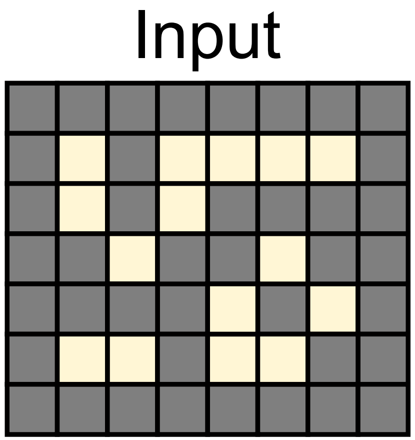
LSL3D: a run-based CCL algorithm for 3D volumes

COMPAS 2022

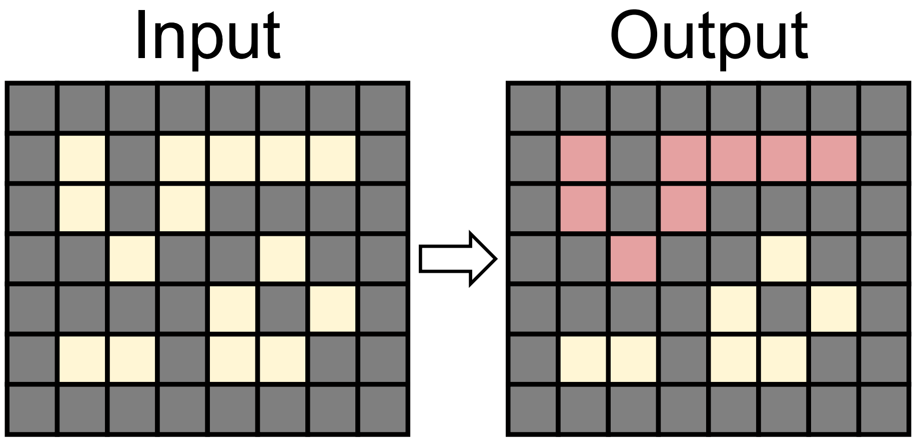
Nathan Maurice, Florian Lemaitre, Julien Sopena, Lionel Lacassagne

July 6, 2022

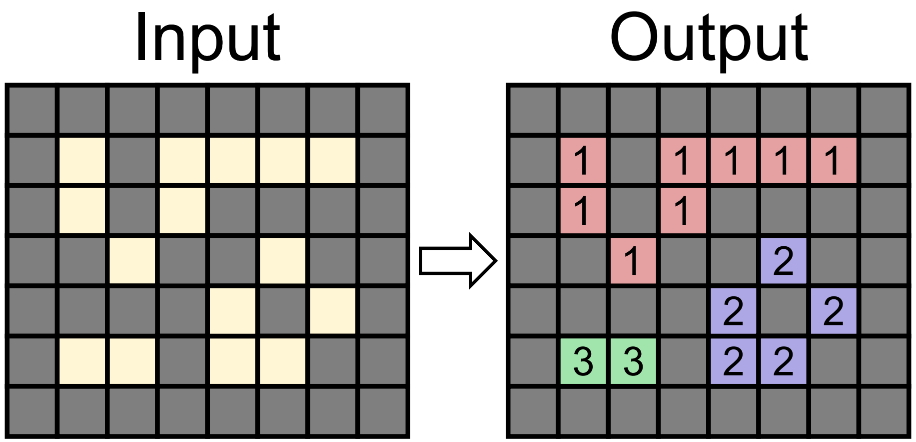
Connected Component Labeling



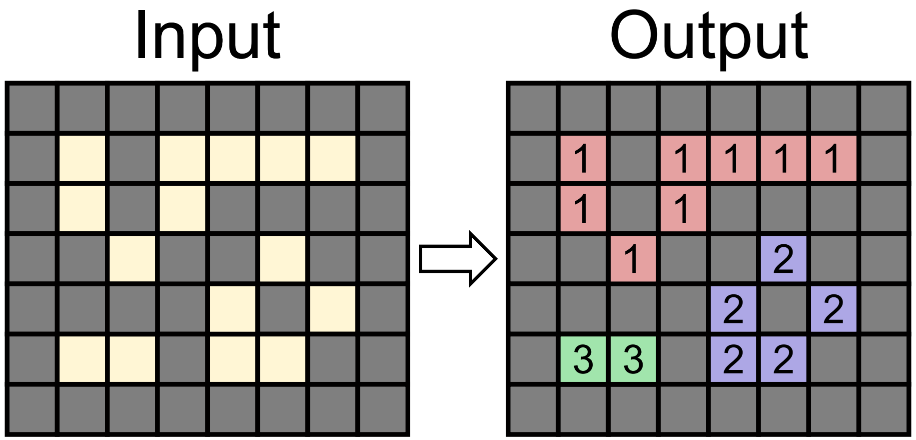
Connected Component Labeling



Connected Component Labeling



Connected Component Labeling



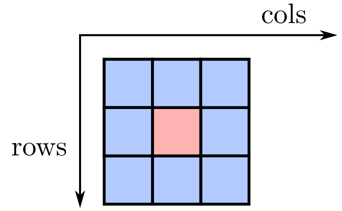
Applications: autonomous driving, biology, pre-processing for *AI*

Goals: ⇒ Performance: for real-time applications
⇒ Regularity: reduce sensitivity to image type

State of the Art: Pixel-based 2D

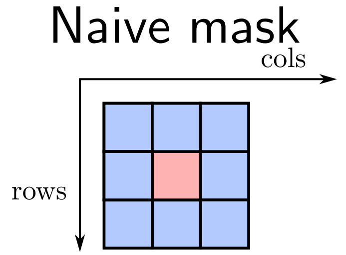
Naive approach: test all neighbouring pixels

Naive mask

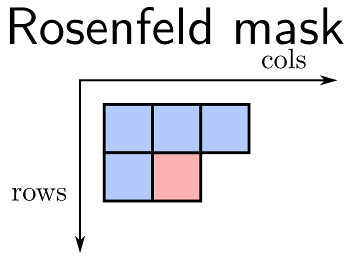


State of the Art: Pixel-based 2D

Naive approach: test all neighbouring pixels

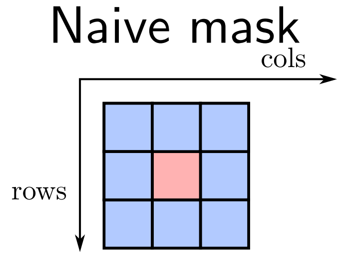


Pixel-based: Rosenfeld [1], *SAUF* [2], *LEB* [3], *PRED* [4]

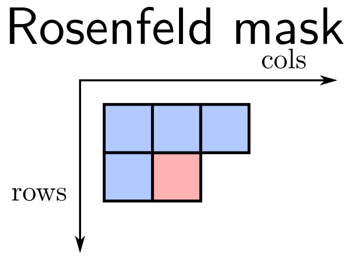


State of the Art: Pixel-based 2D

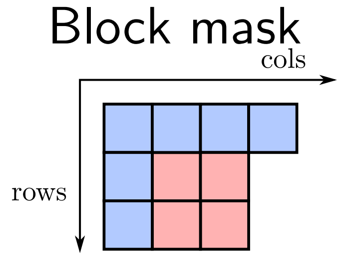
Naive approach: test all neighbouring pixels



Pixel-based: Rosenfeld [1], *SAUF* [2], *LEB* [3], *PRED* [4]



Block-based: *BBDT* [7], *Spaghetti* [8]



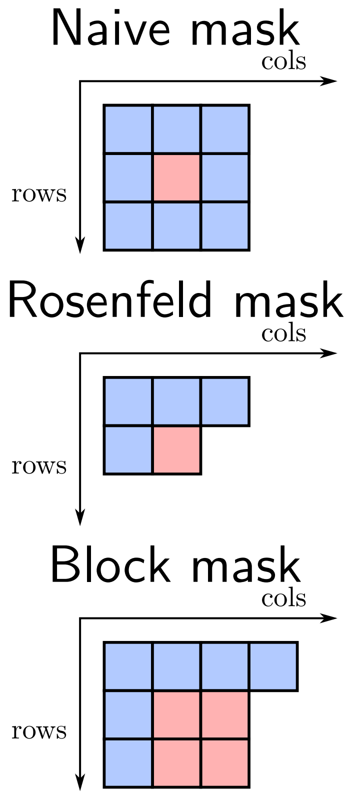
State of the Art: Pixel-based 2D

Naive approach: test all neighbouring pixels

Pixel-based: Rosenfeld [1], *SAUF* [2], *LEB* [3], *PRED* [4]

Block-based: *BBDT* [7], *Spaghetti* [8]

3D algorithms: usually extensions of 2D algorithms



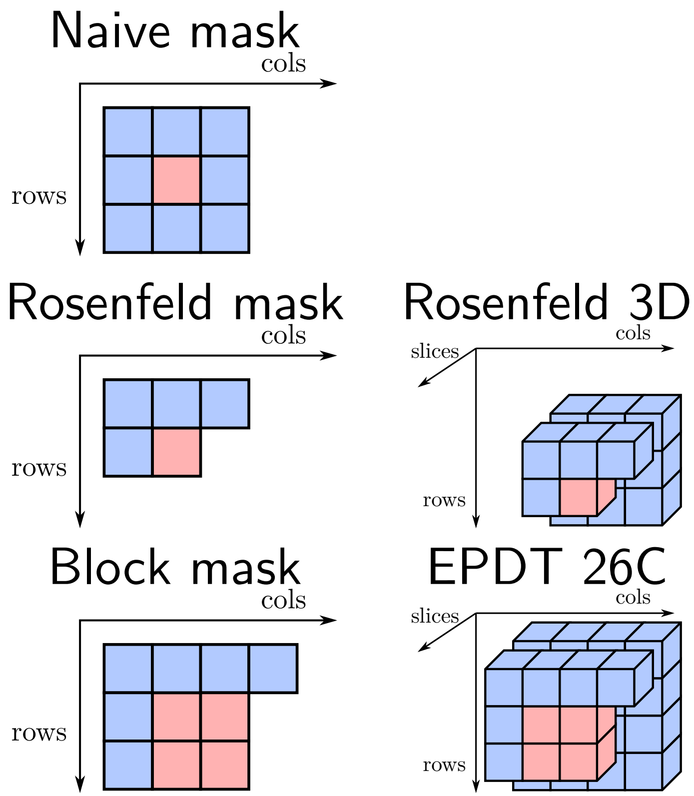
State of the Art: Pixel-based 2D→ 3D

Naive approach: test all neighbouring pixels

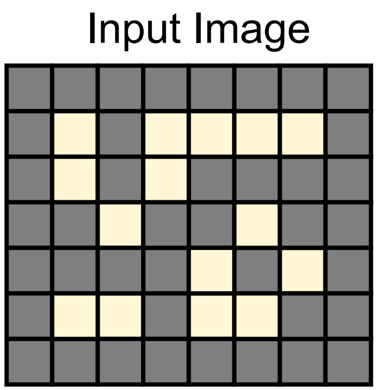
Pixel-based: Rosenfeld [1], *SAUF* [2], *LEB* [3], *PRED* [4]
⇒ Rosenfeld 3D, *SAUF 3D* [5], *LEB 3D* [6], *PRED 3D* [5]

Block-based: *BBDT* [7], *Spaghetti* [8]
⇒ *EPDT* (19C, 22C, 26C) [9]

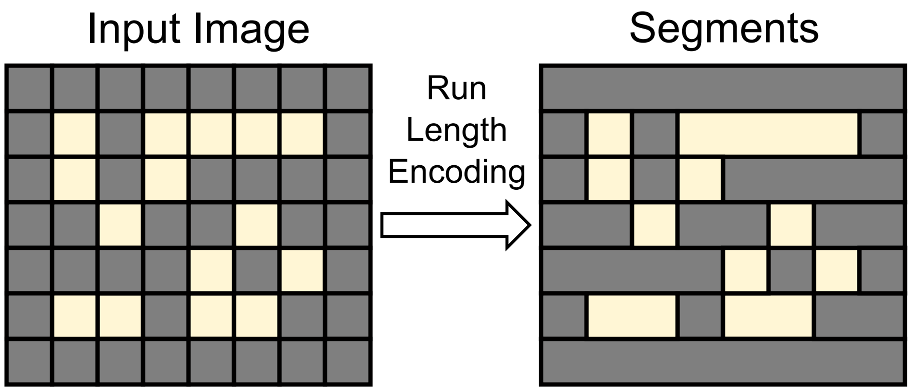
3D algorithms: usually extensions of 2D algorithms



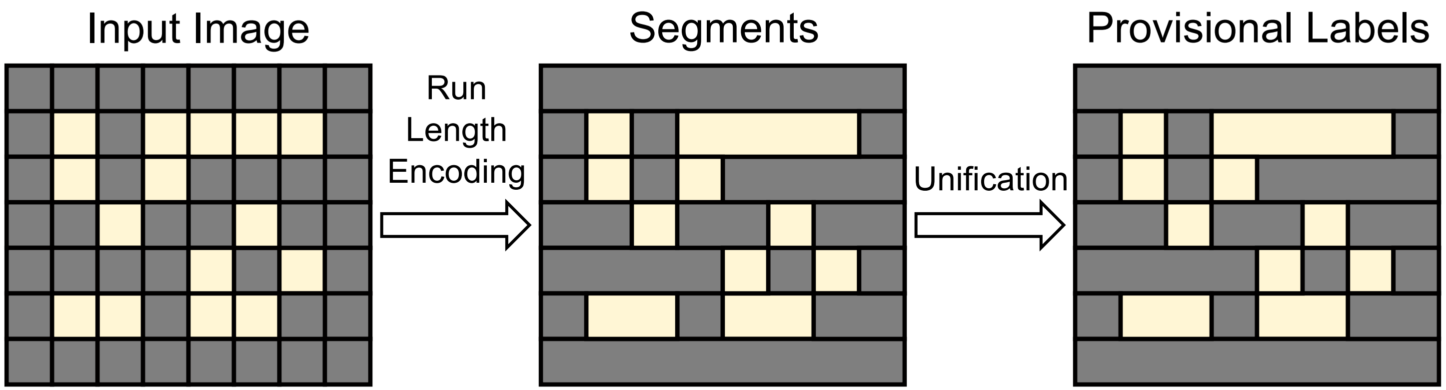
State of the Art (segment-based)



