MAY 15, 2025

Rohan Riaz

Azka Aqeel

Subata Naveen Khan

PARALLEL INTER-IMAGE K-MEANS

A Faster Satellite Imagery Algorithm

Search

Parallel and Distributed Computing
Spring 2025

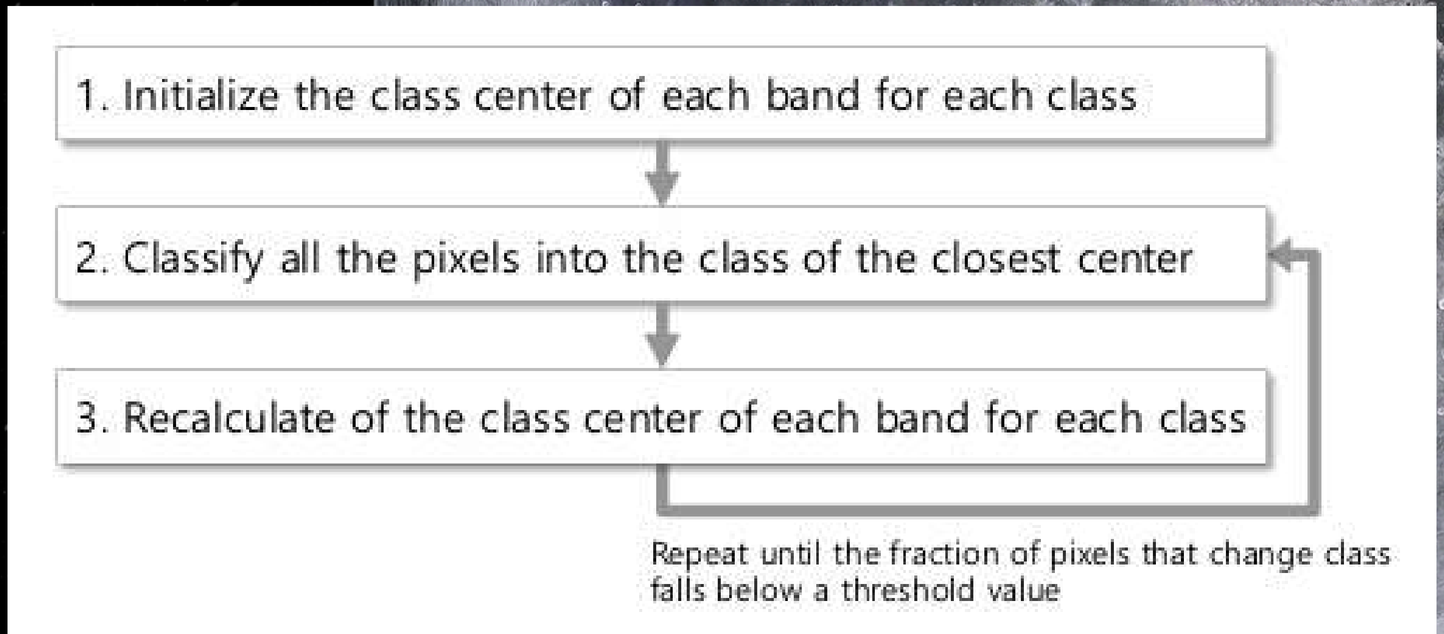
Why Analyze Satellite Images?

- Satellite imagery is vital for tracking land use, urban growth, and environmental change.
- But analyzing them consistently is computationally hard.
- Traditional clustering doesn't always result in consistent labelling

MAY 15 2025

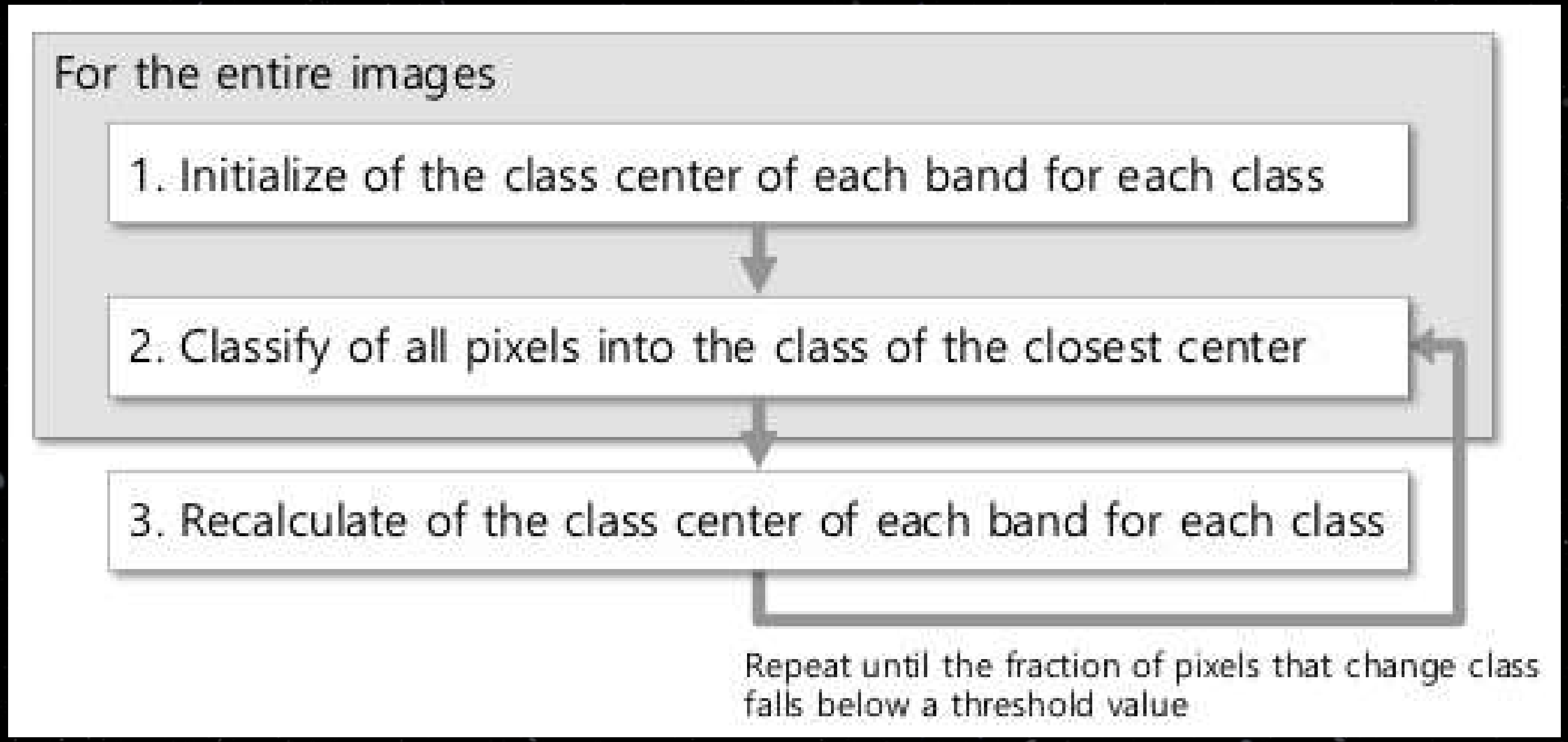
One Scene. One Clustering. No Memory.

