Journal Homepage: -www.journalijar.com
INTERNATIONAL JOURNAL OF ADVANCED RESEARCH (IJAR)

Article DOI:10.21474/IJAR01/9580
DOI URL: http://dx.doi.org/10.21474/IJAR01/9580

RESEARCH ARTICLE

CONNECTED COMPONENT LABELING FOR BINARY IMAGES.

Isha Sehgal[1] and Prof. K. S. Venkatesh[2].
1. CS and IT department Banasthali Vidyapith Rajasthan-304022, India.
2. Department of Computer Vision Indian Institute of Technology Kanpur, India.

Manuscript Info

Manuscript History

Received: 16 June 2019
Final Accepted: 18 July 2019
Published: August 2019

Key words:-
CCA, CCL, Binary Images, Down-Sampling, Dimensions, Parallel.

Abstract

Connected Component Labeling is one of the very important aspects of Image Processing and Computer Vision. Connected Components refers to set of pixels having same value connected to each other in way that there exists a path between every two pixel of the connected component set.This project proposes 3 different algorithms related to different perspectives to solve Connected Component Labelling in Binary Image. These 3 perspectives are: Scaling, N-dimensions, Parallel.

Scaling: This algorithm reduces the resolution of image and then CCA is performed on the low resolution image. After this, label matrix is expanded to high resolution. Then accretion is done to resolve irregular labels. Basic idea behind this approach is that lesser the number of pixels fast is the execution of CCA/L algorithm.

N-dimensions: Algorithm can handle any n-dimensional image, so it works for 1, 2...n-dimensions. This important because we can have higher dimensional images like 20-D or more in near future.

Parallel: Image will be processed simultaneously on separate processors and results will be merged and then sorted to produce a single label matrix corresponding to original image. This consumes less memory and less execution time. It provides results fast for images of varying size and densities.

All the approaches seem to give good performance. They produce accurate results and are efficient in terms of memory consumption and speed.

Introduction:-

Connected Component Labelling is one of the very important aspects in computer vision and image processing field. Technically, all the image objects are formed with components that in turn are formed of connected pixels. So Connected Component Labelling is one of the very important aspects of Image Processing and Computer Vision.

Input to this procedure is a binary image and output is a symbolic image i.e. labeled image where all the components are labeled. All connected pixels with same value are assigned an identification label. Connected Components refers to set of pixels having same value and are connected to each other in way that there exists a path between every two pixel of the connected component set.

Corresponding Author:- Isha Sehgal.
Address:-CS and IT department Banasthali Vidyapith Rajasthan-304022, India.

Connected components: Two pixels p(x, y), q(s, t) are said to be connected if there exists a path between them. Here path refers to sequence of distinct pixels with co-ordinates(x_1, y_1) ……. (x_n, y_n) where $(x_1, y_1)=(x, y)$ and $(x_n, y_n)=(s, t)$ and n=length of path.

Classic Approach of CCL:-
Every image is composed of pixels. For binary image, the value of pixels can be either 1 or 0. If the Image is Gray-scale image or RGB image, it is first converted into binary image using one of the several methods available. Most commonly used is thresholding method. In Thresholding method, a threshold level is set say 100, then all the pixel values greater than 100 are replaced with 1 and pixels with value less than 100 are replaced with 0 or vice-versa.

The Basic approach for Connected Component Analysis/Labelling for every image of size m x n is as follows:-
1. It scans the image pixel by pixel and updates the Label matrix by assigning a label to current pixel being scanned. Scanning checks each and every pixel.
2. For every pixel being scanned, it first checks if it is assigned any label or not. If not, then it proceeds to next step.
3. If the pixel has no label assigned, it then checks for the immediate 4-neighbours neighbours as per 8-connectivity rule.

In fig1a:- Immediate 4-neighbours of pixel b are:

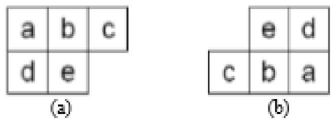


Fig. 1: The labeling mask of eight connected components. (a) Forward raster scan; (b) Backward raster scan

1. connectivity rule. Few algorithms also use this rule.
2. If pixel has no connected-labelled neighbours ,then current pixel is assigned a new label
3. Else if it has connected neighbours with labels assigned, then minimum of all labels is assigned to current pixel.

After all the pixels are labelled, Label matrix goes through second scan to ensure that all the pixels has exactly once label assigned and all the irregularities among labels is resolved. This will provide accurate number of connected components by assigning a different label for each connected components.

All algorithms assume pixel with value 0 to be background pixel and pixel with value 1 to be foreground pixel. Our proposed algorithms have used 8-connectivity rule. Flood fill algorithm uses immediate 4 neighbours as per 8-connectivity rule as in figure Fig(1a) and Label resolution algorithm uses immediate 8 neighbours as per 8-connectivity rule as in figure Fig(1c).