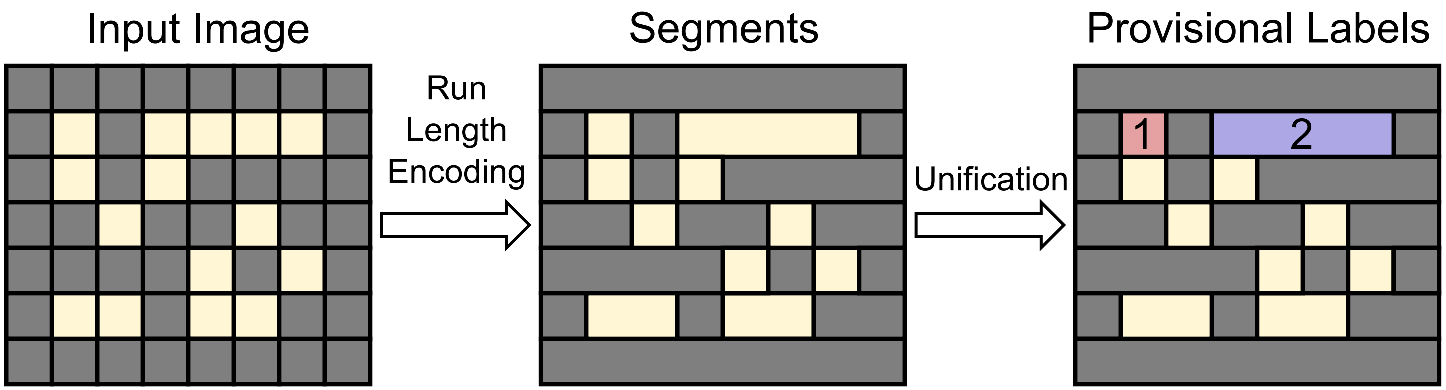
State of the Art (segment-based)



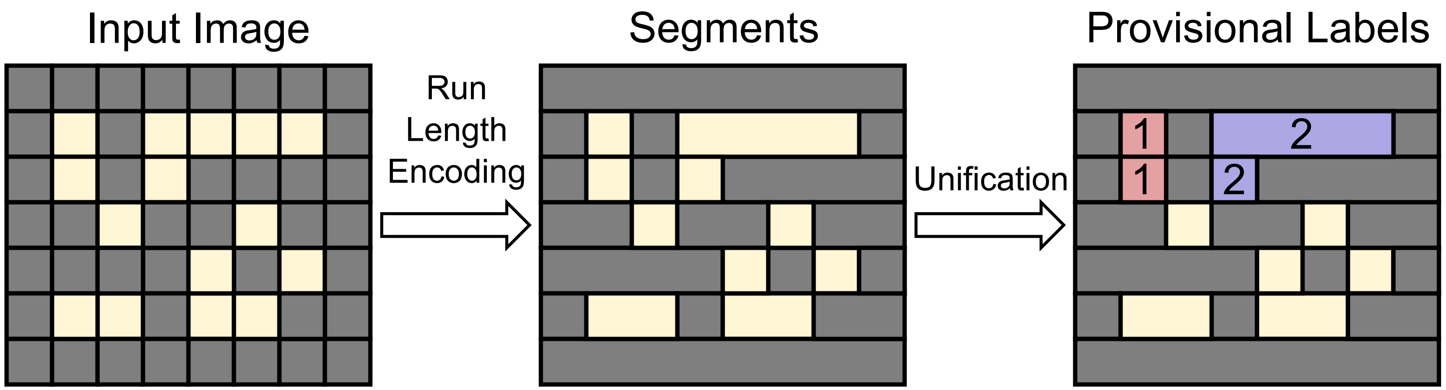
State of the Art (segment-based)



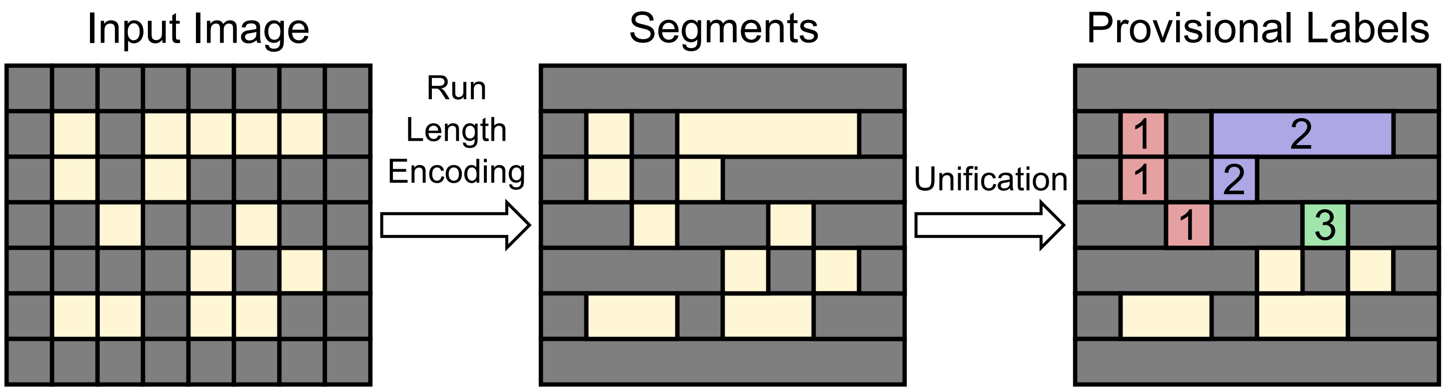
State of the Art (segment-based)



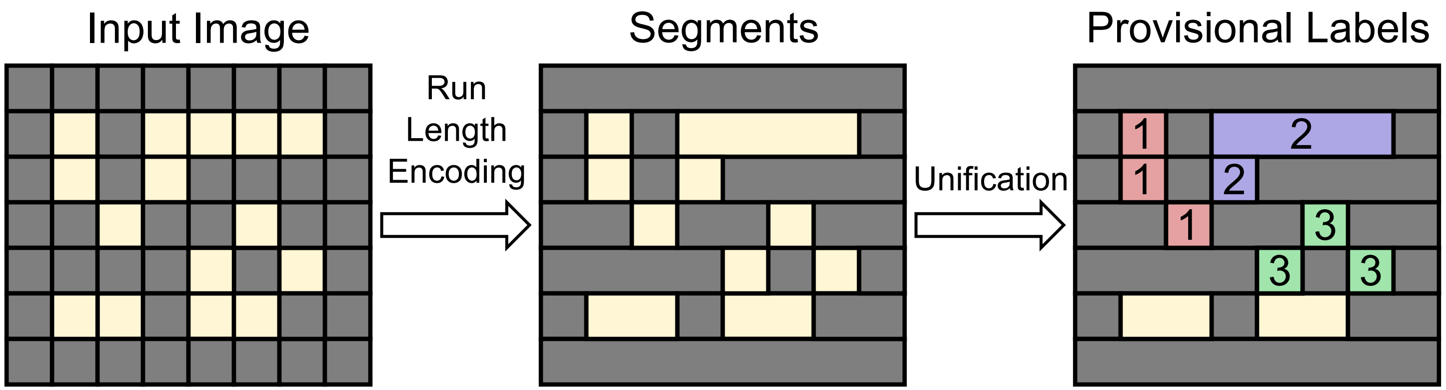
State of the Art (segment-based)



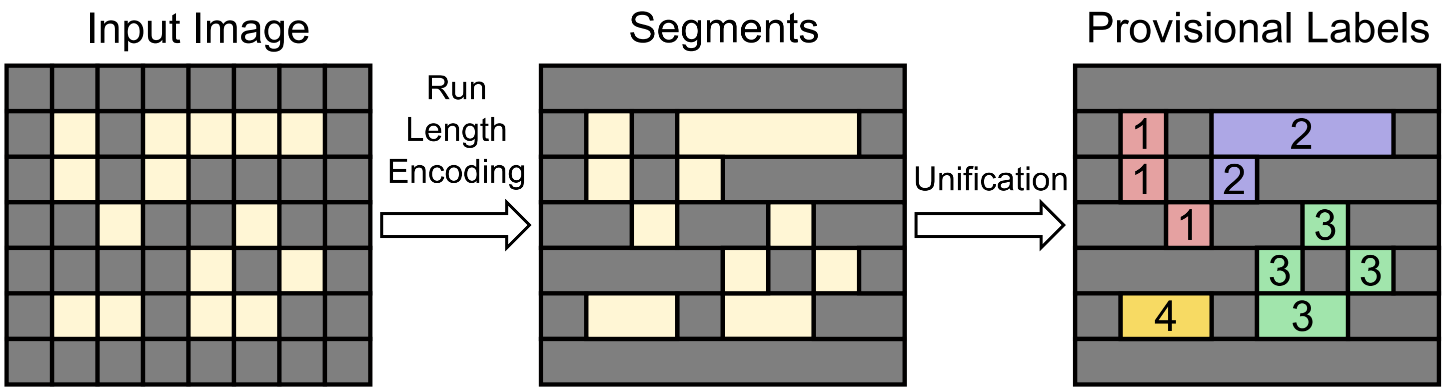
State of the Art (segment-based)



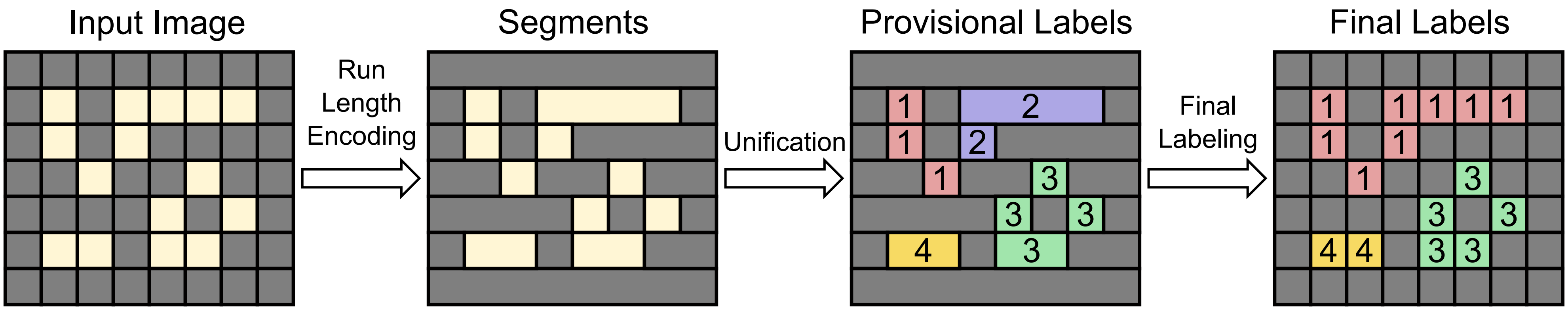
State of the Art (segment-based)



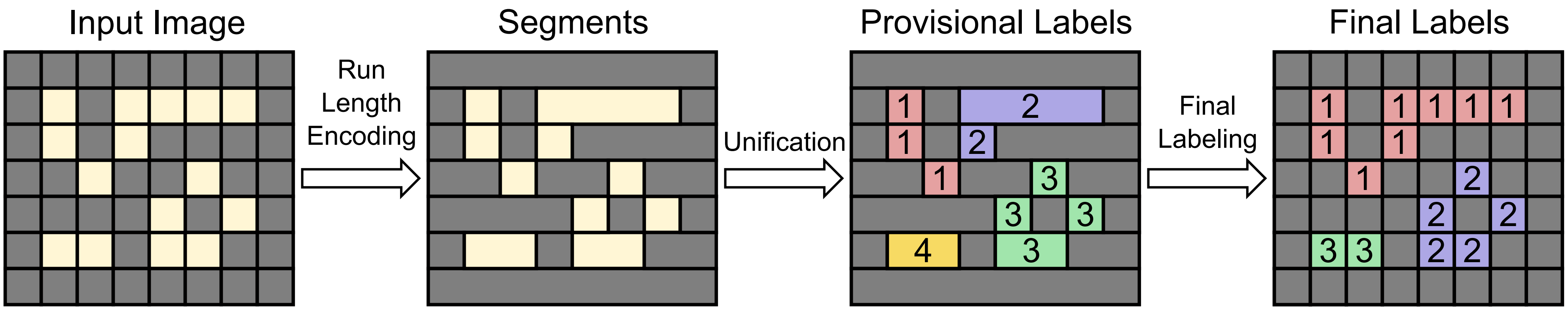
State of the Art (segment-based)



State of the Art (segment-based)

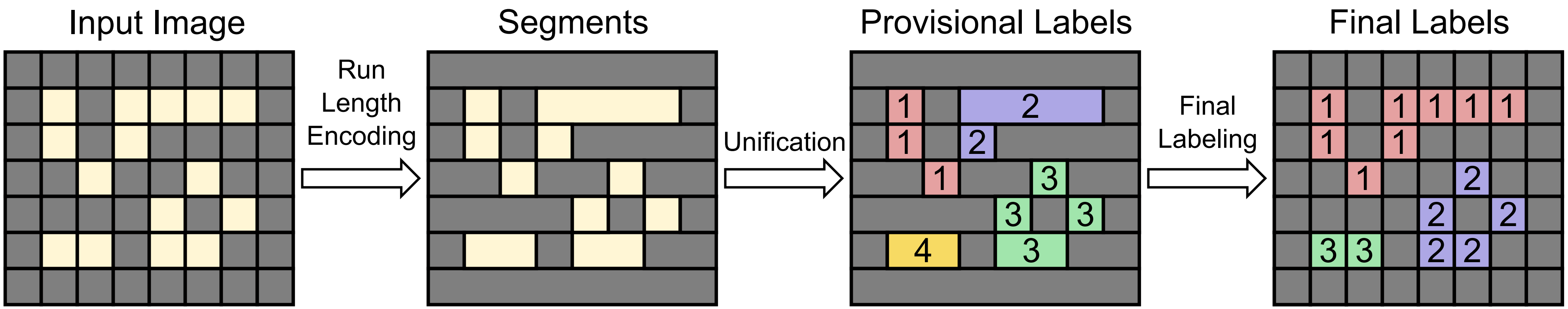


State of the Art (segment-based)



Segment-based: *RBTS* [10] ⇒ *RBTS 3D* [6]
LSL [11][12] ⇒ *LSL3D* is missing

State of the Art (segment-based)



Segment-based: *RBTS* [10] ⇒ *RBTS 3D* [6]
LSL [11][12] ⇒ *LSL3D* is missing

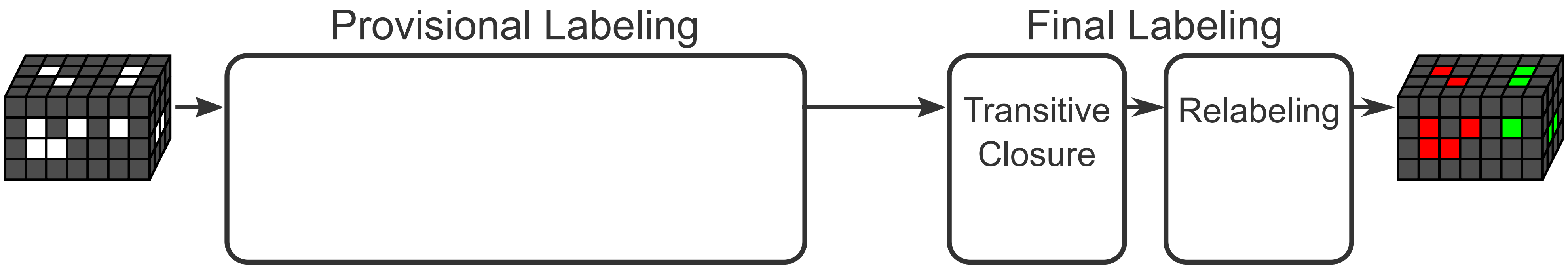
This contribution: LSL3D, a new segment-based algorithm