- k-means clusters each scene *independently*, leads to **label drift**
- Same forest might be green in Scene 1, red in Scene 2
- There's no **temporal consistency**



General Workflow of kMC

IIkMC: Inter-Image k-Means Clustering



Clusters all scenes together, ensuring label consistency across time

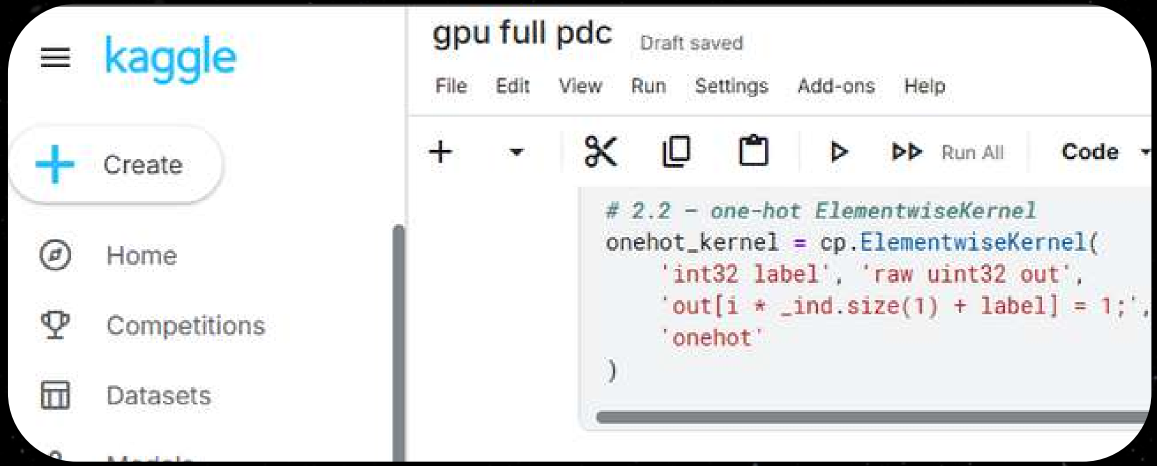
Proposed by *Han & Lee (2024)*

But: it's **computationally intense** → millions of pixels per scene!

ELEMENT SPACE

Our Goal

- Make IIkMC **fast** and **scalable**
- Handle **>200M** pixels total
- Run on *RAM-limited platforms* like Colab/Kaggle
- Compare sequential, CPU-parallel, and GPU-parallel versions



Dataset & Tools

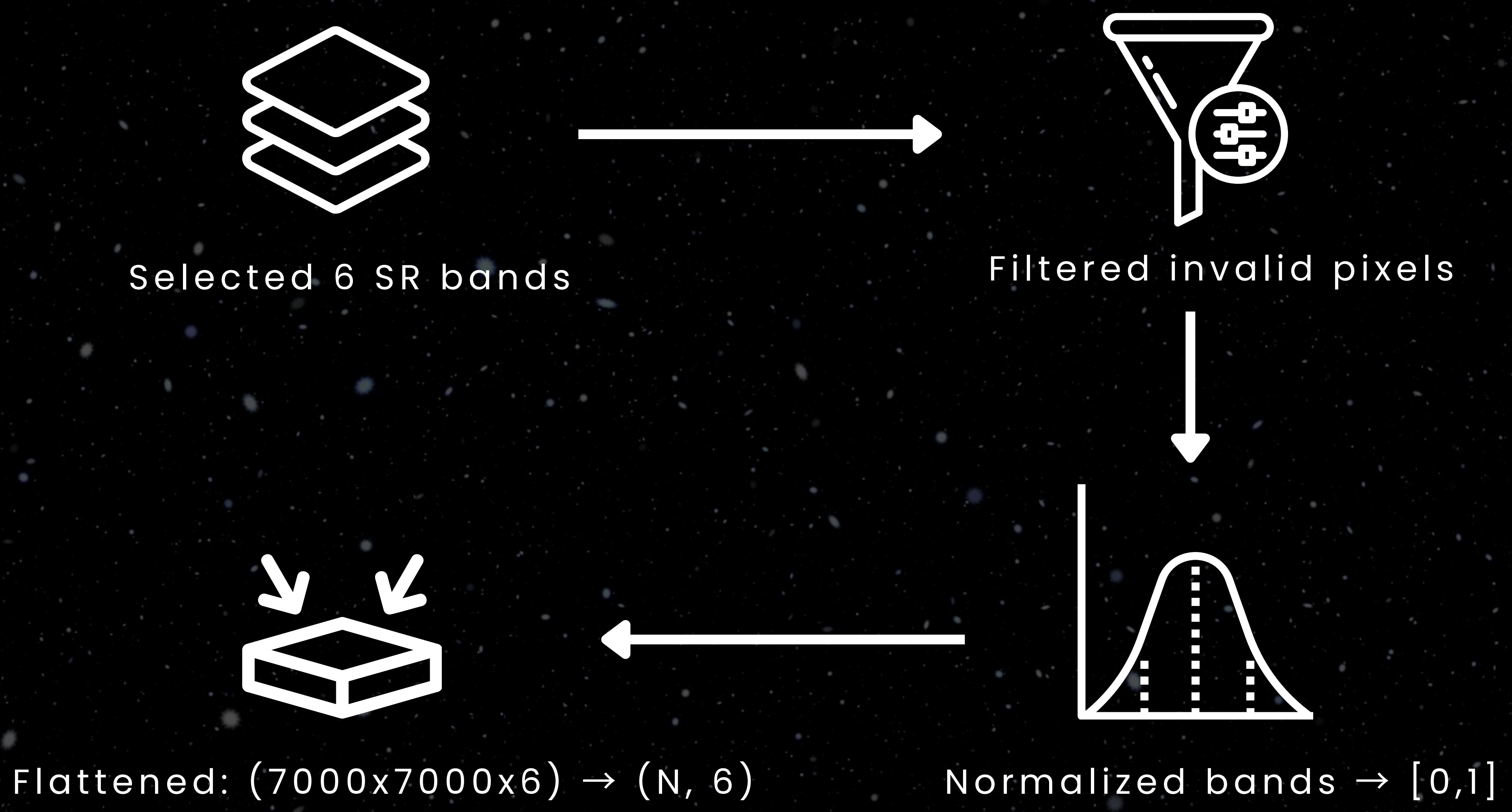
- 5 Landsat 8/9 scenes of Karachi (April 2025)
- 6 spectral bands (B2–B7)
- ~40M valid pixels per scene
- Platforms: Colab (T4 GPU) + Kaggle (P100 GPU)

Search

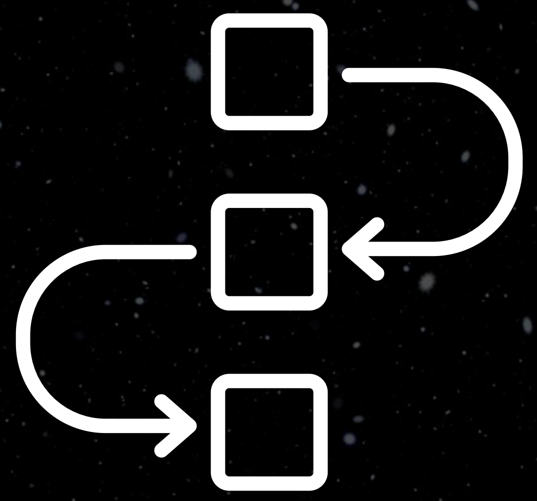
Scientific Project

ELEMENT SPACE

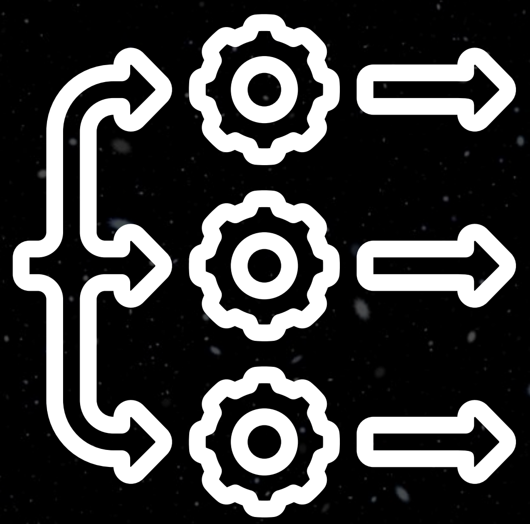
Preprocessing Pipeline



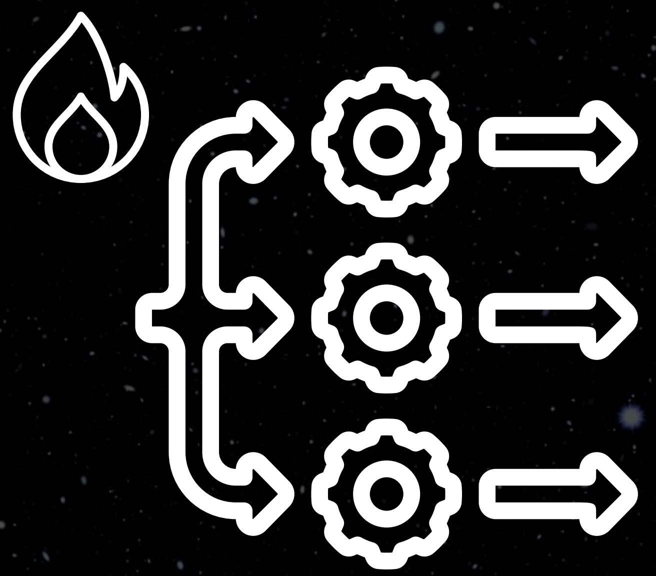
Implementation Overview



Sequential
(numpy)



