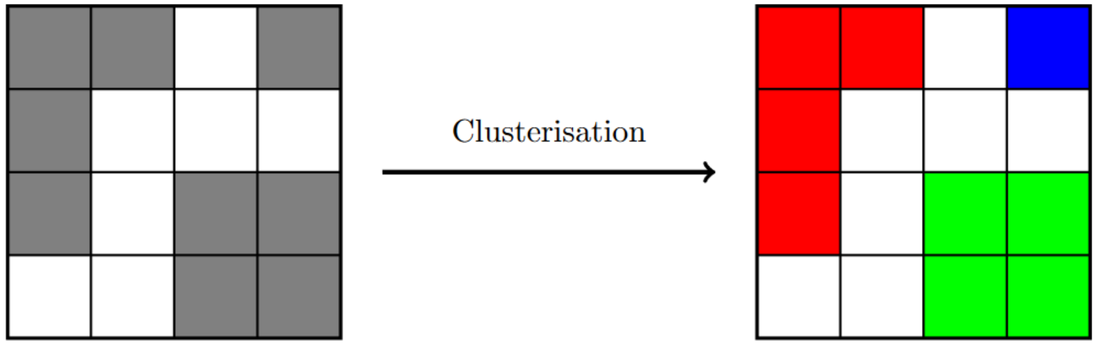
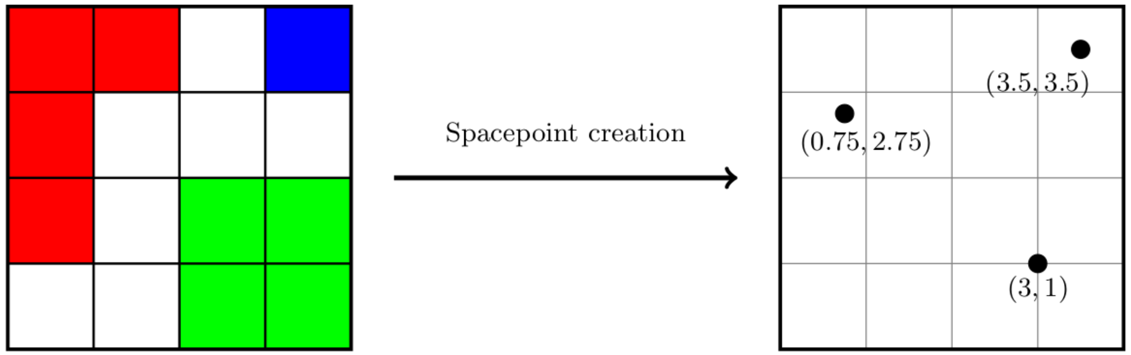
Justus Rudolph (MPhys Project)

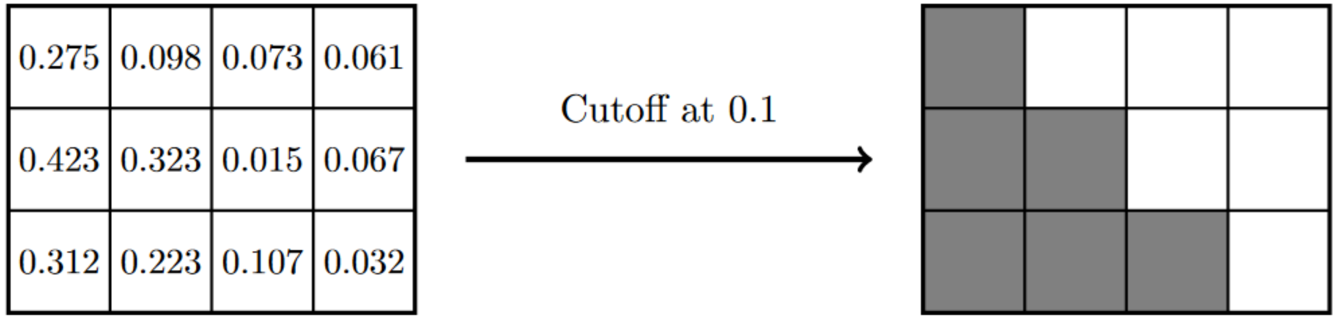
Supervised by Dr. Ben Wynne

ATLAS Phase 2 Upgrade: Cell-wise parallelisation of the clusterisation algorithm in traccc

Clusterisation



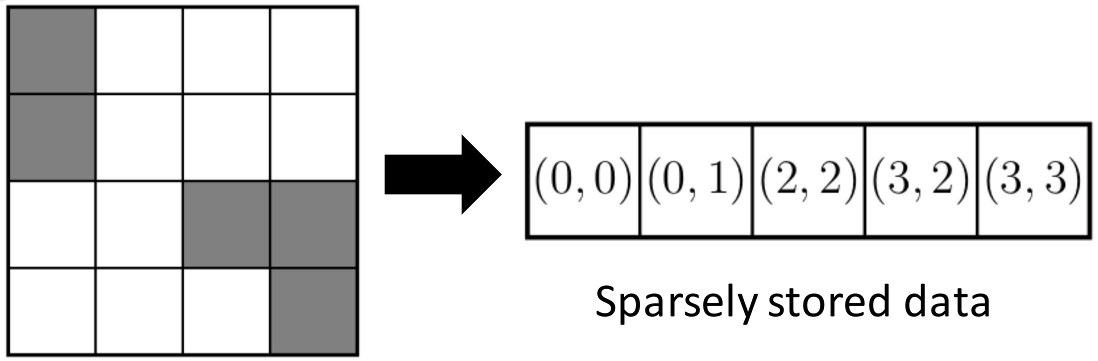




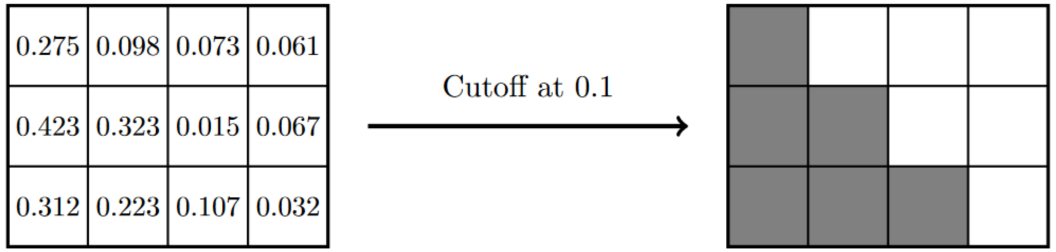
0.275	0.098	0.073	0.061
0.423	0.323	0.015	0.067
0.312	0.223	0.107	0.032

The traccc project – Parallelising Tracking

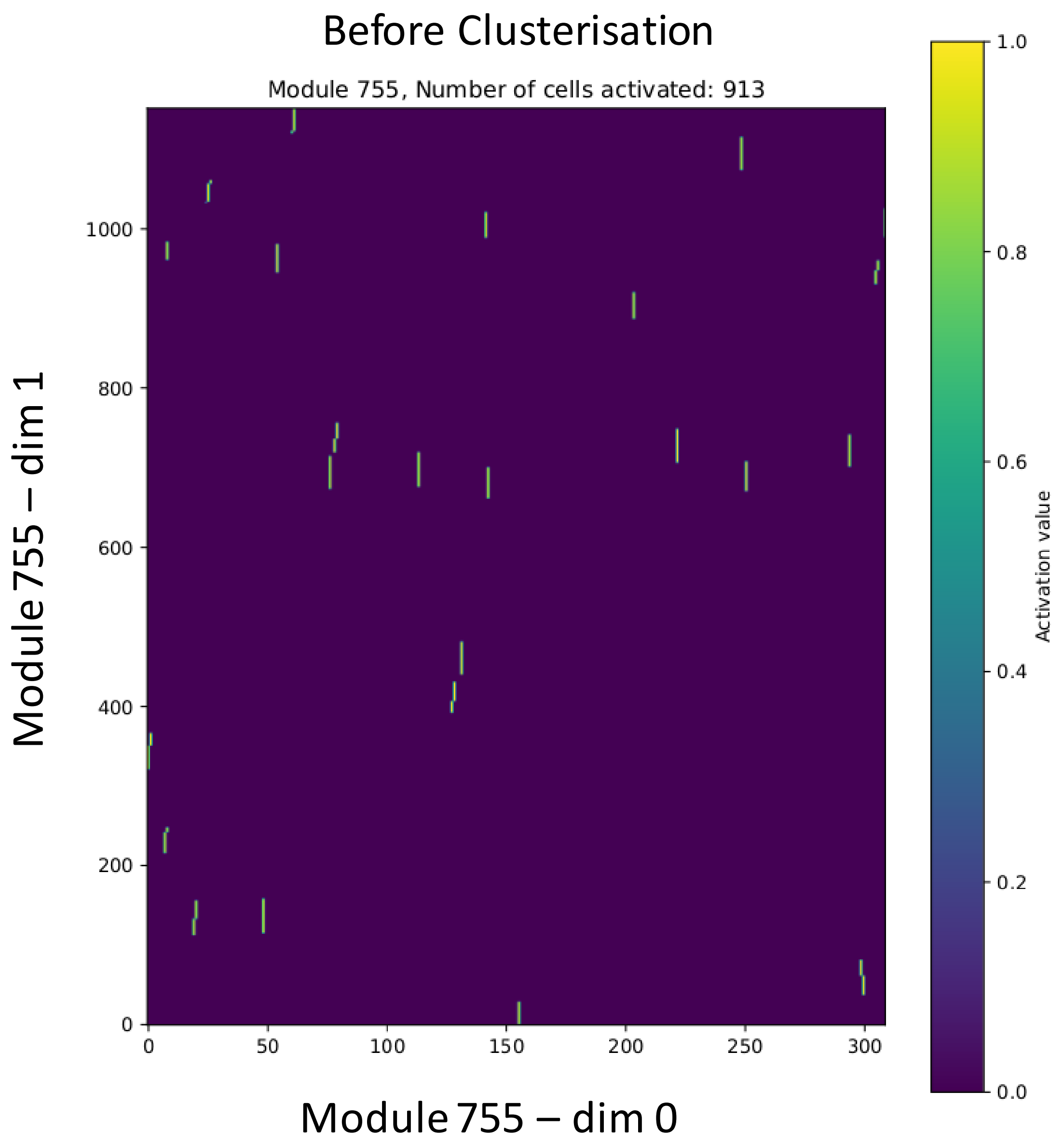
- Some GPU Terminology:
  - **G**raphics **P**rocessing **U**nit
  - Optimised for SIMD: Single Instruction, Multiple Data
  - The whole GPU is called a ***grid***, which is made up out of ***block***s.
    - Each instance is called a ***thread***, keyword: "multithreading"
    - ***Block***s contain ***thread***s
      - Optimised to be a multiple of 32
      - User specifiable
      - Inter-block communication is slow
      - Intra-block communication is fast
  - e.g. Change activation values to binary "Activated"/"Not activated"
  - Cell data in *traccc* is stored sparsely

