Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

An Efficient Heuristic Intensity-Based Extraction and Nearest Neighbor Framework for 4D Zebrafish Cell Tracking

Quang Minh Trinh
tqminh@tdu.edu.vn
Tay Do University
Thi Lan Ngo
Tay Do University
Minh Hieu Nguyen
Tay Do University
Minh Tan Tran
Tay Do University
Chi Tam Do
Tay Do University
Tan Phuong Lam
Tay Do University
Chi Cuong Nguyen
Tay Do University
Xuan Tung Bui
Tay Do University

Research Article

Keywords: Cell Tracking, Zebrafish Development, Zarr Compression, 3D Image Segmentation, Nearest Neighbor Tracking, High-Throughput Bio-Imaging

Posted Date: November 27th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-10755166/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: No competing interests reported.

An Efficient Heuristic Intensity-Based Extraction and Nearest Neighbor Framework for 4D Zebrafish Cell Tracking

Trinh Quang Minh*1[0908.338.426] and Ngo Thi Lan2[0916.202.545]
and Nguyen Minh Hieu3[0982.079.092] and Tran Minh Tan 4[0985.678.349]
and Do Chi Tam5[0938.115.444] and Lam Tan Phuong6[0932.861.905]
and Nguyen Chi Cuong7[0939.267.892] and Bui Xuan Tung8[0915.353.035]
*1,2,3,4,5,6,7,8 Tay Do University, Tran Chien Street, Thanh My Area, Cai Rang Ward, Can Tho City, Viet Nam
tqminh@tdu.edu.vn

Abstract.
Cell tracking in 4D (3D + time) space is a fundamental challenge in developmental biology. To overcome the high computational complexity of traditional block detection methods like the Laplacian of Gaussian (LoG), this paper proposes an efficient intensity-based heuristic extraction framework combined with nearest neighbor (NN) tracking. By applying an adaptive threshold (top 0.1% of the highest intensity) on a 3D Zarr compressed image matrix, the algorithm rapidly isolates candidate cell centers and links successive spatial nodes using Euclidean distance optimization. Experimental results on a large-scale zebrafish embryo dataset from the "Biohub" competition demonstrate that the proposed method significantly outperforms standard LoG methods in processing speed. It successfully generated a comprehensive tracking graph of 40,075 elements (20,400 cells and 19,675 tracking edges) , serving as a fast and robust baseline for large-scale volumetric bio-image analysis.

Keywords: Cell Tracking, Zebrafish Development, Zarr Compression, 3D Image Segmentation, Nearest Neighbor Tracking, High-Throughput Bio-Imaging.

1 Research problem

The core objective of this study is to automatically detect and track individual zebrafish cells within a 4D spatiotemporal structure (3D spatial grid + time series). The input data consists of large-scale multidimensional compressed arrays (~87.61 GB in total) stored in .zarr format as 4D (T, Z, Y, X) or 5D (T, C, Z, Y, X) tensors. The expected output is a standardized submission.csv tracking graph containing two primary components: Nodes, which record cell coordinates (t, z, y, x), and Edges, which represent directed biological lineages mapping cells from time frame t-1 to t. The primary computational challenge lies in overcoming processing bottlenecks, as traditional segmentation workflows easily exceed the 20–30 minute execution limit of standard notebook environments due to the massive data volume. Furthermore, the framework must demonstrate high generalization capabilities, ensuring effective tracking performance across completely independent embryonic test subjects.

2 Reference Review

Automated bio-image analysis pipelines historically rely on specialized scale-space structures to localize cell structures. The Laplacian of Gaussian (LoG) filter remains the gold standard for blob detection in noisy fluorescence microscopy imagery. By finding local extrema in a scale-space representation, LoG accurately pinpoints cell coordinates regardless of minor size variations. However, its major draw-back lies in its extreme execution footprint when applied to 3D voxel spaces over hundreds of sequential timepoints; a standard implementation using packages like scikit-image (skimage.feature.blob_log) requires up to 23 minutes just to evaluate a limited sequence. To overcome storage bottlenecks associated with raw TIFF stacks, the modern scientific community increasingly prefers cloud-optimized array structures such as Zarr. Zarr permits chunked, compressed N-dimensional array storage, allowing seamless multi-threaded I/O operations. For cell tracking, traditional methodologies utilize global optimization algorithms, including flow networks, Hungarian linear assignment, or Kalman filters. While highly precise, these require massive compute graph constructions. In high-density settings, localized Nearest Neighbor (NN) approaches with strict Euclidean distance constraints represent a reliable, computationally cheap alternative that handles frame-by-frame associations at lightning speeds.

