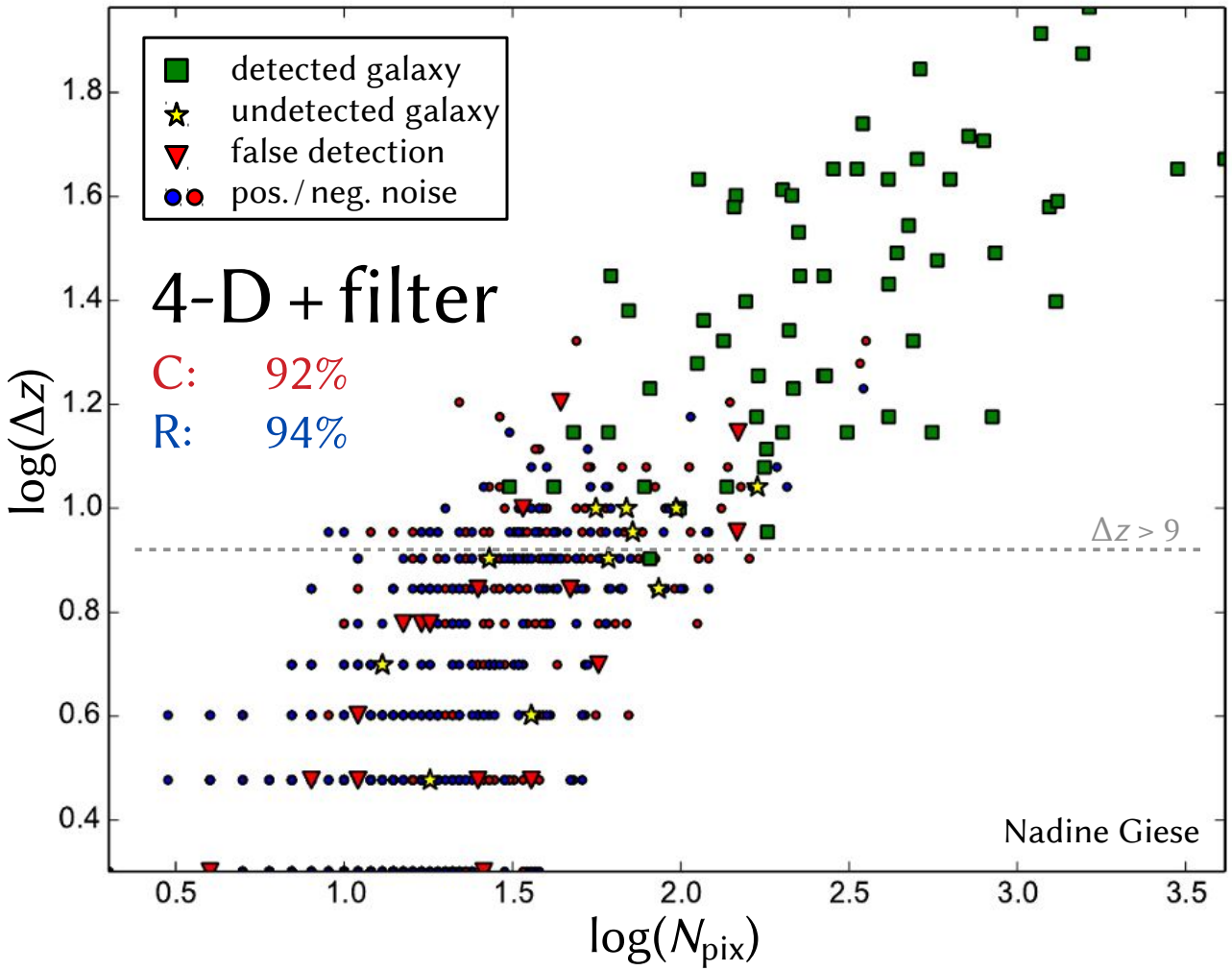


- ★ Improving reliability
  - ▸ 3-D ⟶ 4-D parameter space
    - $N_{pix} - \Delta z - SNR_{int} - SNR_{peak}$



