# Pyramidal Connected Component Labeling of Image

**Majid Banaeyan**[1]
**Pattern Recognition and Image Processing Group**
**193/03, TU Wien, Vienna, Austria**
majid@prip.tuwien.ac.at

**Wlater G. Kropatsch**
**Pattern Recognition and Image Processing Group**
**193/03, TU Wien, Vienna, Austria**
krw@prip.tuwien.ac.at

**Rashid Zamanshoar، High Performance Computing, Farmaniyeh, Tehran,Iran**
zamanshoar@imp.ir

## Abstract
In this paper, a new logarithmic-time algorithm is presented which simultaneously assigns labels to all connected components of a binary image. The main advantage of the proposed algorithm is to propagate information in the logarithmic order by using the graph pyramid structure. The irregular combinatorial pyramid is employed to construct the hierarchy and the maximum independent edge sets (MIES) are used to create this pyramid structure in parallel. To assign a label to each connected component, instead of the common linear-time raster scan techniques, only two traversings of the existing pyramid are needed. First, contracting each connected component to a single vertex maintaining all connectivity relations at the top of the pyramid and assign a new label to each vertex. Second, to go top-down and to propagate this unique label into each individual pixel of the binary image. In addition, no relabeling is needed throughout the whole process as it is needed by other algorithms. Finally, the experimental results show the proposed algorithm outperforms the other state-of-the-arts for large images.

**Keywords**: Combinatorial Pyramid, Parallel Processing, Invariant Topology, Connected Component Labeling, image graph pyramid, image processing.

1- Majid Banaeyan and Rashid Zamanshoar are supported by Pattern Recognition andImage Processing (PRIP) group via PRIP-Club and High Performance Computing IPM-HPC, respectively.

## 1. Introduction

Connected Component Labeling (CCL) is a fundamental task in computer vision whether over a binary image to distinguish between background or foreground components or in image segmentation procedure to assign a unique label to each different region. It employs in many image processing fields such as image analysis, pattern recognition and image understanding. The role of the CCL is to assign a same label into each individual pixel of a region. There are plenty of algorithms which consider this task from different viewpoints [1-3]. However, such algorithms mostly can be divided into two main categories [4] which are based on label-propagation [5-7] or label-equivalence-resolving [8, 9]. The linear-time complexity is the common property in both techniques. In other words, such algorithms may differ from one-scan or two-scan searching through the entire image but the real fact is that all are in the order of image size, O($N^2$). Therefore, we propose our new algorithm using the pyramidal structure [10, 11] which reduces the complexity into logarithmic order. Moreover, in our connected component technique the combinatorial pyramid is employed that preserves the topological property between connected regions [12].

The remaining of this paper is organized as follows: In the upcoming two subsections combinatorail Pyramid (CP) and extended canonical encoding are briefly recalled. Section 2 explains the parallel processing and its different architectures. Section 3 describes our proposed algorithm in details. Moreover, there is a discussion about isolated vertices in section 4. In section 5, the GPU implementation and the experimental results are described. Finally, the conclusion and future work are presented at section 6.

### 1.1 Combinatorial Pyramid

Combinatorial Pyramid (CP) includes a stack of successively reduced graph [13]. Each smaller graph is built from the graph below by contracting or removing a set of edges. The basic elements in combinatorial pyramid are darts or half-edges and two permutation functions over each dart, $\alpha$(d) and $\sigma$(d), encode the space around each vertex and between adjacent ones [14]. The CP preserves topological information such as inclusion relations and multiadjacency and itcan be extended to higher dimensions. Moreover, they consider global and local properties through the same data structure. They extensively used in image segmentation algorithms [15].

### 1.2 Extended Canonical Encoding

The canonical encoding [16] stores the combinatorial pyramid in a one dimensional array which takes the memory space as much as the base level which is almost 4 times of the image size. This representation of the pyramid data structure keeps the history of the constructed pyramid in the passive part which can later use them in the reconstruction procedure. During the construction of the pyramid selected contraction kernels first belong to the active part, however, after every contraction they transfer into the passive part. The contraction procedure will be terminated whenever there is no pair dart in the active part of the array. The canonical algorithm is a very efficient method for representing the graph pyramid structure. However, it is in the sequential manner. Therefore, in our proposed algorithm we extend the same idea and modify the algorithm for parallel usage. In fact, the canonical array is partitioned into different part as large as each connected component and then because the dart sets belong to each CC are independent to each other, therefore, each CC has its own passive and active part through the one dimensional array.