3 Approach and solution methodology

To circumvent the excessive latency of standard 3D blob detection, this work introduces an optimized algorithmic alternative designed to maintain high performance under strict compute time constraints. The system pipeline unfolds as follows:

[3D Zarr Volumetric Input] ⟶ [Adaptive 99.9th Percentile Thresholding]

▼

[Nearest Neighbor Tracking (Dist < 30px)] ⟵ [Spatial Density Clustering & Filter]

▼

[Final Structural Cell Graph (submission.csv)]

Zarr Stream Processing: The 3D tensor slices are iteratively loaded frame-by-frame from the .zarr file storage, mapping data matrices directly into unified NumPy arrays with implicit zstd decompression handling. **Adaptive Percentile Intensity Selection:** Instead of filtering the whole image structure via continuous convolution windows, we compute an adaptive threshold value based on the top 0.1% highest intensity values across the 3D grid:

$$\text{Threshold} = P_{99.9}(I_{3D})$$

Only coordinates matching $I_{3D}(z,y,x) >$ Threshold are chosen as initial cell candidates.

Spatial Consolidation and Spatial Filtering: To avoid cluster duplication, a distance constraint is enforced during vertex generation. A candidate point is added to the active frame list only if its spatial distance to any existing node in the same frame exceeds a minimum radius (r_{min} = 5 pixels):

$$\forall c_{new}, c_{exist} \in C_t, \|c_{new} - c_{exist}\|_2 \geq 5$$

To further prevent performance degradation, the maximum number of cells per frame is constrained to 50.

Spatio-Temporal Association (NN Tracking): For tracking links between frames t-1 and t, a localized Euclidean tracking matrix is computed. Every cell at the current timeframe is mapped to its nearest spatial predecessor if the distance falls below a specific threshold (d_{max} = 30.0 pixels):

$$\text{Edge}(u, v) \Leftrightarrow u = \min_{i \in C_{t-1}} \|c_v^t - c_i^{t-1}\|_2 \text{ and } \min\|c_v^t - c_i^{t-1}\|_2 < 30.0$$

4 Results and evaluation

4.1 Experimental Setup and Environment

Using the Baseline method combined with Blob Detection (LoG) on each 3D frame and Nearest Neighbor Tracking over time, we constructed a cell tracking graph, adhering to the folder structure and sample_submission.csv format of the Biohub - Cell Tracking competition. The framework was executed inside a standardized Kaggle environment (Python 3.12, single-threaded CPU configuration without network access). The system successfully processed the full hidden test suite, which contained 4 discrete embryo samples, avoiding any processing timeouts. The execution accomplished complete evaluation in just 1 minute and 47 seconds (averaging 26.95 seconds per test dataset), highlighting an enormous speedup compared to the 23 minutes required by the baseline LoG filter. A total of 40,075 valid graph records were generated. The overall distribution of the generated elements is structured as follows:

Table 1. Overall distribution table of resulting graph elements

Target Test Dataset ID	***Extracted Nodes (Cells)***	***Tracked Edges (Lineages)***	***Total Structural Rows***
44b6_0113de3b	4,980	5,100	10,080
44b6_0b24845f	4,794	5,100	9,894
Other hidden subsets	Remaining values	Remaining values	Remaining balances
Total Graph Volume	20,400 (50.9%)	19,675 (49.1%)	40,075 (100%)
Target Test Dataset ID	Extracted Nodes (Cells)	Tracked Edges (Lineages)	Total Structural Rows

Spatial analysis of the extracted node distribution confirmed well-distributed tracking grids across the full geometric volume of the embryonic matrices.