CPU-Parallel
(Threaded Block Processing)



GPU-Parallel
(CuPy + RawKernel)

Sequential Version

- *NumPy-only* version
- One pixel at a time, 2 iterations, **3.5 hours total**
- Used midpoint-based initialization, standard convergence

ELEMENT SPACE

CPU-Parallel (Threaded)

- Used *ThreadPoolExecutor*
- Scenes chunked into 1M-pixel blocks
- Per-block Euclidean distance without allocating full (N×k) matrix
- Took 25 minutes
- **~8.2× faster than baseline**

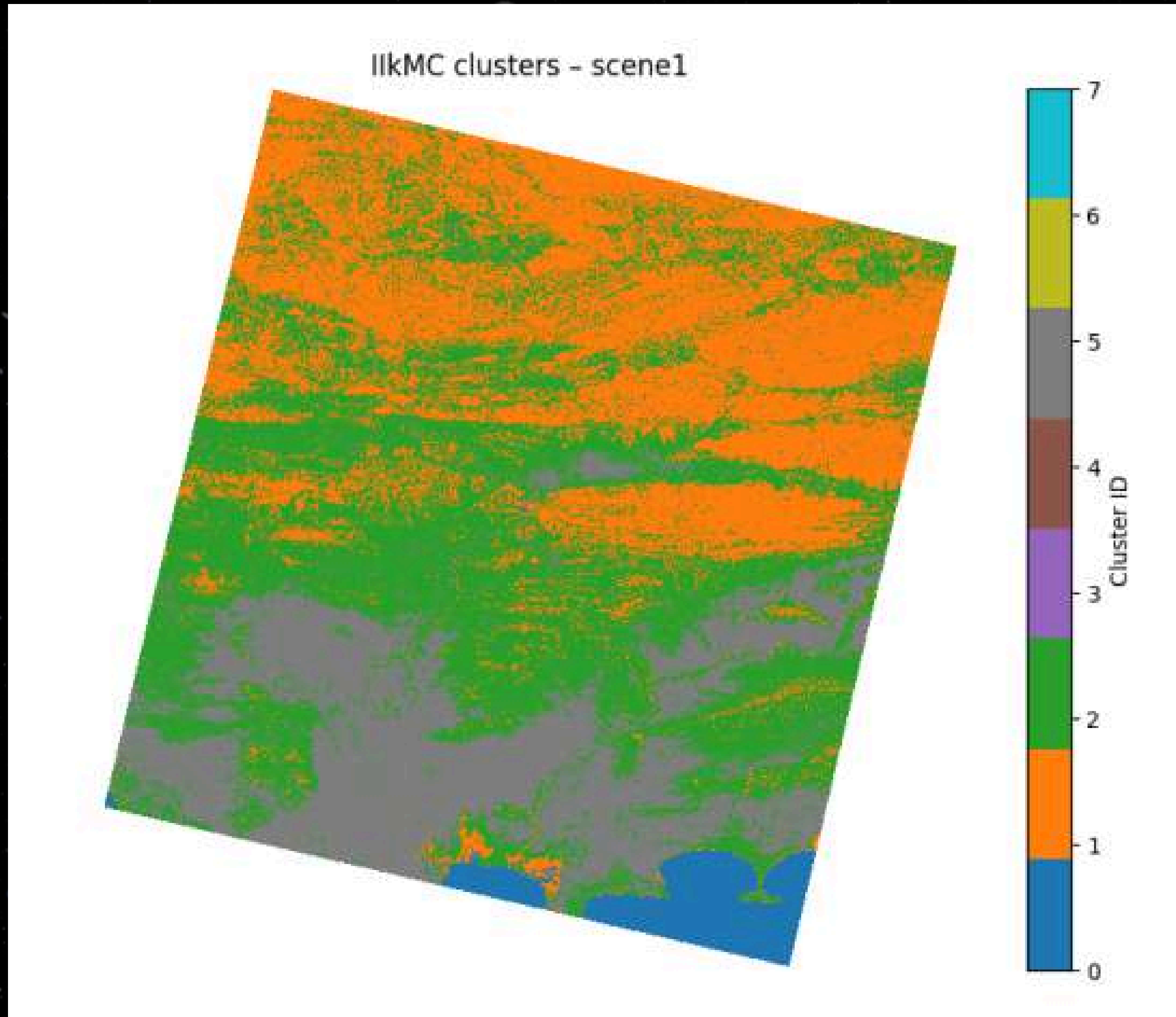
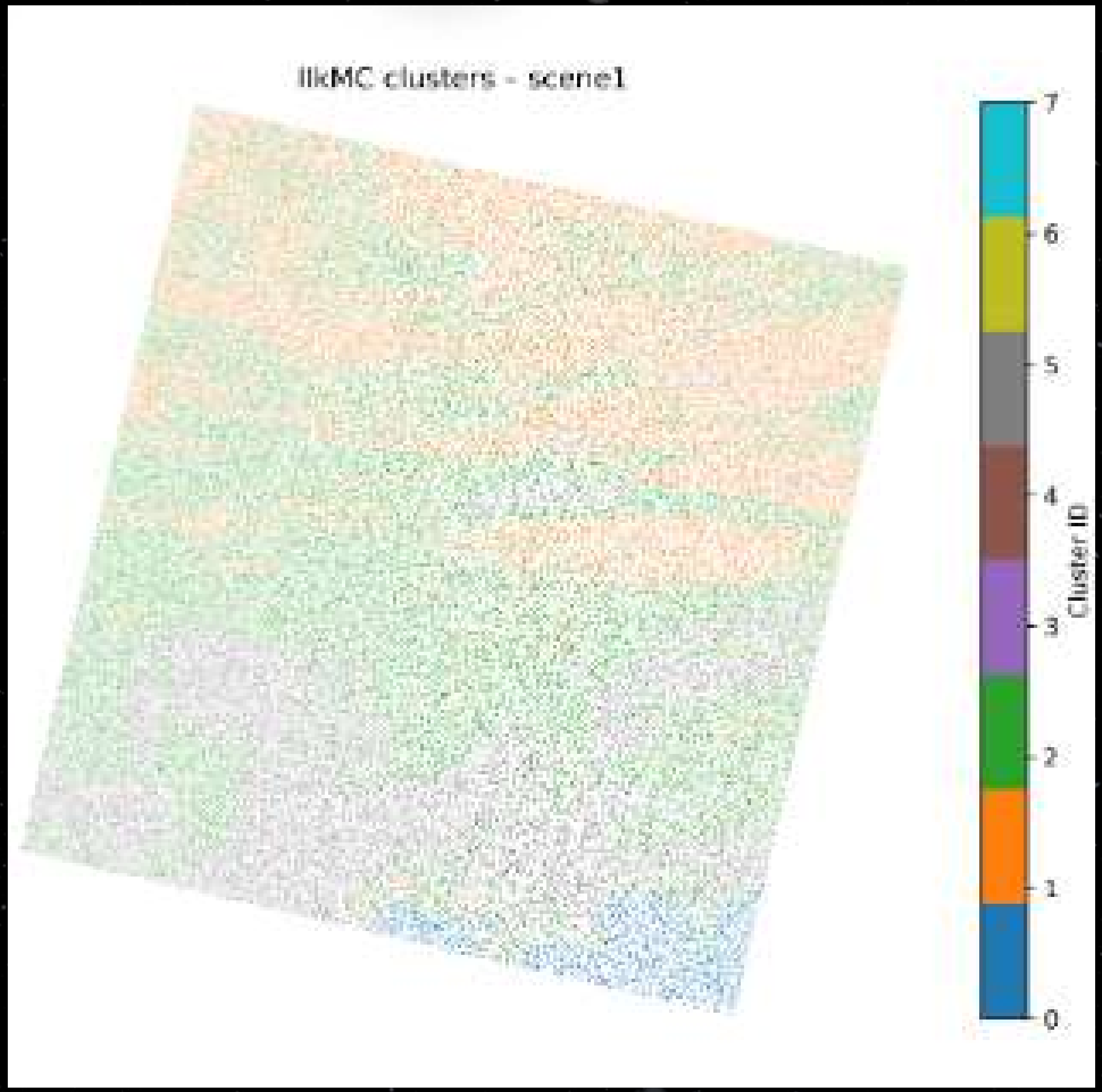


