LSL3D: a run-based Connected Component Labeling algorithm for 3D volumes

Nathan Maurice, Florian Lemaitre, Julien Sopena, Lionel Lacassagne

LSL3D: a run-based Connected Component Labeling algorithm for 3D volumes

Nathan Maurice, Florian Lemaitre, Julien Sopena, and Lionel Lacassagne

LIP6, Sorbonne University, CNRS

Abstract. Connect Component Labeling (*CCL*) has been a fundamental operation in Computer Vision for decades. Most of the literature deals with 2D algorithms for applications like video surveillance or autonomous driving. Nonetheless, the need for 3D algorithms is rising, notably for medical imaging.
While 2D *CCL* algorithms already generate large amounts of memory accesses and comparisons, 3D ones are even worse. This is the *curse of dimensionality*. Designing an efficient algorithm should address this problem. This paper introduces a segment-based algorithm for 3D labeling that uses a new strategy to accelerate label equivalence processing to mitigate the impact of higher dimensions. We claim that this new algorithm outperforms State-of-the-Art algorithms by a factor from $\times 1.5$ up to $\times 3.1$ for usual medical datasets and random images.

1 Introduction

Connected Component Labeling (CCL) has been a fundamental algorithm in Computer Vision for decades [37] [40][15]. It consists of finding connected components (sets of adjacent pixels) in a binary image and assigning them a unique identifier referred to as the *label*.

CCL is used in a wide array of applications, such as autonomous driving [41][11], video surveillance [21][38], medical applications [10][35][1][30][22] and other fields like [34] where a real-time implementation matters.

This article introduces a new 3D labeling algorithm named *LSL3D* and our contributions are twofold: 1) a new Finite State Machine (FSM) to efficiently process segments using Run-Length Encoding (RLE) and 2) a cache mechanism to re-use partial results and reduce computational complexity.

We claim that our new segment-based algorithm is $1.8\times$ to $2.3\times$ faster than State-of-The-Art algorithms for existing medical datasets. Moreover, for random 3D images, which are more stressing at low granularities, we claim that LSL3D is $1.5\times$ to $3.1\times$ faster.

The article is written as follows: Section 2 gives a background on *CCL* and details our benchmark protocol. Section 3 reviews existing literature. Then, Section 4 introduces three strategies for label equivalence management that

dataset	# subsets	size	density	granularity	# runs	# CCs
mitochondria	3	1024×768×165	5.9	26.4	197,878	40
OASIS	373	256×256×128	19.8	4.2	236,718	3,200
random	16	256×256× 256	0 - 100	1 -16		

Table 1: Average characteristics of 3D datasets.

attenuate the *curse of dimensionality*. Finally, Section 5 studies the impact of SIMD on our new *CCL* algorithms.

2 Classical approaches to connected components labeling and their evaluation

Fundamentally, *CCL* algorithms establish equivalences between foreground pixels if they are connected.

Only a single solution for the labeling of a given image exists. Qualitative compromises are therefore impossible, and research on *CCL* algorithms has been focused on the reduction of their execution time.

In this section, we will first detail the basis of modern algorithms, and the evaluation protocols: metrics, datasets and benchmark platform.

2.1 Main principles of modern *CCL* algorithms

Modern algorithms are all derived from historical ones like those from Rosenfeld [37] or Haralick and Shapiro [15]. They are composed 3 steps:

1. a *provisional labeling*, that assigns a temporary label to each pixel and builds label equivalences,
2. label *equivalence solving*, that computes the *Transitive Closure* (TC) of the graph associated to the label equivalence table,
3. *final labeling*, to replace temporary labels with final labels (usually the smallest of each component).

Modern algorithms implement some algorithmic optimizations to accelerate these three steps. Since the bottleneck of these algorithms is usually their *control-flow* rather than memory accesses or calculations, datasets have a major impact on their performance.

2.2 Benchmarking procedure and datasets

To evaluate the algorithms' performance, two medical datasets have been used for the benchmarks: *mitochondria* [31] which includes 3 subsets and *OASIS* [32] which includes 373 subsets. Images from *mitochondria* contain large blobs and several small CCs. On the other hand, images from *OASIS* contain a hollow volume with complex shapes. This translates into a high and low granularity [8]

for *mitochondria* and *OASIS*, respectively (Table 1).