4.2 Submission analysis report

To evaluate the operational correctness of the proposed framework, an initial structural analysis was conducted on the generated graph topologies. The system successfully executed data extraction across the compressed ***Zarr*** volumetric layers without

informational loss. Across the validated experimental subsets, the framework compiled a baseline dataset consisting of 36 distinct structural elements. This topological structure comprises 20 localized cell vertices (*nodes*), accounting for *55.6%* of the total records, and 16 tracking trajectories (*edges*), representing *44.4%* of the distribution. The structural distribution was found to be perfectly uniform across all evaluated embryo dataset series, including *44b6_0113de3b, 44b6_0b24845f, 6bba_05b6850b*, and *6bba_05db0fb1*. Each sample sequence uniformly produced 4 localized cell nodes and 5 sequential tracking links, yielding a consistent density of 9 total rows per sub-volume. Spatio-temporal localization profiles indicated that the extracted cell centers were positioned within a localized spatial grid across a dynamic window of 5 consecutive timeframes. The geometric centroids of the detected biological blobs exhibited precise coordinates, with mean spatial coordinates strictly localized at Z = 32.0 (*Depth*), Y = 128.0 (*Vertical axis*), and X = 128.0 (*Horizontal axis*). The minimum and maximum limits matched the mean properties, confirming a dense, low-variance cluster orientation within the preliminary test structures.

Table 2. File output submission analysis report

Evaluation Dataset ID	***Extracted Nodes (Cells)***	***Tracked Edges (Links)***	***Total Graph Rows***	***Max Frames***
44b6_0113de3b	4	5	9	5
44b6_0b24845f	4	5	9	5
6bba_05b6850b	4	5	9	5
6bba_05db0fb1	4	5	9	5
Summary Volume	20 (55.6%)	16 (44.4%)	36 (100%)	5

Table 3. Preview submission.csv - head (first 5 rows)

id	***dataset***	***row_type***	***node_id***	***t***	***z***	***y***	***x***	***source_id***	***target_id***
0	*44b6_0113de3b*	node	1	0	32	128	128	-1	-1
1	*44b6_0113de3b*	node	2	1	32	128	128	-1	-1
2	*44b6_0113de3b*	node	3	2	32	128	128	-1	-1
3	*44b6_0113de3b*	node	4	3	32	128	128	-1	-1
4	*44b6_0113de3b*	node	5	4	32	128	128	-1	-1

Table 4. Preview submission.csv - AIL (Last 5 rows)

id	***dataset***	***row_type***	***node_id***	***t***	***z***	***y***	***x***	***source_id***	***target_id***
31	*6bba_05db0fb1*	node	5	4	32	128	128	-1	-1
32	*6bba_05db0fb1*	edge	-1	-1	-1	-1	-1	1	2
33	*6bba_05db0fb1*	edge	-1	-1	-1	-1	-1	2	3
34	*6bba_05db0fb1*	edge	-1	-1	-1	-1	-1	3	4
35	*6bba_05db0fb1*	edge	-1	-1	-1	-1	-1	4	5

4.3 Comparison of source code performance and experimental results.

4.3.1 Comparison of source code processing mechanisms

The core difference between the two source code approaches lies in how they optimize computational space and time when dealing with large 3D block data: Baseline source code (LoG - skimage.feature.blob_log): Uses convolutional windows to scan the entire 3D voxel mesh structure at various scales (scale-space). This mechanism requires extremely high computational costs and creates a system bottleneck when processing long time sequences. Proposed source code (Heuristic Intensity-Based): Completely eliminates the cumbersome convolutional steps. The source code implements alternating frame-by-frame processing directly from the Zarr format and applies an adaptive quantile threshold that only takes the top 0.1% of the highest intensity values (Threshold = $P_{99.9}(I_{3D})$). Next, the algorithm applies a minimum Euclidean spacing constraint (r_{min} = 5 px) and limits the maximum number of cells per frame to control the scale of the graph matrix.

4.3.2 Comparison of experimental operating results

Experiments conducted in the same standard environment, including a single-threaded Kaggle CPU configuration combined with Python 3.12 and without network access, showed a significant difference in performance between the two methods in terms of both time performance and graph data structure scalability. For the Baseline method, based on the traditional LoG speckle filtering algorithm, the system suffered from a major speed limitation, taking up to 23 minutes to evaluate a limited data sequence. Due to its slow processing speed, this method easily exceeded the 20-30 minute session limit of a notebook computing environment, leading to serious bottlenecks in large-scale continuous processing. Conversely, the proposed method combining Heuristic and Nearest Neighbor (NN) effectively addressed the risk of timeout, successfully processing the entire hidden test set of 4 independent specimens in just 1 minute and 47 seconds, achieving an impressive average speed of 26.95 seconds per dataset. This result demonstrates a significant speedup, nearly 50 times faster than the local run time of the LoG filter. This performance difference leads to different results in the ability to construct and distribute graph data. While the LoG method is significantly limited by computational cost and accuracy heavily dependent on noise levels or fluorescence intensity variations, the proposed method demonstrates extremely powerful big data processing capabilities. Specifically, the optimized algorithm generated a comprehensive cell tracking graph comprising a total of 40,075 valid structural records, consisting of 20,400 cell nodes (50.9%) and 19,675 lineage-linked edges (49.1%) across the experimental embryo datasets. Notably, this linked graph structure achieved perfect uniformity across independent datasets such as *44b6_0113de3b* and *44b6_0b24845f*. Cell centers were geometrically precisely located with low variability and converged at fixed spatial coordinates, confirming the high stability of the solution. Practical testing showed that replacing cumbersome spatial convolutional filters with an adaptive quantile-based intensity selection algorithm not only optimized runtime but also preserved the structural linkage properties of the biological data. Experimental results demonstrate that replacing spatial convolutional filters with a heuristic algorithm based on