Step 1: Extension of *LSL* to 3D images

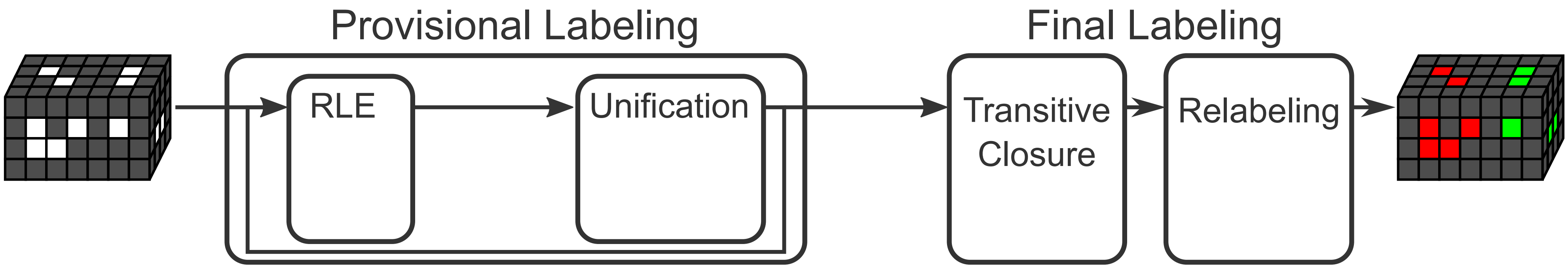
Step 2: Segment overlap detection with Finite State Machine (FSM)

Step 3: Computational re-use & simplification of FSM

Algorithm structure: Direct algorithms



Algorithm structure: LSL3D



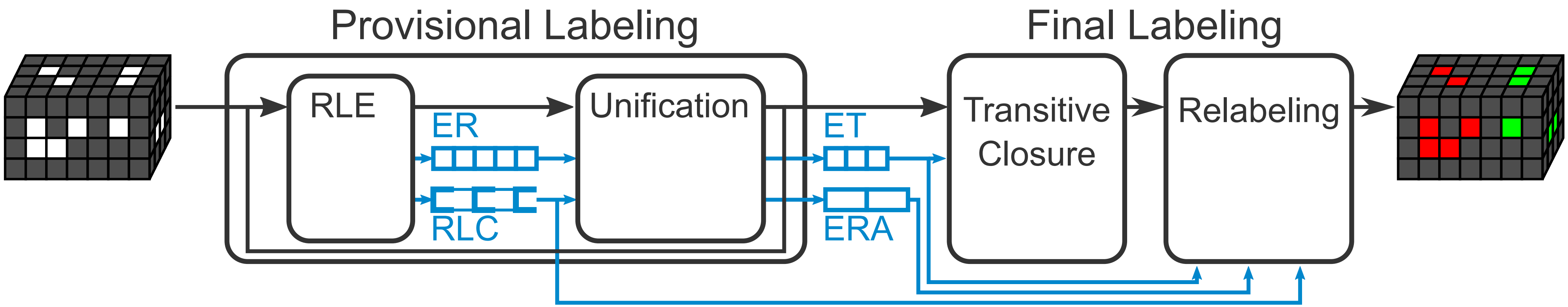
Run-Length Encoding (RLE) algorithm: pixels → segments

Unification: segments → provisional labels

Transitive Closure: provisional labels → final labels

Relabeling: write final labels

Algorithm structure: LSL3D



Run-Length Encoding (RLE) algorithm: pixels → segments

⇒ store segments (start & end) into RLC table & ER table (pixel pos. → segment id)

Unification: segments → provisional labels

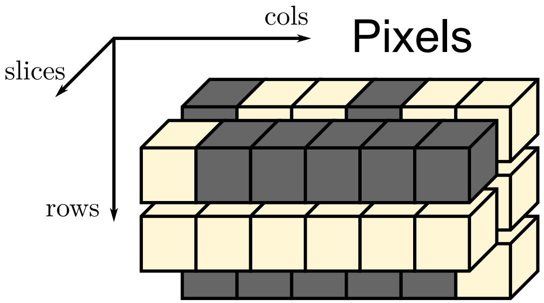
⇒ detect segments overlaps between lines, store provisional labels into ERA table

Transitive Closure: provisional labels → final labels

Relabeling: write final labels

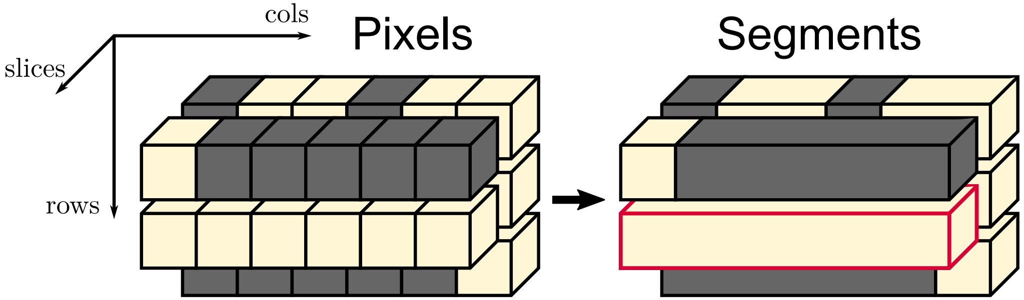
Step 1: Extension of LSL to 3D volumes

RLE algorithm: Same as in 2D
Unification 3D: between 5 lines (vs 2 in 2D)



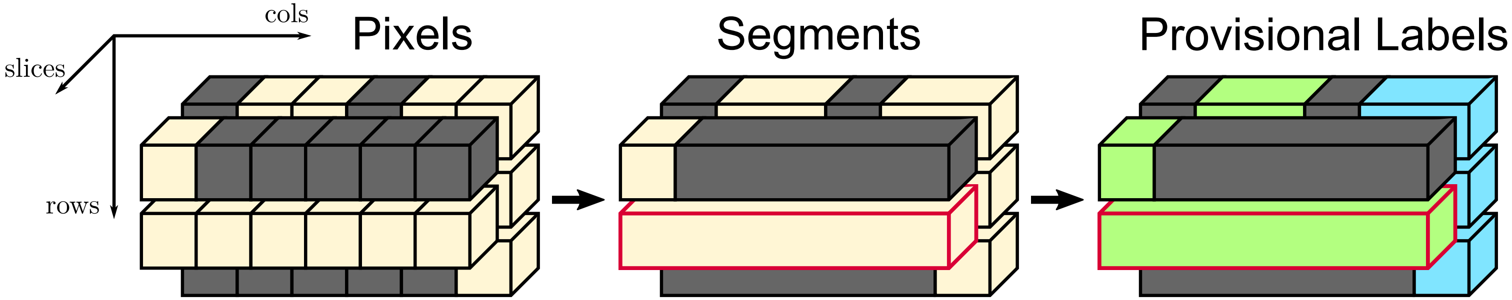
Step 1: Extension of LSL to 3D volumes

RLE algorithm: Same as in 2D
Unification 3D: between 5 lines (vs 2 in 2D)



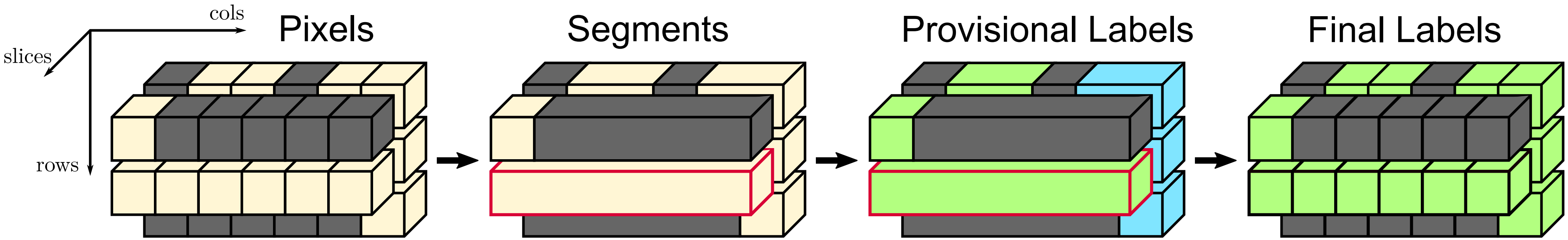
Step 1: Extension of LSL to 3D volumes

RLE algorithm: Same as in 2D
Unification 3D: between 5 lines (vs 2 in 2D)



Step 1: Extension of LSL to 3D volumes

RLE algorithm: Same as in 2D
Unification 3D: between 5 lines (vs 2 in 2D)

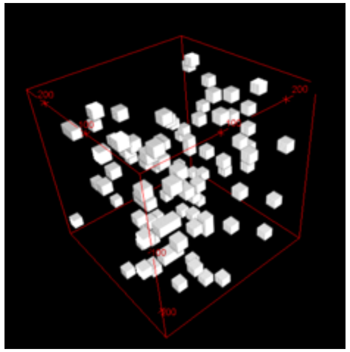


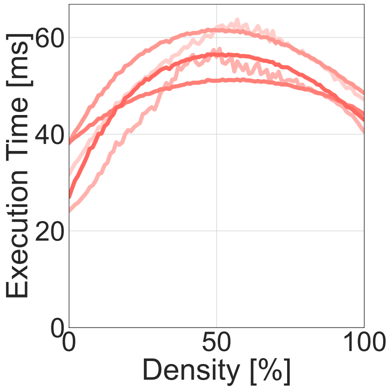
Random datasets: State of the Art

Benchmark: *YACCLAB* [13]
Hardware: Xeon Gold 6126

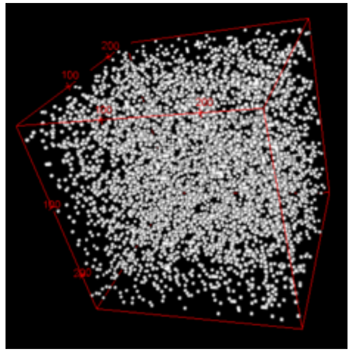
density $= \frac{\text{foreground pixels}}{\text{image size}}$
granularity $=$ cube size

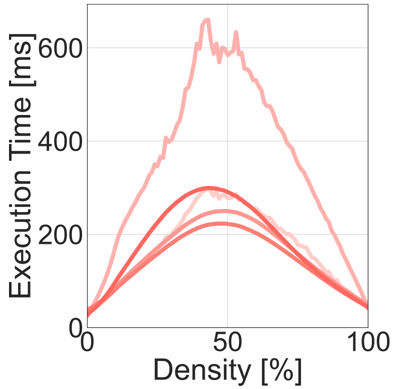
g = 16
(simple)





g = 1
(complex)





- LEB_3D
- RBTS_3D
- SAUFpp_3D
- PREDpp_3D
- EPDT_3D_22c

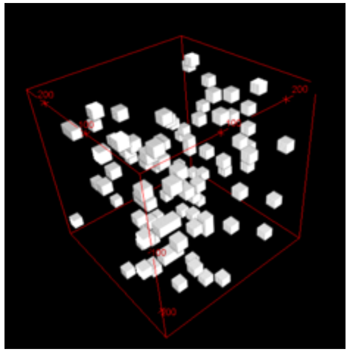
Random datasets: LSL3D

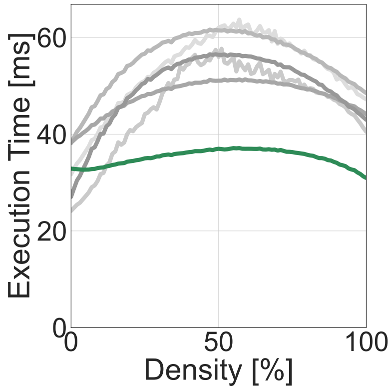
Benchmark: *YACCLAB* [13]
Hardware: Xeon Gold 6126

density $= \frac{\text{foreground pixels}}{\text{image size}}$
granularity $=$ cube size

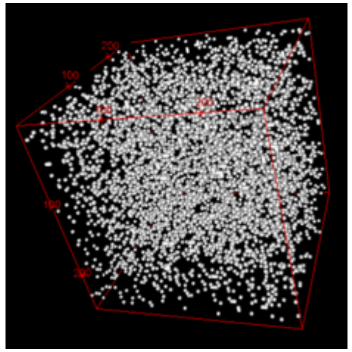
- LEB_3D
- RBTS_3D
- SAUFpp_3D
- PREDpp_3D
- EPDT_3D_22c
- LSL_ER

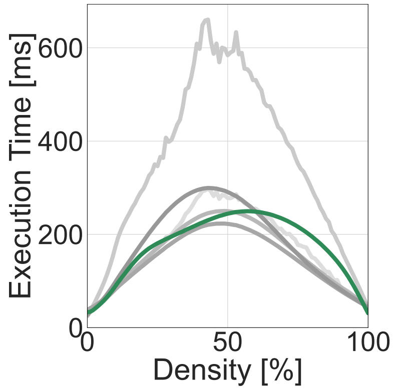
g = 16
(simple)





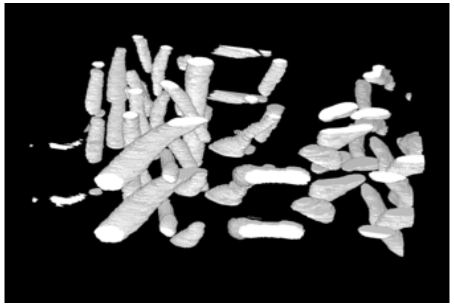
g = 1
(complex)





