# Accelerating automated extraction of radio astronomical sources from observation data with GPU accelerators

by Jarred de Beer

## 1. ABSTRACT

Large scale radio astronomy surveys, such as the Australian Square Kilometre Array Pathfinder (ASKAP) and Meer Karoo Array Telescope (MeerKAT), will generate petabytes of data that will need to be consumed at a rate equal to or faster than what is produced. This rate prohibits the traditional, manual approach to analysing data for HI emissions and new automated solutions need to be developed. Efforts have been made in developing software packages to generalise the source finding process, such as SoFiA, and frameworks for parallelizing source finding algorithms, such as SSoFF. While these packages have been successful in speeding up the source finding process, they are still CPU bound, and an opportunity exists to further accelerate these algorithms by porting them onto the GPU. General purpose GPU processing, in particular cases, has proven itself to be orders of magnitude faster and more power efficient than CPU implementations. An attempt at porting the Gaussian source finder onto the GPU, called the Parallel Gaussian source finder, has achieved speedups of 10x that of a four-threaded CPU implementation. In this document we compare existing software packages and source finding algorithms which have been implemented on the CPU and results from implementing the Parallel Gaussian source finder, K-Means and matrix transposition on the GPU.

## 2. INTRODUCTION

Radio astronomy surveys such as ASKAP and MeerKAT, are in development. 'Meer' in the codename MeerKAT is Afrikaans for 'more' and indicates that more telescopes will be used than have been attempted in the past. These surveys will generate Petabytes of data needing to be consumed at a rate which matches its production. The MeerKAT telescopes are located in a radio-quiet reserve in the Karoo, and will be powered by an RFI-silent (Radio Frequency Interference) grid with backup power [Jonas 2009]. Efficient use of power is necessary in order for the project to scale and data processing should strive to be energy efficient. Traditionally, detection of HI spectral emissions in radio surveys has been conducted manually, but this is no longer possible with the increased size of the ASKAP and MeerKAT surveys and automated solutions need to be developed.

HI spectral emissions are detected from large bodies of HI particles scattered throughout the universe which form spiral galaxies and other celestial bodies. These studies help scientists to understand the cosmic neutral gas density of the universe and its evolution, as well as galaxy evolution, over time [Holwerda and Blyth 2010].

The data is usually stored in the 3D Flexible Image Transport System (FITS), however [Badenhorst 2014] notes that the FITS format is reaching data size limits and may be problematic in the future, but is sufficient for the ASKAP and MeerKAT surveys which should generate spectral datacube sizes of up to 2.5 Terabytes. The 3D data is represented by two spatial dimensions which map to coordinates in the sky and one spectral dimension which maps to the spectral frequency of HI detections.

The data contains a considerable amount of RFI noise which makes the automated detection of these source emissions difficult, and various algorithms have been developed to achieve high levels of completeness and reliability [Popping et al. 2012]. In addition, software packages such as SoFiA [Serra et al. 2015] have been developed to

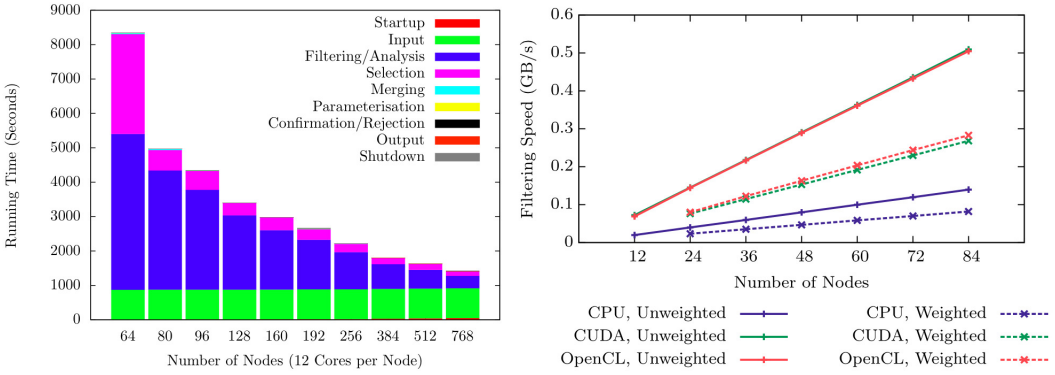


Fig. 1. Acceleration of the parallel Gaussian source finder. The diagram on the left are the results shown in Fig.6 from [Westerlund and Harris 2014] showing the successful speedup from multiple compute nodes with the CPU implementation. The figure to the right are results taken from Fig.12 of [Westerlund and Harris 2015] which compares the speedup of PGSF on the GPU and the CPU