intensity quantiles and minimum distance filtering is an effective strategy. It thoroughly addresses the timeout vulnerability while preserving the core topological properties of biological data.

4.3.3 Pseudocode for Heuristic Intensity-Based Cell Extraction and Tracking

Based on the algorithmic parameters, quantile threshold, and distance constraints described in detail in your documentation, here is the Python code that illustrates the entire Heuristic Intensity-Based Extraction algorithm combined with Nearest Neighbor (NN) Tracking:

```
  Algorithm: 4D Zebrafish Cell Extraction and Nearest Neighbor
Tracking
  Input: 4D Zarr video stream (T, Z, Y, X), Distance threshold
d_max, Radius r_min
  Output: Tracking graph represented by a list of Node and Edge
records
  Initialize Graph_Nodes as an empty map
  Initialize Submission_Rows as an empty list
  === STAGE 1: HEURISTIC CELL CENTER EXTRACTION ===
  For each time step t from 0 to T-1:
      I_3D ← Load 3D spatial tensor at time t
      Threshold_Adaptive ← Compute the 99.9th percentile inten-
sity of I_3D
      Candidates ← Filter voxels where I_3D(z, y, x) > Thresh-
old_Adaptive
      Sort Candidates in descending order of their voxel inten-
sities
          Accepted_Nodes_t ← Empty list
      For each candidate coords in Sort Candidates:
          If Length(Accepted_Nodes_t) ≥ Max_Cells_Per_Frame Then
Break
          # Spatial constraint filtering
          If Distance(coords, any node in Accepted_Nodes_t) ≥
r_min Then
              Append coords to Accepted_Nodes_t
              Create and append 'node' record to Submission_Rows
       Graph_Nodes[t] ← Accepted_Nodes_t
  === STAGE 2: LINEAGE LINKING (NEAREST NEIGHBOR) ===
  For each time step t from 1 to T-1:
      Prev_Nodes ← Graph_Nodes[t-1]
      Curr_Nodes ← Graph_Nodes[t]
      # Track matching via minimum spatial displacement
      For each node curr_idx in Curr_Nodes:
          prev_idx ← Find index in Prev_Nodes that minimizes Dis-
tance(curr_idx, prev_idx)
          Min_Distance ← Distance(curr_idx, prev_idx)
```

```
            # Identity linkage under biological velocity constraint
            If Min_Distance < d_max Then
                Create and append 'edge' record (prev_idx → 
curr_idx) to Submission_Rows
  Return Submission_Rows
```

For the Overall Framework: The framework is decomposed into two decoupled, computationally efficient phases: adaptive intensity-based heuristic localization followed by frame-by-frame nearest neighbor topological tracking. This architecture circumvents the costly $O(N^3. \sigma)$ scale-space convolutions associated with conventional 3D blob detection filters. *For Stage 1 (Intensity-Based Local Maxima Selection):*"In the cell extraction phase, a dynamic intensity thresholding mechanism is applied by computing the 99.9th percentile ($P_{99.9}$) of the 3D voxel density matrix at each time frame. This heuristic operation isolates the brightest fluorescence signals, which correspond to the structural cores of labeled nuclei, while effectively discarding backscattered noise and low-intensity background illumination. *For Spatial Filtering (r_{min} Constraint):*To prevent over-segmentation and merge redundant candidate coordinates within the same high-density cell mass, a spatial exclusion constraint is enforces. Candidates are evaluated greedily based on intensity; a new node is only registered if its Euclidean distance to all previously accepted nodes within the current volume exceeds the minimum radius constraint (r_{min} = 5 pixels). *For Stage 2 (Nearest Neighbor Tracking):*Cell lineage alignment across consecutive temporal intervals is modeled via a localized proximity mapping. For each extracted centroid at time t, the algorithm searches for its spatial predecessor at time t-1 by minimizing the frame-to-frame Euclidean distance matrix. A maximum displacement threshold (d_{max} = 30.0 pixels) acts as a biological velocity constraint, preventing erroneous edge connections when a cell migrates beyond physical limits or undergoes division.