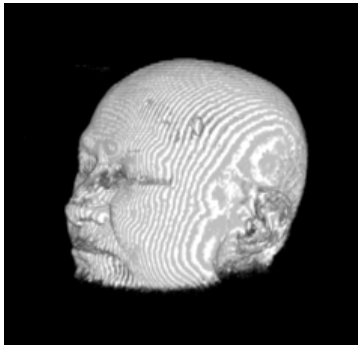
Medical datasets

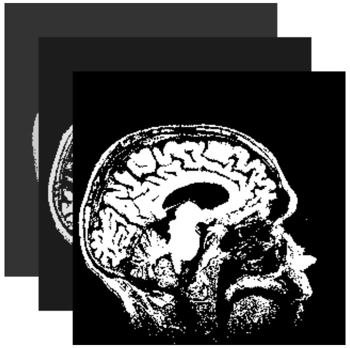
Mitochondria
(3 × 3D images)
1024 × 768 × 165



density = 5.8%
run/MPxls = 1500
CCs/MPxls = 0.31
⇒ simple images

OASIS
(373 × 3D images)
256 × 256 × 128

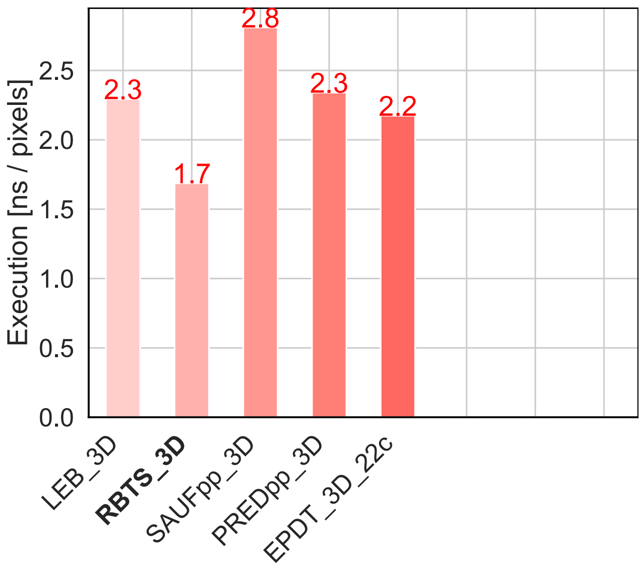


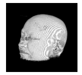


density = 19.8%
run/MPxls = 28 000
CCs/MPxls = 380
⇒ complex images

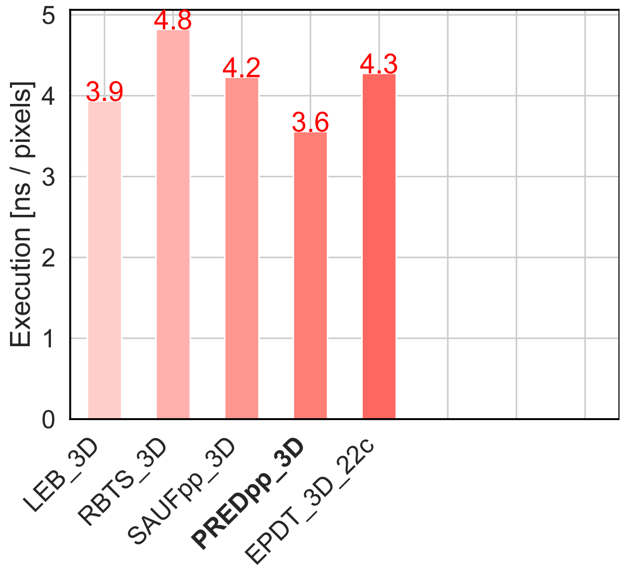
Medical datasets: State of the Art

Mitochondria
(simple)



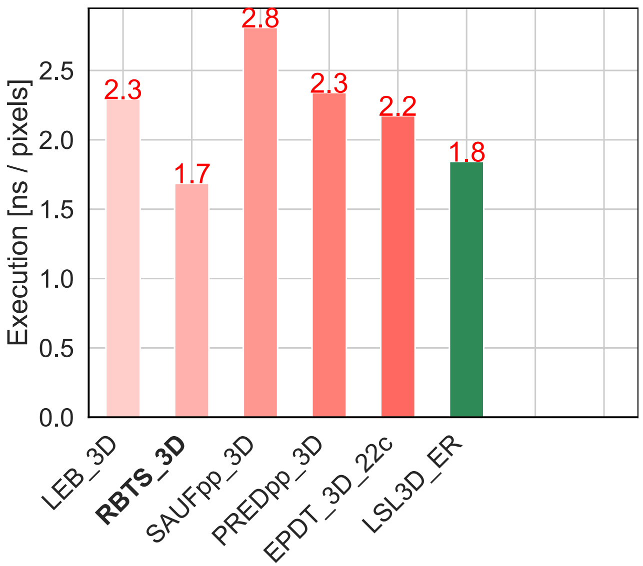


OASIS
(complex)

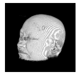


Medical datasets: LSL3D

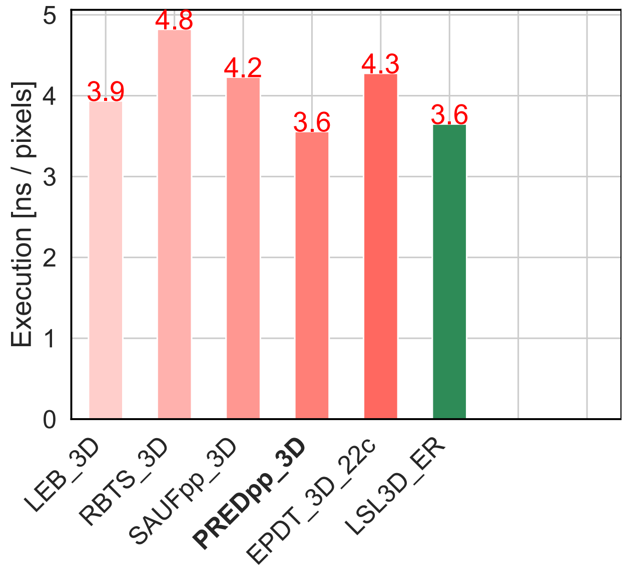
Mitochondria
(simple)



slower by $\times 0.9$ than *RBTS 3D*



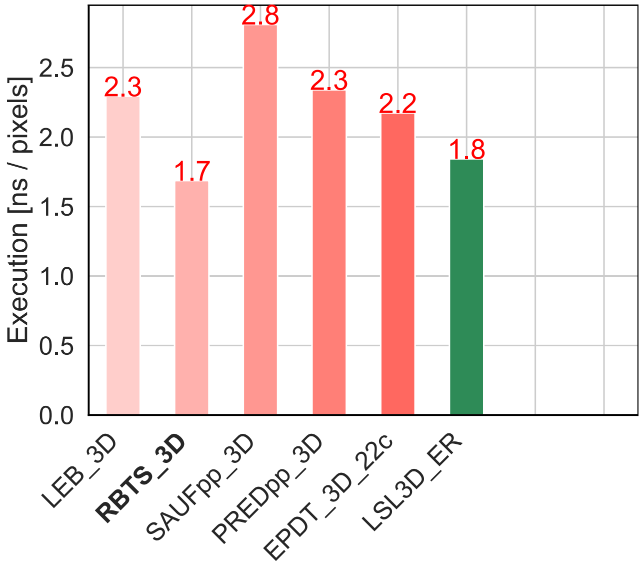
OASIS
(complex)

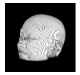


roughly as fast as *PRED++ 3D*

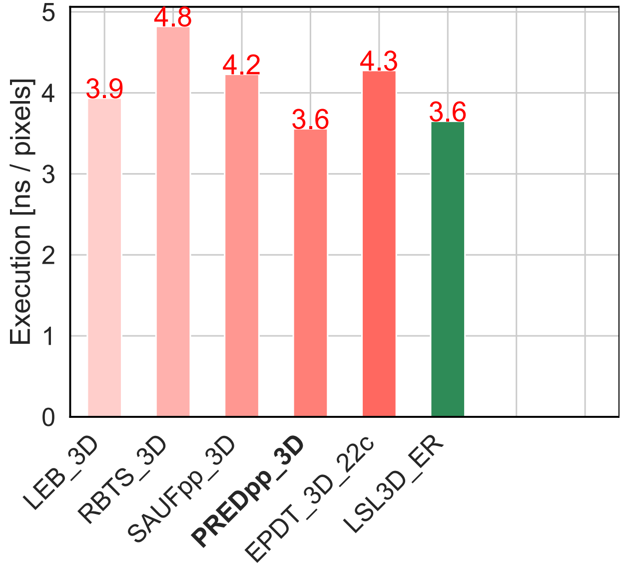
Medical datasets: LSL3D

Mitochondria
(simple)





OASIS
(complex)

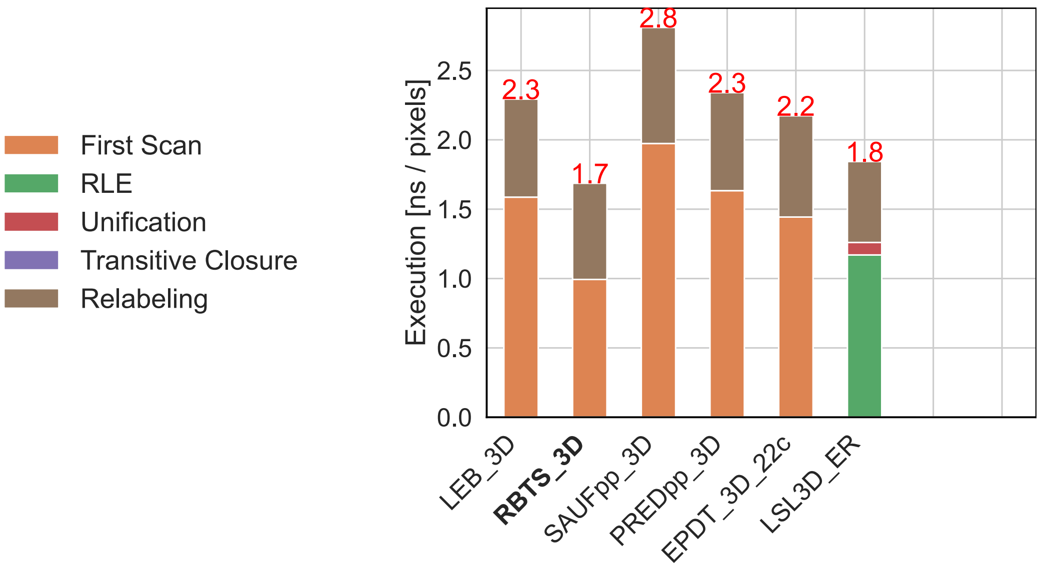


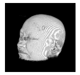
slower by $\times$0.9 than *RBTS 3D*
faster by $\times$ 1.3 than *PRED++*

roughly as fast as *PRED++ 3D*
faster by 1.3 than *RBTS*

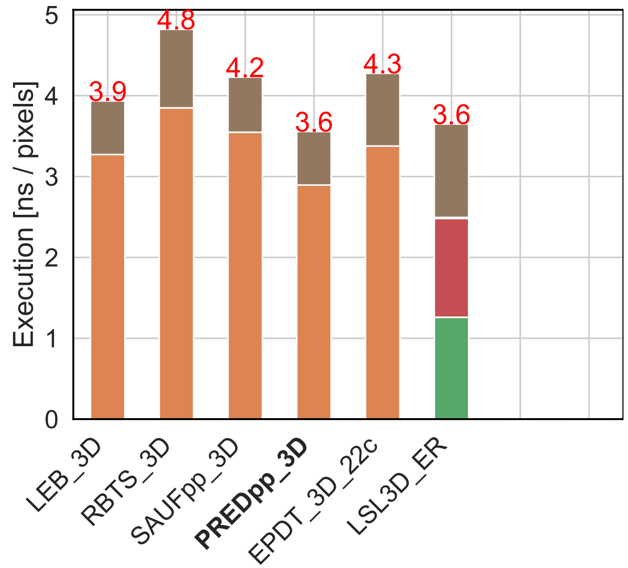
Medical datasets: LSL3D (steps)

Mitochondria
(simple)





OASIS
(complex)



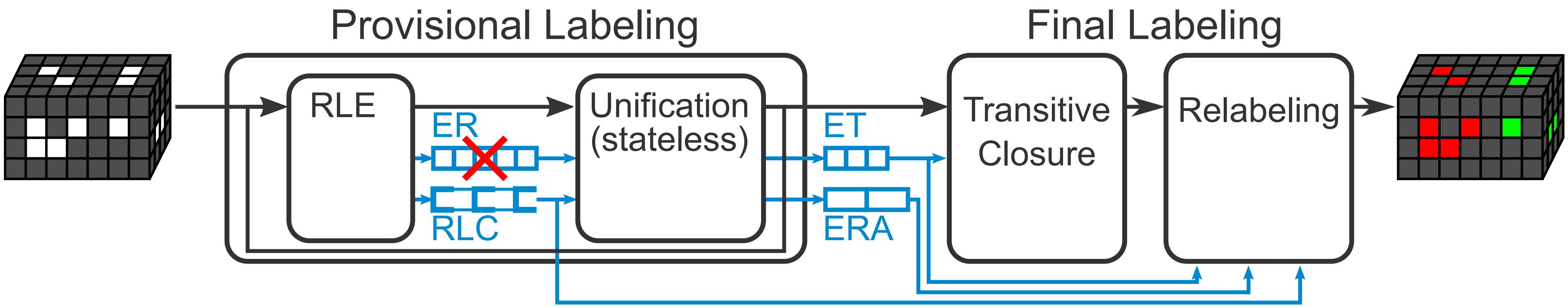
slower by $\times 0.9$ than *RBTS 3D*
faster by $\times$ 1.3 than *PRED++*