## 2. Parallel Processing

Parallel processing is a method in computer sciences for the execution of task on computing platforms containing multiple processors with one or many threads on each one. If the task is not inherently sequential, however, potentially enables to subdivide into several independent sub-parts, it would be possible to employ parallel processing methods for more effcient performance. The number of independent parts, the available hardware and the method of implementation effects on speed up factor in parallel processing.

In [17] four types of parallel processing was introduced, namely, the single-instruction single-data (SISD), the single-instruction multiple-data (SIMD) , the multiple-instruction, single-data (MISD) and the multiple-instruction multiple-data (MIMD) systems. Among them the SMID and MIMD are the most commonly used ones. Here, because of the available GPU specications and the properties of the proposed algorithm the SIMD architecture is employed.

### 2.1. GPU architecture

The general-purpose computing on graphics processing units (GP-GPU) is the concept of using GPUs to execute the tasks which execution of them by CPU is not effcient. The parallel computing in GPUs makes it possible to achieve a signicant computational performance of them. Also, GPUs make to reduce the load on CPUs and liberate other resources of a given computer system. GPUs utilize a massively parallel architecture with a large number of multiprocessors, namely Stream Multiprocessors (SM). Each of SMs containing a number of streaming processors (SP), a memory shared by SPs is called shared memory and a control unit. Moreover, there is a global memory between all the SMs, which is slower than the shared memory and is called Global memory. In addition to the mentioned memory types, there are other types of memory such as texture, constant and etc are existed, each of them are used for specic purposes. See [18] for further details. Figure 1 shows the simplied architecture of the GPU.

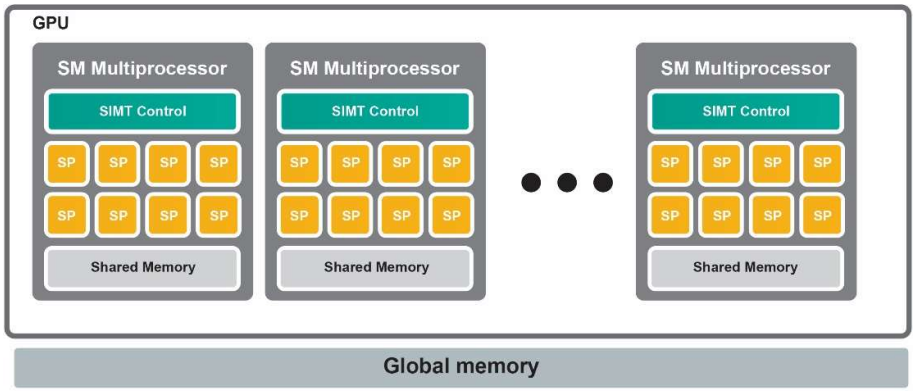


**Figure (1) The simplified architecture of the GPU**

### 2.2 CUDA Programming

Initially, GPGPUs were programming with using APIs (application programming interface) like DirectX and OpenGL. Using these APIs was very dicult in practice and even impossible in many cases because the program had to be tailored to a specic structure. A signicant improvement happened when Compute Unied Device Architecture (CUDA) in 2006 by NVIDIA were introduced [19]. This programming model is designed explicitly for Nvidia devices. Its programming model uses the C/C++ programming language. Although there is another programming model called OpenCL in [20] that can be used for programming on all graphics processors, because of its lower performance, most NVIDIA GPUs users tend to Use the CUDA programming model. CUDA focus on massively data-parallel and task-parallel kernels. It makes GPUs use thousands of threads into a grid of thread blocks. A Streaming Multiprocessor (SM) be able to execute many threads concurrently. GPU threads are grouped into the block of threads and each thread has a unique ID in its block. Moreover, the blocks are grouped in a grid of blocks and also each block has a unique ID on the grid. The SM executes threads in groups of 32 parallel threads called warps [21]. Figure 2-a shows the grid of blocks and block of threads and Figure 2-b shows CUDA memory hierarchy.