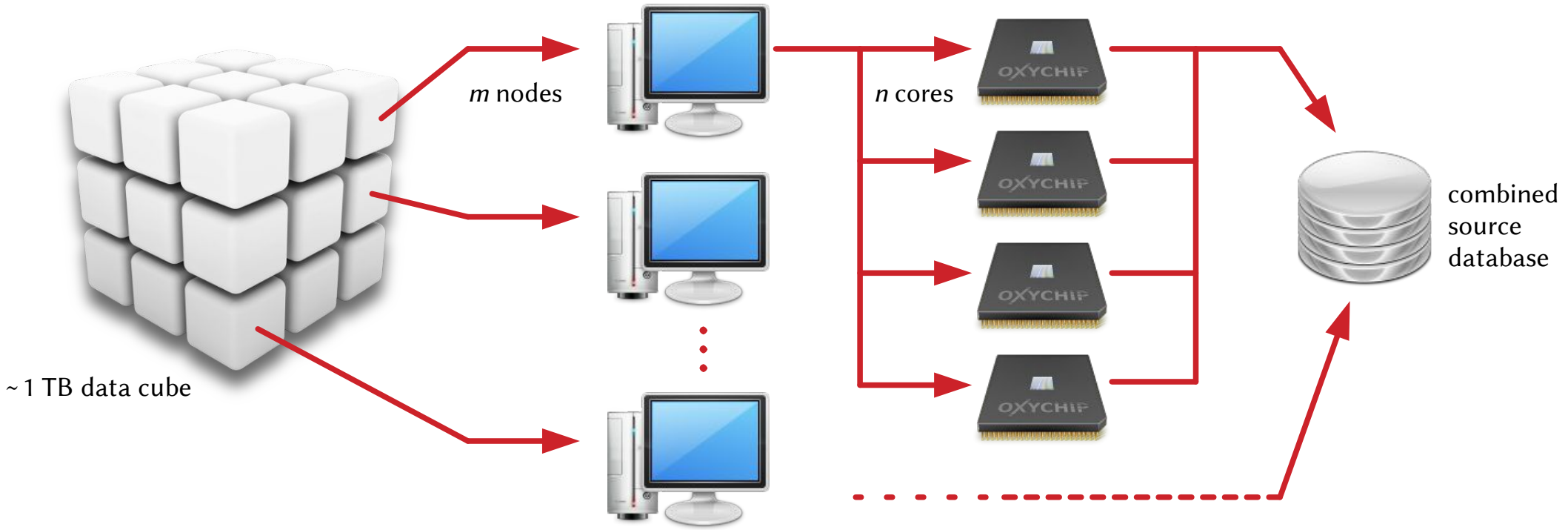
- ★ Improving reliability
  - 3-D ⟶ 4-D parameter space
    - $N_{pix} - \Delta z - SNR_{int} - SNR_{peak}$
  - Improvement of filter used for smoothing in parameter space
    - Narrower filter in Δz direction

More at the next busy week...

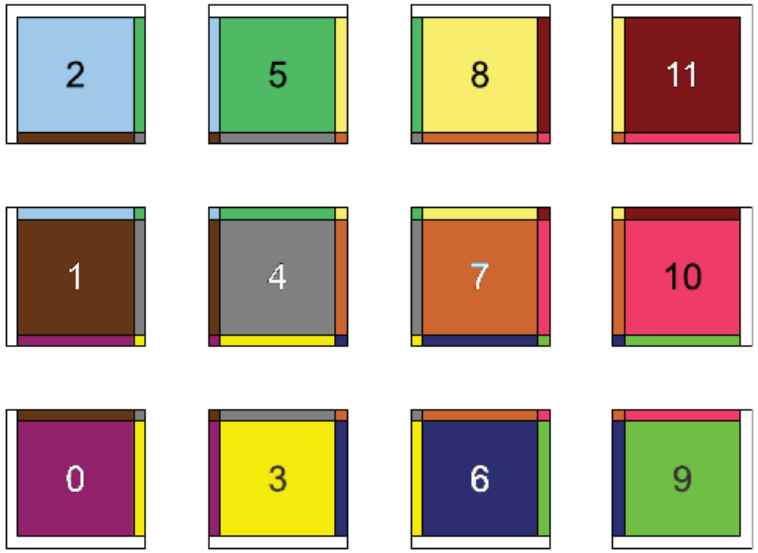


★ Parallelisation of SoFiA

- Limited memory → dissection of data into manageable chunks
- Limited time → speed-up of time-critical algorithms



- ★ Dissection of cube
  - Split into *m* sub-cubes with small overlap region around them:
    - Avoid slicing through sources
    - Buffer for filtering, e.g. smoothing
  - Westerlund & Harris 2014, PASA, 31, 23