Related Work:-
Since early 1980’s fast CCL has always been a research area of interest. Various algorithms have been proposed so far; dealing with either sequential processing or parallel computing. The first algorithm for Connected Component Labelling was proposed in 1966 by Rosenfeld and Pfaltz [3]. They proposed a method which performs two passes of scan over a binary image. Each pixel is scanned only once in the first pass. If the pixel encountered is having value 1, then its immediate 4 neighbouring pixels (left, upper left, top, and upper right) are scanned (same as in Fig:1a). If none of the neighbours are labelled, then the current pixel is assigned a new label else minimum of labels assigned to its neighbouring pixels. For this purpose, a pair of arrays is generated, one array contains all the current labels and the other array consists of minimal equivalent labels of those current labels. Then during the second pass, label replacements are made.

Haralick et al. [8] proposed a method in 1981 to eliminate the extra storage required for the pair of arrays required in the preceding method [3]. Initially, each black pixel is assigned a unique label. Then this labelled image is processed iteratively in two directions. In the first pass (conducted from the top down), each of the labelled pixel is reassigned the smallest label among its immediate four neighbouring pixels. The second pass is similar to the first, except that it is conducted from the bottom up. This process is repeated until no more labels change.

The method proposed by Lumia, Shapiro, and Zuniga [15] proposed a method in 1983 combining the above two methods [3, 8]. During first pass, foreground pixels are labelled as in method [3], and then at the end of scanning of each line, labels on this line are modified to their equivalent minimal labels. The second pass is same as first with the exception that it is initiated from bottom instead of top. After both the passes image is labelled accurately.

Jung-Me Park, Carl G. Looney, and Hui-Chuan Chen [9] proposed an optimization of sequential processing algorithm [3] in 2000 which reduces the size of equivalent matrix. This method involves divide and conquer technique. An image is divided into sub-images, each of which is processed independently. Each sub-image undergoes basic processing steps of scanning and labelling through label-equivalence resolution. And then they are merged and resolved further to produce an accurate labelled image. Though this algorithm was faster than Rosenfeld and Pfaltz algorithm, however, it compromises the accuracy. Images (Fig 1d) of kind shown in below figure are not processed accurately. For such images, two connected components do not belong to same connected component set.

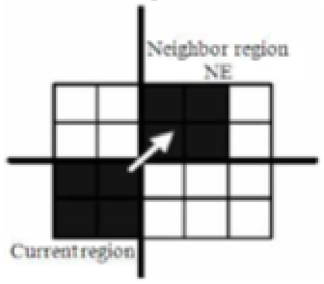


Fig 1d:- Image cannot be processed by algorithm [9]

Yapa and Harada [12] proposed a method in 2008 which makes use of both forward mask and backward mask. This algorithm is same as sequential processing [3] with the difference that it uses backward mask along with forward mask. In first process, each pixel is scanned and if it is having value 1, then its 4-neighbours as in figure Fig (1a) are scanned and minimum of the label is assigned if they are labelled else a new label is assigned. Then second process is carried out which makes use of mask in Fig (1b) and assigns a label. This alternative implementation of forward mask and backward mask is carried out until no label changes.

Suzuki et al. [2] proposed a sequential two-scanning CCL algorithm, which is simple. This algorithm adds a 1-dimensional table called label-connection table to store the labels. Labels are updated on image as well as Label-connection table. It reduces the number of scan and comparatively improves the speed as well. This algorithm is easy to implement and is simple and easy. Though it is not complex at all, however, it is not fast for complex or high density images. Its execution time is proportional to the number of pixels in connected components in an image.

Wu et al. [13] in 2005, proposed an optimization of algorithm proposed by Suzuki [2]. This algorithm uses the neighbour scan theory. It makes use of decision tree to check if the neighbours are related to each other and will assign an identification label to connected components.

Akmal et al. [11] in 2010, proposed the algorithm using a variant of flood-fill algorithm. This algorithm maintains two 2-dimensional arrays: Head table (H) and Component table (C). This algorithm uses the forward raster scan mask. It scans the unlabeled pixel and as soon as it finds the first unlabeled pixel, it is treated as head pixel and is stored in Head table (H) and is labelled a provisional label. It then scans its neighbours and stores it in Component table (C). For each pixel, it labels and scans its neighbours and stores it in Component table C. The process continues till whole image is labelled. After finishing with Component table, it returns to Head table and will proceed further to find another unlabelled pixel. Though it gives good result, yet there is a chance to improve the speed.

Fiorio and Gustedt [4] proposed a union-find algorithm. This method also consists of two passes. In the first pass, a tree is used to represent each set of equivalent label. In the second pass, a relabelling procedure is performed. In this technique, two trees are merged into a single tree if a node in one tree is connected to a node in other tree.