On top of these medical datasets, we use random images, which have been generated using MT19937 [33] for reproducible results. The random images have a density $d \in [0\%; 100\%]$ and a granularity $g \in [1; 16]$ where the granularity describes the detail level of an image (size of individual blocks during generation). The algorithms have been evaluated on an Intel Xeon Gold 6126 using the *YACCLAB* [7] framework.

3 State-of-the-Art of 3D algorithms

Literature on *CCL* algorithms is extensive and has been centered on 2D images. *CCL* on CPUs has been heavily studied and optimized [14][17][6][26]. On GPUs, after an early era of iterative algorithms [43][3][20], a new generation introduced by Komura [23] are now direct; a new way to manage equivalences and reduce memory accesses was introduced by Playne [36] and has become the basis of the fastest *CCL* algorithms [19] [2].

CCL algorithms can be classified according to their neighborhood mask and how they process data: they can be pixel-based, block-based or segment-based.

3.1 Pixel-based algorithms

The extension of the *Rosenfeld* 2D algorithm to 3D is straightforward: the 9 adjacent pixels from the previous slice are added to the mask, for a total of 13 pixels.

The mask-based approach was improved by Wu [42] (*SAUF*). Wu realized that a decision could be taken without accessing all 4 pixels in the neighborhood for 2D images. A decision tree was proposed to access as few neighbors as possible. *SAUF* was later ported to 3D by Bolelli as *SAUF 3D* [4]. The decision tree was further optimized by He *et al.* with the *Label Equivalency Based* (*LEB*) algorithm for 2D [16] and 3D. [29] images.

In [18], He noticed that the value of the previous pixel could simplify the following decision with fewer comparisons and introduced a graph of decision trees. This method was generalized by Grana with the *PRED* algorithm [12], which was later extended to 3D volumes by Bolelli with *PRED 3D*. [4]. The introduction of *Direct Rooted Acyclic Graphs* (*DRAG*) by Bolelli [5] reduced the code footprint. *DRAG* were used to revisit existing algorithms, like with *SAUF++* and *PRED++* by Bolelli [4]. The same paper extended these new *SAUF++* and *PRED++* algorithms for 3D (*SAUF++ 3D* and *PRED++ 3D*).

3.2 Block-based algorithms

Grana [13] proposed a block-based approach (foreground pixels in the same 2×2 block are necessarily in the same component). The decision tree for block-based algorithms was then improved upon by Chabardes [9] with a forest of *decision*

trees, and was later adapted to use a *DRAG* by Bolelli [5].

The block-based approach was extended to 3D volumes by Sochting [39] with the *Entropy Partition Decision Tree*. *EPDT* algorithms handle pixels by blocks of either $2 \times 1 \times 1$ (*EDPT_ 19c* and *EDPT_ 22c*) or 2×2 (*EDPT_ 26c*).

3.3 Segment-based algorithms

While block-based approaches have been shown to be efficient, pixels can also be regrouped as segments. For a given line, it iterates over each column and aggregates consecutive pixels into segments using a *Run Length Encoding* (interval encoding indices). Then, it checks for segment adjacency (overlaps between current segment and segment of the previous line) and performs unions when needed.

A *Run Based Two Scans* strategy (*RBTS*) for 2D images was used by He [28] and later extended to 3D volumes [29]. The segment-based approach has also been proposed by Lacassagne with the *Light Speed Labeling* (*LSL*) [24][25] for 2D labeling. *LSL* also uses a RLE but adds a *line-relative labeling* (ER tables) combined with a table of segments (RLC) to accelerate adjacency detection and equivalence building.