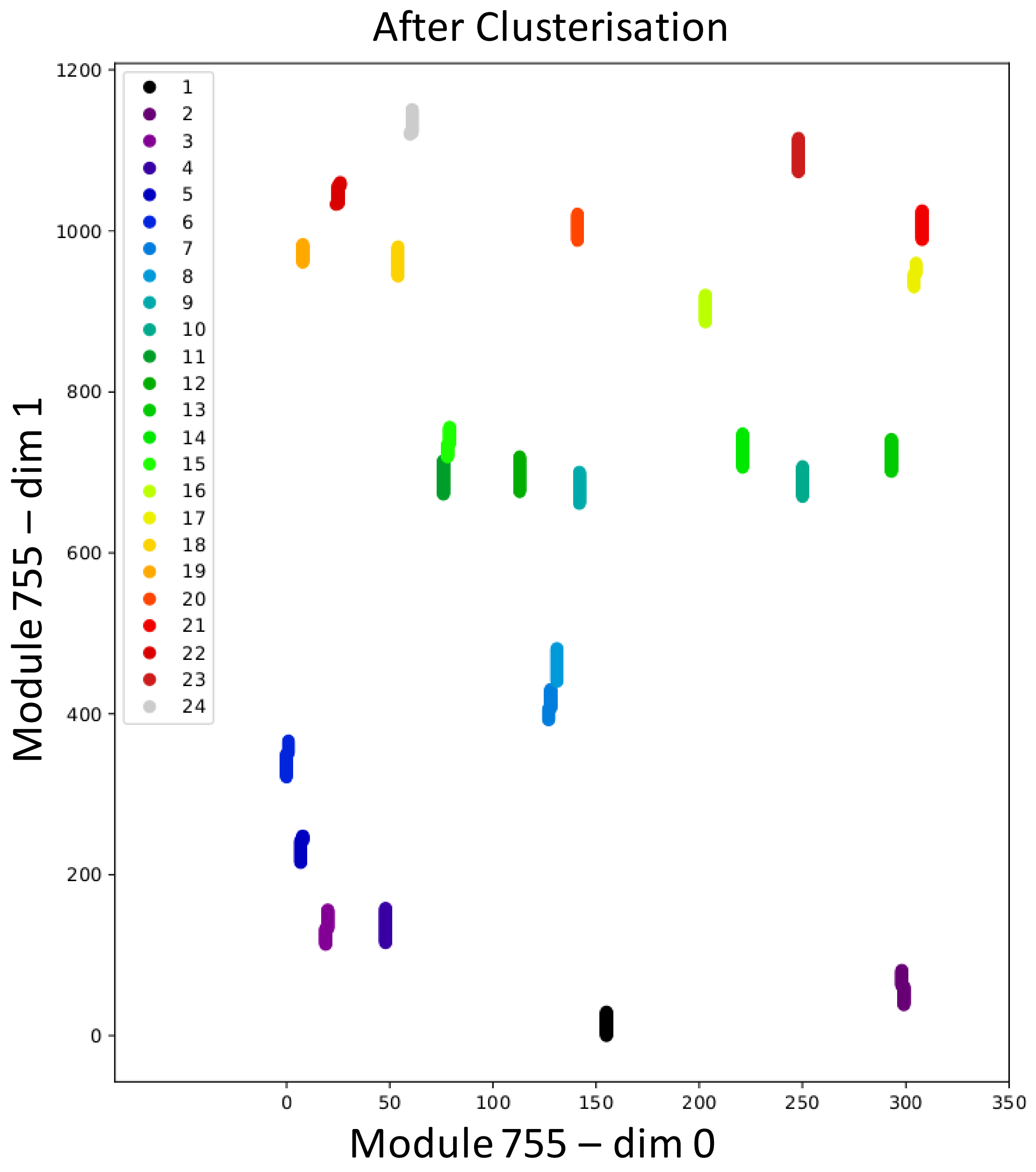


Sparsely stored data



Example from Simulation in *traccc* – Figures

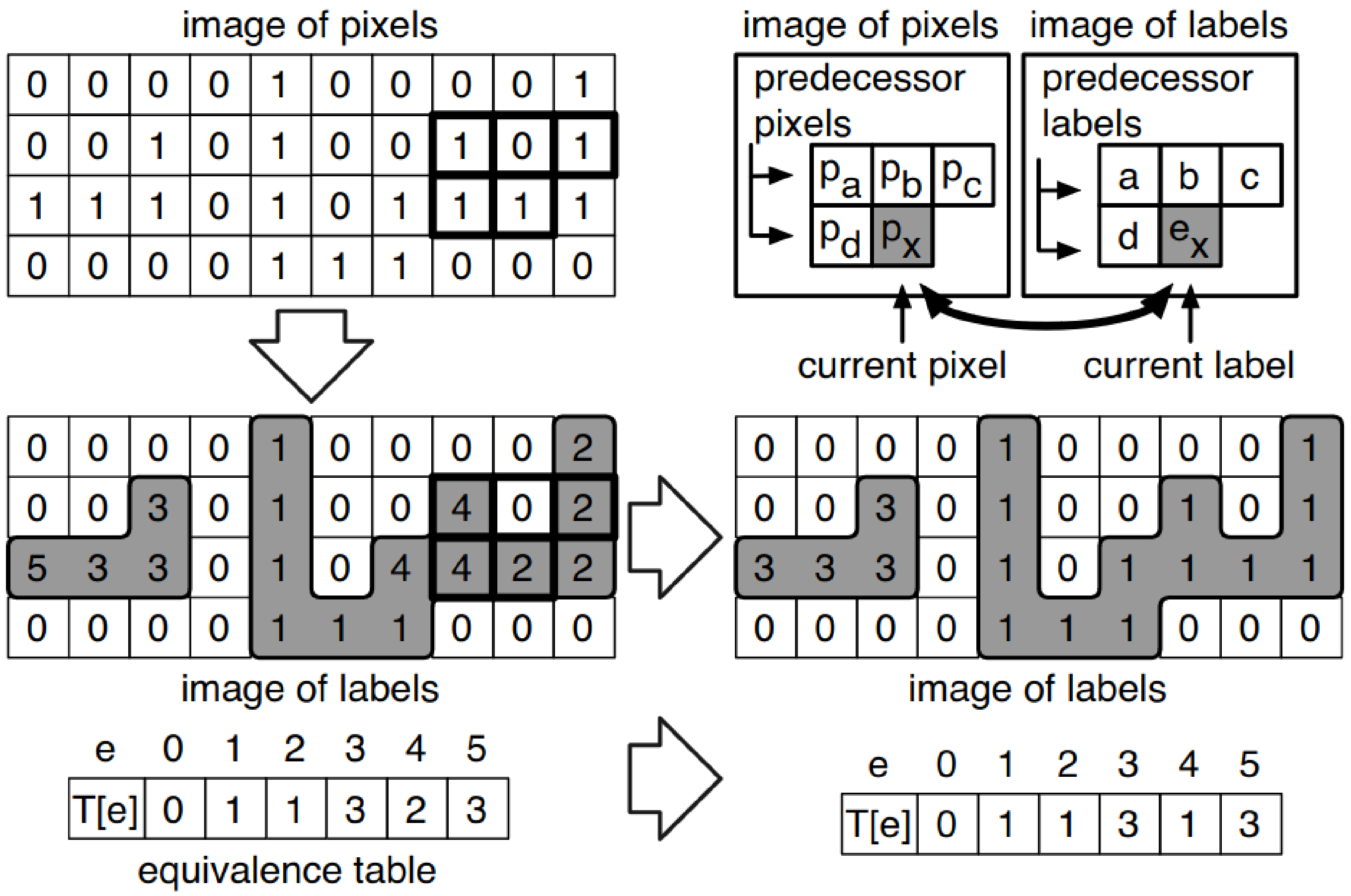




Sparse CCL

Modularly parallelised

- Less memory access required
- Two steps, create lookup table for cluster merging