In the CUDA programming model, at  rst CPU (Host) copy needed data from host memory to GPU Memory that formerly known as global memory. Then GPU (Device), performs the execution of the kernel. At the end of a kernel execution, the CPU can copy back the GPU memory to acquire output data.

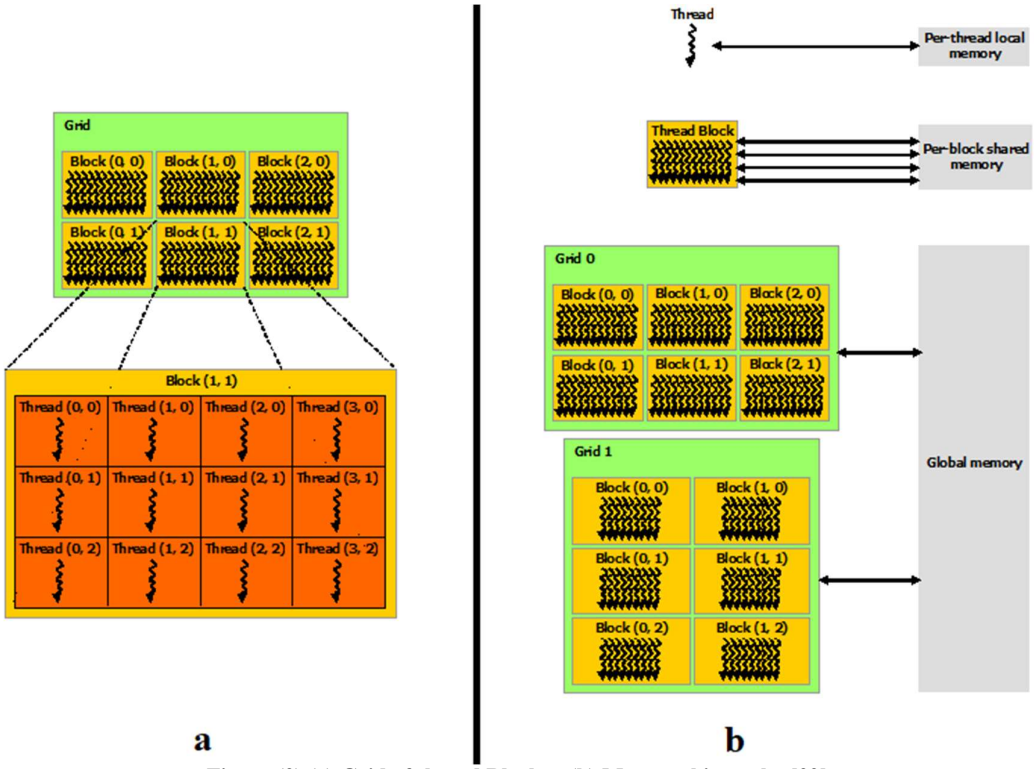


**Figure (2) (a) Grid of thread Blocks - (b) Memory hierarchy [22]**

## 3. Pyramidal Connected Component labeling Algorithm

In this section the proposed algorithm is presented in details. The PCCL consists of two main steps.The first one is to go up through the pyramid and contract all CKs to reach on the top level and assign unique labels to each vertex. The second step is to move down from the top and propagate the assigned labels into each individual nodes at the base level. Figure 3 graphically illustrates such two procedures.
In addition, the steps of the algorithm are shown in the table 1.

### 3.1. Reach to top of the pyramid

There are four steps to move up and to produce the next level of the pyramid. These steps are repeated in every level until we arrive into the top level. These steps are shown in the algorithm (table 1) in the loop and following are explained in details.