Figure 1: Cluster Map – Scene 1

ELEMENT SPACE

CPU- Downsampling

- • Reduced RAM usage, enabling safe runs on 12–30 GB systems.
- • Allowed more iterations (up to 100) for testing convergence behavior.
- • Made interactive visualizations and debugging feasible



Scientific Project

ELEMENT SPACE

CPU- Downsampling

Number of Threads	Total Runtime (seconds)
1	521.06
2	319.00
4	300.91

- In contrast to the full dataset version (which took approximately 25 minutes to converge), the downsampled version consistently completed within 2–6 minutes
- Speedup = 12.5x

GPU-Parallel (CuPy + CUDA)

- RawKernel fused label assignment + flagging
- On-device one-hot encoding
- *47s runtime on 40M pixels*
- **52× speedup** vs baseline

CUDA Kernel in CuPy – Fast Pixelwise Clustering

- Wrote a custom CUDA C kernel (classify_and_flag) using CuPy's RawKernel
- Each GPU thread processes 1 pixel, computes distances to all clusters, and sets label + flag in one pass

Feature	Benefit
Fused label + flag	Single pass, avoids kernel overhead
Coalesced memory access	Faster pixel-center reads
Loop unrolling	Lower branch cost in inner loop
On-device buffers	No slow host–device transfer each step

ELEMENT SPACE

Downsampling for Safety

- Applied 10–20% random sampling for dev/test
- Reduced crashes on Kaggle/Colab
- Final cluster maps → applied trained centers to full-res scenes

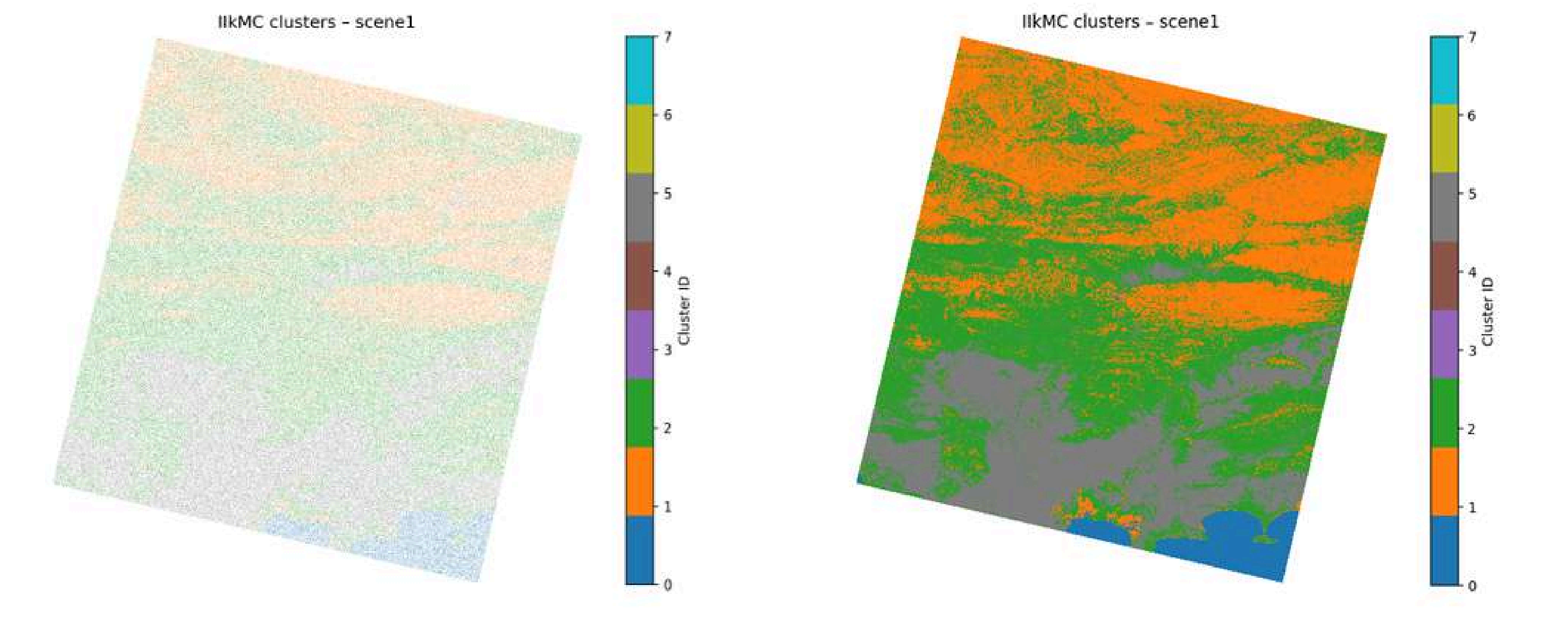
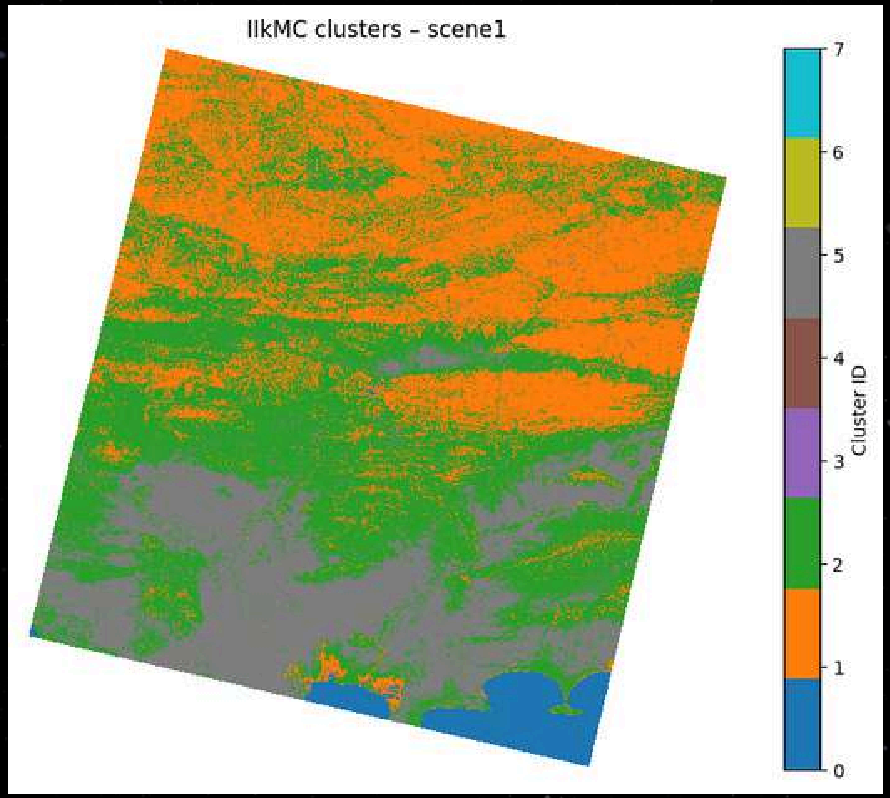
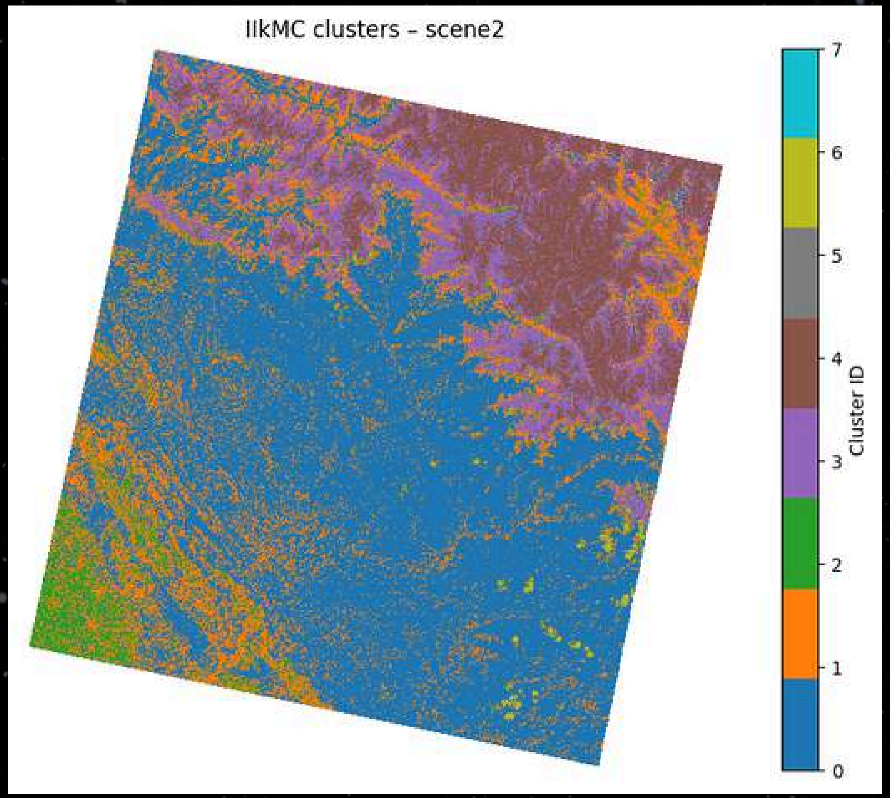
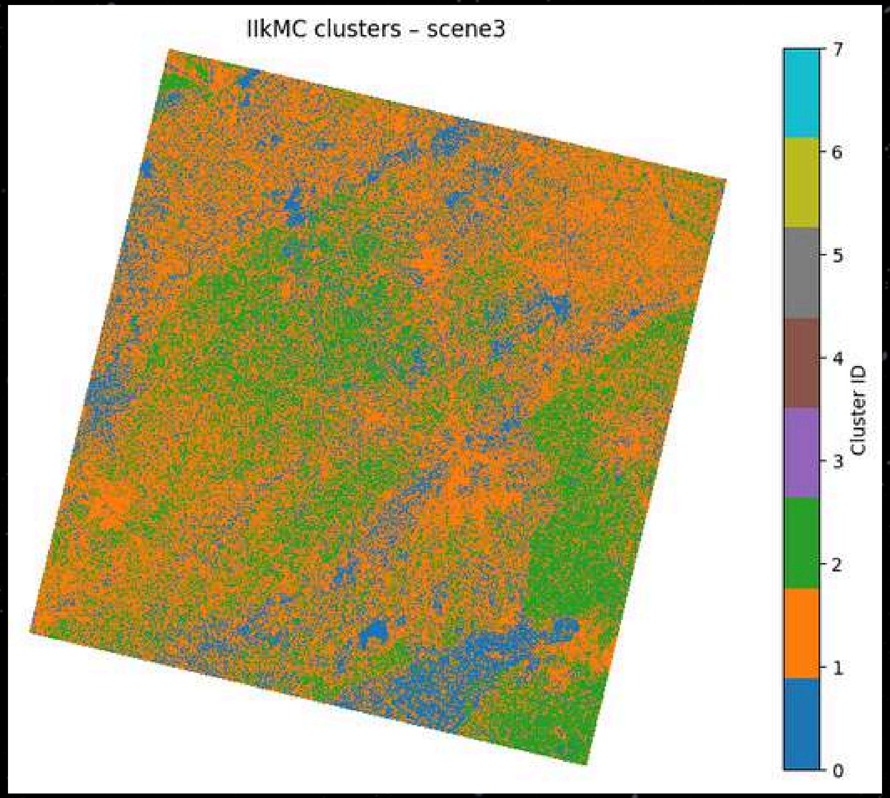