Chang et al. [6] proposed the contour tracing algorithm which offers a better performance in terms of computational speed than [1, 2], however, it consumes more amount of time while accessing the memory pattern of pixels.

Wu et al. [1] proposed the scan based array Union-Find (SAUF) algorithm which optimizes two-pass algorithm. This algorithm is almost 10 times faster than the contour tracing algorithm [6] and other previous methods [2]. However, it is not accurate and efficient for small resolution images.

Hawick et al. [7] proposed a parallel label-equivalence algorithm involving GPUs. The algorithm includes three fundamental steps: scanning, analysis and labelling. A loop involving these three steps executes iteratively until the image is completely labelled. With the parallel implementation of this algorithm, its execution time has been decreased effectively; however, it consumes more amount of memory because of using reference array.

Kalentov et al. [5] proposed two methods. In the first method, a single thread is assigned to each row and column, which scans each row and column in a predetermined direction, and then assigns the minimum label found so far along the scan direction to the pixel. This process is repeated until all the pixels have the minimum label assigned. The second method is the NSZ-LE where a single thread is assigned to each pixel, which searches for immediate 4 neighbours for minimum label, and constructs the label equivalence chain, resolve it and assign it to the pixel. This method seems to give good performance among CUDA-based parallel CCL algorithms.

Youngsung et al. [14] proposed two methods: 8DLS and M8DLS. Both involve parallel approach and were CUDA based. The first algorithm 8DLS labels each pixel by its sequential index value in image. Then it scans each pixel and if pixel value is 1, then it checks all the 8 neighbours and will assign minimum of labels and continues scanning in the direction of connected neighbouring pixels. And the second algorithm is M8DLS which is advanced variant of 8DLS. The only difference is that for M8DLS, after the second iteration, it checks the label of focussed pixel i.e. current pixel being scanned. If it is the smallest label so far, then process is stopped, else, 8DLS is applied.

Methodology:-

Before describing all the three approaches in detail, the two fundamental blocks of the proposed algorithms i.e. Flood-fill approach which is used to label the image and Label-Resolution which is used to resolve label issues are described below:

Flood-fill algorithm:-

This algorithm will check for an unlabelled pixel in the image. Once it founds the unlabelled pixel, it will assign it a new label and will check the neighbours and propagate the same label to all the connected pixels and will continue until any such pixel is encountered which is not connected to the current pixel. The process will continue until all the pixels in the image are labelled accurately.

This algorithm needs a stack for each label which will store all the connected neighbouring pixels that need to be processed and once it is processed, it will be deleted from the stack.

Labelling Resolution:-

This algorithm will scan the entire matrix for a labelled pixel and will then check its value with the value of its immediate 8-neighbours as in fig (1c) thereby assigning the pixel with minimum equivalent label and propagate the label value for all the connected pixels with the same old label value to the entire Image matrix to avoid redundant checking of same pixel and its connected pixels. This will update the label value of all the connected pixels at one go in the entire image.

The above two methods i.e. flood fill and label resolving are the two fundamental methods that we have used in the proposed algorithms concerning three dimensions of our research: Scaling, N-dimensions and Parallelism. We have also tried to combine these three parameters in our research.

All the three proposed approaches and a combination of them will be now discussed in details:-

1. Scaling
2. N-Dimensional
3. Parallel

Scaling:-
The main motto of this algorithm is to reduce the number pixels to be processed as less number of pixels means faster the execution of algorithm.

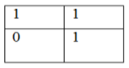
The proposed algorithm down samples the image by a certain scaling factor input by a user. By down sampling the size of image is reduced thus reducing the number of pixels to be scanned and processed.

If the input scaling factor is 1 and image size is m x n, then size reduction is 2^1 i.e. the size of down-sampled is m/2 x n/2 .This means 4 pixels are down sampled to 1 pixel. Similarly, if the input factor is 2, then the size reduction is 2^2 i.e. 4 which means the size of down sampled image is m/4 x n/4. In this case, 16 pixels are down sampled to 1 pixel.

The algorithm is as follows:
Step 1:-
Down sampling In this step, a scaling factor is taken as input and then image is down-sampled as per the scaling factor as mentioned above.

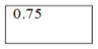
Down-sampling is done by taking the average of pixel values to be down-sampled. This can be explained by taking an example: if scaling factor is 1, then down-sampling of 4 pixels into 1 pixel is performed in the manner shown below where:

Fig 2:- pixels to be down-sampled
Fig 3:-Resulted down-sampled pixel in response to Fig 2.



1	1
0	1

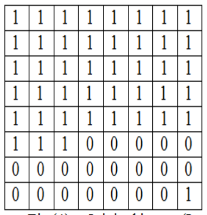
Fig (2): 4 pixels to be down sampled



0.75

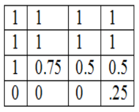
Fig (3): Down sampled pixel by taking average of all the 4 pixels.