**Table 1- Pyramidal Connected Component Labeling Algorithm**

| **PCCL Algorithm** |
|---|
| **Input: G:(d, $\alpha$(d), $\sigma$(d))** |
| **Bottom-up procedure** |
| 1. **while** there are (d, $\alpha$(d)) to contract **do** |
| 1.1 Select the Contraction Kernels (find the maximal matching) |
| 1.2 Contracting the selected CKs simultaneously using identity reduction function |
| 1.3 Select the removal kernels |
| 1.4 Simplication |
| **end while** |
| **Top-down procedure** |
| 2.assign a unique label to each CCs |
| 3.Propagate labels into individual pixels |

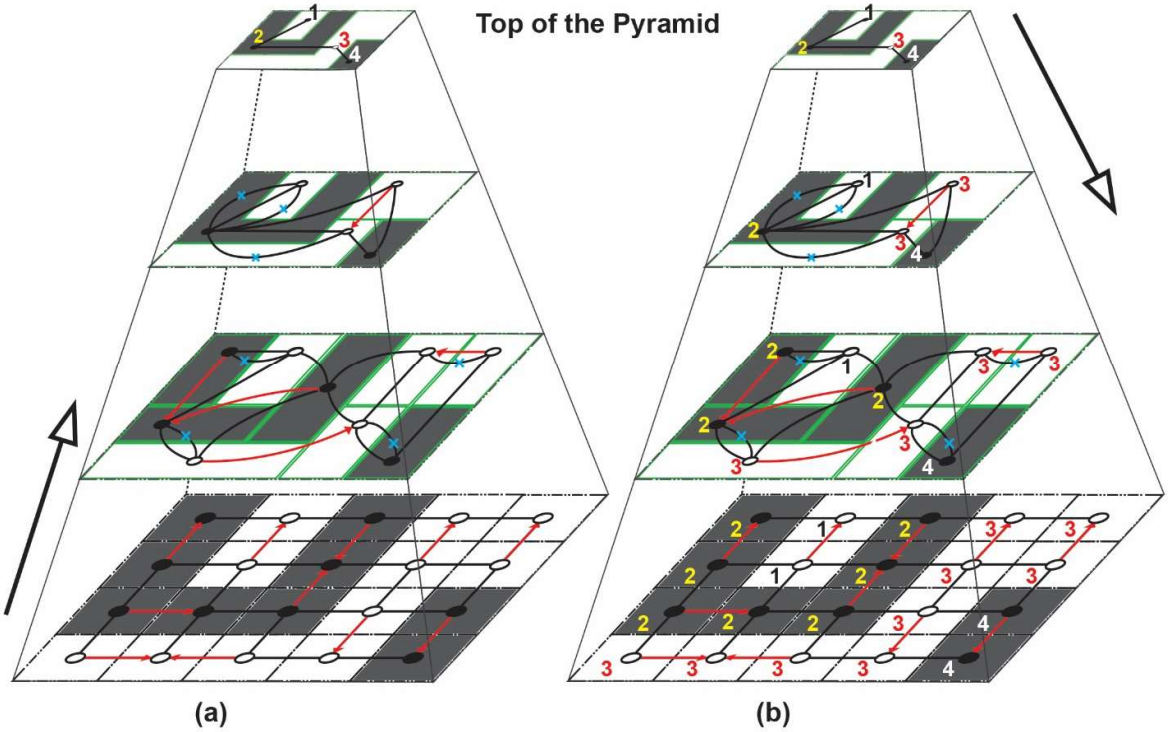


**Figure (3) The illustration of the PCCL algorithm. (a) The bottom-up procedure, (b) the top-down labels propagation**

**Selecting the Contraction Kernels (CKs):** Providing a collection of CKs is the main task in building the pyramid. In base level of the pyramid we have the grid network of vertices with two values. Contraction kernels are selected in a way that only the vertices with the same color value get chosen. In fact, here, in connected component labeling task, two adjacent vertices with different value will never be contracted and the edge between them remain untouched through entire pyramid. It means that the corresponding darts of such an edge must not never selected in the CKs. To this purpose, we assign a value to each dart as its sign by following definition:

**Denition 1.** The sign of the dart is called ***sd*** and defines such bellow which p and q are the vertices between the corresponding dart and the g is the gray value:

$$\begin{cases} sd = 0, & if g(p) = g(q) \\ sd = 1, & if g(p) \neq g(q) \end{cases} \quad (1)$$

To select the CKs we propose a new technique which is similar to MIES[23]. Our algorithm and MIES both have the same goal which is the selecting of the maximal edge sets for the contraction task. However, the MIES works on vertices and edges of the graph while instead we deal with darts of the combinatorial structure. Moreover, in order to find its maximal matching the MIES uses the MIS[23] technique which is in sequential manner.On he other hand, we propose our technique which select the contraction dart sets in parallel. To select a collection of contraction darts sets we define a function which locally selects the proper candidate dart for the contraction task. To define such function first consider the dependent darts around a pair dart which is illustrated in Figure 4.