roughly as fast as *PRED++ 3D*
faster by 1.3 than *RBTS*

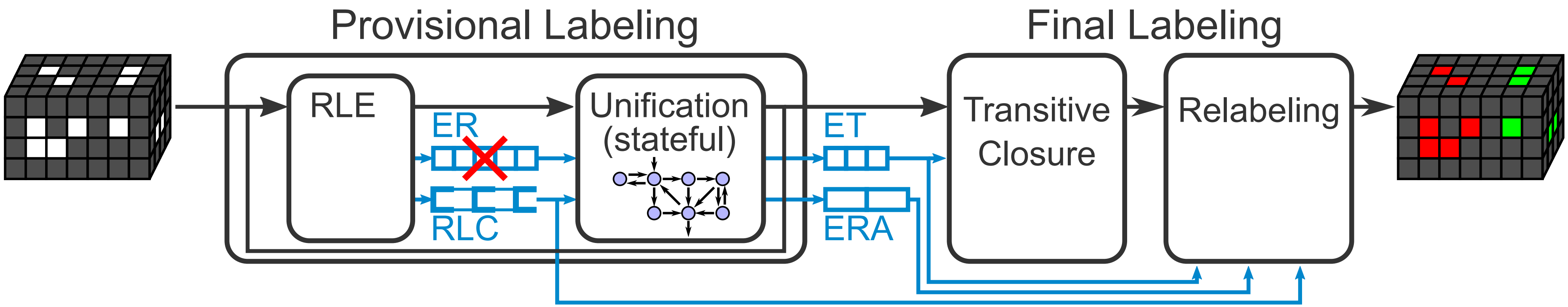
$\Rightarrow$ Performant & Regular

$\Rightarrow$ RLE & Unification expensive

Step 2: Finite-State Machine-based unification

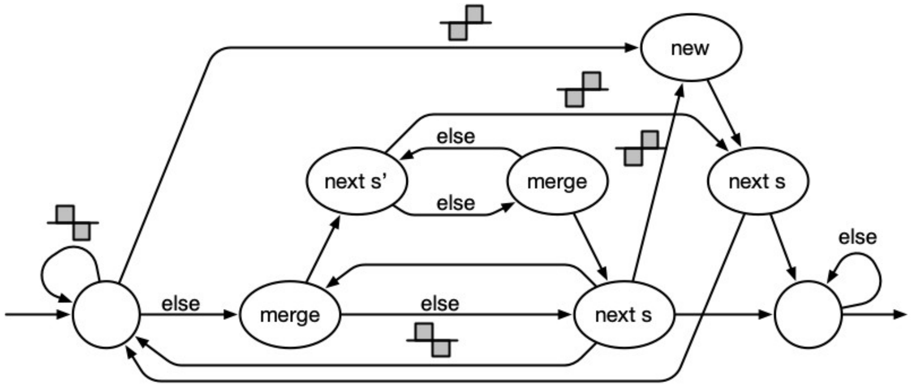


Step 2: Finite-State Machine-based unification



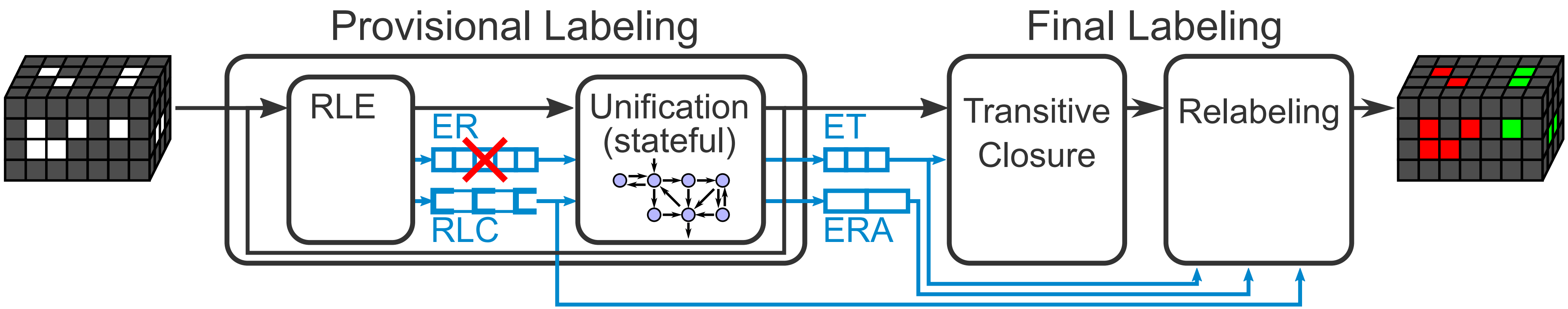
Overlap detection without ER:
⇒ Finite-State Machine (FSM)

transition = segment configuration
merging lines ⇒ iteration



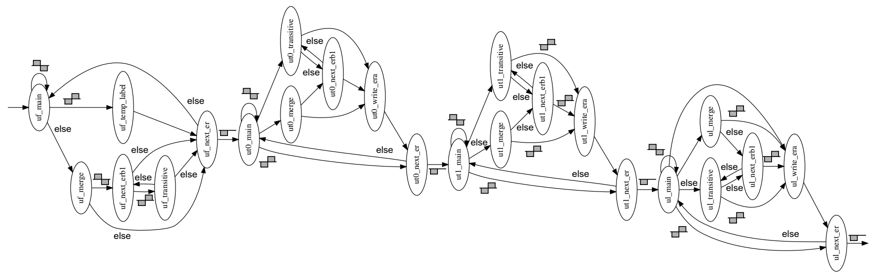
2D FSM (8 states, 14 transitions)

Step 2: Finite-State Machine-based unification



Overlap detection without ER:
⇒ Finite-State Machine (FSM)

transition = segment configuration
merging lines ⇒ iteration

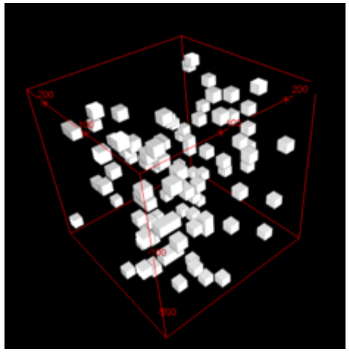


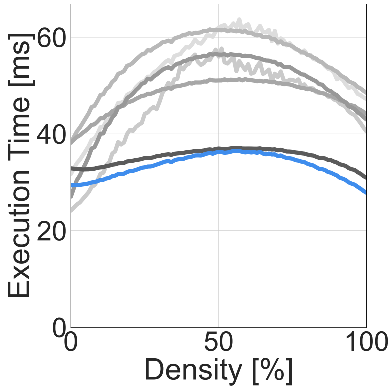
3D FSM (27 states, 55 transitions)

Random datasets: LSL+FSM

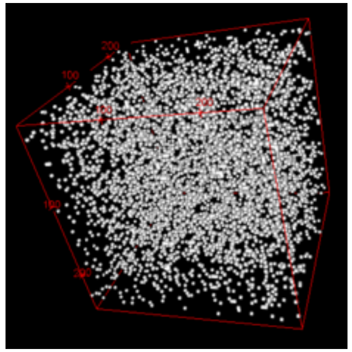
Benchmark: *YACCLAB*
Hardware: Xeon Gold 6126

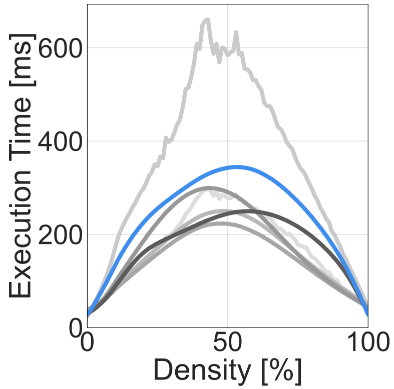
$g = 16$
(simple)





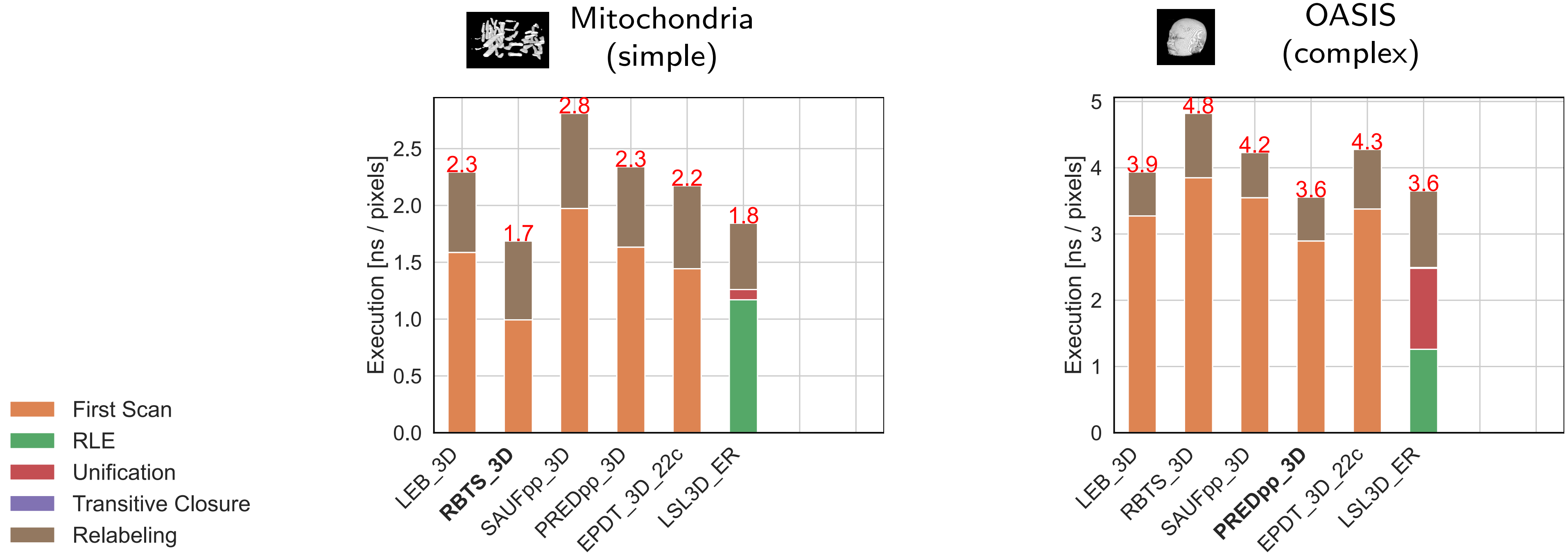
$g = 1$
(complex)



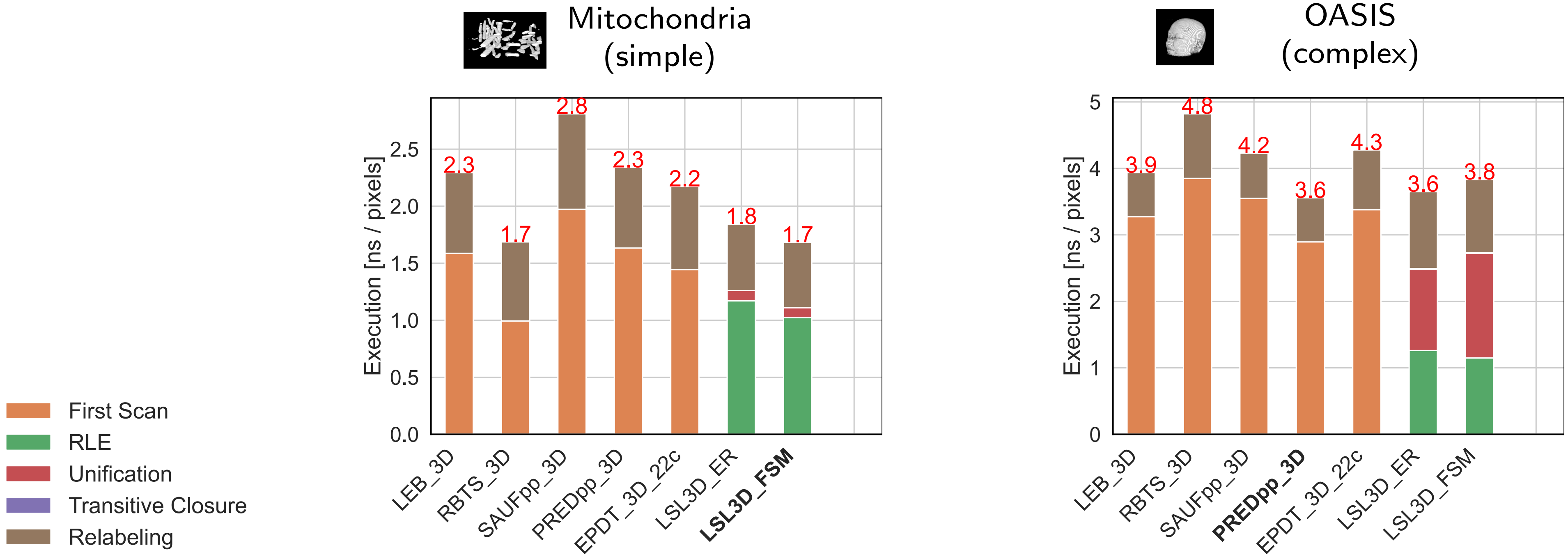


- LEB_3D
- RBTS_3D
- EPDT_3D_19c
- EPDT_3D_22c
- LSL_ER
- LSL_FSM

Medical datasets: LSL+FSM

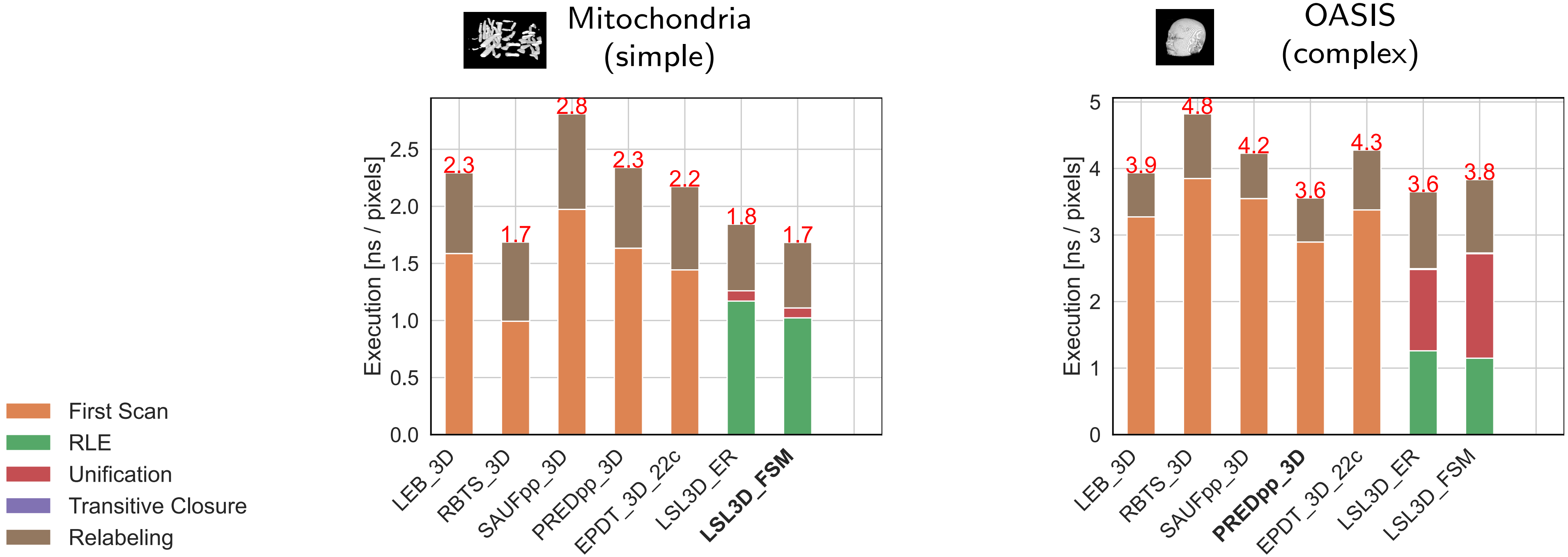


Medical datasets: LSL+FSM



LSL_FSM faster than LSL_ER by $\times$ 1.1   LSL_FSM slower than LSL_ER by $\times$ 0.95

Medical datasets: LSL+FSM



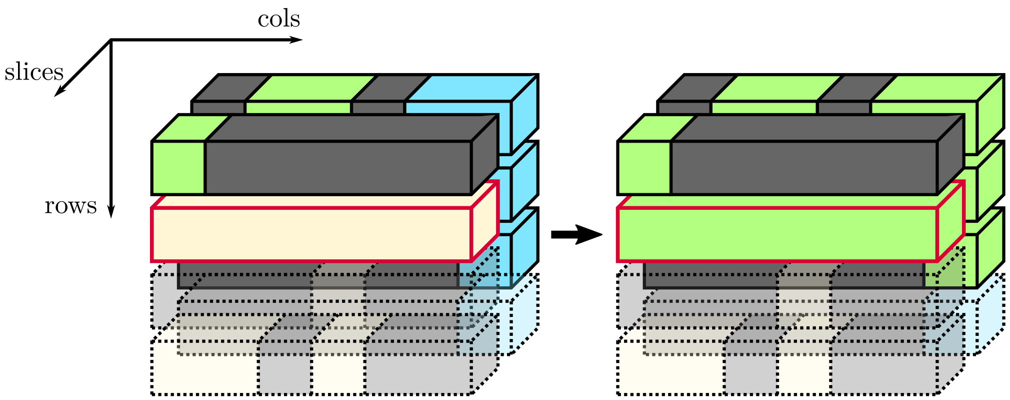
`LSL_FSM` faster than `LSL_ER` by $\times$ 1.1 `LSL_FSM` slower than `LSL_ER` by $\times$ 0.95