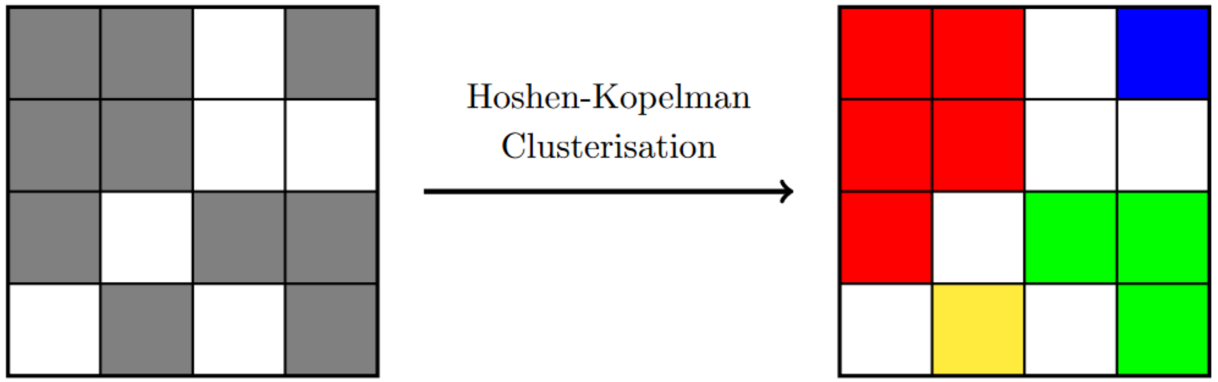


[1]

Hoshen-Kopelman

Every cell belongs to the same cluster as all its nearest neighbours [2]

For each cell:
Check above and left only, no diagonals.

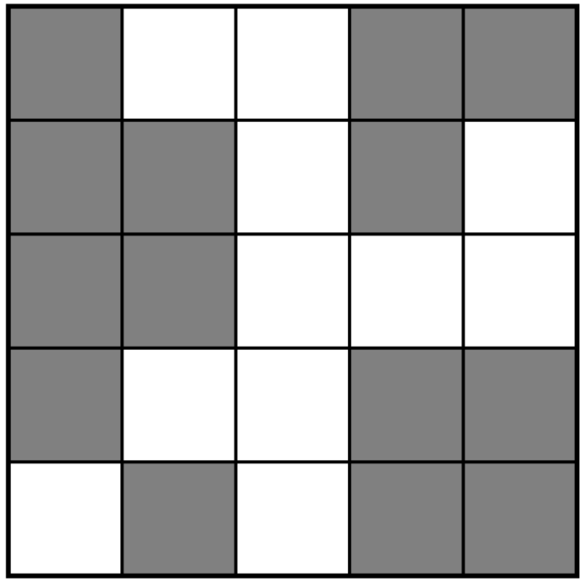


- Extended version:
- Inaccuracies with missing long stripes

➡ Include diagonals

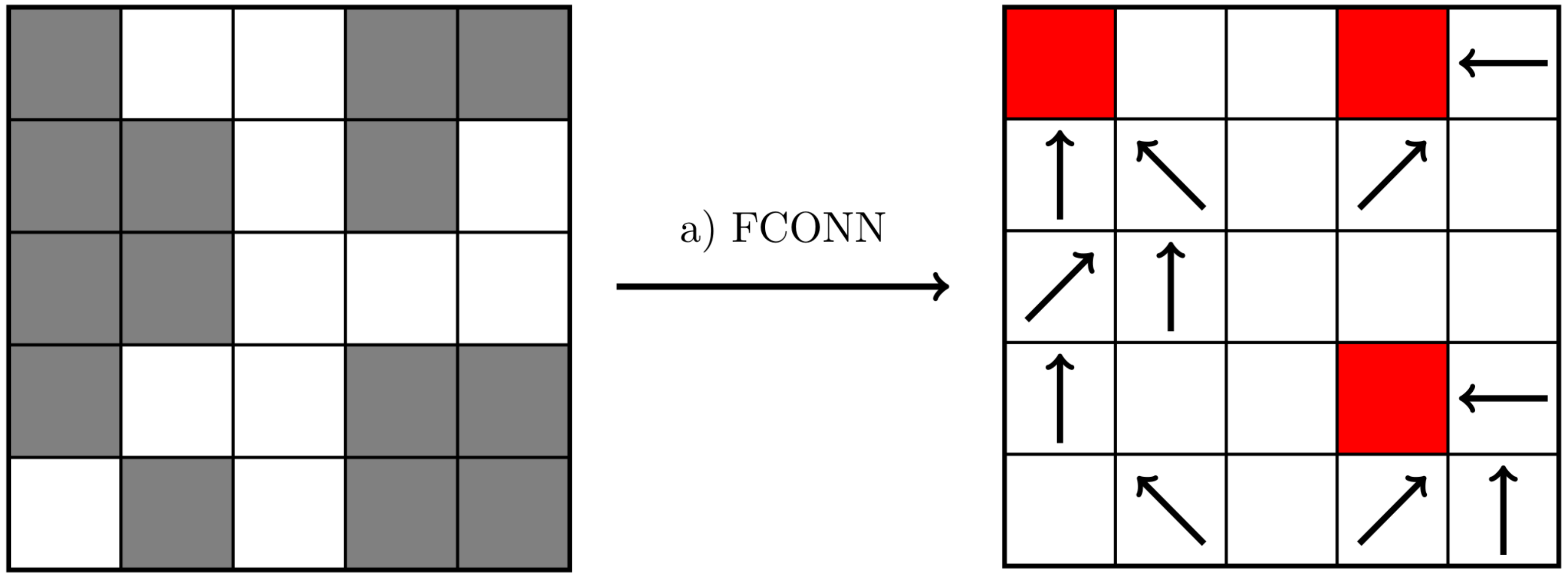
  - Clusterisation accuracy jumps from ≈85 to ≈98.5%