Let's take an example which will be considered while explaining the entire algorithm of scaling. We are taking the scaling factor of 1 for the explanation.
Scaling factor=1.
Fig 4:- Original Image (Img) of size 8 x 8 that needs to be processed for calculating the connected components.
Fig 5:- Resulted down-sampled image (Image2) whose size is reduced to half of the size of Img (Original Image) i.e. the size of down-sampled image is 4 x 4.



1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	1

Fig (4) :Original image (Img)



1	1	1	1
1	1	1	1
1	0.75	0.5	0.5
0	0	0	.25

Fig(5):Image2(low resolution image by taking an average of 2x2 pixels)

Step 2:-
CCA and Label Resolution in down-sampled Image:-
This step will perform CCA through flood-fill algorithm mentioned earlier in the section followed by label resolution through label-resolution algorithm (mentioned above in the same section). Both the procedure will be performed on the down-sampled image obtained as a result from step 1. This step will generate a labelled down-sampled image.

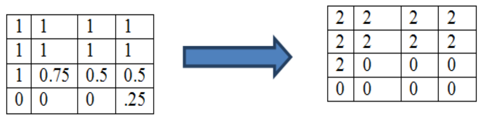
This will require a Label matrix initially with all zero labels with the same size as of Image2 (down-sampled image from step 1, fig 5). Process carried out is as follows:

Flood fill Algorithm:-
This will check for an unlabelled pixel with value "1" and will assign it a new label and then by implementing the flood-fill algorithm, it will propagate the label to all the connected pixels in the entire image at one go.

Label Resolving:-
This will resolve all the label issues by scanning each pixel and comparing its label with the label of immediate 8-neighbouring pixels and will assign the minimum labels and will propagate the resolved label to all the connected pixels in the image. This step can be understood by the images shown below:-

Where the down-sampled image (from previous step Fig 5) is processed and,

Fig 6:- Resulting labelled down-sampled image after applying flood-fill and label-resolving algorithm.



1	1	1	1
1	1	1	1
1	0.75	0.5	0.5
0	0	0	.25

2	2	2	2
2	2	2	2
2	0	0	0
0	0	0	0

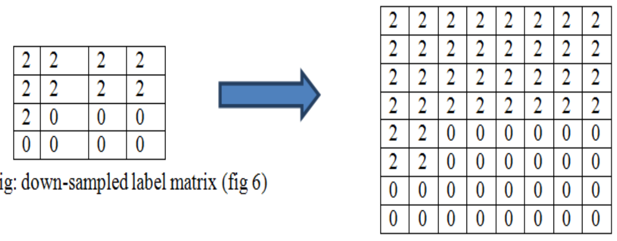
Fig: Down-sampled image from previous step (Fig 5)

Fig 6: Resulted Label matrix for fig 5 (down sampled image)

Here labelling starts from '2' to avoid confusion as 1 and 0 are the values of pixels.

Step 3:-
Expanding the down-sampled labelled matrix (Image2):-
This phase of the algorithm expands the labelled down-sampled image matrix obtained from previous step (Fig 6, in our case) to the original size m x n i.e. the size of image by following the reverse procedure as in Step 1 for down-sampling.

Algo:-
```
m2 x n2=size of down-sampled labelled matrix;
sl1=2^scaling factor;
for ir=1 to m2 iterations do
    for jr=1to n2 iterations do
        in=((ir*sl1)-(sl1-1));
        jn=((jr*sl1)-(sl1-1));
        for u=1 to sl1 iterations do
            for v=1 to sl1 iterations do
                Lab(in+u-1,jn+v-1)=L1(ir,jr);
            end for
        end for
    end for
end for
L1=down-sampled labelled matrix.
Lab=Labelled matrix of original size m x n.
```
This algorithm will replicate the label value of each pixel to the corresponding pixels that were down-sampled to form a single pixel. This step will return an expanded labelled matrix corresponding to original image size.
Considering our example, this step can be understood by following diagram:



2	2	2	2
2	2	2	2
2	0	0	0
0	0	0	0

Fig: down-sampled label matrix (fig 6)

2	2	2	2	2	2	2	2
2	2	2	2	2	2	2	2
2	2	2	2	2	2	2	2
2	2	2	2	2	2	2	2
2	2	0	0	0	0	0	0
2	2	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