FSM issues $\Rightarrow$ FSM is large (27 states, 55 transitions)

$\Rightarrow$ decreased branch predictor accuracy, especially on complex images

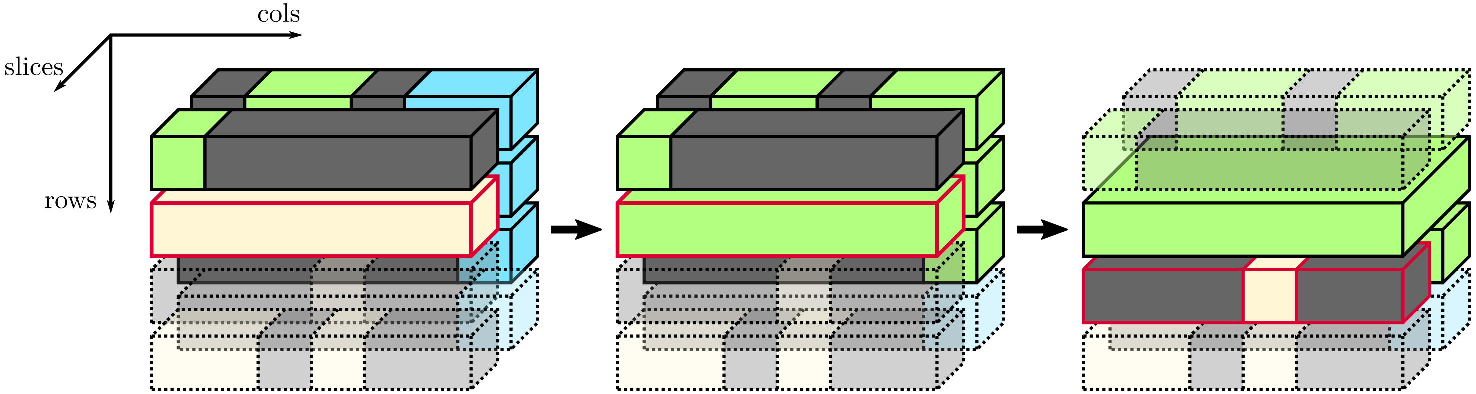
Step 3: Double-Line mechanism

Unification: 3 lines re-processed during next iteration



Step 3: Double-Line mechanism

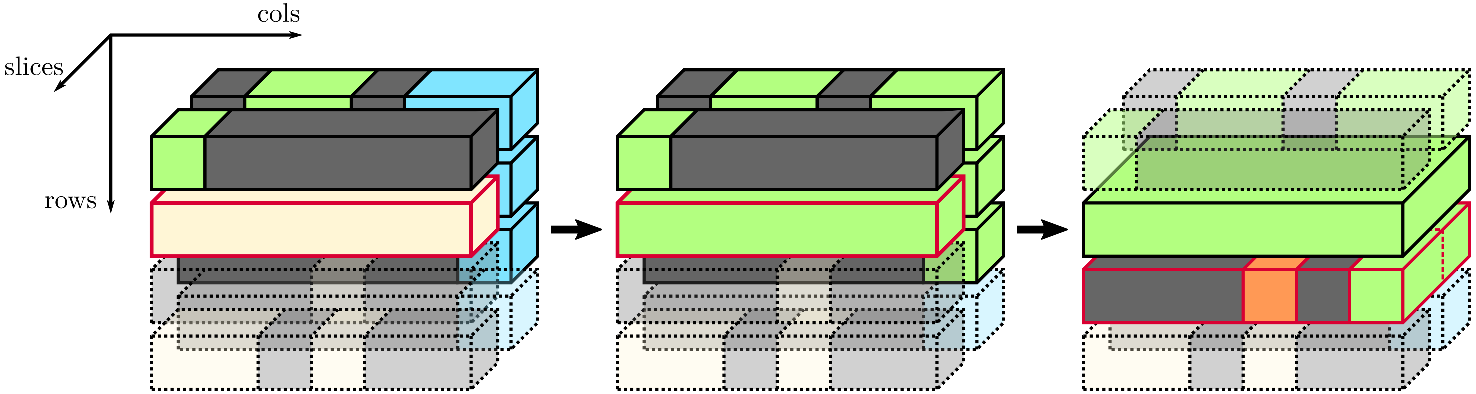
Unification: 3 lines re-processed during next iteration
Idea: Computational re-use by caching partial results (*double-line*):



Step 3: Double-Line mechanism

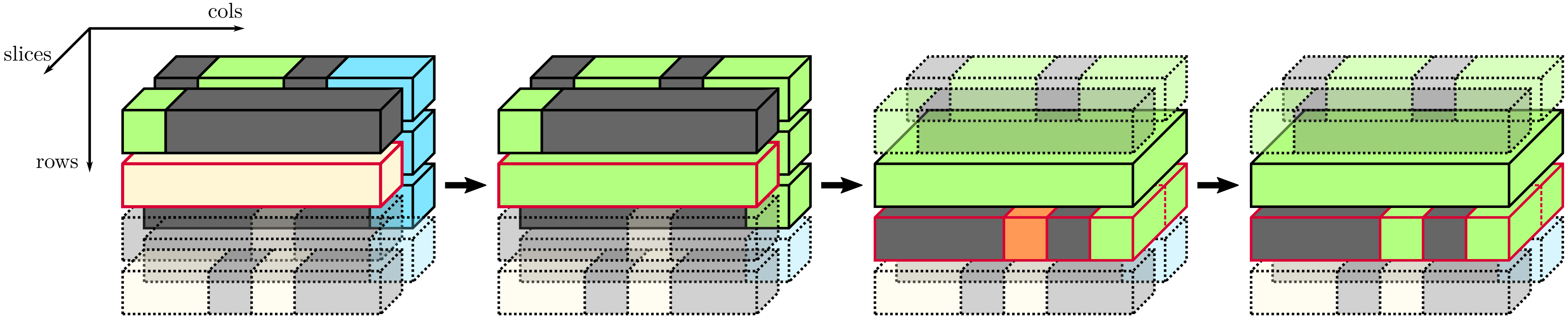
Unification: 3 lines re-processed during next iteration

Idea: Computational re-use by caching partial results (*double-line*):



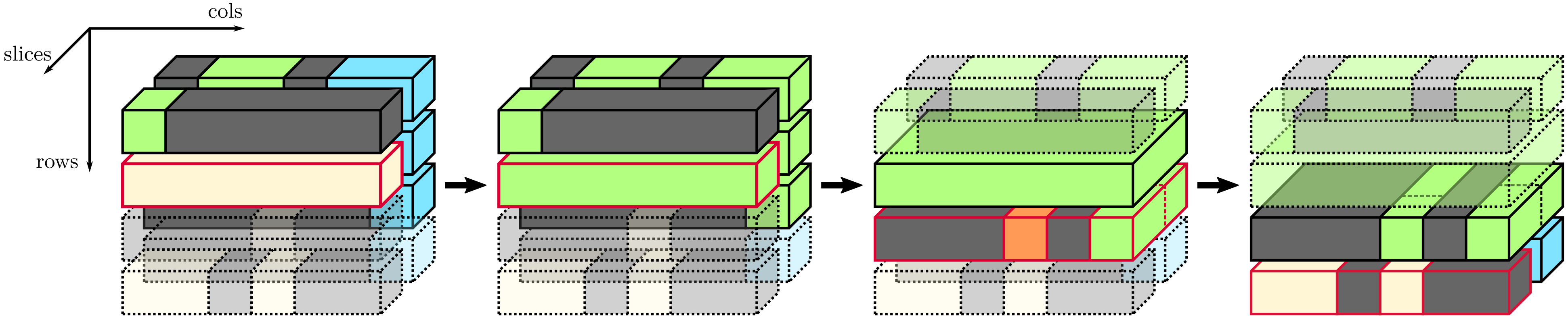
Step 3: Double-Line mechanism

Unification: 3 lines re-processed during next iteration
Idea: Computational re-use by caching partial results (*double-line*):



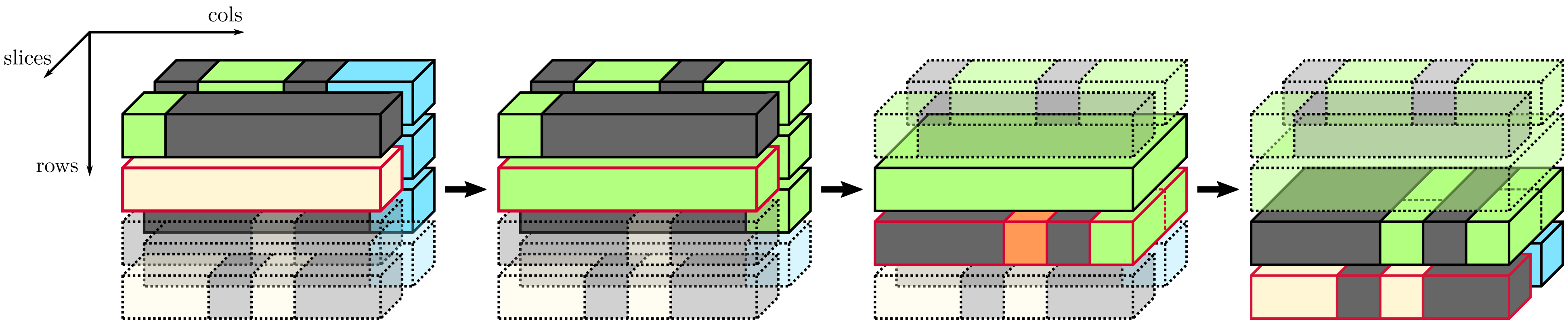
Step 3: Double-Line mechanism

Unification: 3 lines re-processed during next iteration
Idea: Computational re-use by caching partial results (*double-line*):



Step 3: Double-Line mechanism

Unification: 3 lines re-processed during next iteration
Idea: Computational re-use by caching partial results (*double-line*):



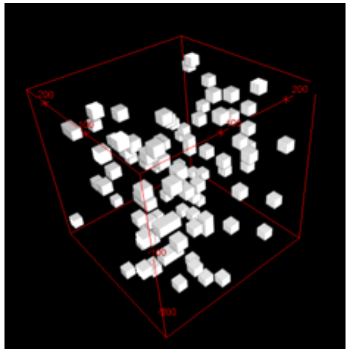
⇒ fewer operations
⇒ simpler FSM (9 states, 18 transitions)

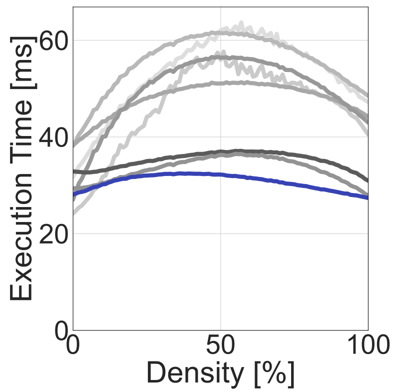
Random datasets: LSL+FSM+DOUBLE

Benchmark: *YACCLAB*
Hardware: Xeon Gold 6126

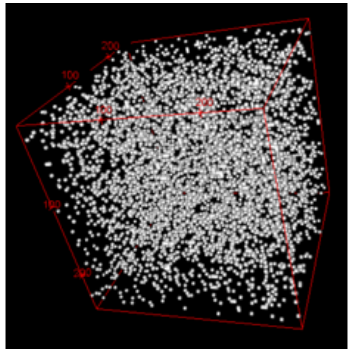
- LEB_3D
- RBTS_3D
- EPDT_3D_19c
- EPDT_3D_22c
- LSL_ER
- LSL_FSM
- LSL_FSM_DOUBLE

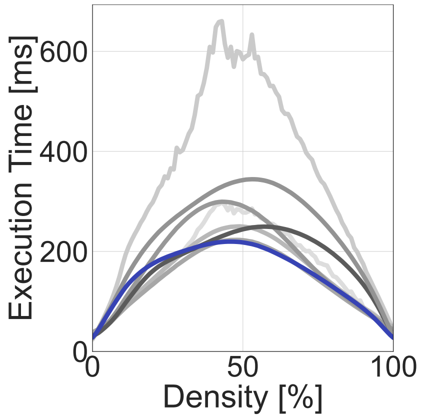
g = 16
(simple)





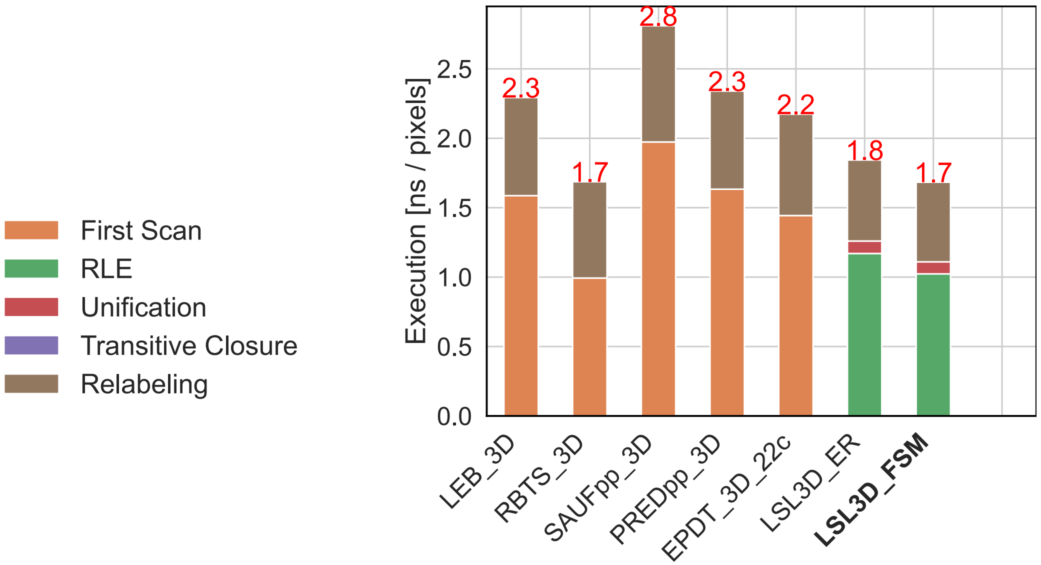
g = 1
(complex)

