A Heuristic-Based Decision Tree for Connected Components Labeling of 3D Volumes: Implementation and Reproducibility Notes

Federico Bolelli(✉), Stefano Allegretti, and Costantino Grana

Dipartimento di Ingegneria "Enzo Ferrari",
Università degli Studi di Modena e Reggio Emilia, Modena, Italy
{federico.bolelli,stefano.allegretti,costantino.grana}@unimore.it

Abstract. This paper provides a detailed description of how to install, setup, and use the YACCLAB benchmark to test the algorithms published in "A Heuristic-Based Decision Tree for Connected Components Labeling of 3D Volumes," underlying how the parameters affect and influence experimental results.

1 Introduction

Although introduced many decades ago [31], the task of labeling objects inside binary images is still employed in several scenarios, whenever an identification of segmented visual objects or image regions is required. This procedure, usually identified as Connected Components Labeling or CCL in short, has a unique and exact solution which provides a description of the objects inside binary images, represented by an output symbolic image where pixels of a connected component are assigned the same integer identifier.

As a matter of fact, many state-of-the-art image processing and computer vision pipelines exploit CCL as a fundamental pre- or post-processing step. The fields of application of such an algorithm range from Object Tracking [18] to Document Restoration [10,25], including Image Segmentation [1,29], Medical Imaging [13,19,30] and many others [6,17]. For this reason, having a fast and efficient algorithm, able to minimize its impact on image analysis tasks, is undoubtedly very advantageous. This is why the research efforts in labeling techniques have such a very long story, full of different strategies and improvements targeting both sequential [8,9,20,22,23,34] and parallel architectures [2,3,5,24,28,32,35]. Among them, some of the most promising techniques that led to major breakthroughs in the field consist in the usage of Decision Trees (DTrees), combined with the 2×2 block-based approach. A detailed description of the algorithms based on these paradigms is provided in [7]. Moreover, algorithmic solutions relying on DTrees have demonstrated their effectiveness even when applied, with the necessary variations, to parallel architectures [4,12].

B. Kerautret et al. (Eds.): RRPR 2021, LNCS 12636, pp. 139–145, 2021.
https://doi.org/10.1007/978-3-030-76423-4_9

Unfortunately, existing techniques for the generation of DTrees become quickly unfeasible when the size of the mask used to scan the input image increases. This prevented the application of block-based trees to 3D scenarios. In order to compensate for this limitation a novel heuristic algorithm, based on decision tree learning and named Entropy Partitioning Decision Tree (EPDT), has been presented in [33]. This algorithm allows to compute near-optimal decision trees for large scan masks, overtaking the limitations of existing approaches.

This paper describes the benchmark used to evaluate the performance of EPDT-generated algorithms, focusing on how to configure it to reproduce the experiments reported in [33].

2 The Evaluation Framework

YACCLAB, Yet Another Connected Components Labeling Benchmark, has been originally released in [21] with the aim of providing a fair comparison and evaluation of CCL algorithms. The benchmark has been later improved with additional datasets, tests and with an extension to 3D and GPU algorithms [5,11]. After its first appearance in 2016, it has been used by many authors [14,15,36] to compare the performance of novel proposals with state-of-the-art solutions, thus setting a *de-facto* standard.

When measuring the performance of an algorithm several details should be taken into account, as they could significantly influence the performance. However, CCL is a well-defined problem and the burden of evaluation can be reduced to the measure of execution "speed".

The main elements that affect execution speed can be resumed as follows: data on which tests are performed, implementation details, hardware capabilities, and code optimization provided by the compiler. YACCLAB takes all these aspects into account; the benchmark is open-source and provides an implementation of state-of-the-art algorithms, directly including the source code released together with the scientific papers whenever available. Given its open-source nature, anyone can verify literature claims testing the algorithms with any combination of hardware architecture, operating system and build tools.

The public dataset provided with the benchmark covers most of CCL fields of application, including 2D images and 3D volumes of both real world and synthetically generated domains. A detailed description of the YACCLAB dataset is available in [5]. Because experimental results reported in [33] concern 3D EPDT-generated algorithms, the general properties of 3D datasets are summarized in Table 1 and a brief description follows:

- *OASIS* is a dataset of medical MRI data taken from the OASIS project [27], binarized with the Otsu threshold;
- *Mitochondria* is the Electron Microscopy Dataset [26], which contains binary sections taken from the CA1 hippocampus;
- *Hilbert* consists of the 3D Hilbert curve, which is a fractal space-filling curve, obtained at different iterations (1 to 6) of the construction method.

