A Hybrid Approach to Parallel Connected Component Labeling Using CUDA

Youngsung Soh, Hadi Ashraf, Yongsuk Hae, and Intaek Kim
Department of Information and Communication Engineering, Myongji University, Yongin, South Korea
Email: soh@mju.ac.kr, nothan111@gmail.com, wise_sunys@nate.com, kit@mju.ac.kr

***Abstract*—Connected component labeling (CCL) is a mandatory step in image segmentation where each object in an image is identified and uniquely labeled. Sequential CCL is a time-consuming operation and thus is often implemented within parallel processing framework to reduce execution time. Several parallel CCL methods have been proposed in the literature. Among them are NSZ label equivalence (NSZ-LE) method and modified 8 directional label selection (M8DLS) method. It was shown that M8DLS outperforms NSZ-LE and M8DLS is by far the best. In this paper we propose a new parallel CCL algorithm termed as HYBRID1 that hybridizes M8DLS and Kernel C method with some modification and show that it runs faster than M8DLS for various kinds of images.**

***Index Terms*—connected component labeling, CUDA, GPU, parallel**

I. Introduction

Connected component labeling (CCL) is a mandatory step in image segmentation where each object in an image is uniquely labeled. Various approaches were proposed in the field of CCL. Wu *et al.* [1] classified CCL methods into 3 groups. They are multi-pass, two-pass, and one-pass methods. Multi-pass method usually assumes some kind of local neighborhood to search for minimum label within that neighborhood and sometimes memorizes label equivalences. This is repeated over multiple iterations. In two-pass method, scanning, analysis, and labeling steps are usually executed. Scanning step assigns an initial label to each pixel and records equivalence among labels if necessary while searching for neighbors. Analysis step tries to find a final label for each label by resolving equivalence chains. Finally labeling step assigns a final label to each label. One-pass algorithm scans the image from left-top to right-bottom only once and gives a new label to unlabeled pixel. Then all the pixels connected to that pixel are searched and are assigned the same label. This is repeated until no more unlabeled pixels are left. The methods mostly used for CCL, regardless of the number of passes they adopt, were sequential [2], [3]. Sequential CCL can be used successfully in a single CPU system that processes a small number of channels of image stream. Though sequential CCL is a computationally expensive operation, increasing power of CPU enables sequential CCL run in real time for a few input channels. However, as the number of channels increases, say up to 32 or 64, it becomes almost impossible to process all input streams in real time with CPU alone. To overcome this difficulty it has been tried to implement CCL in parallel framework using graphics processing units (GPUs).

The usage of GPUs with compute unified device architecture (CUDA) developed by NVIDIA opened a new research field for the parallel implementation of CCL and many other data processing algorithms [4]. The implementation of parallel CCL using CUDA and GPUs drastically reduced computation time. Among many parallel CCL methods using CUDA proposed so far, M8DLS is known to perform best.

In this paper, we present a new hybridized method termed as HYBRID1for parallel CCL using CUDA. We first modify Kernel C method proposed by Hawick et al. [5] to increases efficiency and then combine with M8DLS. Modified Kernel C method will be termed as MKC. In HYBRID1, M8DLS and MKC will be executed at alternate iterations. M8DLS searches for minimum label of each object pixel for 8 directions (east, west, south, north, and 4 diagonal directions) until background pixel is encountered and changes the label of that pixel with minimum label found. If the object pixel has minimum label already, then that pixel is not processed. This process is repeated until all object pixels have minimum labels. In MKC, minimum label for each object pixel is searched in four directions (east, west, south and north) until the background pixel is hit and then the object pixel value is relabeled to the minimum value. If the object pixel has the minimum value already, then that pixel is not processed.

This paper is organized as follows. In Section II, related works for CCL are discussed. Section III presents the proposed method. Results are depicted in Section IV and Section V concludes the paper.

II. Related Works

Since the early 1980s many researchers have been working on fast CCL. Some of these researches were based on sequential processing [1], [2], [6] and some others within parallel framework [5], [7]. Suzuki *et al*. [2] proposed a sequential CCL, which is quite simple and suitable for implementation in hardware. But it is not fast enough since the execution time of their method is proportional to the number of pixels in connected

Manuscript received July 11, 2013; revised November 11, 2013.

doi: 10.12720/ijsps.1.2.130-135

components in an image, hence making it not suitable for images with high resolution.