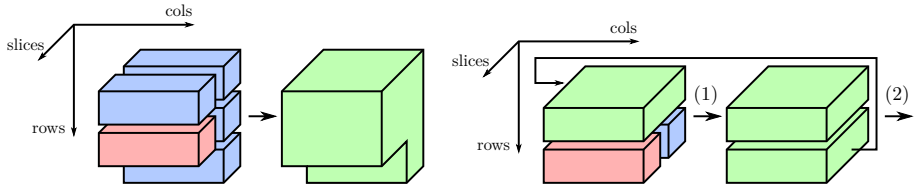
4 LSL3D & efficient unification strategies for 3D volumes

This section presents, step-by-step, the improvement and the transformation of the classical 2D LSL algorithm into an optimized 3D version. Step zero is the extension of the 2D version to 3D, keeping the line-relative labeling (version named `LSL_ER`). It can be viewed as a legacy version for comparison [25]. The first step is the replacement of the ER tables by a Finite State Machine (FSM) (`LSL_FSM`). The second improvement is a cache-reuse mechanism to perform unions/unifications with double-lines (`LSL_FSM_DOUBLE`).

Our successive *LSL* implementations have been tested according to the benchmark in Section 2.2. They are compared to 7 algorithms from the State-of-the-Art: *LEB*, *RBTS*, *PRED++ 3D*, *SAUF++ 3D*, *EDPT_ 19c*, *EDPT_ 22c* and *EDPT_ 26c*. Among the *EPDT* algorithms, we only present the best one (*EDPT_ 22c*). The results are shown on Figures 2 and 3 and will be evaluated throughout the following sections.

4.1 Extension of the segment-based unification for 3D volumes

In order to find overlapping segments without iterating several times on the current line, the first *LSL* algorithm [25] finds overlaps by accessing the ER table. In 2D, two ER tables (current and previous line) are necessary at any given time. On the other hand, due to the raster scan, two planes are required in



(a) Classical 3D Unification (b) 3 Unification with double lines

Fig. 1: Segment-based unification for 3D volumes.

3D (current and previous plane). This can degrade performance on large images: for instance, on *mitochondria*, the ER tables use a total of 3.1 MB of memory and thus do not fit within the 1.0 MB L2 cache of the Xeon.

The performance of our `LSL_ER` implementation in 3D can be seen on Figure 2. `LSL_ER` becomes faster at $g > 4$ on random images and widens the gap at higher granularities (up to $\times 1.3$ for $g = 16$).

For medical images, `LSL_ER` is overall faster than the state of the art (Figure 3): while *RBTS* is $\times 1.1$ faster on *mitochondria*, it is also slower by a factor of $\times 1.3$ on *OASIS*. Similarly, *PRED++ 3D* is as fast on *OASIS* but slower by a factor of $\times 1.3$ on *mitochondria*.

The limits of *LSL* can be explained by the duration of the RLE step, especially on large images (between 60% and 70% of the execution time on *mitochondria*). Not only does it create an array of segments (RLC table) but the initialization of the ER table is costly as it contains one element per pixel.

4.2 A finite-state machine-based unification

Overlapping segments between lines can also be found without ER using a Finite-State Machine (FSM). In the 2D unification [27], each state of the 2D FSM encodes segment configurations between the current and previous lines. Merging two lines involves iterating over both at the same time: a new label is created for each isolated segment, whereas the components of two overlapping segments are merged together.

While the FSM-based implementation of *LSL3D* is efficient on simple images, this is not the case for more complex images. Indeed, Figure 2 shows that `LSL_FSM` is $\times 1.3$ faster than ER for high-granularity images ($g = 16$), but slower by a factor of $\times 0.61$ for $g = 1$.

The execution time of `LSL_FSM` follows a similar trend on medical images, as can be seen on Figure 3: *mitochondria* the FSM-based algorithm improves the execution time compared to the ER-based unification by a factor $\times 1.1$. On the other hand, for *OASIS*, it is slower by a factor of $\times 0.95$.

The overhead of the unification phase explains the results. Despite a lack of ER tables and a faster RLE step, the complexity of the 3D FSM (27 states and 55 transitions in 3D, vs. 8 states and 14 transitions for its 2D counterpart) degrades the accuracy of the branch predictor. This is particularly penalizing on complex

images with many segments and a wider diversity of segments configurations (and thus, with more state transitions being performed).