FCONN: Parallelised Hoshen-Kopelman



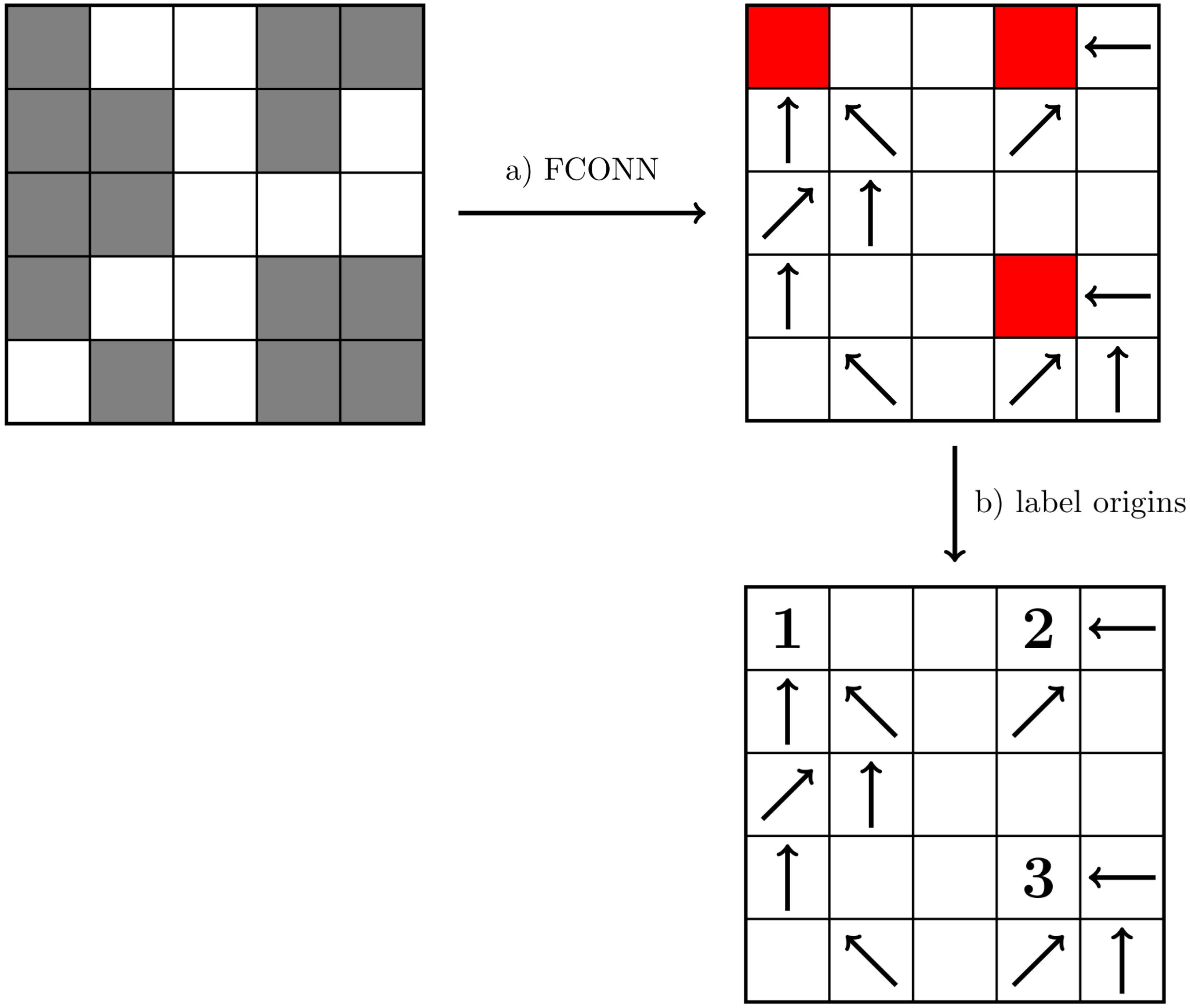
FCONN: Parallelised Hoshen-Kopelman

Step 1



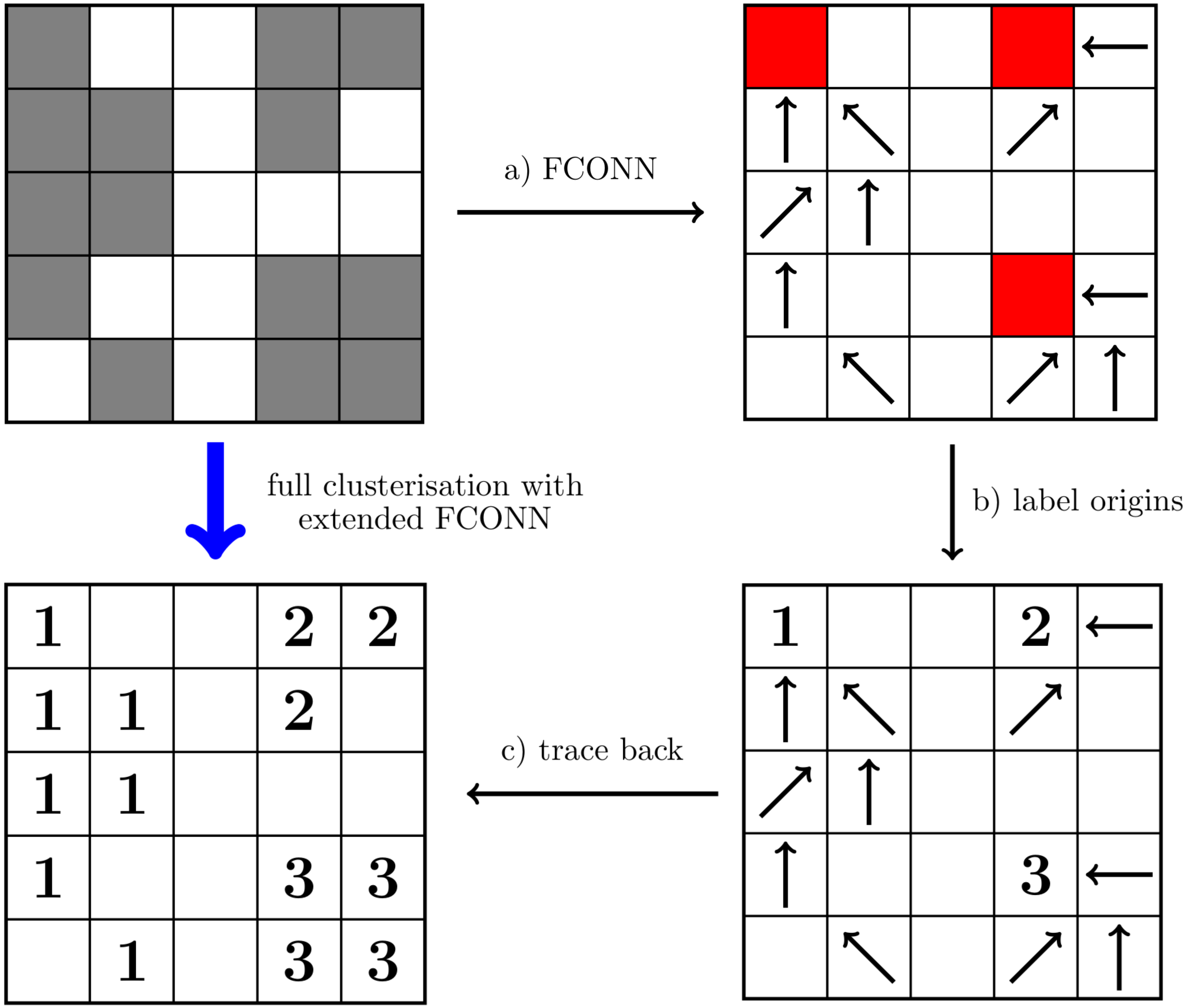
FCONN: Parallelised Hoshen-Kopelman

Step 2



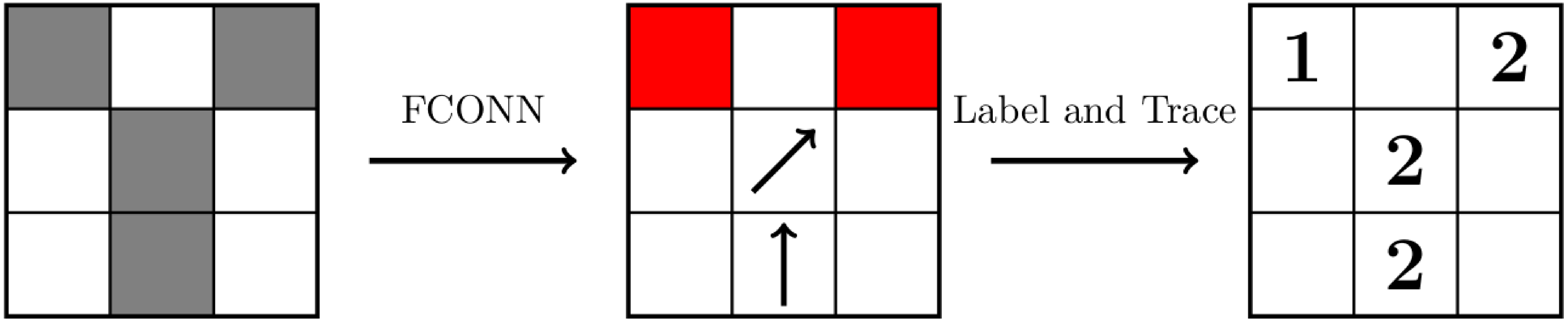
FCONN: Parallelised Hoshen-Kopelman

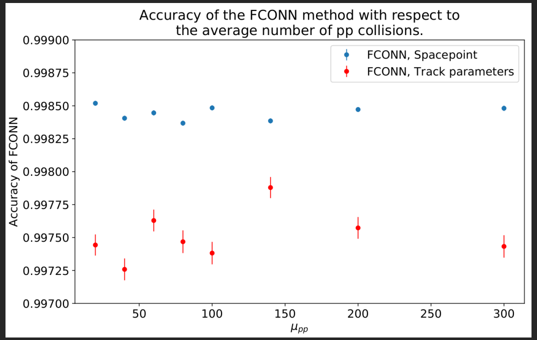
Step 3