Wu *et al*. [1] proposed the scan based array union-find algorithm which is basically an optimized two-pass algorithm. The performance of this algorithm in accessing the memory pattern in CCL has been significantly improved by almost 10 times than that of the contour tracing algorithm [6] and other previous methods [2]. They mentioned that the only drawback of this algorithm is the immense reduction in its efficiency when images with small resolution are processed.

Chang *et al*. [6] proposed the contour tracing algorithm which has a better computational speed than [1], [2] but takes more time in accessing the memory pattern.

Hawick *et al*. [5] proposed parallel version of the label equivalence algorithm using GPUs. Their algorithm consists of three basic steps: scanning, analysis and labeling. These steps are repeated in a loop until all of the connected components are identified and labeled correctly. Due to the parallel implementation of CCL this algorithm decreases the execution time quite effectively but it consumes more memory in using the reference array.

Kalentov *et al*. [7] proposed two methods. The first one is a simple row column unification method where a single thread is assigned for each row and column, scans each row and column in a predetermined direction, and changes the label of each pixel with the minimum label found so far along the scan direction. This process is repeated until all pixels have minimum labels. The second method is the NSZ-LE where a single thread is assigned to each pixel, searches for immediate 4 neighbors for minimum label, constructs the label equivalence chain, and does relabeling by resolving chains.

Soh *et al*. [8], [9] proposed 8 directional label equivalence (8DLS) method where the minimum label is searched in all 8 neighbors rather than 4 immediate neighbors of the focused pixel until background pixel is encountered. Each object pixel is assigned a single thread. This process is repeated until all object pixels have minimum labels.

Soh *et al*. [9] proposed M8DLS, an improved version of 8DLS, where the search space has been decreased by not processing the focused pixel if it already has minimum label.

III. The Proposed Method

The proposed method HYBRID1 is a multi-pass parallel CCL method and is the combination of M8DLS and MKC. First, we describe M8DLS and MKC in detail and then present HYBRID1 below.

A. M8DLS Method

The basic algorithm for M8DLS [9] is same as that of 8DLS [8]. The pseudo code for the M8DLS method is given in Algorithm 1.

Algorithm 1 The M8DLS method in pseudo code

```
for i=1 to n iterations do
  for each pixel p in an image do
    if  p is an object pixel
    then it becomes a focused pixel
          if (i>=2) and (label of p is not the smallest)
          then apply 8DLS
          end if
    end if
   end for
  if no label change for all the object pixels
  then exit
  end if
end for
```

In Algorithm 1, 8DLS is applied if some conditions are met. The pseudo code for the 8DLS algorithm is given in Algorithm 2 [8].

Algorithm 2 The 8DLS method in pseudo code

```
for j =1 to n iterations do
  for each pixel p in an image do
     if p is an object pixel
     then it becomes a focused pixel
       for i = 1 to 8 directions do
         search for minimum label until background pixel is hit
         and put it in min_i
       end for
     end if
     take minimum label m among min_i , 1<= i <= 8 and relabel
     the focused pixel with label m
  end for
  if no label change for all the object pixels
  then exit
  end if
end for
```

M8DLS modifies 8DLS such that after second iteration the label of the focused pixel is checked if it is the smallest so far. If it is not, 8DLS is applied. Otherwise, no further processing is performed, thus saving computation time. Checking for smallest is done as follows. Let initial label image array be LABEL. Then after assigning initial label sequentially to an image, LABEL (i) becomes i, for i = 0 to (total number of pixels in the image) - 1. For focused pixel p having a label j, we check if LABEL (j) is still j. If it is, we say that j is the smallest label so far. Otherwise it is not the smallest since it has already been changed, thus having a possibility of further change. If LABEL (j) is changed to something else later, then we apply 8DLS again to pixels having label j. Algorithm stops when there is no label change for all the object pixels in the image. The running appearance of M8DLS for initial label array is exactly same as that of 8DLS. However, many pixels will not be processed while producing the same results in later iterations. In Fig. 1 we show a running example of M8DLS. Fig. 1(a) is a sample input image that has been uniquely labeled according to the indices of the pixels in the image. Here white pixel is an object pixel and the shaded is a background pixel.