4.3 Computational reuse of merged lines

The complexity of the FSM has led to performance limitations on complex images. To overcome these limitations, we have redesigned the FSM to store and reuse partial results. On top of factorizing calculations, this also simplifies the FSM (from 27 states and 55 transitions to just 9 states and 18 transitions).

More precisely, as shown on Figure 1, two consecutive iterations on the same slice process three lines several times: the current line (in red) and two neighboring lines (in blue) will be re-processed in the following fusion. This redundancy can be removed by caching partial results (in green) into a double-line array.

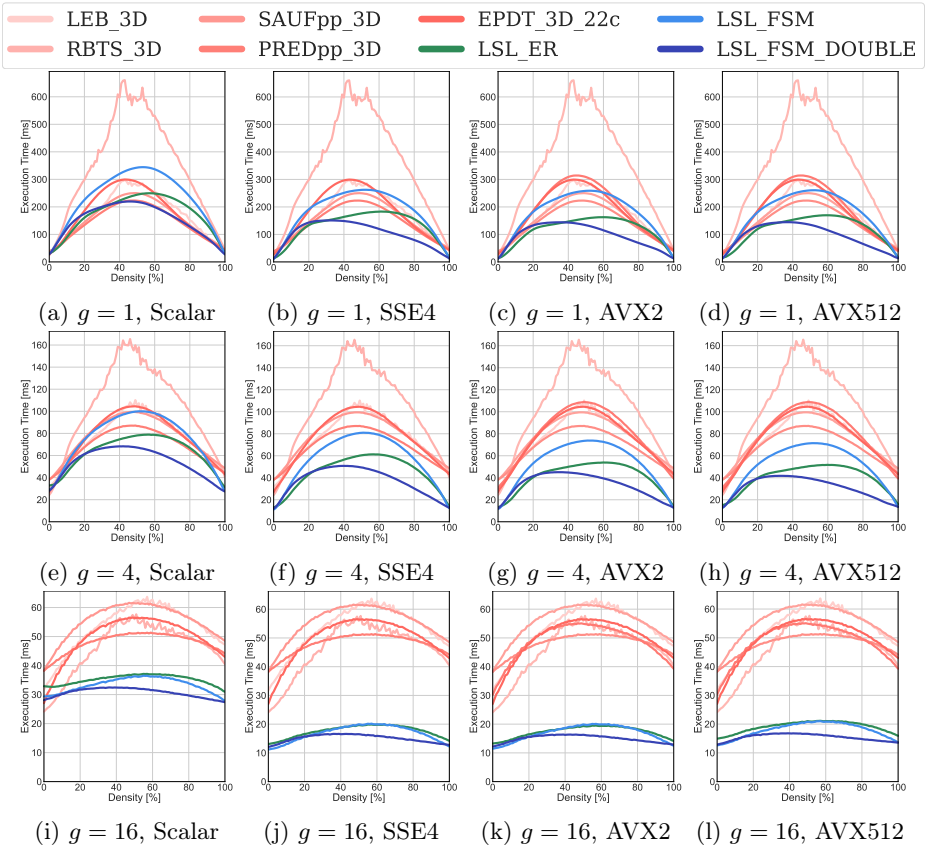
In order to simplify the computational reuse, a 2×2 mask is used for lines, as displayed on Figure 1b. Two phases are required: a first step (1) unifies the current line (red line) and its neighbor (blue line) from the previous slice. It produces a temporary line (green) that contains overlaps from both lines. Then a second step (2) unifies the double-line with the one produced in the previous step. The newly-created double-line is re-used in the next unification to avoid redundant processing. The former double-line is discarded, and its memory is recycled for the next iteration.

The performance of `LSL_DOUBLE` on random images (Figure 2) shows that `LSL_DOUBLE` is on average better than `LSL_ER` and `LSL_FSM` for all granularities: It is indeed $\times 1.3 - 1.5$ faster than the best algorithm for $g = 4$ and $g = 16$ and only $\times 0.94$ the speed of the best.