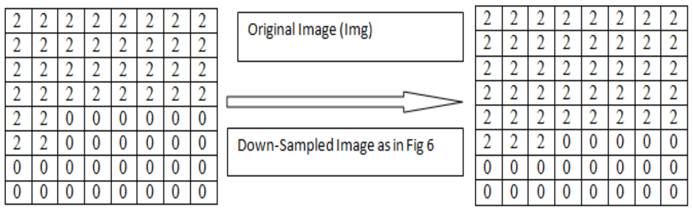
Fig 7: expanded label matrix

Step 4:- Dealing with the boundaries of labelled blocks/pixels

This step will process the expanded labelled matrix obtained from previous step (Fig 7). It will consider both expanded labelled matrix as in Fig 7 and down-sampled labelled matrix as in Fig 6, simultaneously.
In down-sampled labelled matrix, this step will check for labelled pixel and will check it neighbouring pixels(in north, south, east and west direction) which actually corresponds to multiple pixels in each direction in expanded labelled matrix as multiple pixels are down-sampled to single pixel(in our case we have 4 pixels being down-sampled to one). Then after checking, it will assign the label to the neighbouring connected pixels in the expanded label matrix.

```
Algo:
s1=down-sampled image;
[m2 x n2]=size (s1);
Img=original image
for i=1:m2
 for j=1:n2
   if s1 (i, j) =1
      check all the 4 neighbours: east, west, north and south.
   end if
 end for
end for
```

Following diagram shows the working of this step:



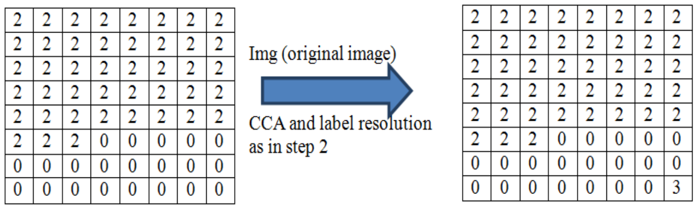
2	2	2	2	2	2	2	2
2	2	2	2	2	2	2	2
2	2	2	2	2	2	2	2
2	2	2	2	2	2	2	2
2	2	0	0	0	0	0	0
2	2	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

2	2	2	2	2	2	2	2
2	2	2	2	2	2	2	2
2	2	2	2	2	2	2	2
2	2	2	2	2	2	2	2
2	2	2	2	2	2	2	2
2	2	2	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Fig 7: Expanded Label Matrix

Fig 8: Expanded labelled matrix after checking boundaries.

Step 5: CCA and Label Resolution in expanded labelled matrix
This step is same as step 2 except that it is carried out in expanded labelled matrix obtained from previous step (fig 8). This step will return final accurate labelled matrix for original image (Img).
Following diagram demonstrates this step:



2	2	2	2	2	2	2	2
2	2	2	2	2	2	2	2
2	2	2	2	2	2	2	2
2	2	2	2	2	2	2	2
2	2	2	2	2	2	2	2
2	2	2	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

2	2	2	2	2	2	2	2
2	2	2	2	2	2	2	2
2	2	2	2	2	2	2	2
2	2	2	2	2	2	2	2
2	2	2	2	2	2	2	2
2	2	2	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	3

Fig 9: Final Label Matrix

Step 6:-
Calculate the number of connected components:-
This step will look for different labels in the labelled matrix obtained after step 5 and will count the number of connected components and store it in a separate array (R).

For Fig 9, Number of Connected Components=2 (2 and 3 labels):-
1. So algorithm can be summarized as:
2. Img=Down-Sampling (Image);
3. Label1=Flood-fill (Img);
4. L1=Label-Resolution (Label1);
5. Lexp1= Expand (L1);
6. Lbound1= Boundaries dealing (Lexp1);
7. FL1= Flood-Fill and Label-Resolution (Lbound1);
8. Compute components (FL1);

Equation of the scaling algorithm can be formulated as:

FinalLabelledImage=∑_(i=1 ,Im=Img)^k▒〖Img=func_scale(Im) 〗+ ∑_█(1≤ i ≤ m1@1≤j≤n1)▒〖Flood-fill+Label-resolution(i,j) 〗+ Func_expand+ Func_boundaries+ ∑_█(1≤ i ≤ m@1≤j≤n)▒〖Flood-fill+Label-Resolution(i,j) 〗
Where $func_{scale}$ is the scaling function, k is the scaling factor, $Func_{expand}$ is the function to expand the down-sampled matrix, and $Func_{boundaries}$ deals with boundaries of labelled pixels.

N-Dimensions:-
This phase of our research deals with N-dimensional image where N can be any positive integers, N=1, 2, 3, 4… N. The need of this algorithm was felt because nowadays higher dimensional images up to 7-D or more are being used. And it is clear that in near future we can have 20-D or more than that. So we need such algorithm that can be used today as well as in future for image processing applications. After understanding the need of this algorithm, the next key point to be explained is the phenomenon about N-dimensional that forms the base of this algorithm. This key point is: How the arrays/matrices are arranged for N-dimensional Images/Arrays/Matrices. N-dimensional array can be thought of size M1 x M2 x M3 ……………. x Mn. Now, let us see for various values of dimensions other than 2 (we are avoiding 2 because it is the ideal dimension value and all other dimensions depend on 2-D):

1-D:-
A 1D array can be thought of as 2-D array with the value of 1-dimension equal to 1. For example, [1, 1, 1, 1, 1, 0, 0, 0, 0] is an array with size 1 x 9.