allow the user to choose which source finding algorithm to run on their dataset and SSoFF [Westerlund and Harris 2014], has been developed to parallelize and speed up its processing.

Most of the software packages and algorithms have been written for the CPU, and efforts to parallelize source finding algorithms, such as in the parallel implementation of the Gaussian source finder (PGSF) with SSoFF, have been successful. However, additional effort has been made to implement PGSF on the GPU, and results show significant speedups over CPU implementations, see fig1. Similar results of GPU and CPU comparisons have been seen in other studies, such as in [Huang et al. 2009] which obtained 25x speedups in processing time and 9x less energy consumption on the GPU than multi-threaded CPU implementations.

The favorable performance and energy consumption of GPU processing makes it an attractive technology for use in radio astronomy surveys. In this document we compare the various software packages and source finding algorithms and consider efforts in implementing such algorithms on the GPU. Our intention is to later identify an algorithm to be implemented in CUDA (Compute Unified Device Architecture) for GPU processing on Nvidia graphics cards.

## 3. SOFTWARE PACKAGES

In this section two CPU bound software packages are discussed, Sofia [Serra et al. 2015] and the Scalable Source Finding Framework (SSoFF) [Westerlund and Harris 2014]. SoFiA has been written for general purpose source finding and allows the user to choose the most appropriate source finding algorithm to use on their data. SSoFF is a framework which can be used to distribute the workload of source finding algorithms among a grid based cluster of machines. Separate effort is required to implement each source finding algorithm using SSoFF.

We also note the existence of the Source Finder Accuracy Evaluator (SFAE) [Westerlund et al. 2012] which provides an automated and deterministic method for determining the accuracy of a source finder, and can be used independently of the source finders implementation.

### 3.1. SoFiA

SoFiA has been developed in preparation for various HI surveys which are going to be carried out at ASKAP. These surveys are: WALLABY, a wide survey covering 3/4

of the sky at z-0.25; DINGO, a deep survey reaching z-0.4; and APERTIF [Serra et al. 2015]. SoFiA has been designed to be modular and allows algorithms to be switched out as needed to accommodate the different ranges of datasets which these surveys will produce.

Key advantages as noted in [Serra et al. 2015] is that SoFiA can search for emissions on multiple scales while also considering variations in noise level. The package is publicly available and can be downloaded on Github.

The pipeline employed by SoFiA is illustrated in a flowchart which we describe here as consisting of four stages. In the first stage data is input and modified according to flags or weights. In the second stage filters are applied to reduce noise, preparing the data for the third stage, source detection, which employs various source detection algorithms. In the fourth stage the sources are merged and parameterised, and in the fifth stage the results are output.

At the time of writing, SoFiA has implemented two filtering and three source finding algorithms which can be chosen by the user. The two filtering algorithms implemented are a convolution with a 3D kernel, and 2D-1D wavelet de-noising [Flöer and Winkel 2012]. Source finding algorithms include Simple threshold, Smooth and Clip (S+C), and Characterised noise HI (CNHI). We detail these algorithms in the following section, while a summary can also be found in [Koribalski 2012].

A merging process needs to be run on the detected sources so that those clustered close enough together will be considered as a single source. SoFiA does not take into account the size of sources and merging needs to be parameterised by the user. It is noted that source finders do exist which take into account individual sizes of sources, namely Clumpfield [Williams et al. 1994] and BLOBCAT [Hales et al. 2012].

All of the source finding algorithms return a binary mask containing the detected sources. This binary mask is then used in identifying sources, and for this purpose SoFiA uses the Lutz one pass algorithm [Lutz 1980] by [Jurek 2012], implemented in C++.