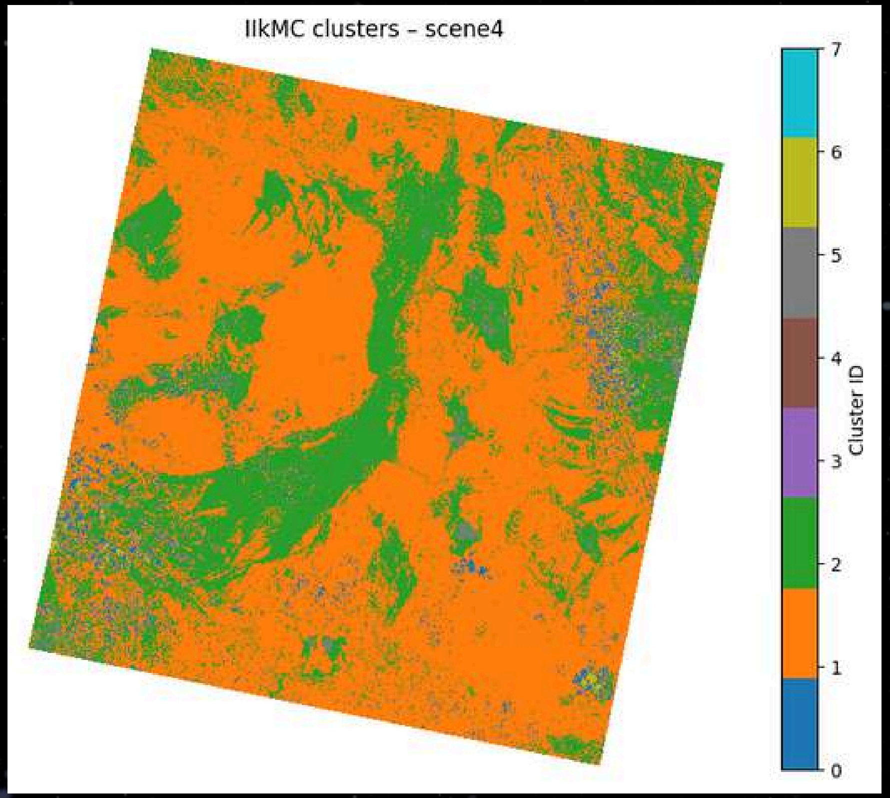
Figure 6: Comparison of cluster maps from downsampled (left) and full-resolution (right) Scene 1.

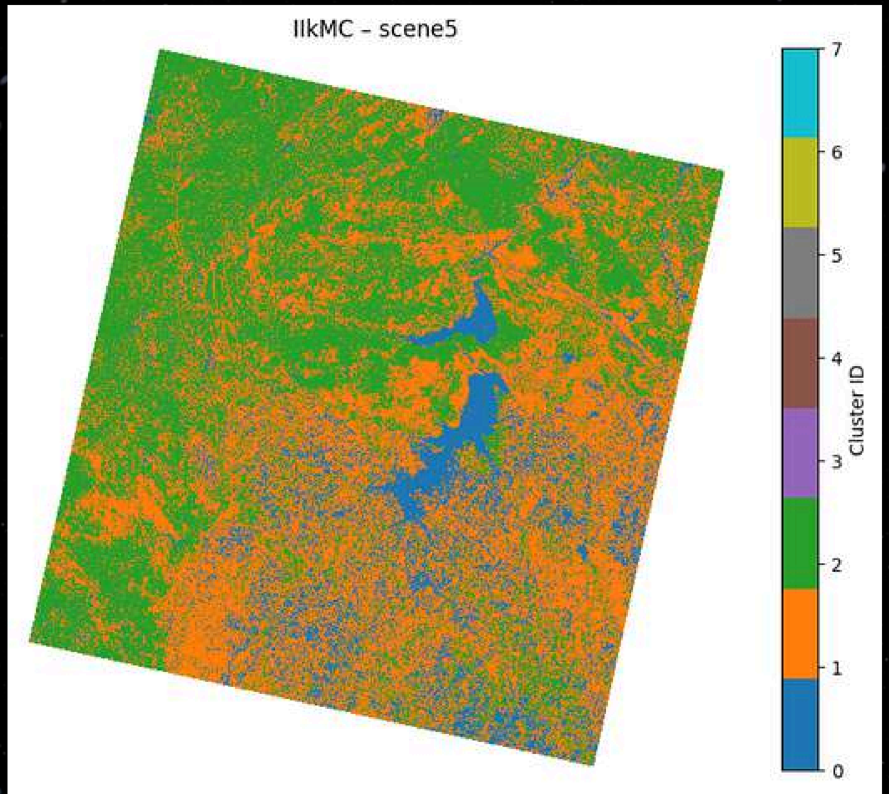
Cluster Maps (full-res)