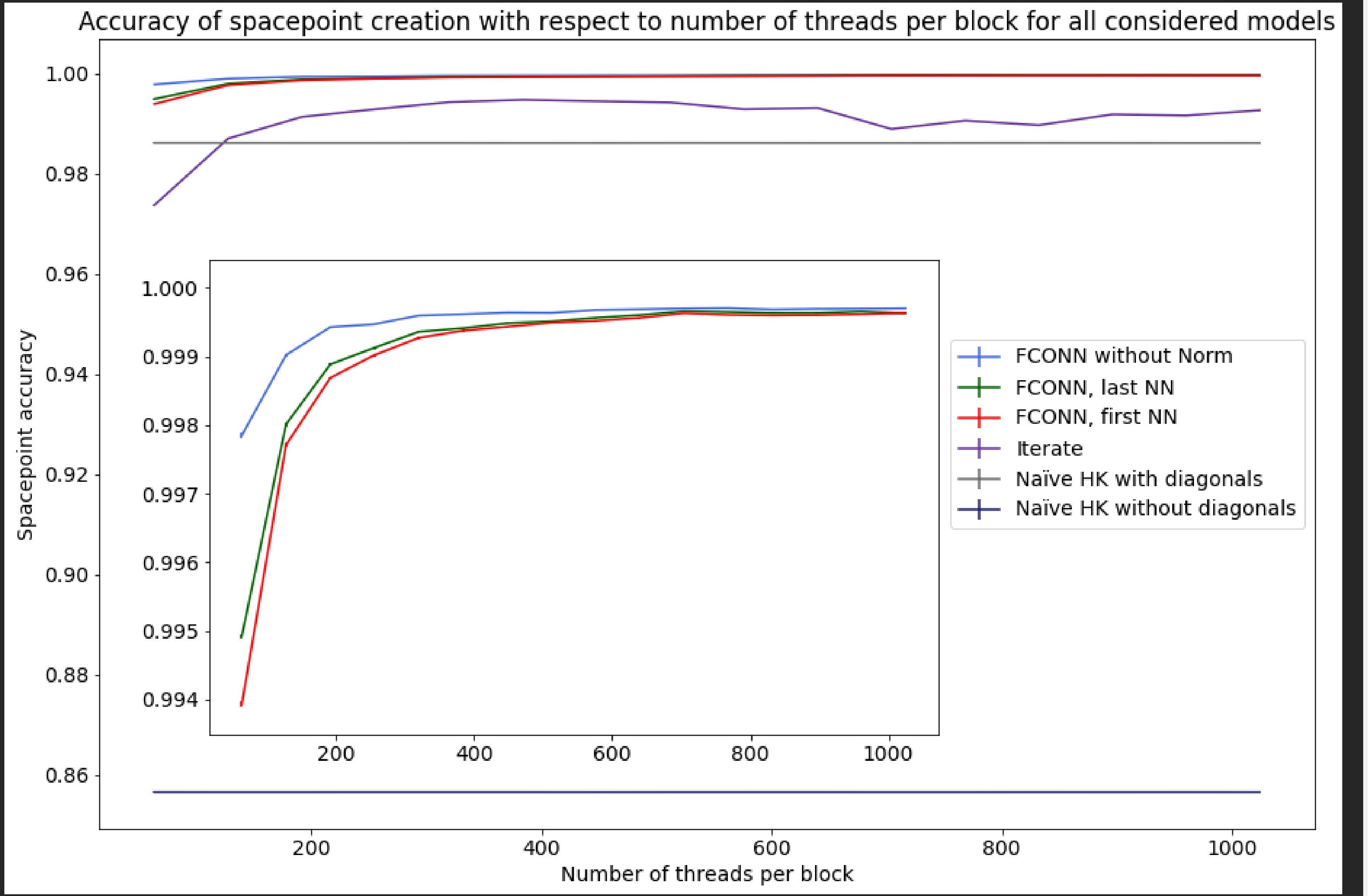


FCONN Inaccuracy or Extra Accuracy?

- Sparse particle hits: Worse
- Dense particle hits: Might be better?

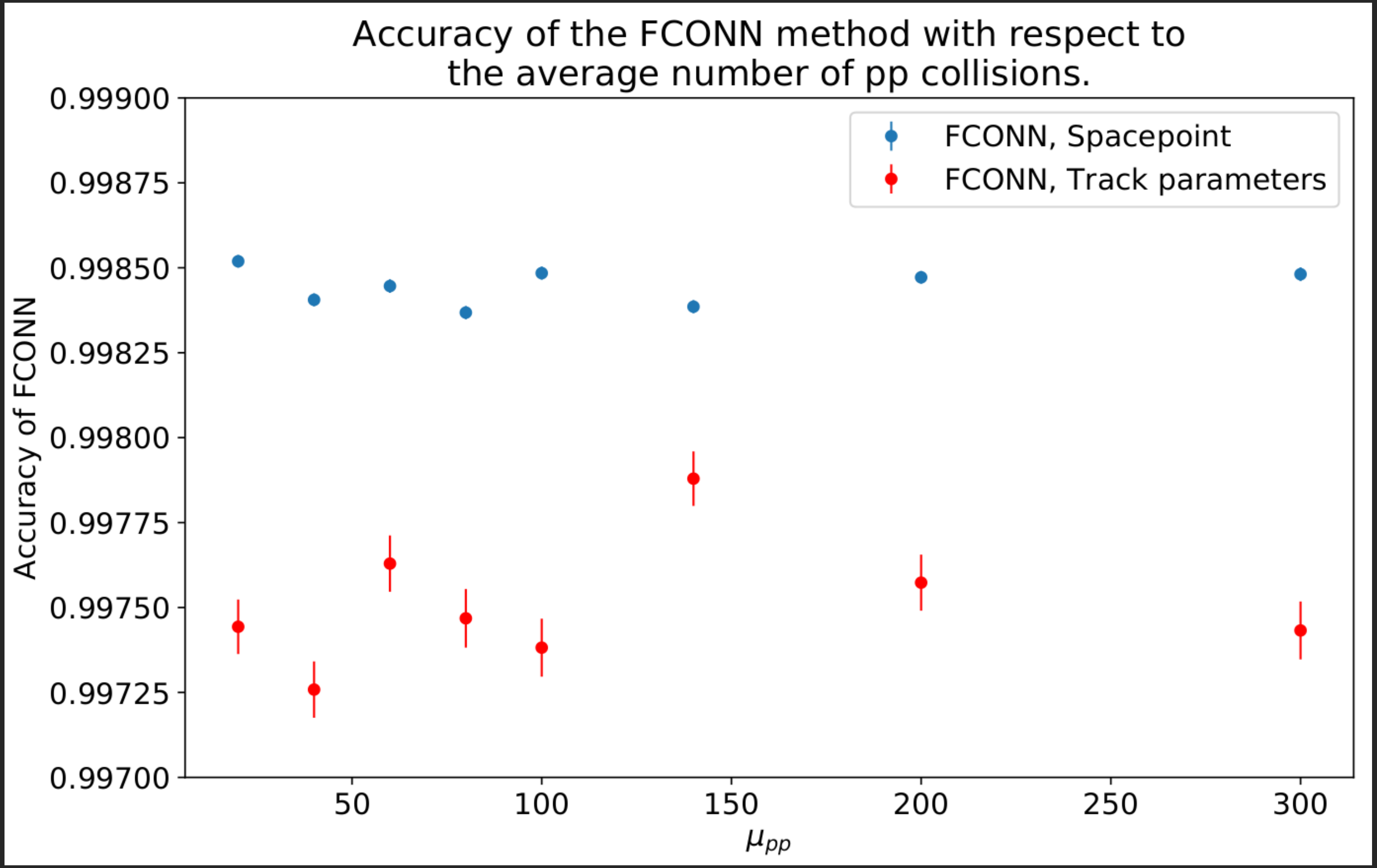


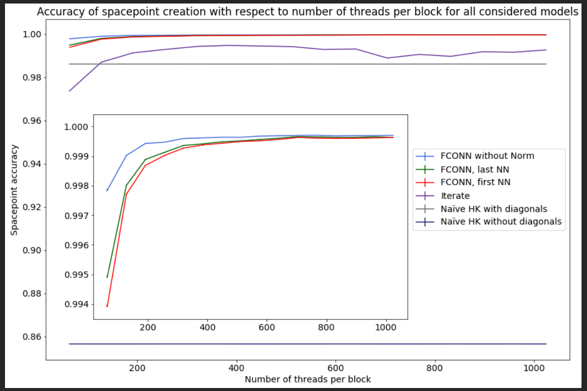




Results – Accuracy

- Smaller benchmarking dataset
- Comparison of methods
- Accuracy performed against the number of threads per block