### 3.2. Scalable Source Finding Framework (SSoFF)

SSoFF [Westerlund and Harris 2014] is used to build distributed source finding algorithms which can be used in an HPC environment. The framework makes use of MPI, MPI-IO and OpenMP libraries to handle communication between processes and machines. The framework takes advantage of the fact that the datacubes can be broken down into smaller sub cubes and processed in parallel by source finding algorithms, with minimal dependencies.

Many of the algorithms apply statistical methods which sample neighbouring voxels, meaning that voxels near the edges of a cube will need to access the neighbouring cube. To avoid this dependency the voxels need to be duplicated and included with the neighbouring cube. This duplicated data is known as Halo data and we need to ensure that the amount of halo data relative to sub cube size is minimised for optimal use of network transfer and memory storage. The minimum amount of halo data is dependent on the radius of sampling performed by the algorithms.

SSoFF distributes the sub cubes, with their halo data, to a 3D grid of processes. This type of grid based parallelism with shared data is very similar to the grid based multithreaded nature of graphics processing used in GPU programming. SSoFF does not appear to make use of GPU processing, but instead uses OpenMP for multithreaded CPU processing. [Westerlund and Harris 2014] notes that significant effort is needed to implement each individual source finding algorithm, and this would be the same with a CUDA implementation.

SSoFF also provides functionality for multi-threaded selection and merging tasks. Both of these make use of a flood fill algorithm, which is performed in a procedure of

steps to accommodate multiple processes which may each contain a piece of the same split source.

### 3.3. Source Finder Accuracy Evaluator (SFAE)

SSoFF was used to implement the Parallel Gaussian Source Finder (PGSF) and the catalogues which were detected from the test data were matched against sources in the Source Finder Accuracy Evaluator, which has been mentioned here because of its possible usefulness in verifying the accuracy of source finding algorithms.

## 4. SOURCE FINDING ALGORITHMS

In this section we list various aspects of existing source finding algorithms. We note how the performance of a source finding algorithm is determined and how they compare to one another.

There are two types of algorithms mentioned in [Jurek 2012], one type of algorithm is based on intensity thresholding, such as DUCHAMP, in which filtering is performed on each voxel by comparing its value against an absolute threshold. Another type of algorithm, such as CNHI, assumes that sources are more likely to have contiguous voxel values along the spectral dimension. [Jurek 2012] also notes an inherent limitation to intensity thresholding algorithms where sources become harder to detect in datasets with a higher resolution. This is due to a decreased contribution of total flux at each voxel because the source is distributed into a larger number of voxels in the higher resolution dataset. Despite this supposed limitation [Popping et al. 2012] reports that DUCHAMP achieved reliability ratings of 99.7% and 72.2% for point source datasets and 60.8% and 69.9% for model galaxies datasets, which is a lot better than CNHI's reliability results of 8.3% and 34.7% for point source and 4.1% and 40.1% for model galaxies datasets. We are not aware from the comparisons in [Popping et al. 2012] what role this limitation has on the results.

[Popping et al. 2012] describes completeness and reliability as two measures of performance in a source finding algorithm. Completeness is defined as the number of detected sources divided by the total number of sources in a datacube. Reliability is defined as the number of true detections divided by the total number of detections. The higher the completeness and reliability ratings are, the lower the rate of false detections. Reliability can potentially be misleading as it can be influenced by the size of the datacube. For example, if the size of a datacube is doubled but the number of true sources remain constant, then the false positive rate will increase because of the additional noise, decreasing the reliability value. The completeness value, however, would not change.

The better the signal-to-noise ratio in the dataset, the better reliability and completeness the source finding algorithms will have. For this reason most source finders, and even the SoFiA package, have a preprocessing step which smooths the noise. [Badenhorst 2014] notes that this decreases the number of false detections and also the number of missed sources.

### 4.1. DUCHAMP source finder

DUCHAMP [Whiting 2012] is an Intensity threshold source finder. Various efforts to accelerate the algorithm with regards to multi-threading, SIMD, memory management have been studied in [Badenhorst 2014]. The Selavy project [Whiting and Humphreys 2012] is also a parallelized implementation of DUCHAMP.

