A Connected Component Labeling Algorithm for Grayscale Images and Application of the Algorithm on Mammograms

Roshan Dharshana Yapa
Information Major, Graduate School of Engineering,
Hiroshima University, Japan.
dharshana@hiroshima-u.ac.jp

Harada Koichi
Information Major, Graduate School of Engineering,
Hiroshima University, Japan
Harada@mis.hiroshima-u.ac.jp

ABSTRACT
A new algorithm for connected component labeling is presented in this paper. This algorithm requires only one scan through an image for labeling connected components. Once this algorithm encounters a starting pixel of a component, it completely traces all the contour pixels and all internal pixels of that particular component. This algorithm recognizes components one at a time in the image while scanning in raster order. This property will be very useful in areas such as image matching, image registration and content-based information retrieval etc. This algorithm is also capable of extracting contour pixels of an image and storing them in the order of clock-wise direction which will provide very useful information in many applications. Also this algorithm assigns consecutive label numbers for different components and hence needs a minimum number of labels. As our main research is on mammography image analysis for diagnosing breast cancers, we applied this algorithm to mammograms and measured performance of the algorithm in terms of processing time. This will be a useful algorithm in medical image analysis as a preprocessing tool.

Keywords
Connected component labeling, Contour-tracing, Grayscale image, Binary image, Mammogram.

1. INTRODUCTION
Connected component labeling is a very important tool in pre and post processing stages of image data analysis [5]. This paper represents a new algorithm for connected component labeling on grayscale images and demonstrates usability of connected component labeling algorithms on mammograms for diagnosing possible malignancies. Almost all known algorithms developed in this matter have been focused on binary images which need to convert grayscale images into binary images prior to the process. This needs additional overheads regarding computing resources. It gets worse when searching for unknown objects in images, such as diagnosing tumors in breast using mammograms.

SAC'07, March 11-15, 2007, Seoul, Korea.

In early stages of breast cancer, it is very difficult to detect abnormalities in a mammogram by human vision system as they have very low contrast. In such situations, it is impossible to predict the required threshold value straight away that is needed to convert from grayscale to binary image with the intention of detecting possible malignancies in mammograms. This leads to repeated conversion of grayscale images into binary images using different threshold values using manual or adoptive threshold value selection methods [2, 3, 8, 13] for detecting possible malignant lesions. It would make this process more convenient and would enable a reduction in computing overheads if we could develop more efficient labeling algorithms which can be directly applied to grayscale images rather than converting them into binary images.

In a previous study [16] we evaluated the performance of three connected component labeling algorithms [9, 10, 15]. These authors claim that their algorithms perform faster for binary images than other existing algorithms. These three algorithms were originally proposed and tested on binary images. All these algorithms were evolutions of conventional component labeling algorithms [4, 6, 7, 12, 14]. We introduced changes to these algorithms to search for a corresponding gray value, around the neighborhood of a given pixel and evaluated their performance directly on grayscale images (digital mammograms). From this study, it was found that our version of Kesheng Wu's [10] algorithm outperformed all other algorithms with respect to processing time. Meanwhile, Kesheng Wu had also suggested that another labeling algorithm based on a contour tracing technique would perform well on binary images when they contain more pixels which belong to the object of interest than the background. This contour tracing algorithm was developed by Chang Fu [1]. In our work we evaluated the possibility of using this algorithm directly on gray scale images. But a bottleneck was found which hindered us extending this algorithm to grayscale images. A brief description about Fu Chang's contour tracing algorithm and about the bottleneck which obstructed us extending this algorithm for grayscale images is described in section 2 of this paper. Our new algorithm is mainly based on the contour tracing algorithm; but, a different approach has been used to overcome the bottleneck found in the Fu Chang's original algorithm.

Section two of this paper presents a brief description on Chang's contour-tracing algorithm, and section three will discuss the problem of extending Chang's algorithm to grayscale images. Section four of this paper represents our new algorithm for connected component labeling on grayscale images. Section five

presents experiment results by comparing performance time of our new algorithm with another faster algorithm. Section six discusses pros and cons of this algorithm, while the last section presents concluding remarks.