3-D:-
3D array can be thought of as stack of 2D arrays. For example, an array of size 3 x 2 x 4 means that 4 slices/layers of array of size 3 x 2 are stacked together or we can say that array of size 3 x 2 is stored in another array of size 4 x 1. Here the 4th dimension will be automatically is set to 1.

4-D:-
4-D array is divided into sub-arrays till we receive 3D arrays each which are processed.
N-D array is subdivided until we get N-1 D array.

This algorithm stores the dimensions in an array, for example for an array of size 3 x 2 x 4, the array will be [3, 2, 4, The process can be described as follows:-

1. Dim=Identify the dimensions.
2. If the dimension is an odd number, like 3D, 5D, etc, then add a last dimension with value one to make it an even number. Thus for 3D, new dimension is 4d. This can be explained by following example; A is a 3D array of size 3x2x4, then its new size will be 3x2x4x1.
3. Now store the size into another array S.
4. For same example A in 2nd step, S= [3, 2, 4, 1].
5. Determine the length of S. In our example, length(S) =4.
6. Now initiating from second last element, perform iteration in reverse direction till it reaches first two dimensions, and perform the flood fill for each iteration.
7. Perform Label-resolution algorithm to solve label issues.

Algo:-

```
i1 and j1 refers to dimensions.
cnt=2;
s=array storing dimensions
    for i1=3:2:i-1
      j1=i1+1;
      for ilel=1 to s(i1) iterations do
        for jlel=1 to s(j1) iterations do
            [Label(:,:,ilel),cnt1]=floodfill(Image,cnt,ilel,Label(:,:,ilel));
        cnt=cnt1+1;
        end for
      end for
    end for
```

FinalLabel=Label Resolution(Image,Label);

Flood fill algorithm basically applies to 2-D images so for 1d and 2d, 3rd and 4th dimension will be set to 1.

Equation for this algorithm can be formulated as:

LabelledImage=∑_(i_n=1)^(M_N)▒〖⋯⋯⋯∑_(i_(2=1))^(M_2)▒∑_(i_(1=1))^(M_1)▒〖Flood-fill+Label Resolution〗〗

Parallel:-

Parallel approach will use multiple processors simultaneously to calculate CCA in an image fast. This approach basically divides an image into 2 or 4 sub-images for 2 and 4 processors respectively and then sub-images are assigned to processors with 1:1 ratio where each processor independently process its sub-image simultaneously. This reduces the overall load on the processors. Then the results from all the processors clubbed together to form a single labelled image matrix. For an image of size m x n, it can be divided into sub-images either row-wise resulting into two sub-images each of size m/2 x n or column-wise which results into two sub-images each of size m x n/2. Here we are dealing with column-wise distribution of image among 2 processors.

This algorithm is designed to take the advantage of multiple processors and compute CCA fast as each processor works simultaneously.

The algorithm is as follows:-

Img=Image of size m x n.

1. Pool of processors open(P1 and P2)
2. Divide image (Img) into 2 column-wise sub-images Img1 and Img2 of size m x n/2.
3. Assign sub-images to processor 1:1 which means one sub-image for one processor (P1←Img1, P2←Img2).
4. Now each processor will carry out following procedure on its assigned sub-image:Label=Flood-fill (Img); Final Label=Label-Resolution (Label);
5. R_FinalLabel=Merge the resulted Final Labels from all the processors.
6. Label1=Label-Resolution(R_FinalLabel);
7. Compute the components from Label1.
8. Pool of Processors close

This can be understood using same example (Fig 4):-

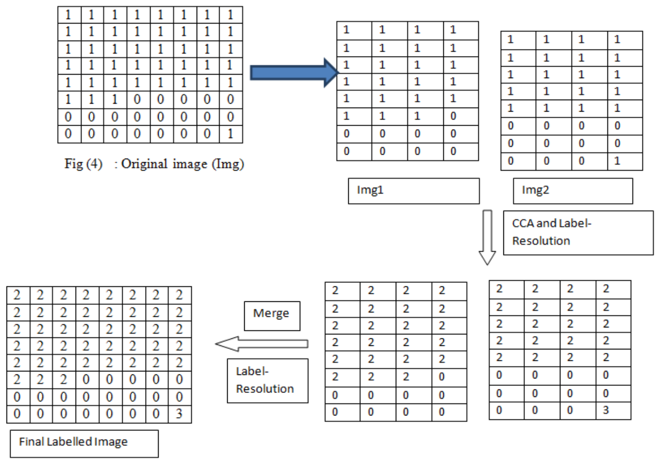


Fig 10:-Complete Parallel Method depicting CCA

Equation can be formulated as:-

$$LabelledImage = \sum_{i=1}^{m}\sum_{j=1}^{\frac{n}{2}} Flood-fill + Label - Resolution(i,j)$$
$$+ \sum_{i=1}^{m}\sum_{j=\frac{n}{2}+1}^{n} Flood-fill + Label - Resolution(i,j)$$
……………….............. (3)

Experimental Results:-

All the algorithms were implemented on a PC-based workstation with following configurations: Intel Core i3 processor, 2.30 GHz CPU, 2 GB RAM, and Windows 8 OS. We used a single core for scaling and dimensional approaches and 2 cores for implementations involving parallel approach which are parallel and a combination of scaling and parallel approach. All the approaches were implemented using MATLAB 2010.