[Badenhorst 2014] notes that the combination of SIMD and CPU multi-threading has terrible performance. However, in the context of GPU processing, [Mahesri et al. 2008] finds that MIMD has superior performance to SIMD. We note that this may be useful in the context of CUDA processing.

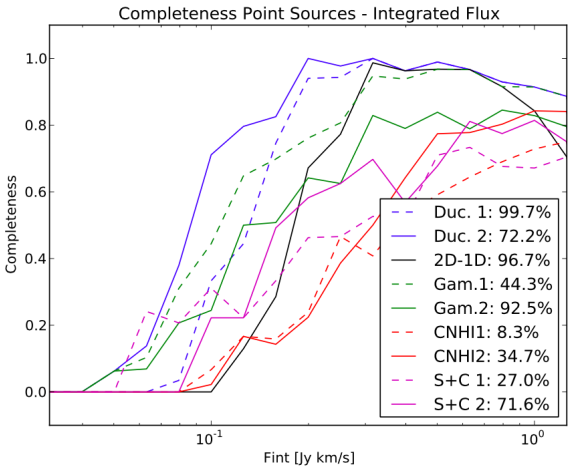


Fig. 2. Figure taken from [Popping et al. 2012] showing the completeness of source finders as a function of integrated flux. We note the table showing the reliability of the source finding algorithms

Results from tests in [Popping et al. 2012], see fig.2, utilising a datacube of point sources show that DUCHAMP has the highest completeness when considered as a function of integrated flux, integrated signal to noise, peak flux, and velocity width. DUCHAMP obtained reliability ratings of 99.7% and 72.2% for the point source datacube test, and reliability ratings of 60.8% and 69.9% for the model galaxies datacube.

### 4.2. CNHI source finder

The Characterised noise HI source finder (CNHI) [Jurek 2012] is an alternative to intensity based source finders which uses a method for automatic detection of sources called matched filtering. It implements the Lutz one-pass algorithm, with a compressed, sparse representation.

[Jurek 2012] lists two core concepts behind the CNHI algorithm. The first is that a datacube is treated as a bundle of HI spectra, as opposed to a collection of voxels. Secondly, contiguous blocks of voxels are tested for sources. Sources are detected by regions which do not appear to be noise, the inverse approach to shape based algorithms.

Noise is detected with comparative statistical tests using the Kolmogorov-Smirnov test [Kendall and Stuart 1979] and the Kuiper test [Kuiper 1960].

CNHI's performance with respect to completeness is presented in [Popping et al. 2012]. In these tests CNHI performed with the worst reliability rating of the compared source finders on both point source and model galaxies datacubes, with reliability ratings of 8.3% and 34.7%, and 4.1% and 40.1% respectively.

### 4.3. The Gamma-Finder

The Gamma-Finder [Boyce 2003] estimates noise variance, known as the Gamma Statistic. A signal-to-noise ratio is then used to clip the data. There are no parameters to the Gamma-Finder and it does not output a binary mask, which makes it problematic for packages such as SoFiA.

### 4.4. 2D-1D Wavelet Reconstruction finder

2D-1D Wavelet Reconstruction [Flöer and Winkel 2012] takes into account that the spectral dimension is not isomorphic to either of the two positional dimensions. The shape of a source along the spectral dimension is therefore different. The algorithm

performs a 2D wavelet transform in all planes of the cube and a 1D wavelet transform at each pixel. The wavelet coefficients from these transformation steps are then thresholded and the coefficients which result from this are noise free.

### 4.5. Smooth plus Clip source finder

This algorithm optimises the signal-to-noise ratio of objects in a datacube. It looks for sources in both the original and the smoothed cube. Smoothing can be done on the sky, velocity, or all three axes. [Serra et al. 2012]

## 5. GPU ACCELERATION

### 5.1. GPU-accelerated Filter-based source finder

In [Westerlund and Harris 2014] PGSF was built on SSoFF and parallelized on multiple nodes using the CPU. It was later implemented in [Westerlund and Harris 2015] for GPU processing, in both OpenCL and CUDA.