2. CHANG'S CONTOUR TRACING ALGORITHM

A brief insight into the Chang Fu's connected component labeling algorithm is given in this section. This algorithm was mainly developed for the labeling of connected components in binary images. This method scans a binary image from left to right and top to bottom respectively. Conceptually, this algorithm can be divided into four main operations. These four operations are graphically represented in figure 1 A~D (reference [1]).

When a new external contour point (*A*) is encountered (figure 1 A), a complete trace over the contour is performed until the current pixel point returns to the starting point *A*. Also a new label is assigned to all the contour points traced during this step.

When a labeled external contour point (*A'*) is encountered (figure 1 B), it follows the scan line to find all subsequent white (white pixels are assumed to belong to the object of interest) pixels and assigns them the same label as pixel *A*.

When a new internal contour point (*B*) is encountered (figure 1 C), *B* is assigned the same label as the external contour of the same component. Then a complete trace is made over the internal contour containing *B* and also the same label as *B* is assigned to all contour points.

When a labeled internal contour point (*B'*) is encountered, it follows the scan line to find all subsequent white pixels and assigns them the same label as *B'*.

This algorithm completes component labeling through only a single pass over the image.

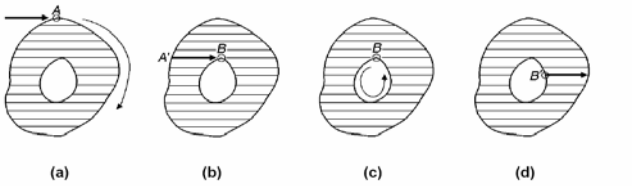

Figure 1: The four major steps in tracing and labeling component points. (reference [1])

When implementing this algorithm, the above mentioned four conceptual points have been reduced to three logical steps. First step deals with the external contour points.

If a pixel (white) *A* is unlabeled and the pixel above it is a black pixel then that pixel must belong to an external contour point. Then the algorithm starts searching all the points belonging to the external contour containing pixel *A*. While tracing external contour points, it marks surrounding black pixels with a negative integer as illustrated in figure 2 (a).

If the pixel below *P* is an unmarked (with a negative integer), then *P* must belong to a newly encountered internal contour (even though the pixel *P* already has been labeled). Then the algorithm starts tracing all the internal contour points of the component which contain pixel *P*. At the same time the algorithm marks surrounding black pixels of the internal contour with a negative integer. Then if the scan line sweeps pixel *Q*, the pixel below *Q* is no longer an unmarked pixel (figure 2 (a)) and hence the algorithm does not start tracing an internal contour containing the pixel *Q*. This ensures that each internal contour is traced only once. Figure 2 (b) illustrates the state of the labeled image after it has completed tracing the internal contour containing pixel P.

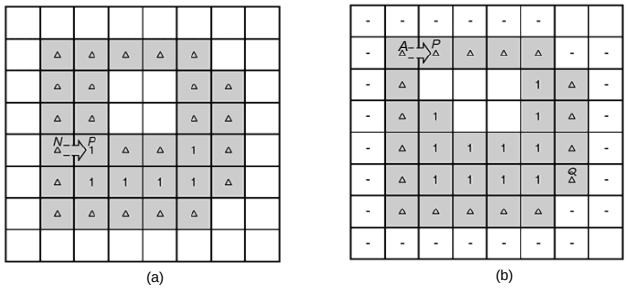

Figure 2: (a) pixel P belongs to both external and internal contours. But Q no longer belongs to an internal contour as the pixel below Q has already been marked with a negative integer. (b) The labeled image after completing the internal contour tracing. (reference [1])

If a pixel *P* does not belong to any of the above two steps it means that *P* does not belong to a contour point. Then *P* is assigned the same label as that of the left neighbor pixel.

To get more concrete information about this algorithm, readers are encouraged to refer to the original paper by Chang [1].

3. A BOTTLENECK WITH THE CHANG'S ALGORITHM

When considering binary images, each region of interest is surrounded by background (black) pixels as illustrated in figure 3. So, marking surrounded pixels of contour points with a negative value will not have any effect on the regions of interest in the image. But it is different with grayscale images. As illustrated in figure 4, a component with one gray value will be surrounded by some other components with different gray values which also belong to the same region of interest, as well as by pixels which belonging to the background.