Table 1. Properties of 3D datasets in terms of foreground pixel density, number of connected components (objects), number of volumes, and resolution.

	Density		*Objects*		*Volumes*	*Resolution*
	μ	σ	μ	σ		
Hilbert	0.055291	0.0873024	1	0	373	$256 \times 256 \times 128$
Mitochondria	0.0588272	0.00599026	40	5.09902	3	$1024 \times 768 \times 165$
OASIS	0.198208	0.0245339	3199	1027.79	6	$128 \times 128 \times 128$

The source code of the EPDT-generated algorithms as well as the benchmarking suite is available at https://github.com/prittt/YACCLAB.

3 How to Test EPDT-Generated Algorithms

In order to correctly install and run the current version of the YACCLAB benchmark, the following packages, libraries and utilities are required:

- CMake 3.13 or higher (https://cmake.org);
- OpenCV 3.0 or higher (http://opencv.org);
- Gnuplot (http://www.gnuplot.info);
- A C++ compiler supporting C++14.

The installation procedure is well detailed in the aforementioned GitHub repository; the main steps can be resumed as follows:

- Clone the repository;
- Generate the YACCLAB project using CMake;
- Set the configuration file `config.yaml` placed in the installation folder;
- Open the project folder, build and run.

When configuring the project through CMake the flags `YACCLAB_ENABLE_3D` and `YACCLAB_ENABLE_EPDT_*` must be enabled in order to set-up the benchmark for 3D algorithms and to include EPDT implementations. The CMake file should automatically find the OpenCV installation path, otherwise it must be manually specified. The flag `YACCLAB_DOWNLOAD_DATASET_3D` must be enabled if the user wants CMake to automatically download the YACCLAB 3D dataset. CMake will automatically generate the C++ project for the selected compiler.

YACCLAB allows to perform multiple tests: *correctness* is an initial validation of the algorithms; *average* runs algorithms on every image of a dataset, measuring the average run-time; *average_with_steps* measures separated run-times for the different steps each algorithm is composed of, including multiple scans over the input image and allocation/deallocation of data structures; *granularity* uses synthetic images to evaluate the performance of different approaches in terms of scalability on the number of pixels, foreground density and pattern granularity; *memory* reports the expected number of memory accesses required by an algorithm on a reference dataset.

YACCLAB stores experimental results in the `output path` specified by the configuration file. Multiple output formats including plain text, bar chart and LaTeX table will be produced.

CCL algorithms are independent of the Union-Find strategy employed. For this reason YACCLAB provides a Union-Find templated implementation for most of the algorithms, thus being able to compare each algorithm (but those for which the label solver is built-in) with different label solving strategies: standard Union-Find (UF), Union-Find with Path Compression (UFPC) [34], Interleaved Rem's algorithm with splicing (RemSP) [16] and Three Table Array (TTA) [22]. This standardization reduces code variability, allowing to separate label solving data structures from CCL strategies, and provides fair comparisons without negatively impacting execution time.

4 Experiments Reproducibility

```
CPU 3D 26-way connectivity:
  execute: true
  perform:
      correctness: true
      average: true
      average_with_steps: true
      density: false
      granularity: false
      memory: true
  algorithms:
      - EPDT_3D_19c_RemSP
      - EPDT_3D_22c_RemSP
      - EPDT_3D_26c_RemSP
      - LEB_3D_TTA
      - RBTS_3D_TTA
```

Listing 1. Excerpt of the YAML configuration file.

The EDPT algorithms were tested on an Intel(R) Core(TM) i7-4790 CPU @ 3.60 GHz with Windows 10.0.17134 (64 bit) OS and the MSVC 19.15.26730 compiler. The benchmark was compiled for x64 architecture with optimizations enabled. It is worth noticing that most compilers need several minutes to build EPDT algorithms; in particular, some of them actually fail to compile EDPT_26c. For these reasons, aforementioned algorithms are optional and must be singularly enabled with CMake, as described in Sect. 3.

The performance of EPDT-generated algorithms have been compared to state-of-the-art solutions over the collection of 3D datasets included in YACCLAB and described in Sect. 2. In order to reproduce the same experiments reported in [33], the `CPU 3D 26-way connectivity` section of the configuration file must have its `execute`, `perform` and `algorithms` fields set as in Listing 1. The other fields can remain as default. Finally, 2D tests can be disabled to avoid useless experiments.

