YACCLAB
Yet Another Connected Components Labeling Benchmark

Costantino Grana, Federico Bolelli, Lorenzo Baraldi and Roberto Vezzani
University of Modena and Reggio Emilia

Problem Statement

The problem of labeling the connected components (CCL) of a binary image is well-defined and several proposals have been presented in the past. Unfortunately, almost none of them were compared on the same data.

Goal: tackles the problem of evaluating the execution speed of different strategies to solve the CCL problem on binary images.

When talking of "speed of execution", one should answer to three questions:

1. On what data?
   ↳ «The YACCLAB dataset»
2. On which machine?
   ↳ «Yours»
3. With which implementation?
   ↳ «The public ones if available, ours otherwise»

The YACCLAB Dataset

Random Noise *MIRflickr* *Hamlet* *3DPeS*

The YACCLAB dataset includes both synthetic and real images and is suitable for a wide range of applications (ranging from document processing to surveillance) and features a significant variability in terms of resolution, image density and number of components.

The YACCLAB Project

YACCLAB is an open source C++ project which runs CCL algorithms and performs tests:

- Correctness test;
- Average run-time tests;
- Density and size tests: check the performance of different CCL algorithms on images with varying foreground density and size.

Test process can be repeated more times to get the minimum execution time for each image: this allows to overlook delays produced by other running processes.

New resources can be easily integrated into the project.

Parameter name	Description
input_path	Folder on which datasets are placed
output_path	Folder on which result are stored
write_n_labels	Whether to report the number of Connected Components in output files
Correctness tests	
check_8connectivity	Whether to perform correctness tests
check_list	List of datasets on which CCL algorithms should be checked
Density and size tests	
ds_perform	Whether to perform density and size tests
ds_colorLabels	Whether to output a colorized version of input images
ds_testsNumber	Number of runs
ds_saveMiddleTests	Whether to save the output of single runs, or only a summary of the whole test
Average execution time tests	
at_perform	Whether to perform average execution time tests
at_colorLabels	Whether to output a colorized version of input images
at_testsNumber	Number of runs
at_saveMiddleTests	Whether to save the output of single runs, or only a summary of the whole test
averages_tests	List of algorithms on which average execution time tests should be run
Algorithm configuration	
CCLAlgorithmsFunc	List of available algorithms (function names)
CCLAlgorithmsName	List of available algorithms (display names for charts)

YACCLAB configuration parameters

Available Algorithms

We included in YACCLAB the following algorithms, with our implementation, if the authors did not provide source code:

- ***CT*** (Contour Tracing): representative algorithm of the contour tracing technique (our implementation);
- ***DiStefano***: requires multiple searches over the equivalence array, leading to a non-linear behavior with respect to the number of labels (our implementation);
- ***BBDT*** (Block Based with Decision Tree): block based (2x2) scanning algorithm with online equivalence resolution that uses an optimal decision tree to scan input image (public implementation);
- ***CCIT***: variation of BBDT Block Based analysis (public implementation);
- ***LSL*** (Light Speed Labeling): a run based algorithms that is not always correct (our implementation).
- ***SAUF*** (Scan Array Union Find): a pixel based scanning algorithm with online equivalence resolution that uses a decision tree for accessing only the minimum number of already scanned labeled pixels (public implementation).