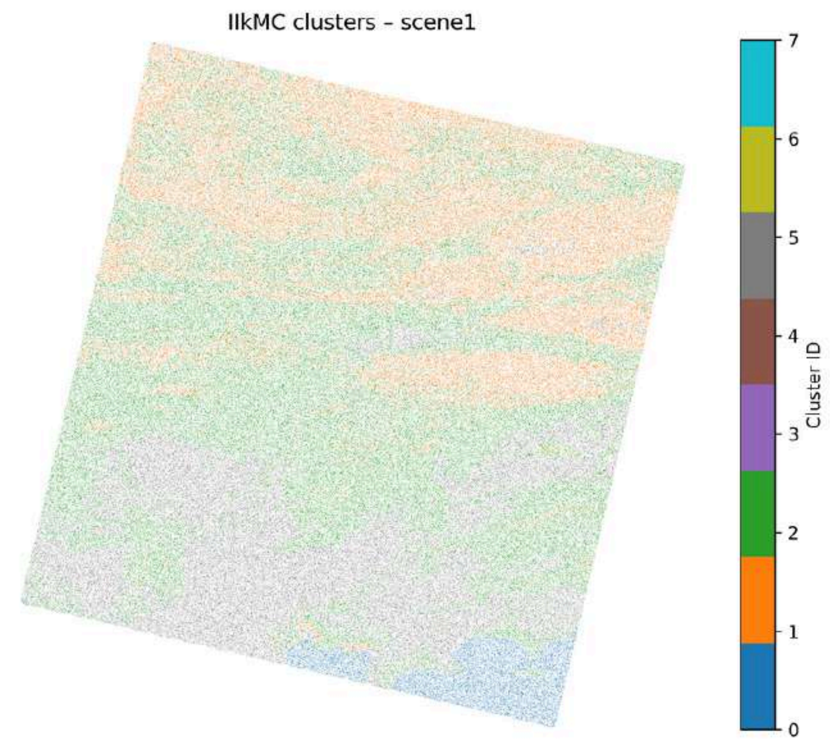


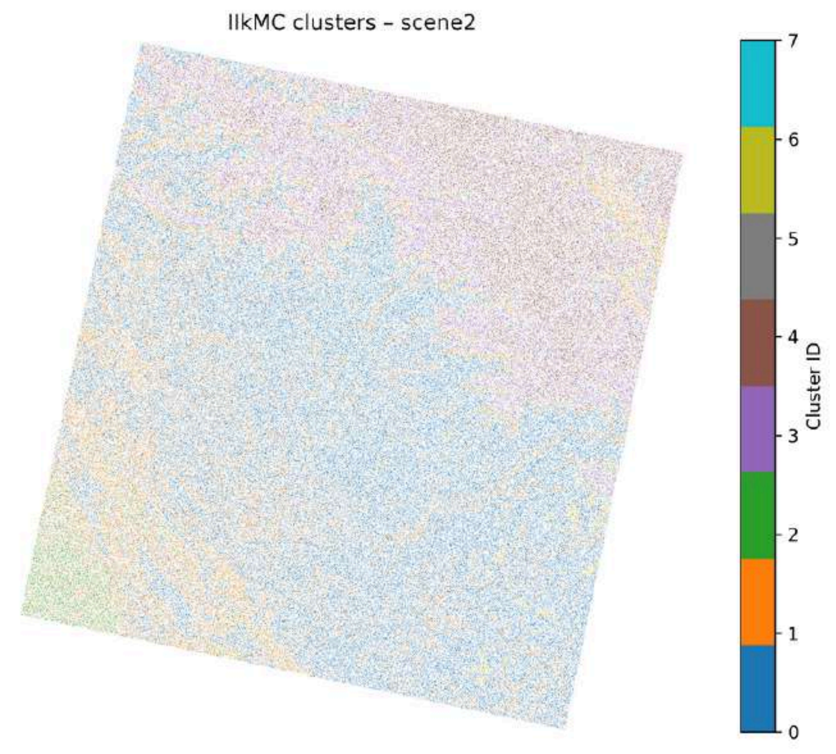


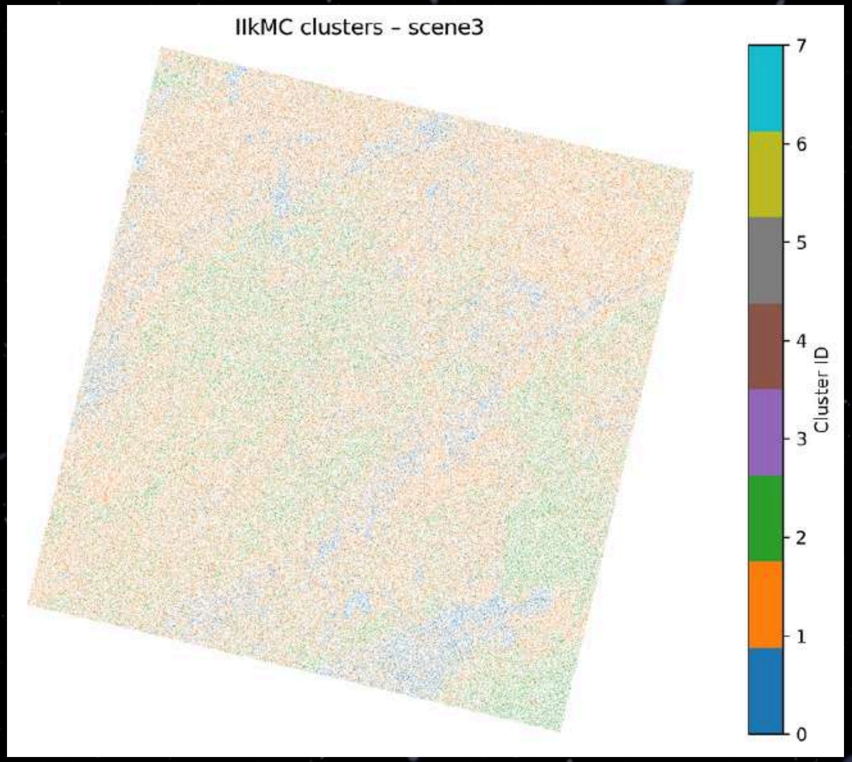


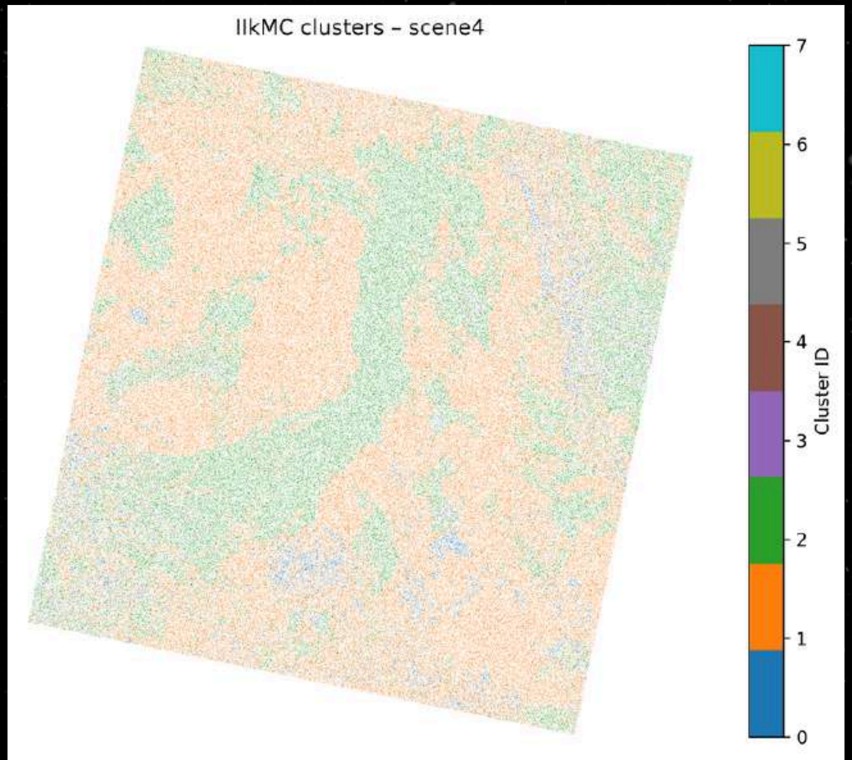


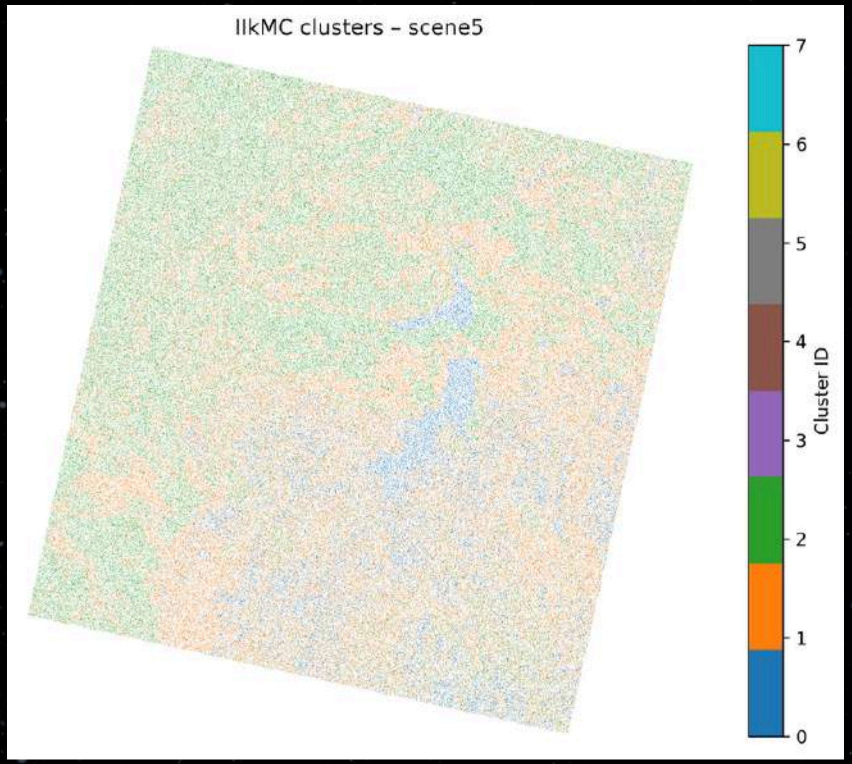
Cluster Maps (CPU)