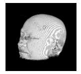




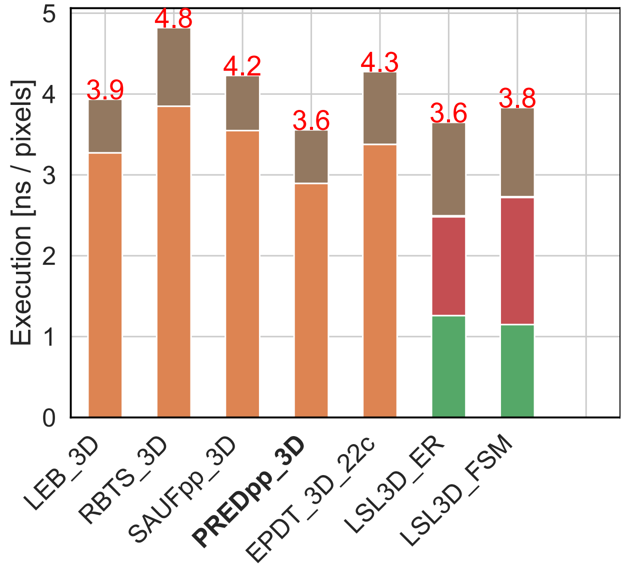
Medical images: LSL+FSM

Mitochondria (simple)





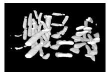
OASIS (complex)

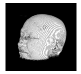


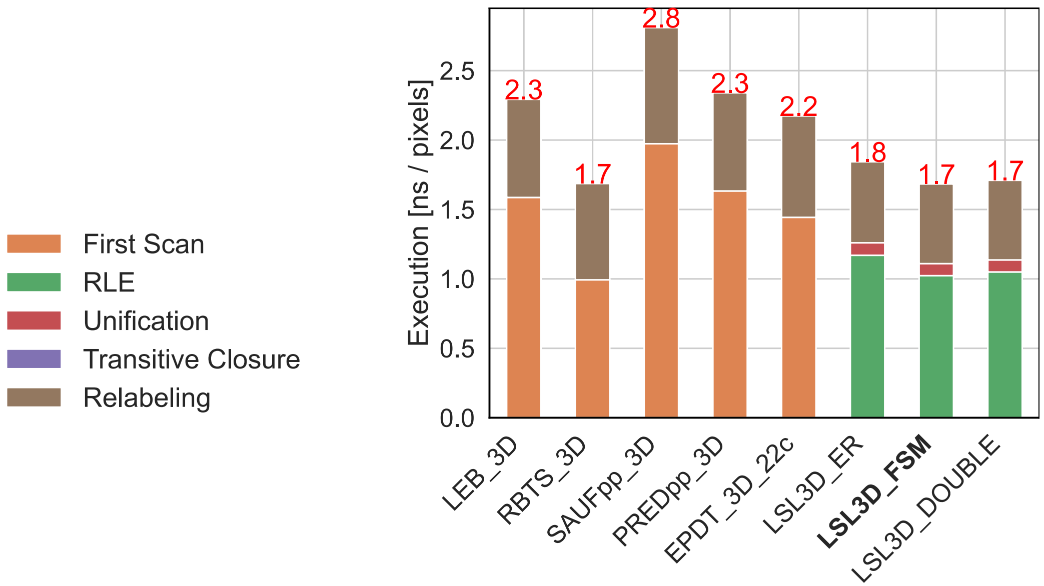
LSL_FSM faster than LSL_ER by × 1.1

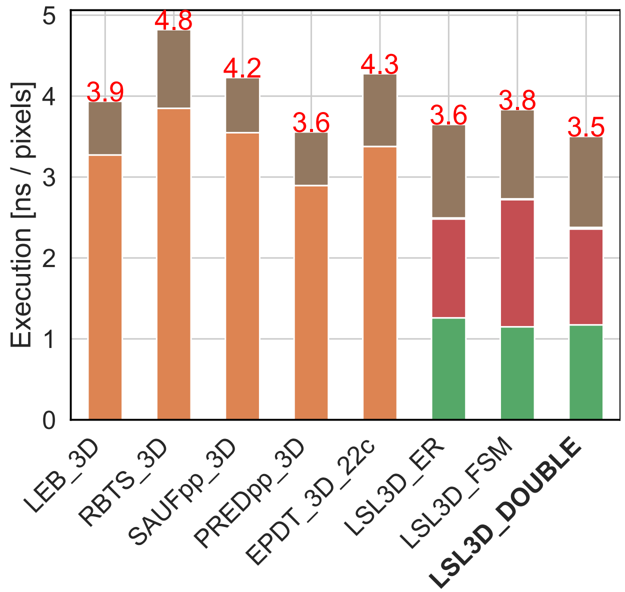
LSL_FSM slower than LSL_ER by × 0.95

Medical images: LSL+FSM+DOUBLE


Mitochondria
(simple)


OASIS
(complex)





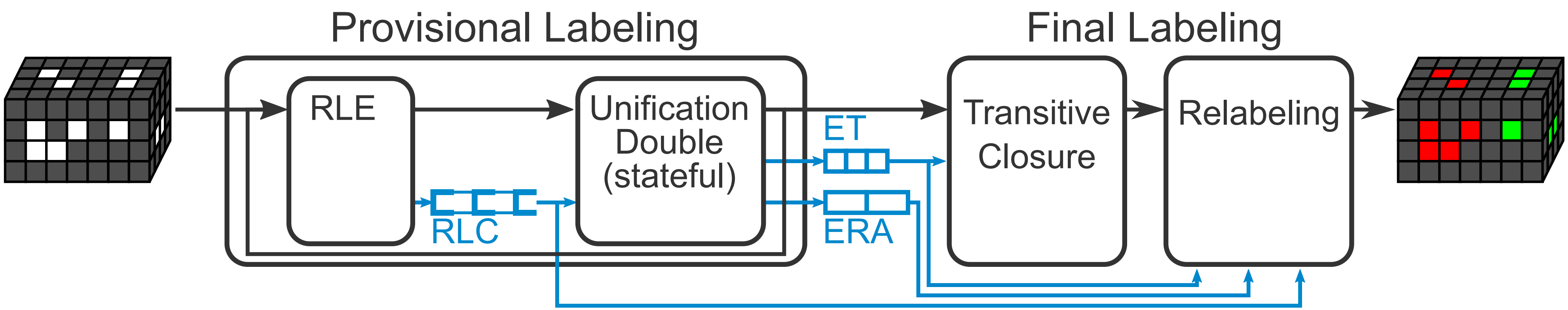
LSL_DOUBLE faster than LSL_ER by $\times$ 1.1

LSL_DOUBLE as fast as LSL_ER

$\Rightarrow$ Faster & Regular

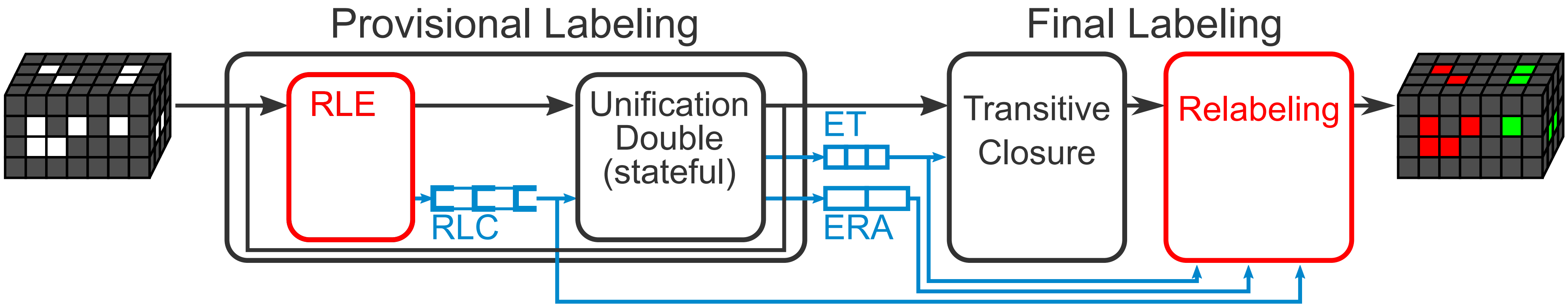
LSL3D with Hardware acceleration

Performance evaluation: RLE, Unification, Transitive Closure, Relabeling



LSL3D with Hardware acceleration

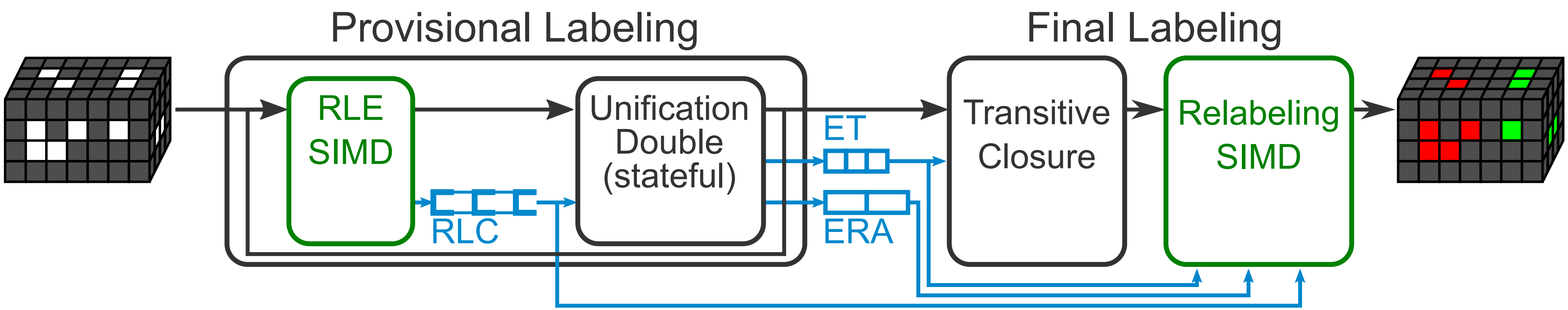
Performance evaluation: RLE, Unification, Transitive Closure, Relabeling
⇒ 70-90% of execution time



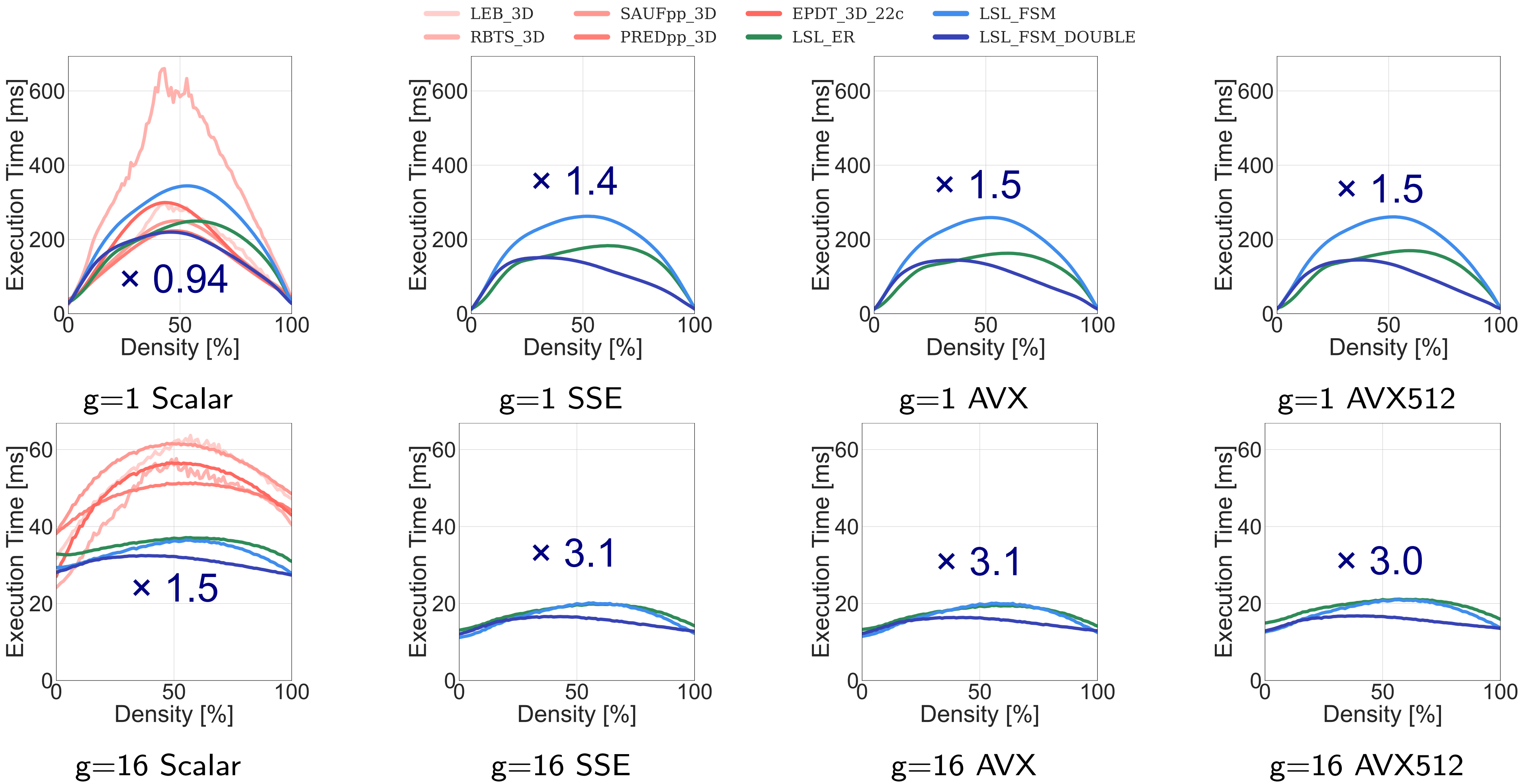
LSL3D with Hardware acceleration

Performance evaluation: RLE, Unification, Transitive Closure, Relabeling
⇒ 70-90% of execution time

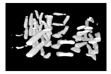
Fortunately: RLE and Relabeling benefit from instruction level parallelism [14]
Single Instruction Multiple Data (SIMD): *SSE4*, *AVX2* and *AVX512*