We have tested the algorithms on 10 images out of which 8 images of varying size ranging from 256x256 to 3000x3000. Some of the images were downloaded from Volume 3 (Miscellaneous) of the image data base of University of South California (USC, database link: http://sipi.usc.edu/services/ database/ database.cgi?volume=misc) and 2 images of 3 dimensional are downloaded from link: http://www.ece.ncsu. edu/ imaging/Archives/Image/Database/Medical/. Some are anonymous downloads through internet. The result for these images in response to N-dimensional approach is shown in Table 2. It shows time (sec) and CC as number of connected components.

Table 1:- shows the results for without scaling or parallel, scaling, and parallel. Table shows the time (in sec) and CC (connected components).

Images	Size	CC	CCL Without scaling and/or parallel(sec)	Scaling (sec/scaling level)	Parallel (sec/no of processors)
Image1: 5.1.13	256x256	2	16.7176	1.4738/3	1.1116/2
Image2: 5.1.14	256x256	48	16.5464	1.1614/3	1.2082/2
Image3: 4.0.7	512 x 512	128	256.7240	2.4844/2	1.7446/2
Image4: 7.1.04	512x512	44	289.3965	2.0840/3	1.7014/2
Image5: 7.2.04	1024x1024	1923	22.5626	4.2737/3	3.2086/2
Image6: 1102 Nasa	1024x1024	5857	51.3355	4.9435/5	2.8238/2
Image7: testpat.1k	1024x1024	1483	3236.5	8.4340/4	4.2095/2
Image8: Nasa Apollo	3000x3000	4636	13000(more than 3 hrs)	27.9165/3	23.5423/2

Graph for Table1:-

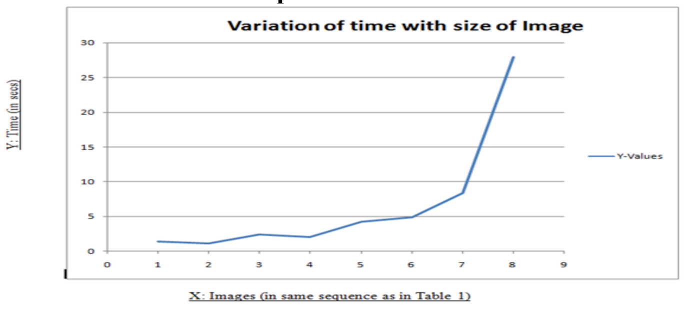


Fig 1 : Graph representing Table 1 data (Image and time in same sequence). As Image size increases, time also increases.

Table 2:- shows the execution time for Image 1102 NASA for various scaling level from 1 to 10.

Image	Scaling Level	Time(s)
1102 Nasa	1	18.0795
	2	7.8600
	3	6.3137
	4	5.9589
	5	4.9435
	6	17.7807
	7	52.4859
	8	52.7351
	9	52.2130
	10	51.5960

Graph corresponding to Table 2 is:-

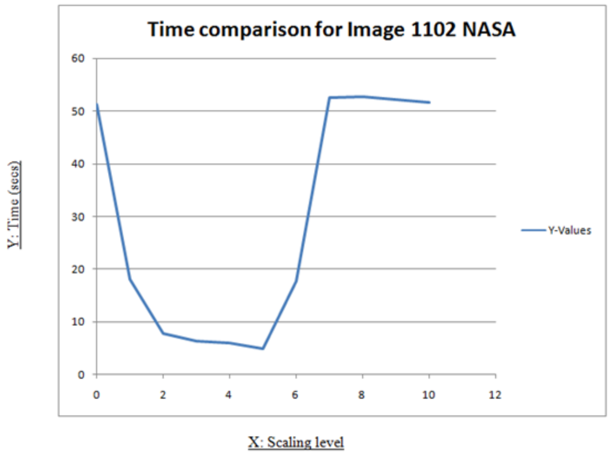


Fig 2 : Graph showing variation in execution time for various scaling levels for Image 1102 NASA

Table 3:-show results for N-dimensional approach. It shows time (sec) and CC as Connected Components.

Images	Size	Cc	Time(sec) (dimension is 3)
melan1.thumb	64x64x5	1	0.9774
lymphoma. thumb	64x64x5	2	0.3907

During testing we encountered that for higher resolution images large scaling gives better results in comparison 1 or 2 scaling factor. However, with very large scaling factor, overhead also increases.