[Westerlund and Harris 2015] describes the sequence of the GPU processing in which small portions of the data cubes are processed by reading data onto the GPU, running the filter and writing back the results. The largest portion of the data is along the frequency axis and is kept coalesced in memory, and a single thread can be assigned to each line of frequency and processed independently from the others.

The kernel begins by mapping the thread's grid coordinates to its voxel in the datacube, then the line of frequency values for that voxel is read from memory, after which it is blocked by a barrier. Each thread then performs filtering on the line and stores the result to local memory before hitting another barrier, after which the results are output. Filter data is stored in constant memory so multiple threads can read values from the same voxel without bank conflicts. Multiple kernel executions are used to avoid the watchdog timer from timing out.

[Westerlund and Harris 2015] reports that the speedup for OpenCL and CUDA was similar, between 1.8 and 2.1 times faster than the CPU implementation, but with OpenCL slightly outperforming CUDA. This was a little surprising, as [Karimi et al. 2010] reports that CUDA noticeably out performs OpenCL in all problem sizes, with OpenCL having end-to-end times 16% to 67% slower than CUDA. [Karimi et al. 2010] notes that CUDA has better performance in transferring data to and from the GPU, and it is possible the data set being used in [Westerlund and Harris 2015] was too small for this to have an affect.

[Westerlund and Harris 2015] details that the statistics functions have not been implemented on the GPU, and now take up a noticeable portion of time the overall runtime.

### 5.2. K-Means

K-Means is an algorithm which is used to find clusters of points. It is iterative and has a guarantee of converging. [Wu et al. 2009] implements the K-Means algorithm on the GPU and achieves a 35x speedup over a four-threaded CPU implementation. The test was performed with a data set fitting on device memory and performance may take a hit when data is too large to fit onto the device.

This type of algorithm is used during the merging process of source finding. It involves transferring a block of data from the CPU onto the GPU, transposing it into a column based layout, computing the cluster for each point, and transferring the results back to CPU. A streaming kernel can be used to asynchronously process multiple blocks at once, allowing for overlap between memory transfer and computation.

[Wu et al. 2009] notes that Texture memory is sometimes slower than global memory in cases where the centroids array is large enough to cause cache misses in texture

memory, and that it can be hard to tell which is better to use. If the centroids array fits into GPU constant memory then 10x speedups were obtained on the GPU over the 8-core CPU implementation.

### 5.3. Matrix Transposition

[Sung et al. 2014] discusses an algorithm for in-place matrix transposition implemented on GPUs. DUCHAMP uses transposition to keep the memory accesses coherent in the de-noising process and we note that it may also be useful, if needed, in CNHI to obtain the contiguous blocks of voxels used along the spectral axis.

Matrix transposition is useful when transforming row-based matrices into column-based matrices for contiguous memory access along columns, and vice versa.

## 6. DISCUSSIONS

SoFiA is a software package which is intended to process both wide and deep surveys, and it allows the user to choose the source finding algorithm most appropriate for their data. Its modular structure allows source finding algorithms to be added, and it comes with CNHI and S+C already integrated. As seen in fig.2 CNHI and S+C give the worst reliability results, and it is interesting that those two were chosen. It may be the case that CNHI and S+C are favorable for SoFiA in that they require fewer parameters from the user and are more consistent between different datasets. SSoFF, on the other hand is a framework for implementing source finding algorithms which are then distributed among compute nodes. It's possible that SSoFF based source finders could be used by SoFiA. This would allow the user to specify which source finder to use in a distributed environment. Significant effort is required to implement each algorithm with SSoFF.

As mentioned, fig.22 shows that the reliability of DUCHAMP is considerably superior to CNHI. Yet DUCHAMP is an intensity thresholding algorithm which [Jurek 2012] reports becomes a limiting factor in high resolution surveys, a limitation which CNHI is not subjected to. It is unclear whether this limitation is purely theoretical, or whether the surveys just weren't of a high enough resolution for this to have an affect. It is also unclear at what resolutions this limitation might start having an affect.