**Defnition 2**. The DDS includes at most six darts which are dependent to each other during the contraction task. In fact, in the canonical representation of combinatorial map a dependent darts set (DDS) is a collection of darts adjacent to a pair dart (d, $\alpha(d)$) which their ordering number need to be updated after each contraction operation and includes the two contracted darts as well.

**Definition 3.** $f(d) = sd(d) + rn(d)$, which the *rn* is a random real positive number less than 1 and therefore the *f(d)* value is a positive real number less than 2.

**Defnition 4.** A selecting dart function (*Sedart*) is a local operation which select the local dart with minimum value of its f in DDS.

$$Sedart(d) = aegmin(f(d), f(\alpha(d)), f(\sigma(d)), f(\sigma^{-1}(d), f\left(\sigma(\alpha(d))\right), f\left(\sigma^{-1}(\alpha(d))\right))$$

The output of the sedart function is the dart which demonstrates the contraction kernel (d,$\alpha$(d)). Essentially, it is a decision function which selects the best dart as a candidate for contraction. Having the sd value in computing the $f$ function results in an interesting property which states that the contraction procedure will continue until there is a dart in the DDS with a sd value 0. In other words, it means whenever the local minimum of a DDS is greater than 1 there are no further darts to be selected as CK and the selection task would be terminated.

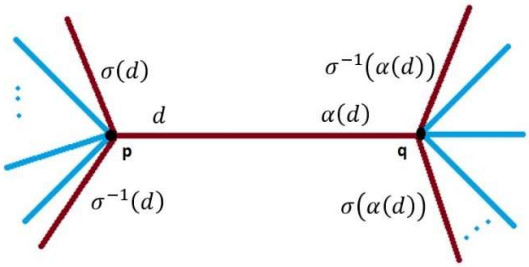


**Figure (4) The dependent darts set (DDS) around a dart pair.**

**Contracting the selected CKs simultaneously:** The next step after selecting the CKs is to contract them with a proper reduction function. Since in this study our goal is to do the labeling task, therefore, we simply consider the identity function as the reduction function. Because there is no contraction between the darts with the sd value of 1, choosing the identity reduction function leads in the same value be inherited by the next upcoming levels until the top of the pyramid. In other word, the vertex value remain the same through entire of the pyramid which assists us to have unique labeling for the connected components. Now, after the selection of the CKs and defining the reduction function we aim at contracting all the CKs at the same time. To this purpose, we use the extended canonical encoding representation which was explained in section 1.

**Selecting the removal kernels**: The resulted graph after contracting CKs usually has a higher complexity and consists of parallel edges and self-loops. These redundant information are non-useful to be used and we are interested in removing them in the produced graph. However, there are not such that we remove them at once. To avoid extra overhead for the complexity of our algorithm, we consider only the self-loops which have the degree less than three in the dual graph [24] and the parallel edges in the different DDSs as removal kernels.

**Simplication:** The simplication task reduces the complexity of the resulted graph and remove the redundant information. It consists of removing the parallel edges and the self-loops which are selected in the previous step of the algorithm. In Figure 4 the parallel edges are marked with blue cross signs and are removed in moving to the next level in pyramid. By doing the simplication task we are ready to go up to the next level of the pyramid. The four steps which explained above are iterated in each level until we reach to the top of the pyramid in logarithmic complexity. In fact, the height of the resulted pyramid in the worst case is equal to the log(image-dimension). This height even get reduced by the log(largest cc dimension) as a result of parallel selection of the CKs.