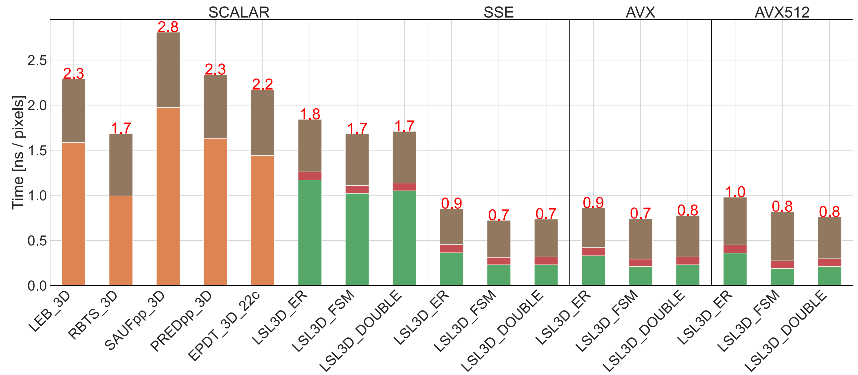
Random datasets: LSL3D with SIMD



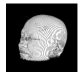
Medical datasets: LSL3D with SIMD



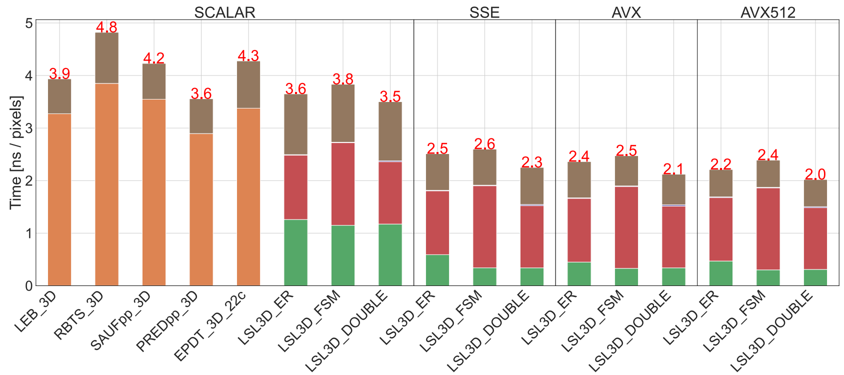
Mitochondria (simple)



- First Scan
- RLE
- Unification
- Transitive Closure
- Relabeling



OASIS (complex)



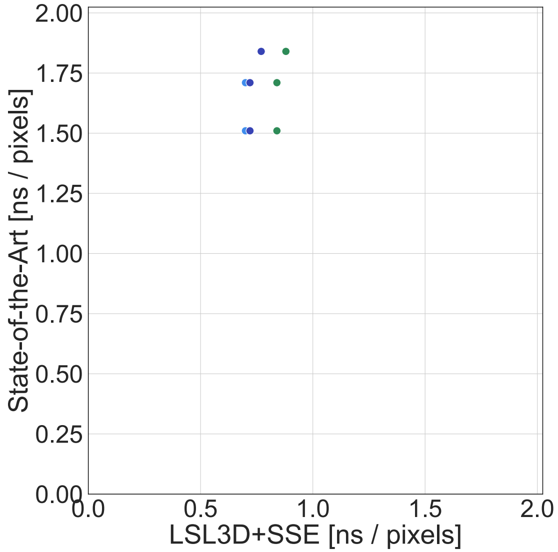
SSE ⇒ faster than State-of-the-Art

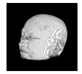
AVX & AVX512: no significant speed-up compared to SSE

Medical datasets: Individual images

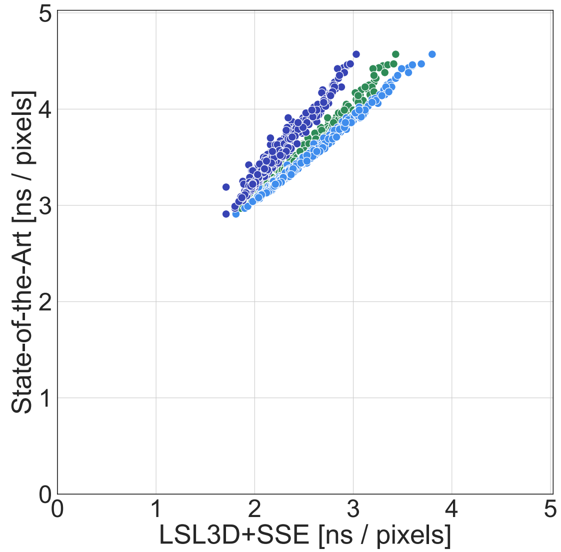
LSL3D vs best results of State-of-the-Art algorithms (1 point = 1 image)

Mitochondria
(simple)





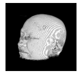
OASIS
(complex)



Medical datasets: Individual images

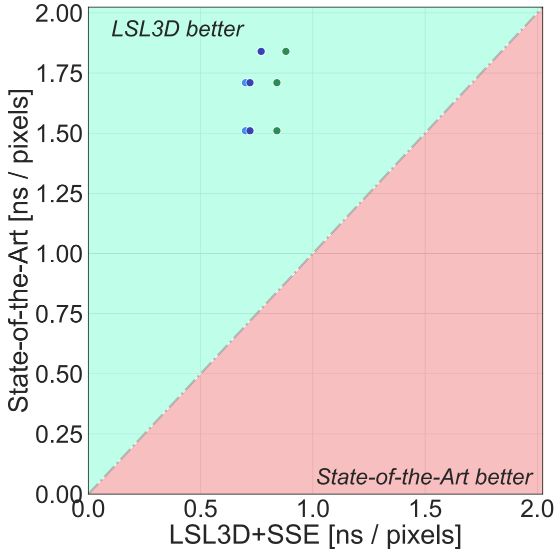
LSL3D vs best results of State-of-the-Art algorithms (1 point = 1 image)

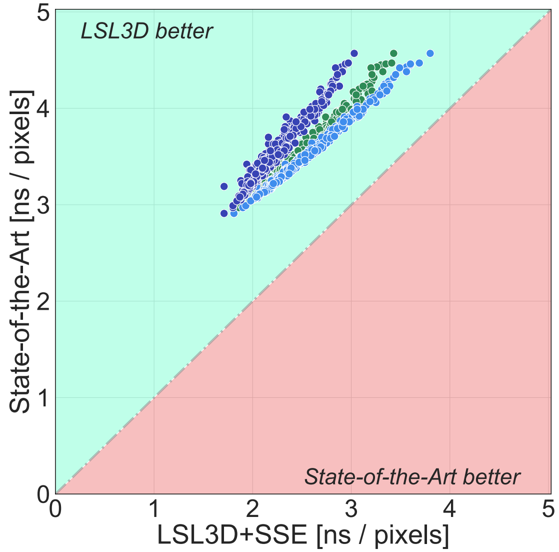
Mitochondria (simple)



OASIS (complex)

- LSL3D_ER
- LSL3D_FSM
- LSL3D_DOUBLE



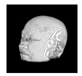


⇒ always faster than best State-of-the-Art

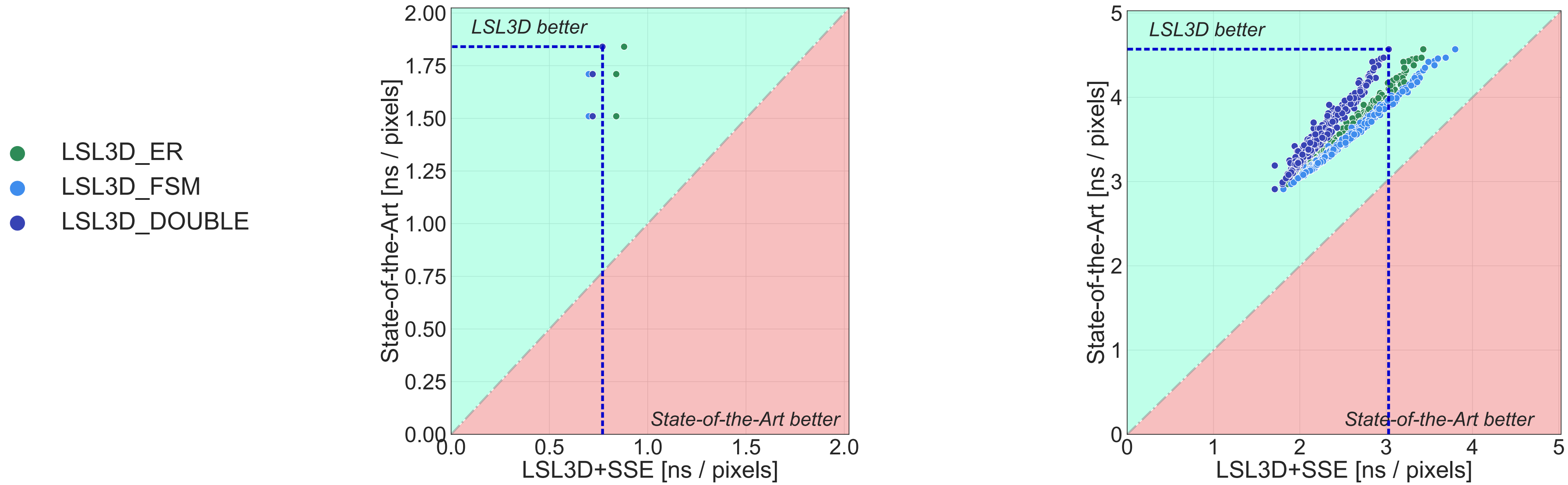
Medical datasets: Individual images

LSL3D vs best results of State-of-the-Art algorithms (1 point = 1 image)

Mitochondria
(simple)



OASIS
(complex)



⇒ always faster than best State-of-the-Art

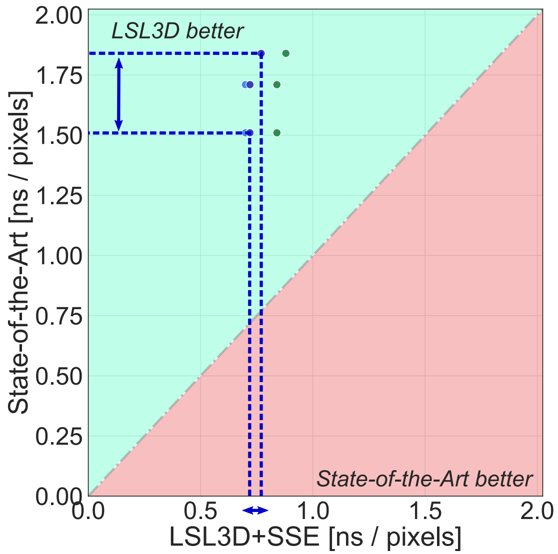
⇒ at least ×1.5 faster than best State-of-the-Art on worst-cases

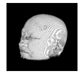
Medical datasets: Individual images

LSL3D vs best results of State-of-the-Art algorithms (1 point = 1 image)

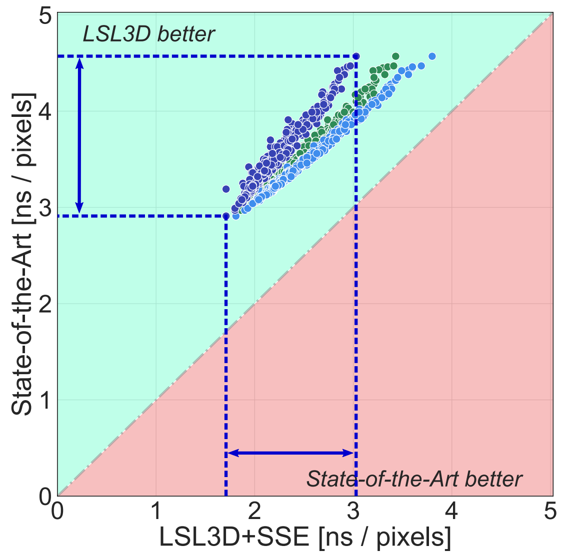
- LSL3D_ER
- LSL3D_FSM
- LSL3D_DOUBLE

Mitochondria (simple)





OASIS (complex)



⇒ always faster than best State-of-the-Art

⇒ at least ×1.5 faster than best State-of-the-Art on worst-cases

⇒ less sensitive to image variations

Conclusion

We propose a new *CCL* algorithm for 3D images that is based upon

1. a segment-based approach
2. an optimized FSM for merging segments with cache re-use mechanism (*double-line*)
3. an efficient SIMD implementation

Goals accomplished ⇒ faster than State-of-the-Art (or equivalent)
⇒ lower sensivity to image characteristics

Future work: parallelization on multi-core CPU and GPU

References I

A. Rosenfeld and J. L. Pfaltz, “Sequential Operations in Digital Picture Processing,” *Journal of the ACM*, vol. 13, pp. 471–494, Oct. 1966.

K. Wu, E. Otoo, and K. Suzuki, “Optimizing two-pass connected-component labeling algorithms,” *Pattern Analysis and Applications*, vol. 12, pp. 117–135, June 2009.

L. He, Y. Chao, and K. Suzuki, “A Linear-Time Two-Scan Labeling Algorithm,” in *2007 IEEE International Conference on Image Processing*, (San Antonio, TX, USA), pp. V – 241–V – 244, IEEE, 2007.

C. Grana, L. Baraldi, and F. Bolelli, “Optimized Connected Components Labeling with Pixel Prediction,” in *Advanced Concepts for Intelligent Vision Systems* (J. Blanc-Talon, C. Distante, W. Philips, D. Popescu, and P. Scheunders, eds.), vol. 10016, pp. 431–440, Cham: Springer International Publishing, 2016.

References II

F. Bolelli, S. Allegretti, and C. Grana, "One DAG to rule them all," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Jan. 2021.

Lifeng He, Yuyan Chao, and K. Suzuki, "Two Efficient Label-Equivalence-Based Connected-Component Labeling Algorithms for 3-D Binary Images," *IEEE Transactions on Image Processing*, vol. 20, pp. 2122–2134, Aug. 2011.

C. Grana, D. Borghesani, and R. Cucchiara, "Optimized Block-Based Connected Components Labeling With Decision Trees," *Transactions on Image Processing*, vol. 19, pp. 1596–1609, June 2010.

F. Bolelli, S. Allegretti, L. Baraldi, and C. Grana, "Spaghetti Labeling: Directed Acyclic Graphs for Block-Based Connected Components Labeling," *IEEE Transactions on Image Processing*, vol. 29, pp. 1999–2012, 2020.

References III

- M. Sochting, S. Allegretti, F. Bolelli, and C. Grana, “A Heuristic-Based Decision Tree for Connected Components Labeling of 3D Volumes,” in *International Conference on Pattern Recognition*, p. 8, 2021.
- Lifeng He, Yuyan Chao, and K. Suzuki, “A Run-Based Two-Scan Labeling Algorithm,” *IEEE Transactions on Image Processing*, vol. 17, pp. 749–756, May 2008.
- L. Lacassagne and A. B. Zavidovique, “Light speed labeling for RISC architectures,” in *IEEE International Conference on Image Analysis and Processing (ICIP)*, 2009.
- L. Lacassagne and B. Zavidovique, “Light speed labeling: Efficient connected component labeling on RISC architectures,” *Journal of Real-Time Image Processing*, vol. 6, pp. 117–135, June 2011.
- C. Grana, “Yacclab `https://github.com/prittt/YACCLAB`,” 2016.

References IV

F. Lemaitre, A. Hennequin, and L. Lacassagne, "How to speed Connected Component Labeling up with SIMD RLE algorithms," in *Proceedings of the 2020 Sixth Workshop on Programming Models for SIMD/Vector Processing*, (San Diego CA USA), pp. 1–8, ACM, Feb. 2020.