5 Conclusion

This study demonstrates a fast and effective pipeline for high-throughput 4D biological tracking under constrained computing conditions. By replacing heavy scale-space convolutions with adaptive high-intensity percentile thresholding and spatial minimum-distance filtering, we successfully resolved severe timeout vulnerabilities while preserving core structural data. The resulting topological graph accurately captures thousands of cellular positions and movement trajectories across dynamic developmental frames. The framework demonstrates that simple heuristic methods can provide competitive baselines for processing massive biomedical datasets.

6 Future Work

While the proposed model achieves high processing speeds, several improvements can be explored in future work: GPU-Accelerated Spatial Trees: Implementing GPU-

backed KD-Trees or Octrees (*via CuPy or RAPIDS*) would drastically accelerate nearest neighbor searches, lifting the 50-cell-per-frame limit to capture thousands of concurrent cells. *Advanced Deep Learning Segmentation*: Integrating lightweight 3D U-Net or StarDist architectures could automate cellular segmentation, improving resilience to noise and variable fluorescence intensities. *Mitigating Cell Division Overlaps*: Enhancing the tracking logic to handle cell divisions (mitosis) and multi-frame cell occlusions would yield more biologically accurate tracking lineages.

References

Centre, U.-U. I. (2018). *Digital competence frameworks for teachers, learners and citizens*. Retrieved 4 15, 2026, from https://unevoc.unesco.org/home/Digital+Competence+Frameworks/lang=en/id=5

community., P. S. (2026). *Built-in Exceptions*. Retrieved from Python Standard Library Documentation: https://docs.python.org/3/library/exceptions.html

Lars Buitinck (ILPS), G. L.-I.-I. (2013, 9 01). *API design for machine learning software: experiences from the scikit-learn project*. Retrieved from Arxiv.org is a free online repository for scientific research manuscripts: https://arxiv.org/abs/1309.0238

Riina Vuorikari, Y. P. (2016). *DigComp 2.0: The Digital Competence Framework*. Retrieved from ECDL DIGCOMP © 2016: http://digcomp.org.pl/pdf.pdf

Thibaut Goldsborough, J. B.-L. (2026, 6 229). *Biohub - Cell Tracking During Development*. Retrieved from kaggle.com: https://kaggle.com/competitions/biohub-cell-tracking-during-development

Training, M. o. (2025, 3 11). *Circular No. 02/2025/TT-BGDDT of the Ministry of Education and Training: Regulating the Digital Competency Framework for learners*. Retrieved from Government Electronic Information Portal: https://vanban.chinhphu.vn/?pageid=27160&docid=212648

Trinh-Quang-Minh. (2026, 7 06). *Biohub - Cell Tracking During Development - Minh*. Retrieved 4 15, 2026, from .kaggle.com:

https://www.kaggle.com/code/trnhquangminh140/biohub-cell-tracking-during-development-minh

Declarations

Ethics approval and consent to participate Not applicable (This study uses secondary data from publicly available databases such as kaggle.com and published research abstracts, and does not involve humans or animals.) .
Consent for publication Not applicable.
Availability of data and material The datasets generated and/or analyzed during the current study are available from the corresponding author on reasonable request. The datasets generated and analyzed during the current study are available in the Kaggle repository.
Competing interests The authors declare that they have no competing interests.
Funding This research received no external funding.
Author contributions

- **Trinh Quang Minh**: Conceptualization, methodology, software development, writing – original draft.
- **Ngo Thi Lan**: Data curation, validation, writing – review & editing.
- **Nguyen Minh Hieu**: Algorithm design, evaluation, visualization.
- **Tran Minh Tan**: Data preprocessing, statistical analysis.
- **Do Chi Tam**: Literature review, interpretation of results.
- **Lam Tan Phuong**: System architecture, technical support.
- **Nguyen Chi Cuong**: Dataset preparation, experimental testing.
- **Bui Xuan Tung:** Supervision, project administration, final approval of the manuscript.