The implementation of above algorithms in MATLAB gives good results; however if it is implemented in C language then timing results are improved dramatically. The execution gets faster almost 10 times and in some case cases more than that. So for real-time applications this should be implemented using C language.

Conclusions:-

We have presented three new approaches and their algorithms to deal with the aspect of CCL/A. These approaches are: Scaling, N-Dimensions, Parallel.

Scaling reduces the size of image by down-sampling and processes it to compute CCL fast. Even with down-sampling, it does not compromise accuracy. It gives fully accurate results. Second approach i.e. N-Dimensions deals with N-Dimensional images. The Third approach, Parallel again tries to achieve fast execution of CCA/L. This divides the task of computing connected components among multiple processors by dividing an image into sub-images and then assigning them to multiple processors, where each handles one sub-image. This approach takes the advantage of multiple processors and fastens the computation of connected components. Experiment results shows that all the 3 approaches give good performance in terms of accuracy, speed and execution time.

Future Enhancements:-

We have tried with three different approaches (Scaling, N-Dimensions, and Parallel) which turn to be very effective and computes connected components fast. We may still try to improve speed further as there is always a scope of improvement. The above three approaches can be combined in various orders and tested to check for better results. We may further modify the algorithm either for software or hardware programming.

Acknowledgment:-

The authors would like to thank the anonymous referees for their valuable comments that improved this paper greatly. A vote of thank to Prof. K.S. Venkatesh for his guidance and supervision to perform this research.

References:-

1. K. Wu, E. Otoo, and K. Suzuki, "Optimizing two-pass connected-component labelling algorithms," Pattern Analysis & Applications, vol.2, pp. 117-135, 2009.
2. K. Suzuki, I. Horiba, and N. Sugie, "Linear-time connected-component labelling based on sequential local operations," Computer Vision and Image Understanding, vol. 1, pp.1-23, 2003.
3. Rosenfeld, A., Pfaltz, J.L., "Sequential Operations in digital Processing," JACM, 13, 471-494, 1966.
4. C. Fiorio and J. Gustedt, "Two linear time Union-Find strategies for image processing," Theoretical Computer Science, 154(1996) 165-181.
5. O. Kalentev, A. Rai, S. Kemnitz, and R. Schneider, "Connected component labelling on a 2D grid using CUDA," J. Parallel Distributed Computing, pp. 615-620, 2011.
6. F. Chang, C. Chen, and C. Lu, "A linear-time Component-labelling algorithm using contour tracing technique," Computer Vision Image Understanding, vol. 2, pp. 206-220, 2004.
7. K. Hawick, A. Leist, and D. Playne, "Parallel graph component labelling with GPUs and CUDA," Parallel Computing, vol. 12, pp. 655- 678, 2010.
8. R. H. Haralick, "Some neighborhood operations, In M. Onoe, K. Preston, and A. Rosenfeld," (Eds.) Real Time/Parallel Computing Image Analysis, 1981, Plenum Press, New York.
9. Jung-Me Park, Carl G. Looney, Hui-Chuan Chen "Fast Connected Component Labelling Algorithm Using A Divide and Conquer Technique", 2000.
10. R.c. Gonzalez, and R.E. Woods, Digital Image Processing, Prentice-Hall, New Jersey, 2002.
11. Akmal Rakhmadi, Nur Zuraifah Syazrah Othman, Abdullah Bade, Mohd Shafry, Mohd Rahim and Ismail Mat Amin, "Connected Component Labelling Using Components Neighbors-Scan Labelling Approach", Journal of Computer Science 6 (10): 1099-1107, 2010.
12. Yapa, R.D. and K. Harada, "Connected component labelling algorithms for gray-scale images and evaluation of performance using digital mammograms. Int. J. Comput. Sci. Network Secur." 8: 33-41, 2008.
13. Wu, K., O. Ekow and S. Arie, "Optimizing connected component labelling algorithms". Proc. SPIE, 5747: 1965-1976. DOI: 10.1117/12.596105,2005.
14. Youngsung Soh, Hadi Ashraf, Yongsuk Hae, Intaek Kim, "FAST PARALLEL CONNECTED COMPONENT LABELLING ALGORITHMS USING CUDA BASED ON 8-DIRECTIONAL LABEL SELECTION" International Journal of Latest Research in Science
15. and Technology, Volume 3, Issue 2: Page No.187-190 ,March-April, 2014
16. R. Lumia, L. Shapiro, and O. Zuniga, A New Connected Components Algorithm for Virtual Memory Computers, Computer Vision, Graphics, and Image Processing, 22(1983) 287-300.
17. http://sipi.usc.edu/services/database/database.cgi?volume=misc
18. http://www.ece.ncsu.edu/imaging/Archives/ImageDatabase/Medical/