CNHI treats datacubes as bundles of HI spectra. This is to take advantage of the distorted shape of sources along the spectral dimension in which the voxels are contiguous. It's possible to accelerate this technique in a multi-threaded or GPU environment by assigning each thread to a voxel in a 2D spatial plane, the spectral lines along each of these points are independent and can be processed in parallel. A deeper look into the Kolmogorov-Smirnov test [Kendall and Stuart 1979] and the Kuiper test [Kuiper 1960] would assist in further understanding how this process can be parallelized. CNHI is unique in its approach of characterising noise and detecting sources by that which is not characterised noise. Techniques for parallelizing this algorithm may as a result also be unique for CNHI, and may be subject to different degrees of acceleration.

In the GPU implementation of PGSF we note that OpenCL performed similarly, but outperformed, CUDA. However, [Karimi et al. 2010] tested OpenCL and CUDA on a multitude of data set sizes and found that OpenCL performed 16% to 67% slower than CUDA. This test was performed in 2010, while the PGSF implementation in [Westerlund and Harris 2015] was conducted in 2015. Either OpenCL's performance has developed at a faster rate than CUDA, which is unlikely, or the implementations in [Westerlund and Harris 2015] differed and the CUDA implementation was less efficient. The latter case is also unlikely, as CUDA and OpenCL have interchangeable APIs and implementing two separate versions would have been unnecessary work. [Karimi et al. 2010] notes that memory transfer to and from the device is faster with CUDA, and memory accesses on larger datacubes should be noticeably faster than

OpenCL. It might be the case that the datasets in [Westerlund and Harris 2015] were too small for this difference to have an affect.

## 7. CONCLUSIONS

Various source finding software packages have been developed to assist with the demands of larger radio astronomy surveys such as ASKAP and MeerKAT, and effort has been made to accelerate the processing of source finding algorithms used in these surveys. We reviewed and compared characteristics of various source finding algorithms as well as speedups achieved on the CPU and GPU with the PGSF source finder. The SoFiA package has implemented S+C and CNHI source finding algorithms. DUCHAMP has been implemented with CPU multi-threading on desktops [Badenhorst 2014] and also a grid of compute nodes such as in Selavy. SSoFF is a framework which can be used to distribute source finding algorithms among a grid based cluster of machines, and separate effort is needed in implementing each individual source finder. GPU processing, as shown in the GPU implementation of PGSF [Westerlund and Harris 2015], K-Means and matrix transposition, achieved 25x speedups in CPU processing and 9x less energy consumption [Huang et al. 2009].

## REFERENCES


Scott James Badenhorst. 2014. *Acceleration of the noise suppression component of the DUCHAMP sourcefinder*. Master's thesis. University of Cape Town.

Peter James Boyce. 2003. *GammaFinder: a java application to find galaxies in astronomical spectral line data cubes*. Ph.D. Dissertation. Citeseer.

Lars Flöer and Benjamin Winkel. 2012. 2d–1d wavelet reconstruction as a tool for source finding in spectroscopic imaging surveys. *Publications of the Astronomical Society of Australia* 29, 3 (2012), 244–250.

Christopher A Hales, Tara Murphy, James R Curran, Enno Middelberg, Bryan M Gaensler, and Ray P Norris. 2012. BLOBCAT: software to catalogue flood-filled blobs in radio images of total intensity and linear polarization. *Monthly Notices of the Royal Astronomical Society* 425, 2 (2012), 979–996.

Benne Willem Holwerda and Sarah-Louise Blyth. 2010. Trumpeting the Vuvuzela: UltraDeep HI observations with MeerKAT. *arXiv preprint arXiv:1007.4101* (2010).

Song Huang, Shucai Xiao, and Wu-chun Feng. 2009. On the energy efficiency of graphics processing units for scientific computing. In *Parallel & Distributed Processing, 2009. IPDPS 2009. IEEE International Symposium on*. IEEE, 1–8.

Justin L Jonas. 2009. MeerKATThe South African array with composite dishes and wide-band single pixel feeds. *Proc. IEEE* 97, 8 (2009), 1522–1530.

Russell Jurek. 2012. The Characterised Noise Hi source finder: Detecting Hi galaxies using a novel implementation of matched filtering. *Publications of the Astronomical Society of Australia* 29, 3 (2012), 251–261.