### 3.2. Labeling and moving down of the pyramid

So far, we have reached to the top of the pyramid. The main property of the top level is that there are no two adjacent vertices with the same color. It means each connected component step by step through the pyramid is merged and nally is represented as only one vertex, or lable, on the top.

**Assign unique labels to CCs:** Each single vertex on the top of the pyramid is the candidate for its correspondence CC in base level. Now, to assign a unique label and simultaneously to all single vertices on the top of the pyramid, we use the dart index stored in canonical array for each individual vertex. Since, through construction the combinatorial map the initialization task assign a unique index to each dart therefore the uniqueness of such assignment is guaranteed. In the Figure 3 left pyramid shows four different labels are assigned to the candidate vertices.

**Propagate labels to individual vertices**: Preserving the history of the passive CKs in the canonical array enables us to go down through the pyramid structure and step by step propagates the assigned label into each corresponding vertices of the CKs. This propagation is illustrated in the Figure 3 in the right pyramid. Finally, once all the vertices got labelled one can simply copy such a labeling to the corresponding binary image.

**4. Discussion, isolated vertices**

As it was mentioned in section 3, during selecting the CKs some isolated vertices are generated. Generally, two types of such isolated vertices exist. The first type are those which due to the topology of the shape must remain untouched. In fact, they represent the inclusion relationship with their adjacent regions. The second category belongs to those which are produced as a result of producing a random value of the f function. The number of these isolated vertices may differ in different runs of the algorithm. The MIES algorithm offers to connect these isolated vertices to the adjacent CK and then do the correction procedure when the resulted new CK tree has a length bigger than two edges. However, such a correction task requires sequential actions and may increase the general complexity of the PCCL.

In fact, here, the number of produced isolated vertices play a key role to demonstrate how much overhead complexity resulted through the selection of CKs. To this aim, we did the experimental tests over binary images with different size as are represented in table 2. The results show that for all tests the percentage of the isolated vertices is less than 3 and 6 percentage of the entire nodes for the low and high frequency images, respectively. Therefore, in our algorithm we do not the correction of selecting the CKs and instead the isolated vertices remain untouched. This means as long as they are not belonged to the first type they will be selected in the next iterations through reaching to the top of the pyramid. The similar trade of exists as well in dealing with the removal contraction. We have two choices between the sequentially select the removal kernels and accepting the extra overhead or remaining the complicated connections untouched and selecting the rest simple removal kernels in parallel. In such a case we again decide to remain them untouched because although there will be a lot of nonuseful information until the top the number of vertices will be decreased in a logarithmic factor which is more important.

**5. GPU implementation and the Experimental results**

In the proposed algorithm each level of the pyramid is implemented in a parallel way in GPUs and the only sequential steps are those which we move from one level to another. Moreover, each thread is dedicated to each contraction kernel.As it was mentioned already in section 2, the shared memory is always faster than the global memory, therefore, the local processes are mostly employed by the shared memory. In other words, the neighbor CKs and the initialization of the base level in pyramid are loaded in the shared memory and global memory, respectively.

In large images the number of CKs in the below levels of the pyramid is usually greater than the number of the threads in GPU. Therefore, in such a case, the threads must be waited until a busy SM finishes its task and is replaced by another contraction. However, on the upper levels, due to the logarithmic reduction of the vertices, all CKs can be assigned into each thread at once.

**Tabel 2- Percentage of produced isolated vertices in different binary images (1000 iterations)**

| Type of image | Image size | Percentage of isolated vertices |
|---|---|---|
| Low frequency | $256 \times 256$ | 2.6 |
| Low frequency | $512 \times 512$ | 2.4 |
| Low frequency | $1024 \times 1024$ | 2.3 |
| High frequency | $256 \times 256$ | 4.3 |
| High frequency | $512 \times 512$ | 4.8 |
| High frequency | $1024 \times 1024$ | 5.9 |

Our experiments were performed on the SUPERMICRO motherboard with Intel, E5-2697 v3, 128 GB DDR4 2133MHZ RAM and NVIDIA GeForce GTX 2080 TI.