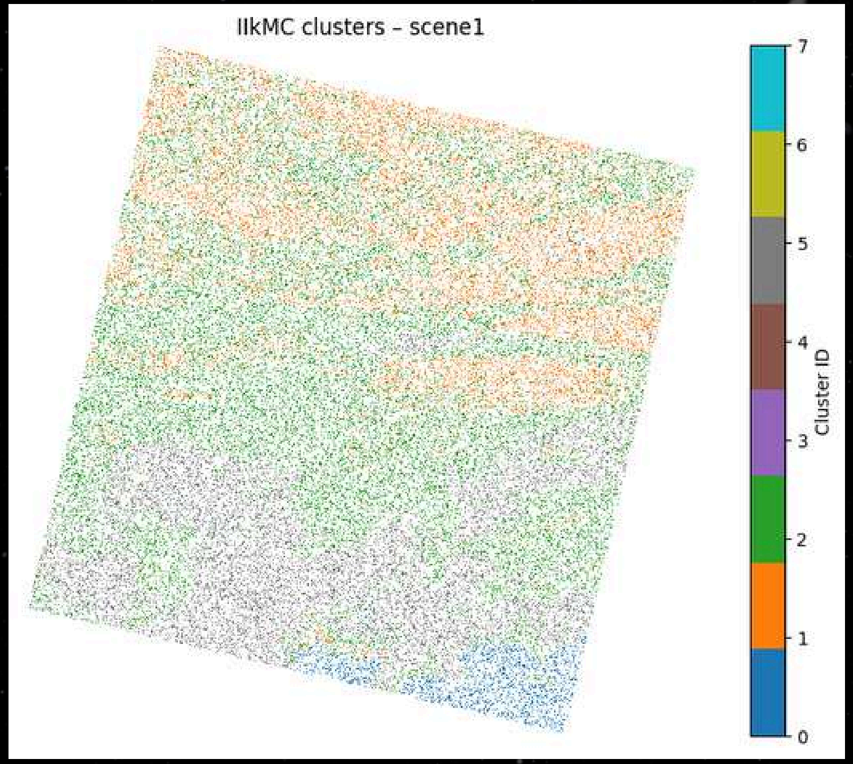


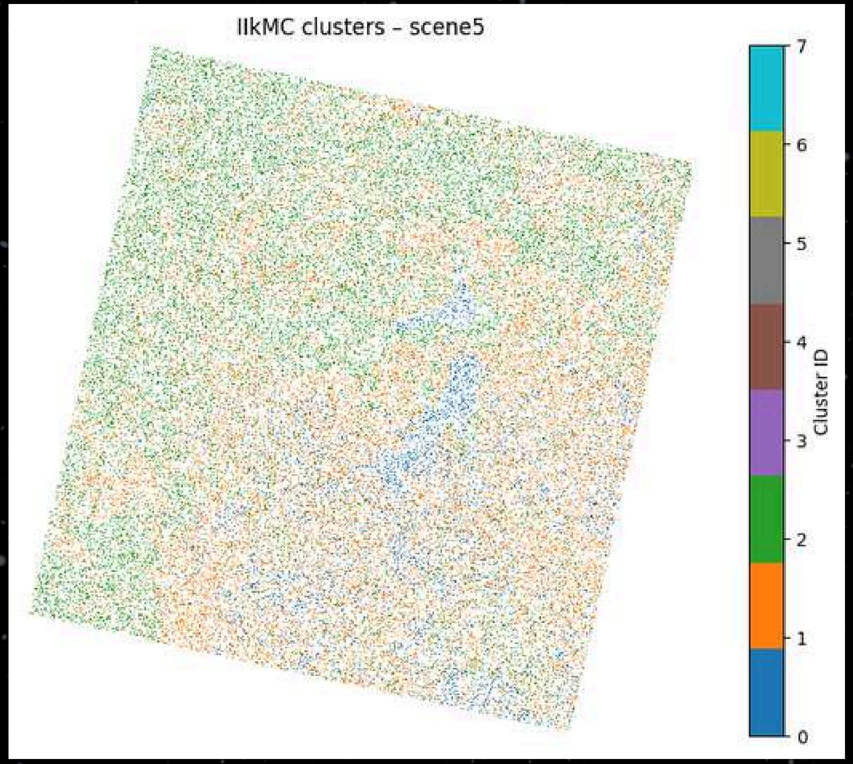


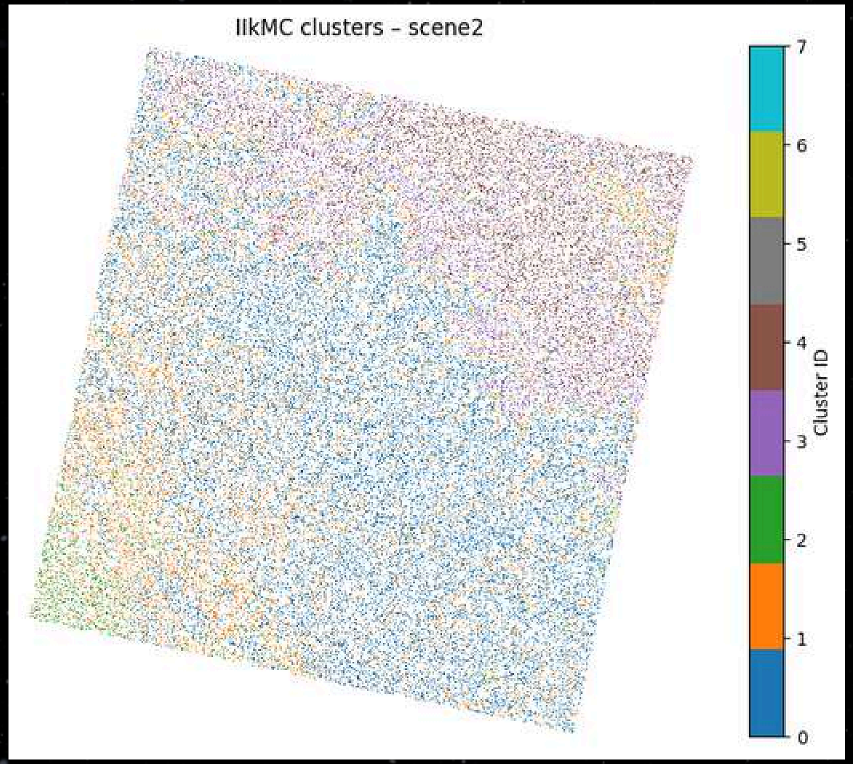


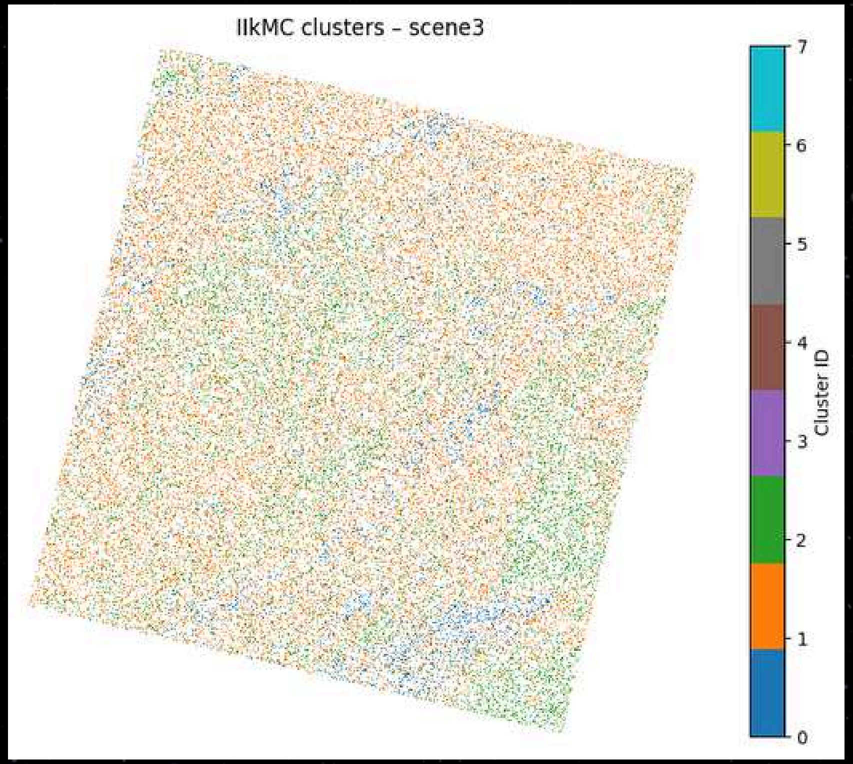


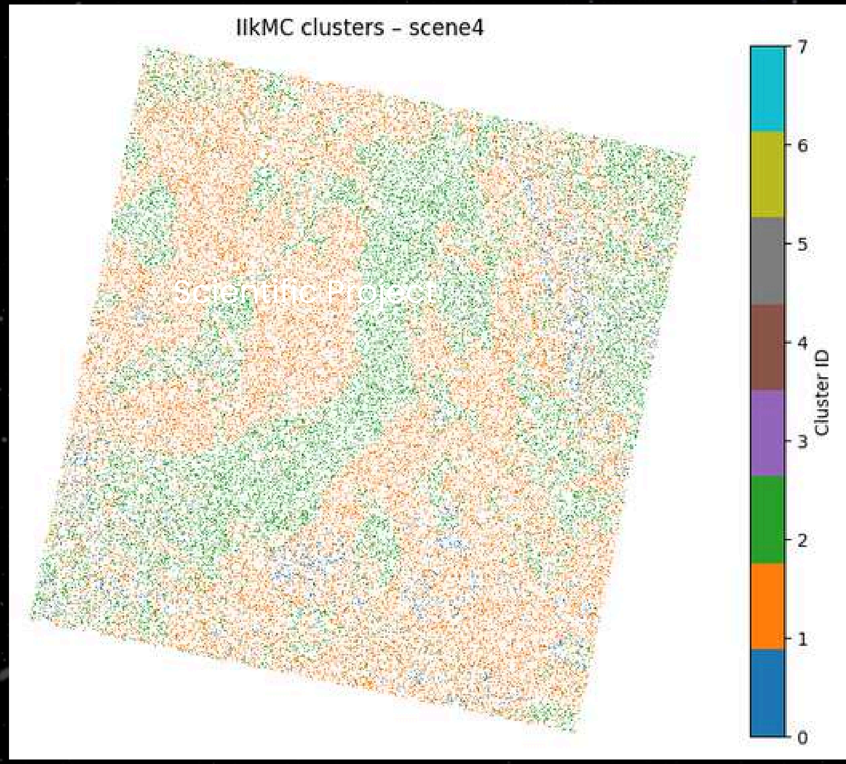
Cluster Maps (GPU)











Final Comparison

All benchmarks were performed on Kaggle's NVIDIA P100 GPU (12–16GB RAM) and 4vCPUs, using Landsat data with 20% downsampling ($N \approx 40.65M$, $B = 6$, $k = 8$).

Version	Time	Speedup	Silhouette
Seq	2494s	1x	0.48
CPU	301s	8.2x	0.47
GPU	47s	52.8x	0.49

Final Comparison

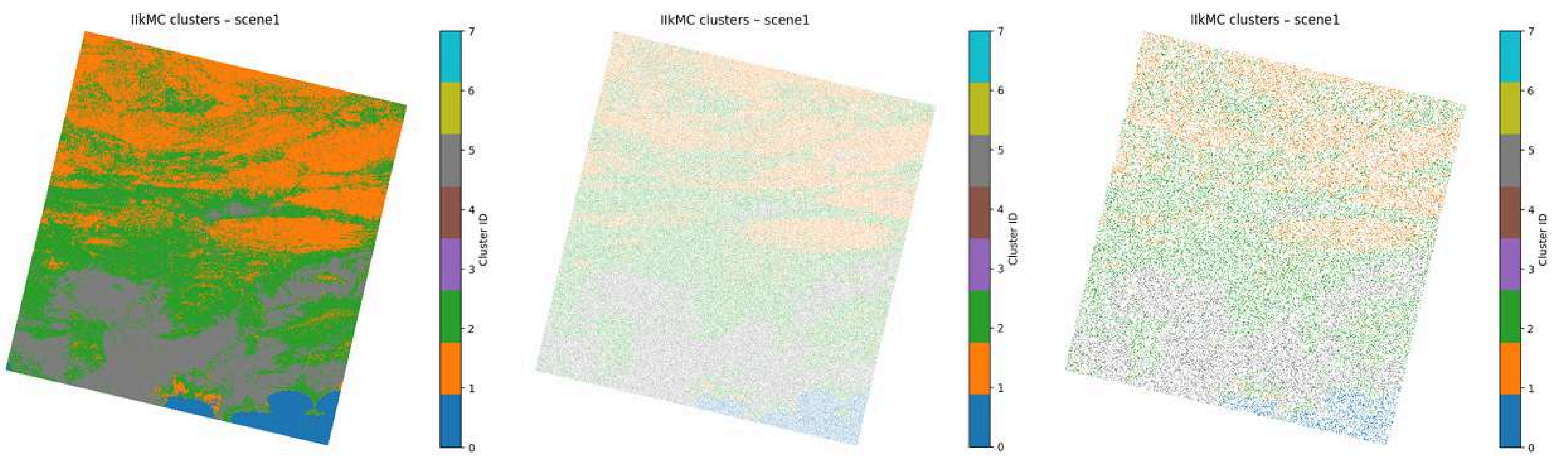


Figure 17: Cluster mapss. Left: full resolution; Middle: CPU; Right: GPU.

the higher mean silhouette score (0.49) achieved by the GPU implementation in dicates slightly improved cluster cohesion compared to CPU-parallel (0.47) and sequential (0.48) runs, likely due to more consistent numerical precision in the fused kernel.

Conclusion

- We successfully implemented a parallel version of the IlkMC algorithm using both CPU and GPU resources
- Compared to the baseline, our versions achieved 8.2× and 52.8× speedups respectively
- This enables practical deployment of temporally consistent unsupervised classification for remote sensing data.

ELEMENT SPACE

Planets

Astronomy

Space

THANK YOU

For Your Attention

Search

Scientific Project