Assuming 8-connectivity, there are two objects. A single thread is assigned to each pixel and the algorithm is performed only on object pixels. Fig. 1(b), Fig. 1(c), and Fig. 1(d) show the results obtained after first, second, and third iterations respectively. To see how it works, let us consider the pixel with label 27 in Fig. 1(a). We search for 8 directions and find that the label 9(underlined) in 45˚ diagonal direction is the minimum. Thus the label 27 is changed to 9 as in Fig. 1(b) at the same location. For this example 3 iterations were enough to correctly label all the object pixels.

0	1	2	3	4	5	6	7	8	9
10	11	12	13	14	15	16	17	18	19
20	21	22	23	24	25	26	27	28	29
30	31	32	33	34	35	36	37	38	39
40	41	42	43	44	45	46	47	48	49
50	51	52	53	54	55	56	57	58	59

(a)

0	1	2	3	4	5	6	7	8	8
10	11	1	3	14	15	16	6	8	19
20	3	12	23	24	25	8	9	28	18
30	31	12	22	34	35	36	17	27	39
40	32	12	43	44	45	37	17	26	49
32	33	12	53	54	55	29	57	38	26

(b)

0	1	2	1	4	5	6	7	6	8
10	11	1	1	14	15	16	6	6	19
20	1	1	23	24	25	6	6	28	18
30	31	1	12	34	35	36	6	9	39
40	12	1	43	44	45	17	6	8	49
12	12	1	53	54	55	17	57	17	8

(c)

0	1	2	1	4	5	6	7	6	6
10	11	1	1	14	15	16	6	6	19
20	1	1	23	24	25	6	6	28	6
30	31	1	1	34	35	36	6	6	39
40	1	1	43	44	45	6	6	6	49
1	1	1	53	54	55	6	57	6	6

(d)

Figure 1. Running example of M8DLS method. (a) initial label, (b) after first iteration, (c) after second iteration, and (d) after third iteration

In Fig. 1(c), we put the underline for object pixels that pass the "smallest" check described above. For left and right objects, 9 out of 13 and 8 out of 16 pixels were not processed respectively, thus saving a great deal of computation time.

In M8DLS, "smallest" label check is performed after two iterations. This number was chosen empirically. We conducted many experiments for various numbers of iterations and for various kinds of test data and found that after two iterations most of object pixels already has smallest label they ought to have due to deep 8-directional search characteristic of our method.

B. MKC Method

The basic algorithm for MKC is that of Kernel C method proposed by Hawick *et al*. [5]. The pseudo code for the MKC method is given in Algorithm 3.

Algorithm 3 The MKC method in pseudo code

```
for j=1 to n iterations do
  for each pixel p in an image do
    if  p is an object pixel
    then it becomes a focused pixel
      if (i>=2) and (label of p is not the smallest)
        for i = 1 to 4 directions do
          search for minimum label until background pixel is hit
          and put it in min_i
        end for
        take minimum label m among min_i , 1<= i <= 4 and
         relabel the focused pixel with label m
      end if
    end if
  end for
  if no label change for all the object pixels
  then exit
  end if
end for
```

Kernel C method proposed by Hawick *et al*. [5] assigns a single thread to each row and column and searches for minimum label of each pixel, and relabel that pixel if necessary. The modification adopted in MKC is twofold. First, we assign a single thread to each object pixel instead of each row and column, thus increasing the degree of parallelism. Second, similarity criterion used in M8DLS is employed to reduce search space. Fig. 2 shows a running example of the MKC method. We used the same image sample as in Fig. 1(a). As can be seen in Fig. 2(d), 3 iterations are not enough to get correct labels with MKC alone. However, if it is mixed with M8DLS, the result gets better drastically.