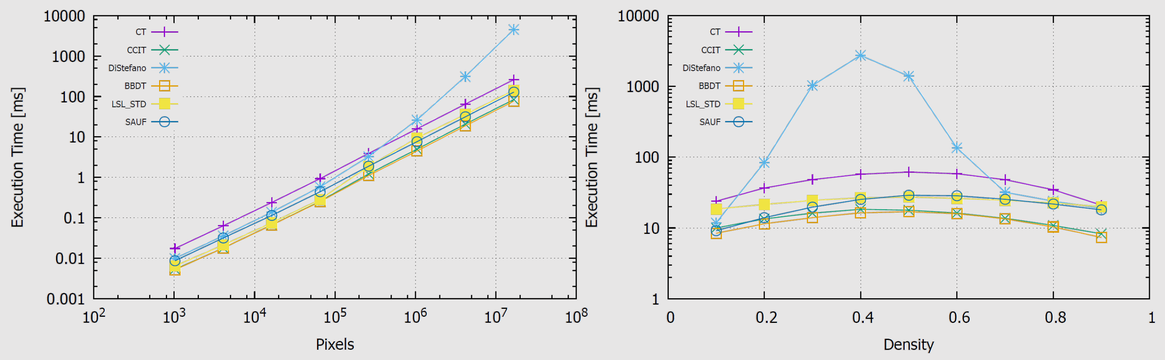
Experimental Analysis

Test environments:

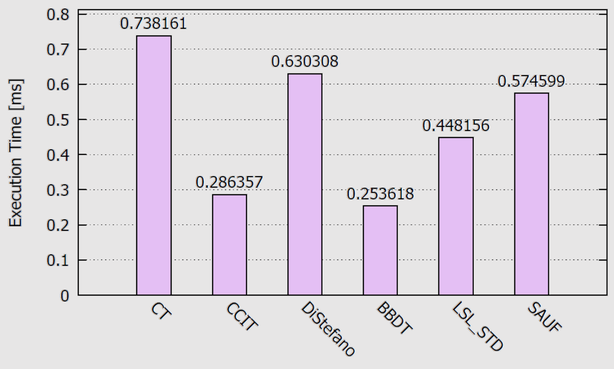
- a Windows PC with a i7-4790 CPU @ 3.60 GHz and Microsoft Visual Studio 2013;
- a Linux workstation with a Xeon CPU E5-2609 v2 @ 2.50GHz and GCC 5.2;
- an Intel Core Duo @ 2.8 GHz running OS X with X Code 7.2.1:

	CT	CCIT	DiStefano	BBDT	LSL	SAUF
MIRflickr	1.51	0.67	1.29	**0.61**	1.06	0.89
Tobacco800	30.71	26.41	21.79	**17.39**	60.53	22.01
3DPeS	1.64	1.01	1.34	**1.00**	3.43	1.03
Hamlet	21.47	14.98	14.01	**10.23**	34.06	13.12

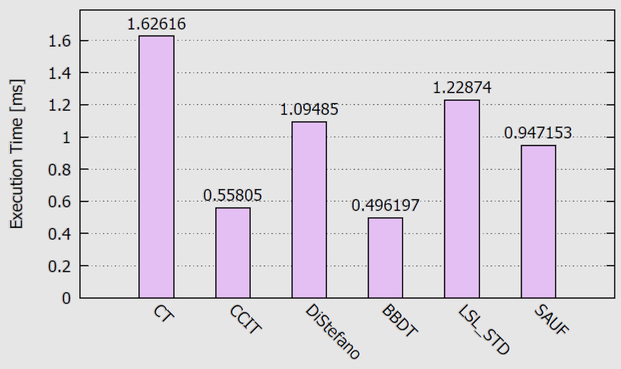
Average results in ms with OS X



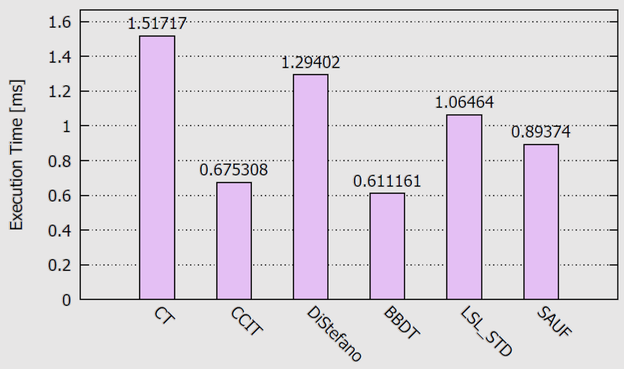
Density and size test on Windows



Average run-time test on MIRflickr with Windows



Average run-time test on MIRflickr with Linux



Average run-time test on MIRflickr with OS X

Check out the project website and join us on GitHub