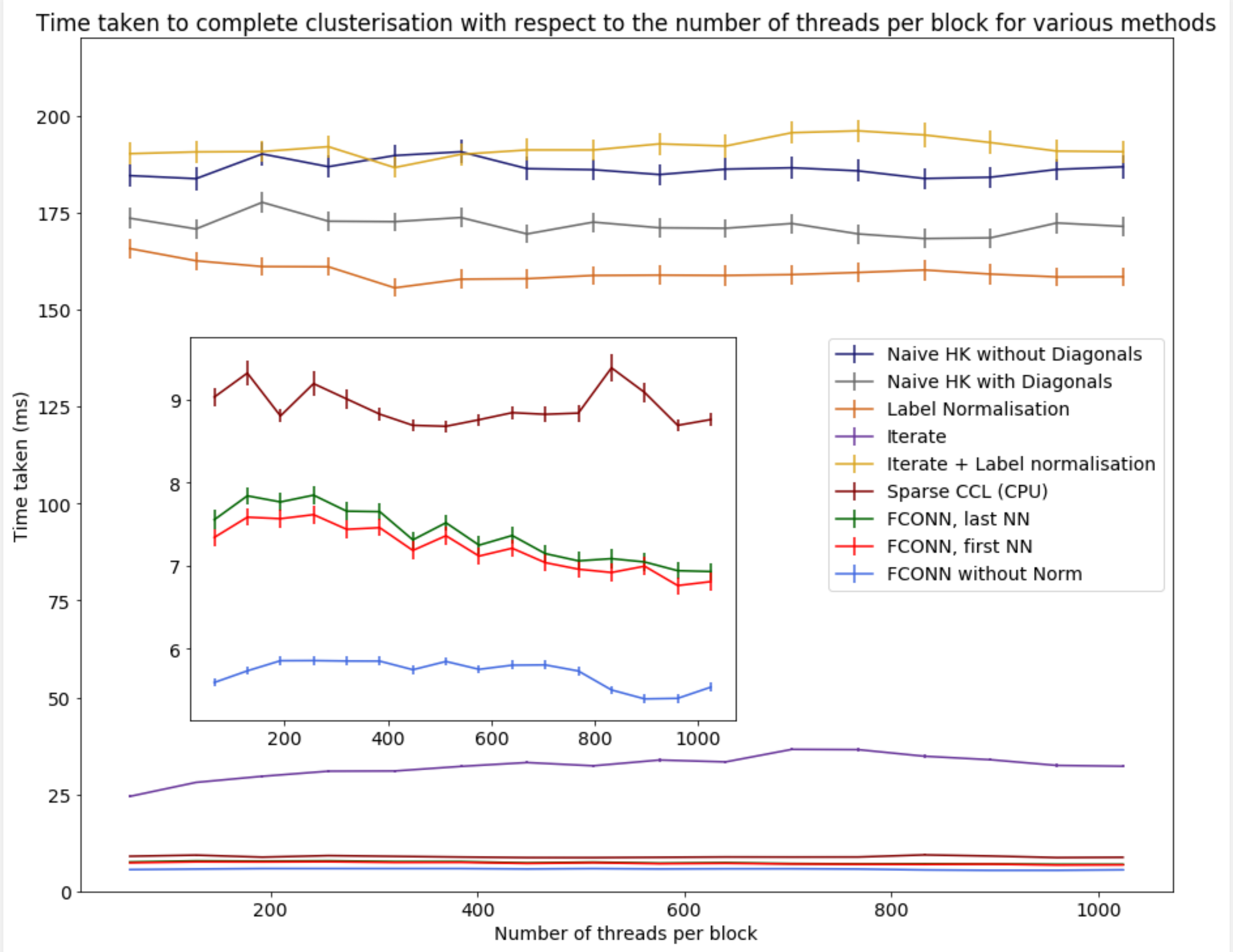


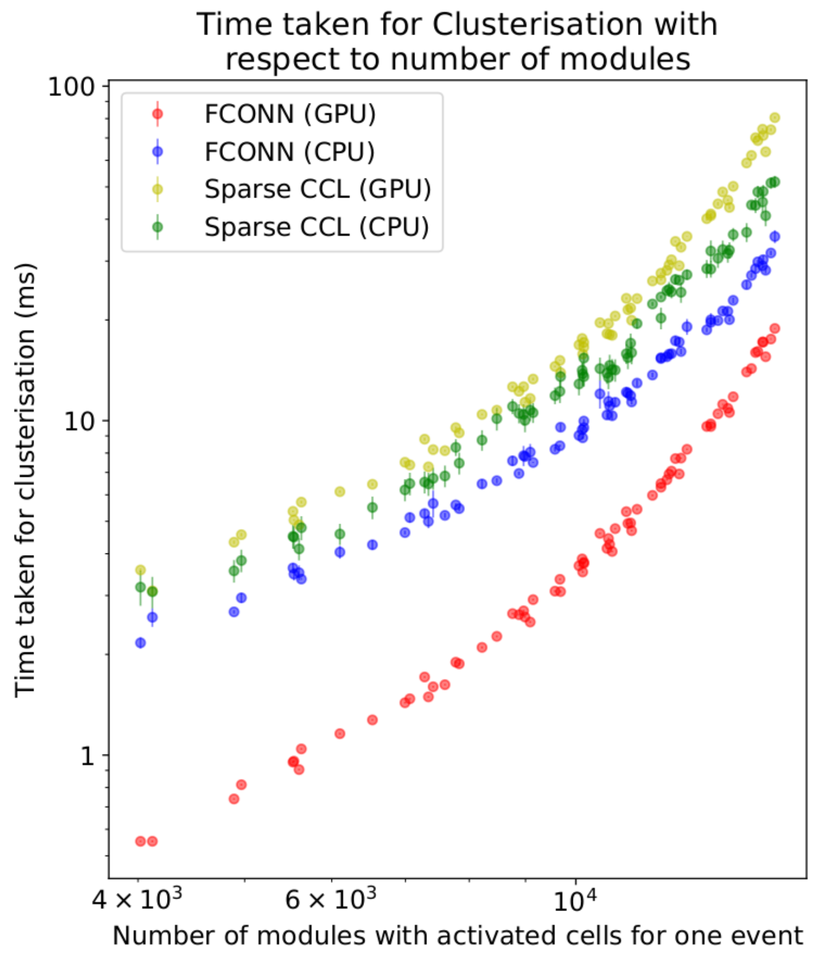
Results – Accuracy

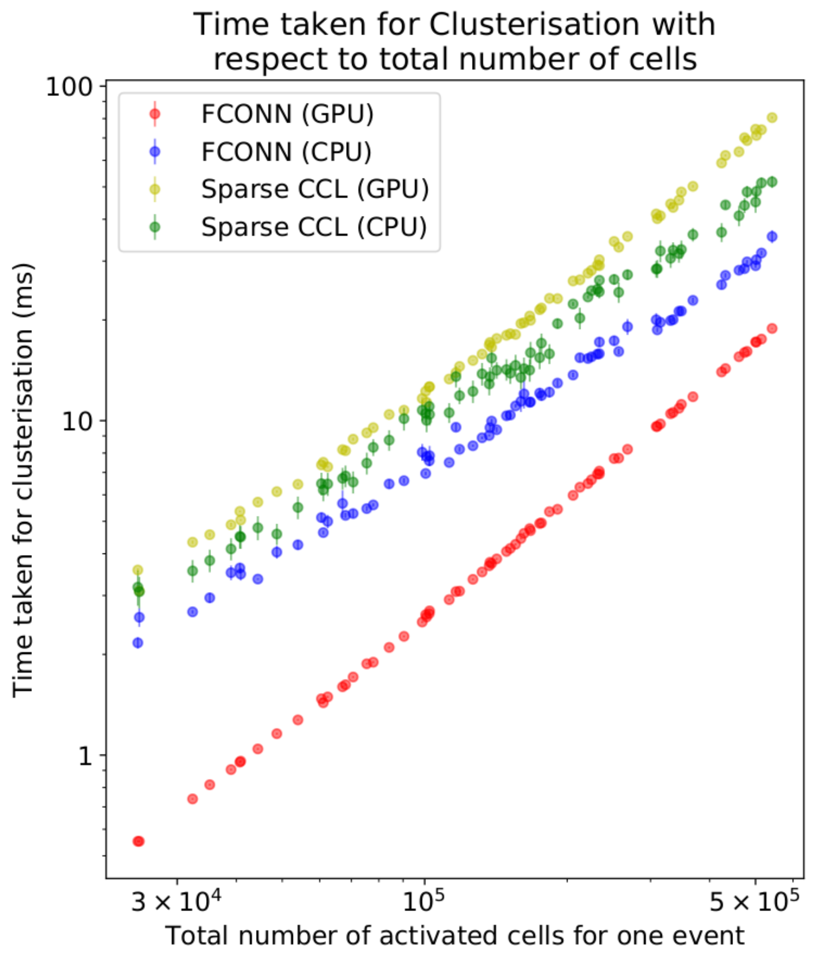
- Extended datasets with varying μ
- Comparison of Sparse CCL with FCONN only