5 Conclusion

We described how to reproduce the experimental results reported in [33]. The environment employed for testing the algorithms can significantly affect performance. Cache size and RAM speed can change absolute results while preserving relative performance. Operative System and compiler are likely to heavily influence the outcome.

References

1. Abramov, A., Kulvicius, T., Wörgötter, F., Dellen, B.: Real-time image segmentation on a GPU. In: Keller, R., Kramer, D., Weiss, J.-P. (eds.) Facing the Multicore-Challenge. LNCS, vol. 6310, pp. 131–142. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-16233-6_14
2. Allegretti, S., Bolelli, F., Cancilla, M., Grana, C.: Optimizing GPU-based connected components labeling algorithms. In: 2018 IEEE International Conference on Image Processing, Applications and Systems (IPAS), pp. 175–180. IEEE (2018)
3. Allegretti, S., Bolelli, F., Cancilla, M., Grana, C.: A block-based union-find algorithm to label connected components on GPUs. In: Ricci, E., Rota Bulò, S., Snoek, C., Lanz, O., Messelodi, S., Sebe, N. (eds.) ICIAP 2019. LNCS, vol. 11752, pp. 271–281. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-30645-8_25
4. Allegretti, S., Bolelli, F., Cancilla, M., Pollastri, F., Canalini, L., Grana, C.: How does connected components labeling with decision trees perform on GPUs? In: Vento, M., Percannella, G. (eds.) CAIP 2019. LNCS, vol. 11678, pp. 39–51. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-29888-3_4
5. Allegretti, S., Bolelli, F., Grana, C.: Optimized block-based algorithms to label connected components on GPUs. IEEE Trans. Parallel Distrib. Syst. **31**, 423–438 (2019)
6. Berka, T.: The generalized feed-forward loop motif: definition, detection and statistical significance. Procedia Comput. Sci. **11**, 75–87 (2012)
7. Bolelli, F., Allegretti, S., Baraldi, L., Grana, C.: Spaghetti labeling: directed acyclic graphs for block-based connected components labeling. IEEE Trans. Image Process. **29**(1), 1999–2012 (2019)
8. Bolelli, F., Allegretti, S., Grana, C.: One DAG to rule them all. IEEE Trans. Pattern Anal. Mach. Intell. 1–12 (2021)
9. Bolelli, F., Baraldi, L., Cancilla, M., Grana, C.: Connected components labeling on DRAGs. In: International Conference on Pattern Recognition, pp. 121–126 (2018)
10. Bolelli, F., Borghi, G., Grana, C.: XDOCS: an application to index historical documents. In: Serra, G., Tasso, C. (eds.) IRCDL 2018. CCIS, vol. 806, pp. 151–162. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-73165-0_15
11. Bolelli, F., Cancilla, M., Baraldi, L., Grana, C.: Toward reliable experiments on the performance of Connected Components Labeling algorithms. J. Real-Time Image Proc. **17**(2), 229–244 (2018). https://doi.org/10.1007/s11554-018-0756-1
12. Bolelli, F., Cancilla, M., Grana, C.: Two more strategies to speed up connected components labeling algorithms. In: Battiato, S., Gallo, G., Schettini, R., Stanco, F. (eds.) ICIAP 2017. LNCS, vol. 10485, pp. 48–58. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-68548-9_5
13. Canalini, L., Pollastri, F., Bolelli, F., Cancilla, M., Allegretti, S., Grana, C.: Skin lesion segmentation ensemble with diverse training strategies. In: Vento, M., Percannella, G. (eds.) CAIP 2019. LNCS, vol. 11678, pp. 89–101. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-29888-3_8
14. Chabardès, T., Dokládal, P., Bilodeau, M.: A labeling algorithm based on a forest of decision trees. J. Real-Time Image Proc. **17**(5), 1527–1545 (2019). https://doi.org/10.1007/s11554-019-00912-8
15. Chen, J., Nonaka, K., Sankoh, H., Watanabe, R., Sabirin, H., Naito, S.: Efficient parallel connected component labeling with a coarse-to-fine strategy. IEEE Access **6**, 55731–55740 (2018)