0	1	2	3	4	5	6	7	8	9
10	11	12	13	14	15	16	17	18	19
20	21	22	23	24	25	26	27	28	29
30	31	32	33	34	35	36	37	38	39
40	41	42	43	44	45	46	47	48	49
50	51	52	53	54	55	56	57	58	59

(a)

0	1	2	3	4	5	6	7	8	8

10	11	12	3	14	15	16	17	8	19
20	21	12	23	24	25	26	17	28	29
30	31	12	32	34	35	36	17	37	39
40	41	12	43	44	45	46	17	38	49
50	41	12	53	54	55	46	57	38	58

(b)

0	1	2	3	4	5	6	7	8	8
10	11	3	3	14	15	16	8	8	19
20	12	12	23	24	25	17	17	28	29
30	31	12	12	34	35	36	17	17	39
40	12	12	43	44	45	17	17	17	49
12	12	12	53	54	55	46	57	37	38

(c)

0	1	2	3	4	5	6	7	8	8
10	11	3	3	14	15	16	8	8	19
20	12	3	23	24	25	17	8	28	29
30	31	3	12	34	35	36	8	17	39
40	12	3	43	44	45	17	8	17	49
12	12	3	53	54	55	17	57	17	37

(d)

Figure 2. Running example of MKC method. (a) Initial label, (b) After first iteration, (c**)** After second iteration, and (d) After third iteration

C. HYBRID 1 Method

As was mentioned earlier, HYBRID1 is the combination of M8DLS and MKC methods. Two methods are executed at alternate iterations until all object pixels are correctly labeled. The pseudo code for HYBRID1 is described in Algorithm 4.

Algorithm 4 The HYBRID1 method in pseudo code

```
for i=1 to n iterations do
  for each pixel p in an image do
    if  p is an object pixel
    then it becomes a focused pixel
        if i % 2 == 0
        then apply M8DLS
        else apply MKC
        end if
    end if
  end for
  if no label change for all the object pixels
  then exit
  end if
end for
```

In the algorithm, M8DLS and MKC are executed at even and odd iterations, respectively. Algorithm stops when there is no label change for all the object pixels. The working example of HYBRID1 is shown in Fig. 3.

0	1	2	3	4	5	6	7	8	9
10	11	12	13	14	15	16	17	18	19
20	21	22	23	24	25	26	27	28	29
30	31	32	33	34	35	36	37	38	39
40	41	42	43	44	45	46	47	48	49
50	51	52	53	54	55	56	57	58	59

(a)

0	1	2	3	4	5	6	7	8	8
10	11	1	3	14	15	16	6	8	19
20	3	12	23	24	25	8	9	28	18
30	31	12	22	34	35	36	17	27	39
40	32	12	43	44	45	37	17	26	49
32	33	12	53	54	55	29	57	38	26

(b)

0	1	2	3	4	5	6	7	8	8
10	11	1	1	14	15	16	6	6	19
20	3	1	23	24	25	8	6	28	18
30	31	1	12	34	35	36	6	17	39
40	12	1	43	44	45	17	6	17	49
12	12	1	53	54	55	29	57	26	26

(c)

0	1	2	1	4	5	6	7	6	6
10	11	1	1	14	15	16	6	6	19
20	1	1	23	24	25	6	6	28	6
30	31	1	1	34	35	36	6	6	39
40	1	1	43	44	45	6	6	6	49
1	1	1	53	54	55	6	57	6	6

(d)

Figure 3. Running example of HYBRID1 method. (a) initial label, (b) after first iteration (M8DLS), (c) after second iteration (MKC), and (d) after third iteration (M8DLS)

Here it can be seen that M8DLS and MKC are applied alternatively per iteration. Fig. 3(a) is the initial label image which is same as the one in Fig. 1(a). Fig. 3(b), 3(c) and 3(d) are labeled images after applying M8DLS, MKC and M8DLS, respectively.

IV. EXPERIMENTAL RESULTS

The system specification used for the experiment is as follows.

- **CPU**: Intel i7, 3.40 GHz
- **OS**: Windows 7

- **GPU**: NVIDIA Geforce GTX 550 Ti with 192 cores.