Kamran Karimi, Neil G Dickson, and Firas Hamze. 2010. A performance comparison of CUDA and OpenCL. *arXiv preprint arXiv:1005.2581* (2010).

M Kendall and A Stuart. 1979. The Advanced Theory of StatisticsVolume 2 Inference and Relation-ship, (London: Charles Griffin). *Kendall4The Advanced Theory of Statistics* 2 (1979).

Bärbel S Koribalski. 2012. Source Finding and Visualisation. *Publications of the Astronomical Society of Australia* 29, 3 (2012), 213–213.

Nicolaas H Kuiper. 1960. Tests concerning random points on a circle. In *Indagationes Mathematicae (Proceedings)*, Vol. 63. Elsevier, 38–47.

RK Lutz. 1980. An algorithm for the real time analysis of digitised images. *Comput. J.* 23, 3 (1980), 262–269.

Aqeel Mahesri, Daniel Johnson, Neal Crago, and Sanjay J Patel. 2008. Tradeoffs in designing accelerator architectures for visual computing. In *Proceedings of the 41st annual IEEE/ACM International Symposium on Microarchitecture*. IEEE Computer Society, 164–175.

Attila Popping, Russell Jurek, Tobias Westmeier, Paolo Serra, L Flöer, Martin Meyer, and Baerbel Koribalski. 2012. Comparison of potential ASKAP HI survey source finders. *Publications of the Astronomical Society of Australia* 29, 03 (2012), 318–339.

Paolo Serra, Tom Oosterloo, Raffaella Morganti, Katherine Alatalo, Leo Blitz, Maxime Bois, Frédéric Bournaud, Martin Bureau, Michele Cappellari, Alison F Crocker, and others. 2012. The ATLAS3D project–

XIII. Mass and morphology of H i in early-type galaxies as a function of environment. *Monthly Notices of the Royal Astronomical Society* 422, 3 (2012), 1835–1862.

Paolo Serra, Tobias Westmeier, Nadine Giese, Russell Jurek, Lars Flöer, Attila Popping, Benjamin Winkel, Thijs van der Hulst, Martin Meyer, Bärbel S Koribalski, and others. 2015. SoFiA: a flexible source finder for 3D spectral line data. *arXiv preprint arXiv:1501.03906* (2015).

I-Jui Sung, Juan Gómez-Luna, José María González-Linares, Nicolás Guil, and Wen-Mei W Hwu. 2014. In-place transposition of rectangular matrices on accelerators. In *Proceedings of the 19th ACM SIGPLAN symposium on Principles and practice of parallel programming*. ACM, 207–218.

Stefan Westerlund and Christopher Harris. 2014. A Framework for HI Spectral Source Finding Using Distributed-Memory Supercomputing. *Publications of the Astronomical Society of Australia* 31 (2014), e023.

Stefan Westerlund and Christopher Harris. 2015. Performance analysis of GPU-accelerated filter-based source finding for HI spectral line image data. *Experimental Astronomy* 39, 1 (2015), 95–117.

Stefan Westerlund, Christopher Harris, and Tobias Westmeier. 2012. Assessing the Accuracy of Radio Astronomy Source-Finding Algorithms. *Publications of the Astronomical Society of Australia* 29, 3 (2012), 301–308.

Matthew Whiting and Ben Humphreys. 2012. Source-finding for the australian square kilometre array pathfinder. *Publications of the Astronomical Society of Australia* 29, 3 (2012), 371–381.

Matthew T Whiting. 2012. duchamp: a 3D source finder for spectral-line data. *Monthly Notices of the Royal Astronomical Society* 421, 4 (2012), 3242–3256.

Jonathan P Williams, Eugene J De Geus, and Leo Blitz. 1994. Determining structure in molecular clouds. *The Astrophysical Journal* 428 (1994), 693–712.

Ren Wu, Bin Zhang, and Meichun Hsu. 2009. Clustering billions of data points using GPUs. In *Proceedings of the combined workshops on UnConventional high performance computing workshop plus memory access workshop*. ACM, 1–6.