Results – Time performance

- Smaller benchmarking dataset
- Comparison of methods



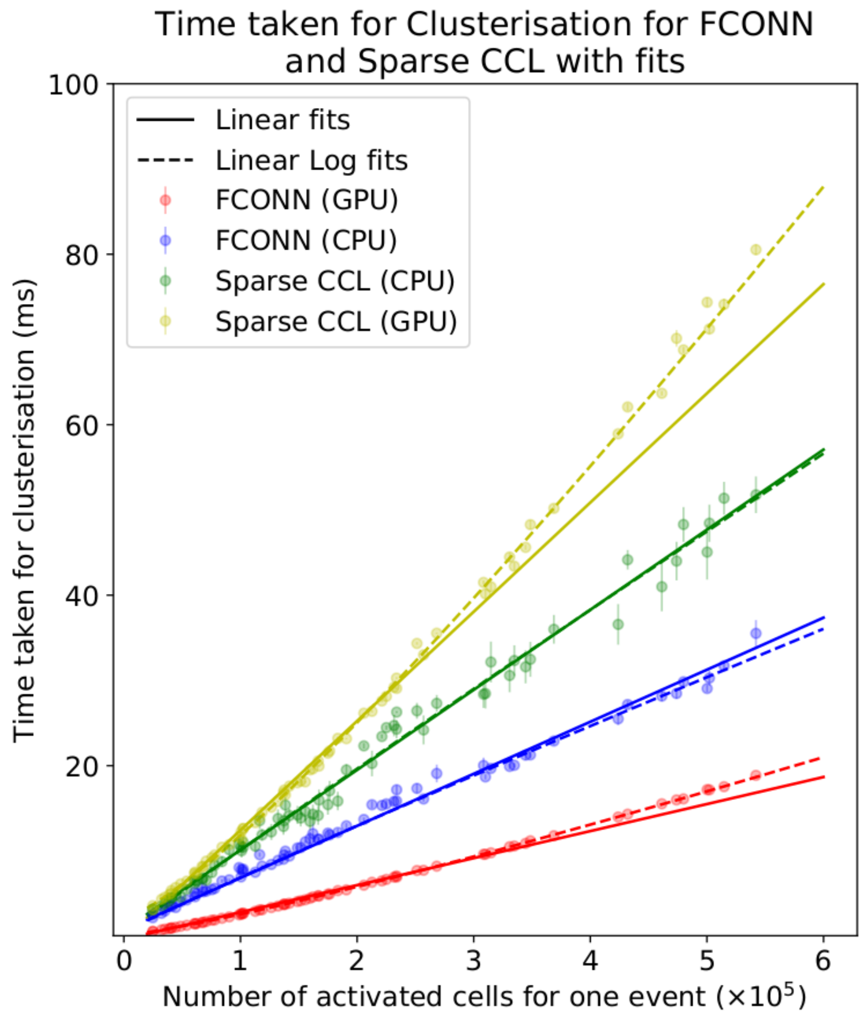


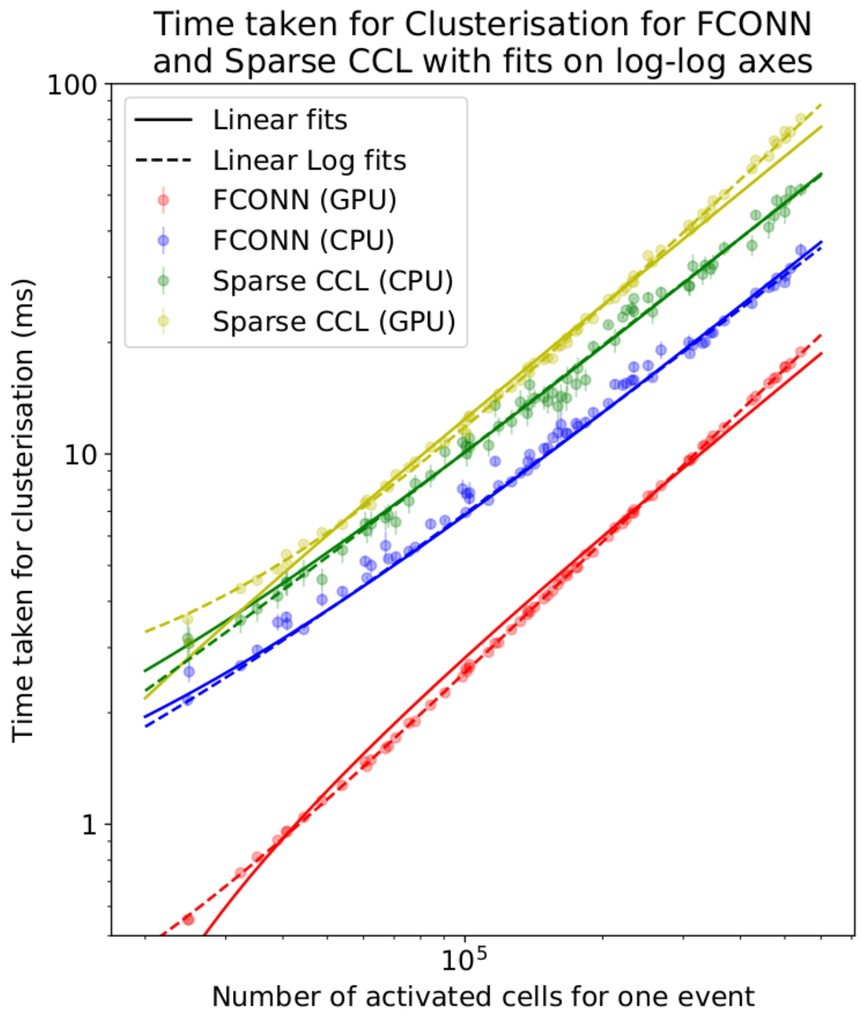


Results – Time performance

- Extended datasets with varying μ
- Comparison of Sparse CCL with FCONN only

Result Fitting – Cells





GPU implementation: O(nlog(n))

CPU implementation: O(n)

Why?
Clusterisation an inherently iterative problem?

CPU hardware (cache focused) outperforms GPU hardware (data processing focused)

Summary and Future extensions

- Clusterisation is a GPU bottleneck
  - FCONN improves time performance compared to Sparse CCL for all probed μ
    - Factor 3.22 ± 0.29 vs CPU for μ = 200
    - Factor 4.37 ± 0.44 vs GPU for μ = 200
  - FCONN retains high accuracy throughout
    - Stays constant as a function of μ
  - However: GPU concedes a factor of log(n) in runtime with respect to number of cells. Is it possible to reduce the asymptotic time performance on GPU?
- Possible improvements
- Takeaways

Summary and Future extensions

- Clusterisation is a GPU bottleneck
- Possible improvements
  - FCONN
    - Reduce the number of lookups in global memory
      - GPU: Fast processing, slow lookup: Focus on processing
      - CPU: Slow processing, fast lookup: Focus on caches
    - More in depth analysis with NVIDIA profiling tool to see where time inefficiencies arise
  - General
    - Study the theoretical limit: Why are the GPU implementations asymptotically slower than the same implementations on CPU? Is it because of the memory management or is it more inherent and will exist in other methods too?
    - Study a potential interplay of CPU and GPU:
      - Modules with smaller number of activated cells: GPU
      - Modules with larger number of activated cells: CPU
- Takeaways

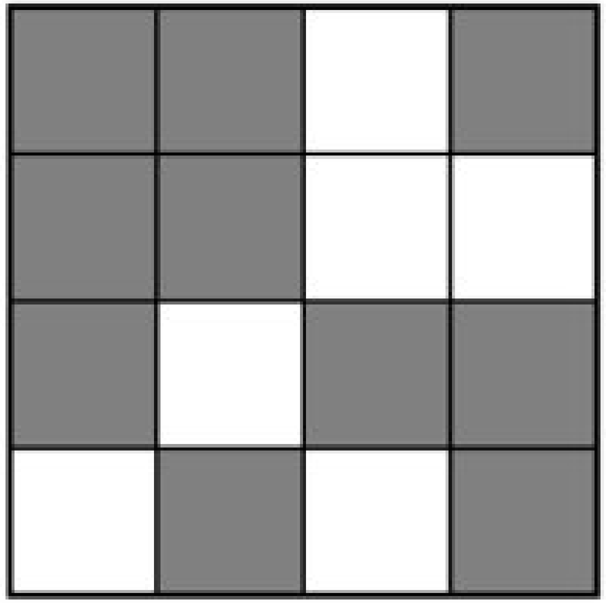
Summary and Future extensions

- Clusterisation is a GPU bottleneck
- Possible improvements
- Takeaways
  - GPU methods valuable for track reconstruction
  - FCONN is an improvement in the μ ranges that are experimentally relevant
  - Viability of GPU programming methods proven: Future development essential

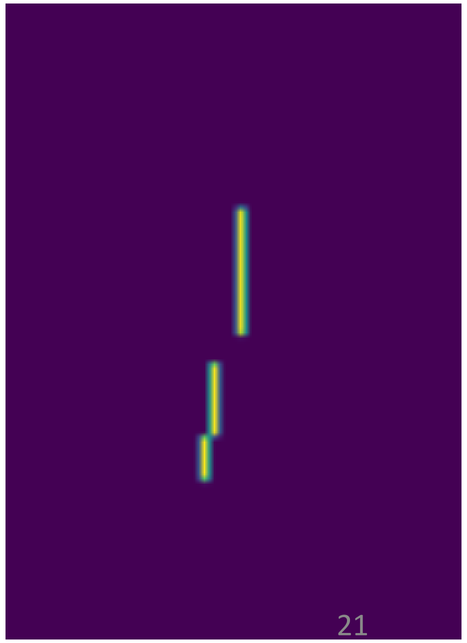
References

- [1] *Arthur Hennequin, Ben Couturier, Vladimir V. Gligorov, and Lionel Lacassagne*. Sparse-CCL: Connected components labeling and analysis for sparse images. In 2019 Conference on Design and Architectures for Signal and Image Processing (DASIP), pages 65–70, 2019.
- [2] *Hoshen, J.; Kopelman, R. (15 October 1976). "Percolation and cluster distribution. I. Cluster multiple labeling technique and critical concentration algorithm". Phys. Rev. B.* ***14*** *(8): 3438–3445. Bibcode:1976PhRvB..14.3438H. doi:10.1103/PhysRevB.14.3438 – via APS.*