For the experiment, 8 types of images were used. All are of size 320 x 240 with 8 bits/pixel. They are,

- **B1, B2, B3, B4, and B5**: Images with occupancy ratio of 0.07, 0.17, 0.27, 0.36 and 0.46 respectively
- **Spiral**: Image with spiral pattern
- **Random1 and Random2**: Images that were generated programmatically using the random number generator and have the occupancy ratio of 0.1 and 0.5 respectively.

We compared the performance of NSZ-LE [7], M8DLS [9] and the proposed method (HYBRID1) in terms of execution time and the number iterations required. Table I shows the comparison results of execution time. We run each algorithm 100 times on each test image and take the average execution time. Irrespective of the methods tested, when we go from B1 to B5, execution time increases due to increasing occupancies. The same observation can be made when we go from Random1 to Random2. Among all the images, Spiral requires far more computation due to its complex shape.

For B1 through B5, Random1, and Random2, M8DLS performs around 3 times faster than NSZ-LE. For Spiral, M8DLS performs more than 10 times faster than NSZ-LE. HYBRID1 shows even better performance than M8DLS for all kinds of images with 2%~3% average speedup. This seemingly small improvement will count when image size gets larger and we have fewer GPU cores. This improvement of HYBRID1 over M8DLS comes from the fact that it searches only 4-connected neighbors at alternate iterations instead of 8-connected neighbors and this alternate searching is enough for various kinds of images to produce the same result obtained by M8DLS.

TABLE I. COMPARISON OF EXECUTION TIME (UNIT: SECOND)

Images	NSZ-LE	M8DLS	HYBRID1
B1	0.026	0.00793	0.00772
B2	0.028	0.00821	0.00813
B3	0.030	0.00851	0.00832
B4	0.031	0.00874	0.00858
B5	0.034	0.00913	0.00899
Spiral	0.135	0.0099	0.00887
Random1	0.0297	0.00803	0.00782
Random2	0.0304	0.00813	0.00796

Table II shows the number of iterations that were taken by each method for different test images. M8DLS and HYBRID1 take exactly the same number of iterations, whereas NSZ-LE consumes far more iterations. This is because NSZ-LE only looks at 4 immediate neighbors, whereas M8DLS considers all 8-connected neighbors and HYBRID1 processes 8-connected and 4-connected neighbors alternatively. Spiral shows extreme performance difference. NSZ-LE iterates around 22 times more than the proposed method and this causes far more computation time.

In conclusion, HYBRID1 shows better performance than M8DLS in execution time while maintaining the same number of iterations.

TABLE II. COMPARISON OF NUMBER OF ITERATIONS REQUIRED

Images	NSZ-LE	M8DLS	HYBRID1
B1	6	4	4
B2	41	6	6
B3	41	6	6
B4	41	6	6
B5	41	6	6
Spiral	344	16	16
Random1	6	6	6
Random2	6	6	6

V. DISCUSSION

CCL is an important step in image segmentation and is often implemented in parallel framework to reduce execution time. Soh *et al.* [9] compared NSZ-LE [7] with 8DLS [8] and M8DLS, and showed that both 8DLS and M8DLS outperformed NSZ-LE and M8DLS performs better than 8DLS. In this paper, we proposed a hybridized parallel CCL method HYBRID1 where we modify Kernel C method [5] and combine it with M8DLS. We showed that HYBRID1 performs better than M8DLS for various kinds of images.

ACKNOWLEDGEMENT

This work (Grants No. C0005448) was supported by Business for Cooperative R&D between Industry, Academy, and Research Institute funded by Korea Small and Medium Business Administration in 2012.

REFERENCES

[1] K. Wu, E. Otoo, and K. Suzuki, "Optimizing two-pass connected-component labeling algorithms," *Pattern Analysis & Applications* vol. 12, no. 2, pp. 117-135, 2009.
[2] K. Suzuki, I. Horiba, and N. Sugie, "Linear-time connected-component labeling based on sequential local operations," *Computer Vision and Image Understandingm,* vol. 89, no. 1, pp. 1-23, 2003.
[3] A. Rosenfeld and A. Kak, *Digital Picture Processing*, Orlando: Academic Press, 1982.
[4] R. Farber, *CUDA Application Design and Development*, Waltham: Elsevier, 2011.
[5] K. Hawick, A. Leist, and D. Playne, "Parallel graph component labeling with GPUs and CUDA," *Parallel Computing*, vol. 36, no. 12, 2010.