Besides these good results, Figure 2 also shows that `LSL_DOUBLE` is more resistant to increasing densities (gap between green a purple lines, beyond 25% density). Indeed, the number of segments at these densities is statistically high, which slows down segments-based algorithms. In `LSL_DOUBLE`, the phenomenon is reduced by the fusion of segments within double-lines: more segments implies more overlaps, which leads to more fusions and fewer segments in the double-line. This makes the double-line strategy particularly relevant for complex images. On *mitochondria* (Figure 3a), `LSL_DOUBLE` is as fast as `LSL_FSM`. However, unlike `LSL_FSM`, it is as fast as `LSL_ER` on *OASIS*. These results make the double-line algorithm at least as fast as the *best* algorithm on both *OASIS* and *mitochondria* (Figure 3).

5 Architecture-specific optimizations of Run-Length Encoding on 3D images

As seen in the previous section, the double-line unification reduces the execution time of *LSL3D*: the unification (which does not need extra ER tables) becomes highly optimized. The RLE and relabeling steps thus become the main bottlenecks. ($\approx 90\%$ and $\approx 70\%$ of the execution time for *mitochondria* and *OASIS*).



(a) $g = 1$, Scalar (b) $g = 1$, SSE4 (c) $g = 1$, AVX2 (d) $g = 1$, AVX512

(e) $g = 4$, Scalar (f) $g = 4$, SSE4 (g) $g = 4$, AVX2 (h) $g = 4$, AVX512

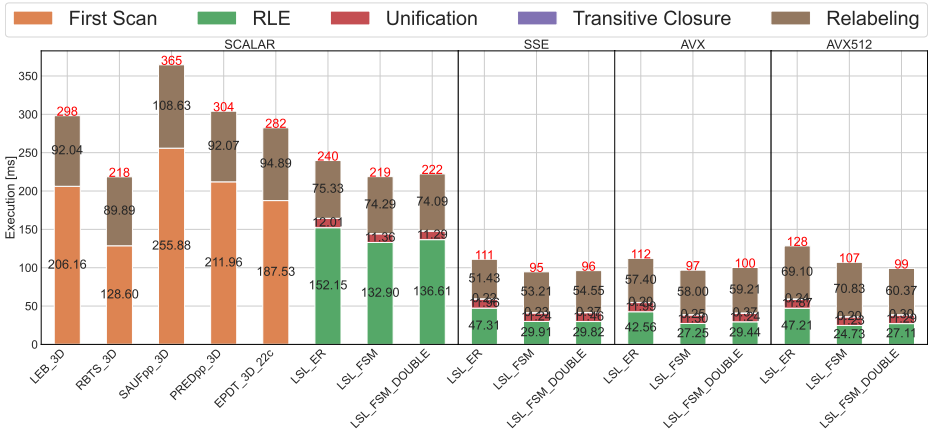
(i) $g = 16$, Scalar (j) $g = 16$, SSE4 (k) $g = 16$, AVX2 (l) $g = 16$, AVX512

Fig. 2: Execution time of algorithms on random images for different granularities and densities on Xeon

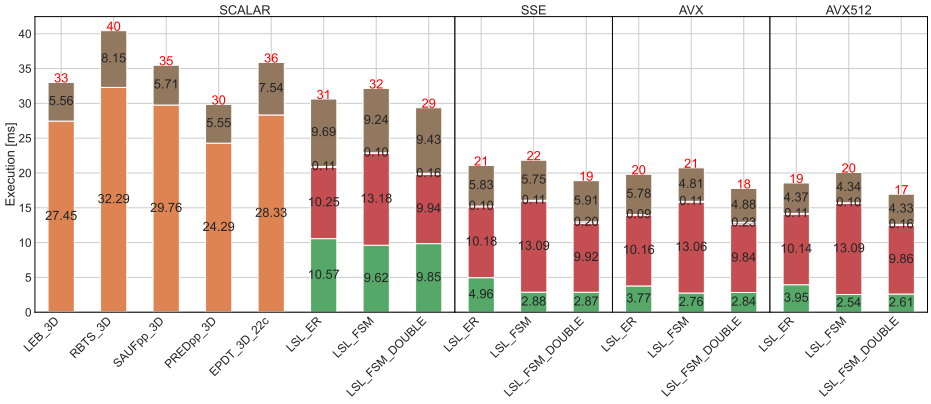
Fortunately, these steps lend themselves well to instruction level parallelism with SIMD [27]. Several SIMD implementations of the RLE and relabeling have been tested: `SSE4`, `AVX2` and `AVX512`.