Troubles with parallelisation

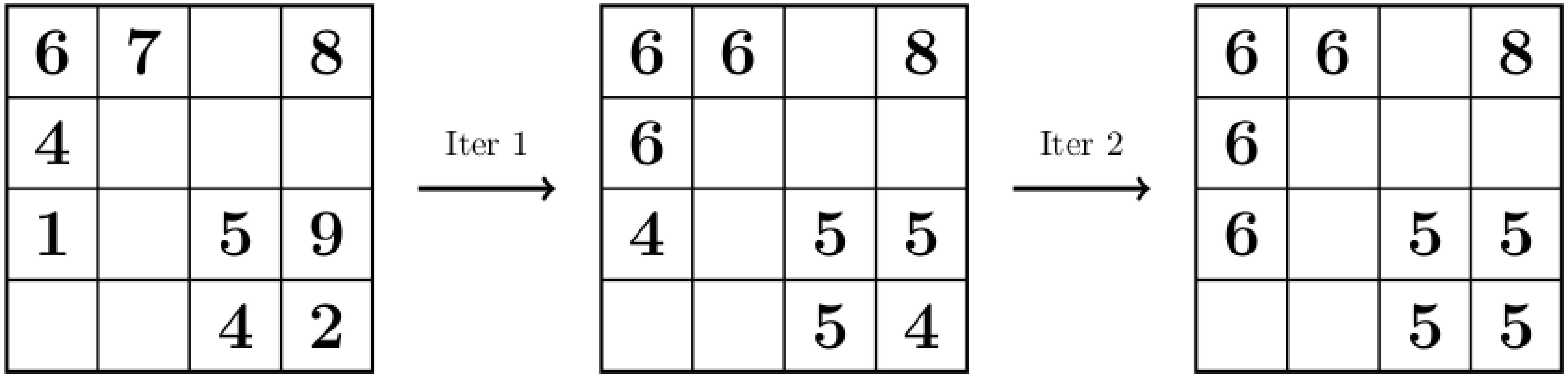


- Inconsistent number of cells per module
  - Cannot effectively launch 2D threads
  - Need lookup table for module by cell
    - Global Memory access -> Spatial and time inefficiency
- Cells cannot interact efficiently
  - Purpose of effective parallelism is SIMD
    - "Single Instruction, Multiple Data"
  - Checking for neighbours for every cell is inconsistent
  - Overwriting neighbours is dangerous
- Solve by just checking half the neighbours and iterating
  - Only overwrite "yourself" -> No cross-writing issues
  - Slow, many threads idling



Iterative Hoshen-Kopelman

- Choose neighbour above/left with diagonals
  - Consistent neighbour selection rule, e.g. first, last, lowest label etc.
    - Example below is with lowest neighbour label selection rule
  - Always write from that neighbour
  - Iterate until no more changes between states

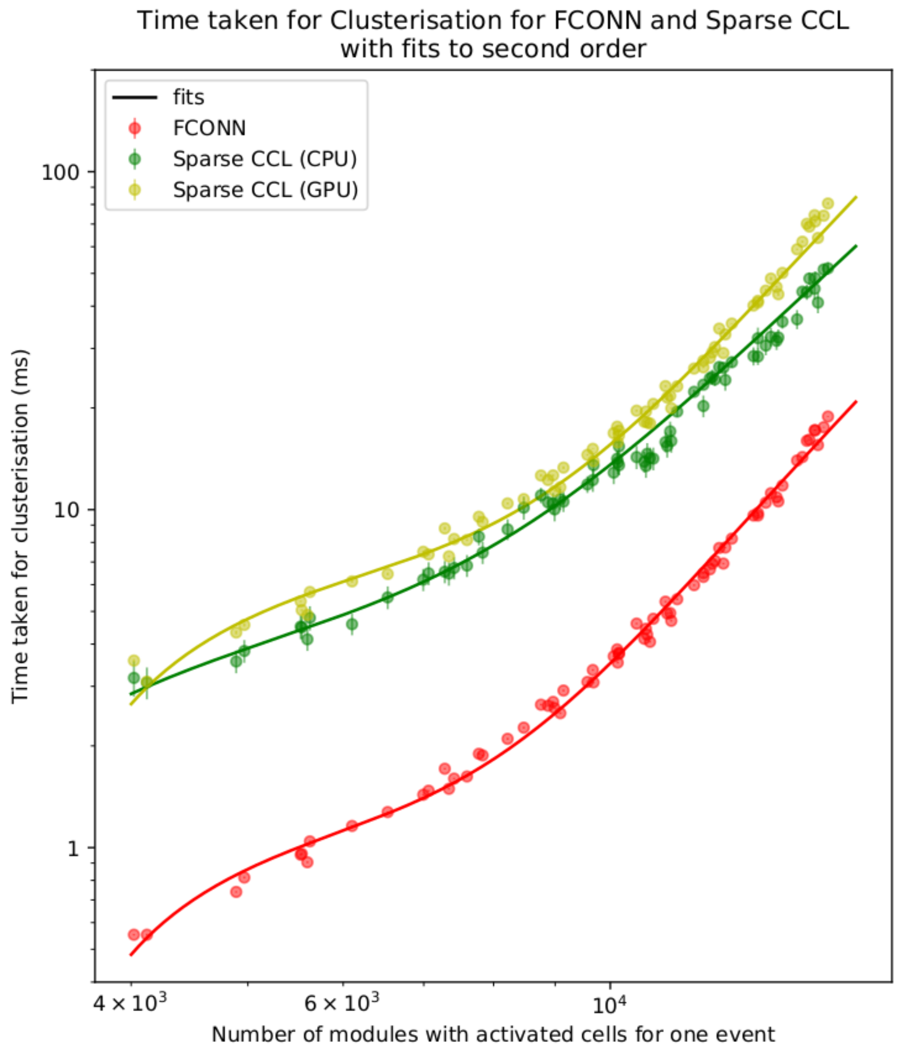


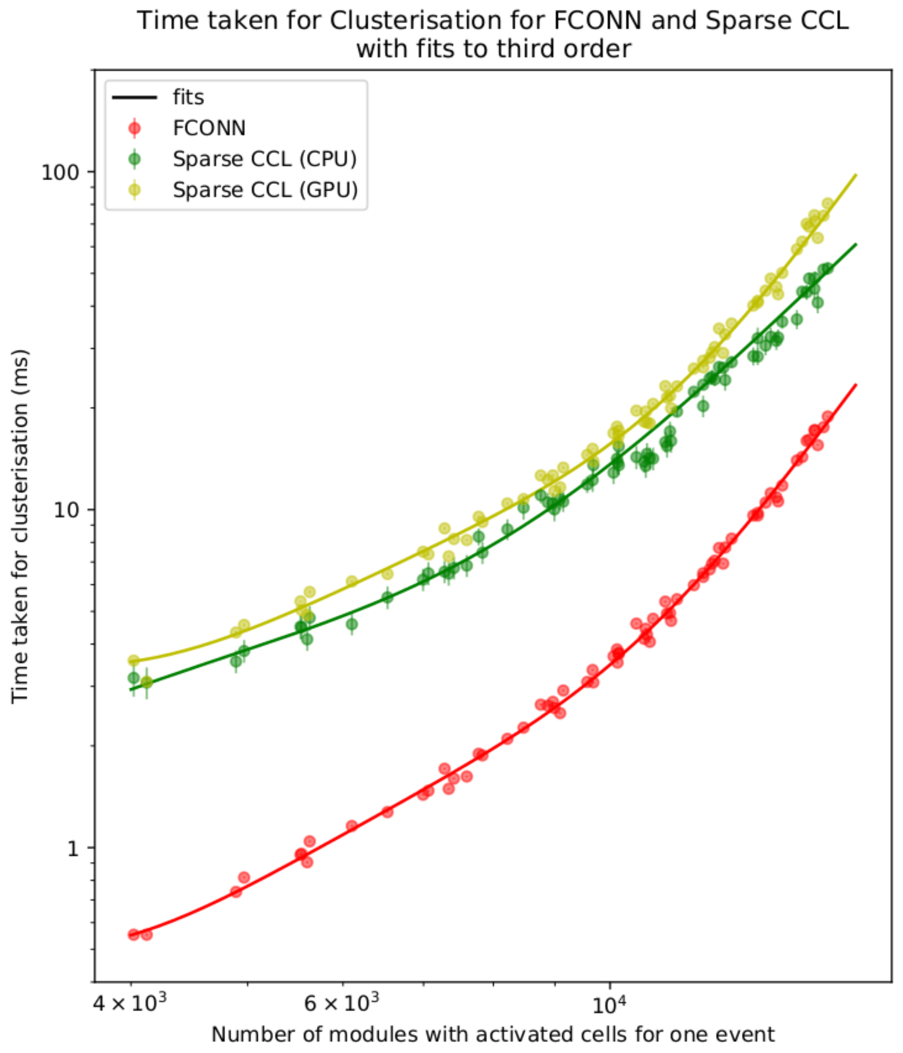
6	7		8
4			
1		5	9
		4	2

6	6		8
6			
4		5	5
		5	4

6	6		8
6			
6		5	5
		5	5

Fitting – Modules


Time taken for Clusterisation for FCONN and Sparse CCL with fits to second order
fits
FCONN
Sparse CCL (CPU)
Sparse CCL (GPU)
Time taken for clusterisation (ms)
4×10^3
6×10^3
10^4
Number of modules with activated cells for one event


Time taken for Clusterisation for FCONN and Sparse CCL with fits to third order
fits
FCONN
Sparse CCL (CPU)
Sparse CCL (GPU)
Time taken for clusterisation (ms)
4×10^3
6×10^3
10^4
Number of modules with activated cells for one event