[6] F. Chang, C. Chen, and C. Lu, "A linear-time Component-labeling algorithm using contour tracing technique," *Computer Vision and Image Understanding*, vol. 93, no. 2, pp. 206-220, 2004.
[7] O. Kalentev, A. Rai, S. Kemnitz, and R. Schneider, "Connected component labeling on a 2D grid using CUDA," *J. Parallel Distributed Computing*, vol. 71, pp. 615-620, 2011.
[8] Y. Soh, H. Ashraf, Y. Hae, and I. Kim, "A simple and fast parallel connected component labeling using CUDA," in *Proc. International Conference on Computer Applications and Information Processing Technology*, 2013, pp. 61-64.
[9] Y. Soh, H. Ashraf, Y. Hae, and I. Kim, "Fast parallel connected component labeling algorithm in CUDA based on 8-directional label selection," *Journal of Parallel and Distributed Computing*, Elsevier, unpublished.

Hadi Ashraf was born in Lahore, Pakistan in Dec. 8 1988. He got BS in electrical engineering in 2010 from Govt. University in Lahore, Pakistan. He entered a Master course in information and communication engineering in Myongji University in 2012.He joined a software company in Pakistan in Aug. 2010 and worked as a software engineer till April 2012.His current interest of research includes object tracking, stereo vision and parallel algorithms for image processing.

Youngsung Soh was born in Seoul, Korea in Mar. 4 1956. He got BS in electrical engineering in 1978 from Seoul National University in Seoul, Korea. He obtained MS and PhD in computer science from the University of South Carolina in Columbia, South Carolina, USA in 1986 and 1989, respectively. He served in the Korean army from June 1980 to Sept. 1982. He worked in Systems Engineering Research Institute in Korea as a senior researcher from Sept. 1989 to Feb. 1991. He joined Myongji University in Korea from Mar. 1991 and is currently a full professor in the Dept. of Information and Communication Engineering. Some of his publications are:Y. Soh and Y. Hae," A New Depth Image Based Rendering with Local Texture Analysis", Advanced science letters, Vol. 9, 2012, pp.173-178.Y. Soh and J. Song," New Methods for 3D Measurement for PTZ Camera Control on Road Surface with Linear and Curved Slopes", Sensor letters, Vol. 10, No. 5-6, 2012, pp.1320-1325 His current interest of research includes object tracking, stereo vision, and parallel algorithms for image processing.Prof. Soh is a member of Korea Information Processing Society and Korea Signal Processing Systems Society.

Yongsuk Hae was born in Mokpo, Korea in August 8 1981. He got BS and MS in information and communication engineering from Myongji University in Yongin, Korea in 2009 and 2012, respectively. He entered a PhD course in information and communication engineering in Myongji University in Yongin, Korea in 2012. He served in the Korean army from August 2001 to October 2003. Since December 2008 he has been working in Nain information company in Korea as a researcher. Some of his publications are:Y. Soh and Y. Hae," A New Depth Image Based Rendering with Local Texture Analysis", Advanced science letters, Vol. 9, 2012, pp.173-178 Y. Soh, Y. Hae, and I. Kim," Spatio-temporal Gaussian Mixture Model for Background Modeling", Proceedings of 2012 IEEE International Symposium on Multimedia, 2012 pp.360-363. His current interest of research includes object tracking, stereo vision, and parallel algorithms for image processing.

Intaek Kim was born in Seoul, Korea in 1960. He received BS and MS in electronics engineering from Seoul National University in Seoul, Korea in 1980 and 1984 respectively. He obtained PhD in electrical engineering from Georgia Institute of Technology in Atlanta, Georgia, USA in 1992.He worked for Goldstar central research lab from 1993 to 1995 as a senior engineer and joined Myongji University from 1995. He is now a professor in the Dept. of Information and Communication Engineering. His recent publications deal with the area of face recognition, hypersepctral image and MR imaging.His research interest includes pattern recognition, image processing and smart grid area. Prof. Kim is a member of Korean Institute of Electronics Engineer.