As can be seen on Figure 2, the use of SIMD accelerates the execution of all *LSL* versions on random images. In fact, `SSE4` alone is enough to make `LSL_ER` faster than State-of-the-Art algorithms on random images by a factor of $\times 1.1$ to $\times 2.6$ (Figure 2). This is also true for medical images, with a speedup of $\times 1.4$ to $\times 2.0$ compared to the State-of-the-Art algorithm on *mitochondria* and *OASIS* (Figure 3).

The use of more complex instruction sets such as `AVX2` or `AVX512` do not provide additional speedups over `SSE4` on simple images such as *mitochondria* ((Figure 3a), but nonetheless improves execution times by 10% on complex images (*OASIS*). For the `AVX2` version, the lack of dedicated *compress* instructions makes the speedup constrained by look-up table accesses in the RLE step. On the



(a) *mitochondria*, Scalar



(b) *OASIS*, Scalar

Fig. 3: Execution time on *mitochondria* and *OASIS* on Intel Xeon

other hand, the `AVX512` *compress* instructions on the Xeon are only available for 32-bits elements. A conversion of elements to 16-bits for processing the segments (RLC table encodes segments using 16-bits numbers) is thus required and adds an overhead.

The combination of SIMD with the double-line mechanism gives an even greater acceleration: on random images, `LSL_DOUBLE`+`AVX512` is on average $\times 1.5$ to $\times 3.0$ faster than the best algorithm from the State-of-the-Art, whereas it is faster by a factor $\times 1.7$ and $\times 2.2$ on natural images.

The speedup of the best *LSL3D* version compared to the best State-of-the-Art can be found on Table 2.

dataset	mitochondria	oasis	random g=1	random g=4	random g=16
speedup	× 2.3	× 1.8	× 1.5	× 2.2	× 3.1

Table 2: Best speedups of LSL3D versus best State-of-the-Art algorithms

6 Conclusion and future work

This article introduces a new algorithm, *LSL3D*, that combines a unification approach based on a finite state machine to improve its efficiency on simple images and a *double*-line mechanism and computational reuse for complex ones. On top of a scalar extension, we use SIMD (`SSE4`, `AVX2`, `AVX512`) instructions to accelerate the RLE compression and decompression steps.

Evaluation of performances on medical datasets and random images shows that *LSL3D* outperforms State-of-the-Art algorithms by a factor ×1.5 up to a factor × 3.1, on an Intel Xeon.

Future works will address the parallelization of this algorithm for multi-core CPUs and GPUs.

References

1. Abuzaghleh, O., Barkana, B. D., and Faezipour, M. Noninvasive real-time automated skin lesion analysis system for melanoma early detection and prevention. *IEEE journal of translational engineering in health and medicine 3* (2015), 1–12.
2. Allegretti, S., Bolelli, F., and Grana, C. Optimized Block-Based Algorithms to Label Connected Components on GPUs. *IEEE Transactions on Parallel and Distributed Systems 31*, 2 (Feb. 2020), 423–438.
3. Barnat, J., Bauch, P., Brim, L., and Češka, M. Computing strongly connected components in parallel on CUDA. In *2011 IEEE International Parallel Distributed Processing Symposium* (2011), pp. 544–555.
4. Bolelli, F., Allegretti, S., and Grana, C. One DAG to rule them all. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (Jan. 2021).
5. Bolelli, F., Baraldi, L., Cancilla, M., and Grana, C. Connected Components Labeling on DRAGs. In *2018 24th International Conference on Pattern Recognition (ICPR)* (Beijing, Aug. 2018), IEEE, pp. 121–126.
6. Bolelli, F., Cancilla, M., Baraldi, L., and Grana, C. Toward reliable experiments on the performance of connected components labeling algorithms. *Journal of Real-Time Image Processing (JRTIP)* (2018), 1–16.
7. Bolelli, F., Cancilla, M., Baraldi, L., and Grana, C. Toward reliable experiments on the performance of Connected Components Labeling algorithms. *Journal of Real-Time Image Processing 17*, 2 (Apr. 2020), 229–244.
8. Breen, E. J., and Jones, R. Attribute Openings, Thinnings, and Granulometries. *Computer Vision and Image Understanding 64*, 3 (Nov. 1996), 377–389.
9. Chabardès, T., Dokládal, P., and Bilodeau, M. A labeling algorithm based on a forest of decision trees. *Journal of Real-Time Image Processing 17*, 5 (Oct. 2020), 1527–1545.