16. Dijkstra, E.W.: A Discipline of Programming. Prentice-Hall, Englewood Cliffs (1976)
17. Dinneen, M.J., Khosravani, M., Probert, A.: Using OpenCL for implementing simple parallel graph algorithms. In: Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA) (2011)
18. Dubois, A., Charpillet, F.: Tracking mobile objects with several Kinects using HMMs and component labelling. In: Workshop Assistance and Service Robotics in a Human Environment, International Conference on Intelligent Robots and Systems, pp. 7–13 (2012)
19. Eklund, A., Dufort, P., Villani, M., LaConte, S.: BROCCOLI: software for fast fMRI analysis on many-core CPUs and GPUs. Front. Neuroinform. **8**, 24 (2014)
20. Grana, C., Baraldi, L., Bolelli, F.: Optimized connected components labeling with pixel prediction. In: Blanc-Talon, J., Distante, C., Philips, W., Popescu, D., Scheunders, P. (eds.) ACIVS 2016. LNCS, vol. 10016, pp. 431–440. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-48680-2_38
21. Grana, C., Bolelli, F., Baraldi, L., Vezzani, R.: YACCLAB - yet another connected components labeling benchmark. In: 2016 23rd International Conference on Pattern Recognition (ICPR), pp. 3109–3114. Springer (2016)
22. He, L., Chao, Y., Suzuki, K.: A linear-time two-scan labeling algorithm. In: International Conference on Image Processing, vol. 5, pp. 241–244 (2007)
23. He, L., Zhao, X., Chao, Y., Suzuki, K.: Configuration-transition-based connected-component labeling. IEEE Trans. Image Process. **23**(2), 943–951 (2014)
24. Komura, Y.: GPU-based cluster-labeling algorithm without the use of conventional iteration: application to the Swendsen-Wang multi-cluster spin flip algorithm. Comput. Phys. Commun. **194**, 54–58 (2015)
25. Lelore, T., Bouchara, F.: FAIR: a fast algorithm for document image restoration. IEEE Trans. Pattern Anal. Mach. Intell. **35**(8), 2039–2048 (2013)
26. Lucchi, A., Li, Y., Fua, P.: Learning for structured prediction using approximate subgradient descent with working sets. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1987–1994. IEEE (2013)
27. Marcus, D.S., Fotenos, A.F., Csernansky, J.G., Morris, J.C., Buckner, R.L.: Open access series of imaging studies (OASIS): longitudinal MRI data in nondemented and demented older adults. J. Cognitive Neurosci. **22**(12), 2677–2684 (2010)
28. Perri, S., Spagnolo, F., Corsonello, P.: A parallel connected component labeling architecture for heterogeneous systems-on-chip. Electronics **9**(2), 292 (2020)
29. Pollastri, F., Bolelli, F., Paredes, R., Grana, C.: Improving skin lesion segmentation with generative adversarial networks. In: IEEE 31st International Symposium on Computer-Based Medical Systems (CBMS), pp. 442–443. IEEE (2018)
30. Pollastri, F., Bolelli, F., Paredes, R., Grana, C.: Augmenting data with GANs to segment melanoma skin lesions. Multimed. Tools Appl. **79**(21), 15575–15592 (2019). https://doi.org/10.1007/s11042-019-7717-y
31. Rosenfeld, A., Pfaltz, J.L.: Sequential operations in digital picture processing. J. ACM **13**(4), 471–494 (1966)
32. Spagnolo, F., Frustaci, F., Perri, S., Corsonello, P.: An efficient connected component labeling architecture for embedded systems. J. Low Power Electron. Appl. **8**(1), 7 (2018)
33. Söchting, M., Allegretti, S., Bolelli, F., Grana, C.: A heuristic-based decision tree for connected components labeling of 3D volumes. In: 2020 25th International Conference on Pattern Recognition (ICPR). IEEE (2021)

34. Wu, K., Otoo, E., Suzuki, K.: Two strategies to speed up connected component labeling algorithms. Technical report. LBNL-59102, Lawrence Berkeley National Laboratory (2005)
35. Zavalishin, S., Safonov, I., Bekhtin, Y., Kurilin, I.: Block equivalence algorithm for labeling 2D and 3D images on GPU. Electron. Imaging **2016**(2), 1–7 (2016)
36. Zhang, D., Ma, H., Pan, L.: A gamma-signal-regulated connected components labeling algorithm. Pattern Recogn. **91**, 281–290 (2019)