**Tabel 3- Execution time for different size of images in Pyramid construction**

| Image size | Execution time(ms) |
|---|---|
| $128 \times 128$ | 0.2 |
| $256 \times 256$ | 0.8 |
| $512 \times 512$ | 3.3 |
| $1024 \times 1024$ | 8.9 |

**Tabel 4- Execution time for different size of images in label propagation**

| Image size | Execution time(ms) |
|---|---|
| $128 \times 128$ | 0.02 |
| $256 \times 256$ | 0.07 |
| $512 \times 512$ | 0.10 |
| $1024 \times 1024$ | 0.54 |

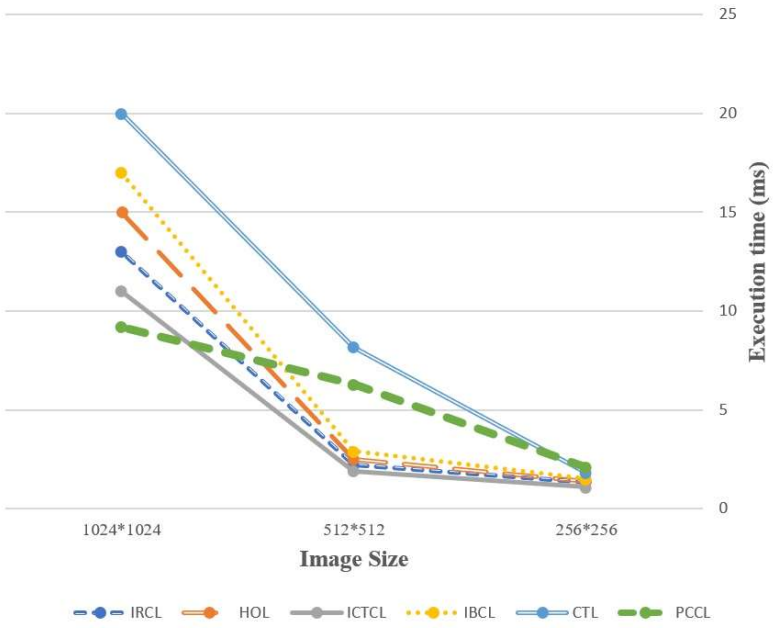


**Figure (5) The connected component labeling execution time over different image size**

In the experimental tests the image sets of the University of Southern California [25], ColumbiaUtrecht Reflectance and Texture Database [26] are used which the former includes different categories such as landscape, text image, finger-print and the latter consists of seven textural images. The result of experimental test on the above GPU and the comparison with the other state-of-the-arts in [4] shows that our proposed PCCL algorithm outperforms the others as the size of the tested image get larger and larger.

## 6. Conclusion

In this paper a new pyramidal connected component labeling was presented. The main advantage of proposed algorithm is its logarithmic complexity which much faster than common linear-time raster scan techniques labels all connected components in a few steps of traversing up and down through the pyramid. Moreover, using combinatorial pyramid structure preserves the topology of the connected components and enables us to extend the proposed algorithm for higher dimensions (nD). Finally, the parallel implementation of the algorithm on GPUs shows that the experimental results outperforms the state of the art in this field.

## References

[1] Q. Gu, T. Takaki, and I. Ishii, "A fast multi-object extraction algorithm based on cell-based connected components labeling," IEICE transactions on information and systems, vol. 95, no. 2, pp. 636-645, 2012.
[2] F. Zhao and Z.-y. Zhang, "Hardware acceleration based connected component labeling algorithm in real-time atr system," in Fifth International Conference on Machine Vision (ICMV 2012): Algorithms, Pattern Recognition, and Basic Technologies, vol. 8784, p. 87841S, International Society for Optics and Photonics, 2013.
[3] P. Sutheebanjard, "Decision tree for 3-d connected components labeling," in 2012 International Symposium on Information Technologies in Medicine and Education, vol. 2, pp. 709-713, IEEE, 2012.
[4] L. He, X. Ren, Q. Gao, X. Zhao, B. Yao, and Y. Chao, "The connected-component labeling problem: A review of state-of-the-art algorithms," Pattern Recognition, vol. 70, pp. 25-43, 2017.
[5] F. Chang, C.-J. Chen, and C.-J. Lu, \A linear-time component-labeling algorithm using contour tracing technique," computer vision and image understanding, vol. 93, no. 2, pp. 206-220, 2004.
[6] J. Martin Herrero, \Hybrid object labelling in digital images," Machine Vision and Applications, vol. 18, no. 1, pp. 1-15, 2007.
[7] L. He, Y. Chao, and K. Suzuki, "Two ecient label-equivalence-based connected-component labeling algorithms for 3-d binary images," IEEE Transactions on Image Processing, vol. 20, no. 8, pp. 2122-2134, 2011.
[8] K. Suzuki, I. Horiba, and N. Sugie, "Linear-time connected-component labeling based on sequential local operations," Computer Vision and Image Understanding, vol. 89, no. 1, pp. 1-23, 2003.