10. Chen, W., Giger, M. L., and Bick, U. A fuzzy c-means (FCM)-based approach for computerized segmentation of breast lesions in dynamic contrast-enhanced mr images. *Academic radiology 13*, 1 (2006), 63–72.
11. Farhat, W., Faiedh, H., Souani, C., and Besbes, K. Real-time embedded system for traffic sign recognition based on ZedBoard. *Journal of Real-Time Image Processing 16*, 5 (2019), 1813–1823.
12. Grana, C., Baraldi, L., and Bolelli, F. 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. Springer International Publishing, Cham, 2016, pp. 431–440.
13. Grana, C., Borghesani, D., and Cucchiara, R. Optimized Block-Based Connected Components Labeling With Decision Trees. *Transactions on Image Processing 19*, 6 (June 2010), 1596–1609.
14. Gupta, S., Palsetia, D., Patwary, M. A., Agrawal, A., and Choudhary, A. A new parallel algorithm for two-pass connected component labeling. In *Parallel & Distributed Processing Symposium Workshops (IPDPSW)* (2014), IEEE, pp. 1355–1362.
15. Haralick, R. Some neighborhood operations. In *Real-Time Parallel Computing Image Analysis* (1981), Plenum Press, pp. 11–35.
16. He, L., Chao, Y., and Suzuki, K. A Linear-Time Two-Scan Labeling Algorithm. In *2007 IEEE International Conference on Image Processing* (San Antonio, TX, USA, 2007), IEEE, pp. V – 241–V – 244.
17. He, L., Ren, X., Gao, Q., Zhao, X., Yao, B., and Chao, Y. The connected-component labeling problem: a review of state-of-the-art algorithms. *Pattern Recognition 70* (2017), 25–43.
18. He, L., Zhao, X., Chao, Y., and Suzuki, K. Configuration-Transition-Based Connected-Component Labeling. *IEEE Transactions on Image Processing 23*, 2 (Feb. 2014), 943–951.
19. Hennequin, A., Lacassagne, L., Cabaret, L., and Meunier, Q. A new direct connected component labeling and analysis algorithms for GPUs. In *2018 Conference on Design and Architectures for Signal and Image Processing (DASIP)* (2018), IEEE, pp. 76–81.
20. Hwu, W. W., Ed. *GPU Computing Gems*. Morgan Kaufman, 2001, ch. 35: Connected Component Labeling in CUDA.
21. Joshi, K. A., and Thakore, D. G. A survey on moving object detection and tracking in video surveillance system. *International Journal of Soft Computing and Engineering 2*, 3 (2012), 44–48.
22. Khan, N., Ahmed, I., Kiran, M., Rehman, H., Din, S., Paul, A., and Reddy, A. G. Automatic segmentation of liver & lesion detection using h-minima transform and connecting component labeling. *Multimedia Tools and Applications 79*, 13 (2020), 8459–8481.
23. Komura, Y. GPU-based cluster-labeling algorithm without the use of conventional iteration: application to swendsen-wang multi-cluster spin flip algorithm. *Computer Physics Communications* (2015), 54–58.
24. Lacassagne, L., and Zavidovique, A. B. Light speed labeling for RISC architectures. In *IEEE International Conference on Image Analysis and Processing (ICIP)* (2009).
25. Lacassagne, L., and Zavidovique, B. Light speed labeling: Efficient connected component labeling on RISC architectures. *Journal of Real-Time Image Processing 6*, 2 (June 2011), 117–135.