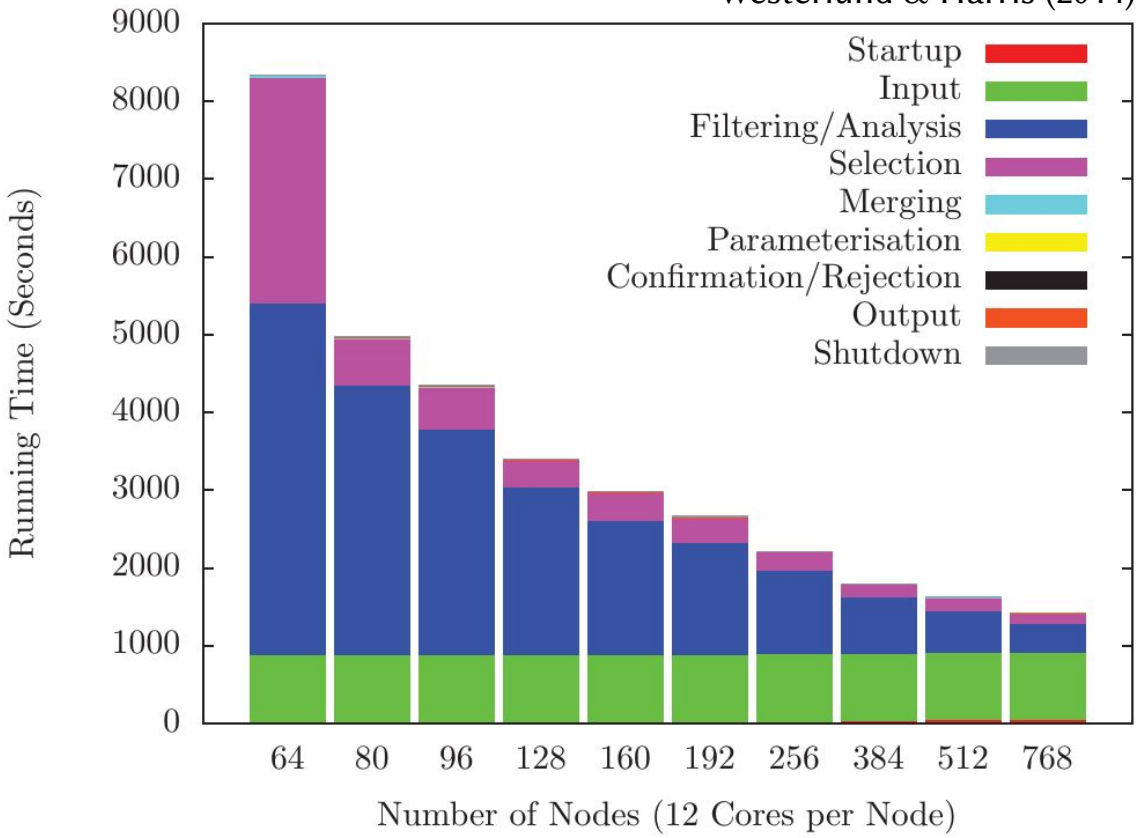


Westerlund & Harris (2014)

★ Dissection of cube

- Split into *m* sub-cubes with small overlap region around them:
  - Avoid slicing through sources
  - Buffer for filtering, e.g. smoothing
- Westerlund & Harris 2014, PASA, 31, 23

Westerlund & Harris (2014)



Era of "Big Data"

★ Parallelisation of time-critical algorithms

- CPU-based:
  - OpenMP — Compiler instructions
    Easy to use; no Python support
  - MPI — Message-passing library
    More complicated; available for Python
- GPU-based:
  - CUDA — Parallel computing platform
    Based on C++; for Nvidia GPUs
  - OpenCL — Parallel programming standard
    Based on C; for various devices

★ Speed-up of algorithms

- Optimisation of algorithms
- Conversion of Python modules to C / C++

Current Activities

- Collaboration with *Sarah Blyth* and *Michelle Kuttel* (UCT)
  - Parallelisation of SoFiA's S+C finder module
  - Honours project of *Jarred de Beer*
- Source finding busy week
  - 8 – 12 February 2016, ICRAR, Perth
  - Main tasks: parallelisation and reliability
- Volunteers welcome
  - Algorithm optimisation and parallelisation
  - Ideas for new techniques

SoFiA Download and Information

- For more information
  - SoFiA website on GitHub:
    - https://github.com/SoFiA-Admin/SoFiA/
  - SoFiA tutorial:
    - https://github.com/SoFiA-Admin/SoFiA/wiki/SoFiA-Tutorial
  - SoFiA paper:
    - Serra, Westmeier, Giese, et al., 2015, MNRAS, 448, 1922
  - SoFiA mailing list:
    - sofia-request@atnf.csiro.au with "subscribe" in the e-mail body (subject will be ignored)