[9] U. H. Hernandez-Belmonte, V. Ayala-Ramirez, and R. E. Sanchez-Yanez, "A comparative review of two-pass connected component labeling algorithms," in Mexican International Conference on Articial Intelligence, pp. 452-462, Springer, 2011.
[10] W. G. Kropatsch, Y. Haxhimusa, and P. Lienhardt, "Hierarchies relating topology and geometry," in Cognitive Vision Systems, pp. 199-220, Springer, 2006.
[11] Y. Haxhimusa, R. Glantz, M. Saib, G. Langs, and W. G. Kropatsch, "Logarithmic tapering graph pyramid," in Joint Pattern Recognition Symposium, pp. 117-124, Springer, 2002.
[12] R. Gonzalez Diaz, W. G. Kropatsch, M. Cerman, and J. Lamar Leon, "Characterizing congurations of critical points through lbp extended abstract," Image-A: Applicable Mathematics in Image Engineering, 4 (7), 2015.
[13] L. Brun and W. Kropatsch, "Introduction to combinatorial pyramids," in Digital and image geometry, pp. 108-128, Springer, 2001.
[14] L. Brun and W. Kropatsch, "Combinatorial pyramids," in Proceedings 2003 International Conference on Image Processing (Cat. No. 03CH37429), vol. 2, pp. II-33,IEEE, 2003.
[15] T. Wang, G. Dai, B. Ni, D. Xu, and F. Siewe, "A distance measure between labeled combinatorial maps," Computer Vision and Image Understanding, vol. 116, no. 12, pp. 1168-1177, 2012.
[16] Torres F and Kropatsch WG (2014), "Canonical Encoding of the Combinatorial Pyramid", In Proceedings of the 19th Computer Vision Winter Workshop 2014. Křtiny, CZ, February, 2014. , pp. 118-125.
[17] M. J. Flynn, "Some Computer Organizations and Their Effectiveness," IEEE Transactions on Computers, vol. C-21, pp. 948-960, Sept. 1972.
[18] X. Mei and X. Chu, "Dissecting GPU Memory Hierarchy Through Microbench marking," IEEE Transactions on Parallel and Distributed Systems, vol. 28, pp. 72-86, Jan. 2017.
[19] D. Kirk, \NVIDIA cuda software and gpu parallel computing architecture," in Proceedings of the 6th international symposium on Memory management - ISMM '07, (Montreal, Quebec, Canada), pp. 103-104, ACM Press, 2007.
[20] N. Trevett, "Opencl overview," Vortrag bei SIGGRAPH Asia. Zugri am, vol. 26, p. 2016, Jan. 2012.
[21] S. Cook, CUDA programming: a developer's guide to parallel computing with GPUs. Amsterdam ; Boston: Elsevier, MK, 2013. OCLC: ocn773025100.
[22] D. Guide, "Cuda c programming guide," NVIDIA, July, 2013.
[23] W. G. Kropatsch, Y. Haxhimusa, Z. Pizlo, and G. Langs, "Vision pyramids that do not grow too high," Pattern Recognition Letters, vol. 26, pp. 319-337, Feb. 2005.
[24] Y. Haxhimusa, A. Ion, W. Kropatsch, and L. Brun, "Hierarchical image partitioning using combinatorial maps," in Joint Hungarian-Austrian Conference on Image Processing and Pattern Recognition, pp. 179-186, 2005.
[25] "SIPI Image Database."
[26] "Department of Computer Science, Columbia University."