26. Lemaitre, F., Hennequin, A., and Lacassagne, L. How to speed connected component labeling up with SIMD RLE algorithms. In *ACM Workshop on Programming Models for SIMD/Vector Processing* (2020), pp. 1–8.
27. Lemaitre, F., Hennequin, A., and Lacassagne, L. 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, Feb. 2020), ACM, pp. 1–8.
28. Lifeng He, Yuyan Chao, and Suzuki, K. A Run-Based Two-Scan Labeling Algorithm. *IEEE Transactions on Image Processing 17*, 5 (May 2008), 749–756.
29. Lifeng He, Yuyan Chao, and Suzuki, K. Two Efficient Label-Equivalence-Based Connected-Component Labeling Algorithms for 3-D Binary Images. *IEEE Transactions on Image Processing 20*, 8 (Aug. 2011), 2122–2134.
30. Litjens, G., Sánchez, C. I., Timofeeva, N., Hermsen, M., Nagtegaal, I., Kovacs, I., Hulsbergen-Van De Kaa, C., Bult, P., Van Ginneken, B., and Van Der Laak, J. Deep learning as a tool for increased accuracy and efficiency of histopathological diagnosis. *Scientific reports 6* (2016).
31. Lucchi, A., Li, Y., and Fua, P. Learning for Structured Prediction Using Approximate Subgradient Descent with Working Sets. In *2013 IEEE Conference on Computer Vision and Pattern Recognition* (Portland, OR, USA, June 2013), IEEE, pp. 1987–1994.
32. Marcus, D. S., Fotenos, A. F., Csernansky, J. G., Morris, J. C., and Buckner, R. L. Open Access Series of Imaging Studies: Longitudinal MRI Data in Nondemented and Demented Older Adults. *Journal of Cognitive Neuroscience 22*, 12 (Dec. 2010), 2677–2684.
33. Matsumoto, M., and Nishimura, T. Mersenne twister web page: http://www.math.sci.hiroshima-u.ac.jp/ m-mat/MT/emt.html.
34. Millet, M., Rambaux, N., Petreto, A., Lemaitre, F., and Lacassagne, L. A new processing chain for detection and tracking of meteors from space. In *International Meteor Conference* (Sept. 2021).
35. Nazlibilek, S., Karacor, D., Ercan, T., Sazli, M. H., Kalender, O., and Ege, Y. Automatic segmentation, counting, size determination and classification of white blood cells. *Measurement 55* (2014), 58–65.
36. Playne, D. P., and Hawick, K. A new algorithm for parallel connected-component labelling on GPUs. *IEEE Transactions on Parallel and Distributed Systems 29*, 6 (2018), 1217–1230.
37. Rosenfeld, A., and Platz, J. Sequential operator in digital pictures processing. *Journal of ACM 13,4* (1966), 471–494.
38. Salau, J., and Krieter, J. Analysing the space-usage-pattern of a cow herd using video surveillance and automated motion detection. *Biosystems Engineering 197* (2020), 122–134.
39. Sochting, M., Allegretti, S., Bolelli, F., and Grana, C. A Heuristic-Based Decision Tree for Connected Components Labeling of 3D Volumes. In *International Conference on Pattern Recognition* (2021), p. 8.
40. Veillon, F. One pass computation of morphological and geometrical properties of objects in digital pictures. *Signal Processing 1,3* (1979), 175–179.
41. Weng, H.-M., and Chiu, C.-T. Resource efficient hardware implementation for real-time traffic sign recognition. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (2018), IEEE, pp. 1120–1124.
42. Wu, K., Otoo, E., and Shoshani, A. Optimizing connected component labeling algorithms. In *Medical Imaging* (San Diego, CA, Apr. 2005), J. M. Fitzpatrick and J. M. Reinhardt, Eds., p. 1965.

43. Ziegler, G., and Rasmusson, A. Efficient volume segmentation on the GPU. In *GPU Technology Conference* (2010), Nvidia, Ed., pp. 1–44.