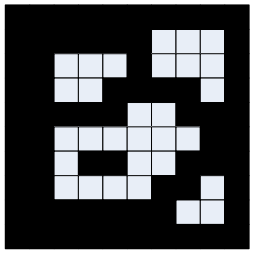

Figure3: Each region of interest (white pixels) is surrounded by background (black) pixels.

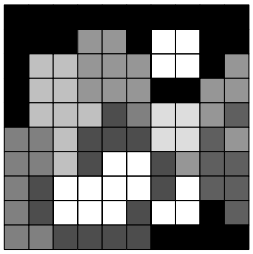

Figure 4: Components of different gray values are surrounded by other components of different gray values which belong to the same region of interest.

Based on this argument, in grayscale images, it is not possible to mark surrounded pixels of a contour with a negative value, as those surrounded pixels are overridden by pixels belonging to some other component in the same region of interest. Hence Chang's algorithm cannot be used with grayscale images for connected component labeling.

4. A NEW ALGORITHM FOR COMPONENT LABELING IN GRAYSCALE IMAGES

We propose a new algorithm which can be used with grayscale images. Even though this algorithm is also based on contour tracing, we have used a different approach to completely trace pixels which belong to a particular component by overcoming the above mentioned bottleneck. In this algorithm we scan a grayscale image in raster scan order. Mainly this algorithm uses a threshold value to differentiate the background region from the region of interest of an image. We assume that if a particular pixel has a gray value less than or equal to a given threshold value, the corresponding pixel belongs to the background region, or else it belongs to the region of interest. Using mammogram images in MIAS database [17] we found that, on average, gray values greater than 51 represent breast area while other values represent background. Hence we used a gray value of 51 as the threshold value to contrast breast area from the background. For the simplicity of the algorithm, we add a frame of width of one pixel around the image, padded with black pixels. This will eliminate the neighbor pixel selection problem at the boundaries of an image due to the possible overlapping of the scan mask and the image. We accompany a labeled image *L* with the original image document *I*. Also we use a queue data structure to store internal pixel information of the current component being traced. After completing contour tracing, information stored in queue is used to recognize internal pixels of the current component. Let *C* be the label index for newly encountered components. Initially *C* is assigned one.

This algorithm maintains a queue of data structure named as Point. The data structure Point is defined as follows.

```
struct  Point {
        int col;
        int row;
        int linkIndex;
}
```

The queue is referred to as *queuePoint*. The main goal of maintaining a *queuePoint* is to keep track of pixels to be searched in later iterations by the algorithm. The pixels belonging to the current component which is being traced are stored in *queuePoint*. The *col* and *row* fields of the structure denote row and column address of a pixel *N*, belonging to the current component which will be traced later by the algorithm. The *linkIndex* field represents index value of the pixel (method of indexing will be presented later in section 4.1) which is used to identify the corresponding pixel N as belonging to the current component.

This algorithm mainly consists of two main steps.

4.1 Step one

This step is a bit similar to the external contour tracing part of Chang's algorithm. When a scan line sweeps an unlabeled pixel *P*, in which gray value is grater than the threshold value and also the pixel above *P* is not equal to the gray value of pixel *P*, then *P* should be in a newly encountered external contour point. Then *P* is assigned to the next label index *C* and *C* is increased by 1. Then execute external contour tracing which is a procedure for tracing external contour points which is similar to contour tracing procedure in Chang's algorithm. Listing 1 explains flow of the main connected component labeling algorithm.

```
Main Algorithm {

Add a frame of width of one pixel around the image padded with black
pixels;
Initialize current pixel position currRow=1 and currCol=1;
Initialize current label value C=1;
Define label image maxtrix L (width and height equals to image matrix I);
Initialize all elements of L into 0;

for currRow=1 to maximum width of the image I {
      for currCol=1 to maximum height of the image I {
            if (current pixel value > threshold) AND
               (current pixel in image L is unlabeled) AND
               (pixel value at (currRow-1, currCol) != pixel value of
                current pixel) {
                  //Then the current pixel is a starting point of a newly
                  //encountered contour
                  Assign label value C into current pixel position in
                  label image L;
                  C = C + 1;
                  call externalContourTracing procedure
            }
      }
}
}
```

Listing 1: The main connected component labeling algorithm

External contour tracing (listing 2) procedure is dedicated to trace external contour points for a given point *S*. Always external contour tracing procedure is called when algorithm encounters a starting point of a new contour. This procedure first calls another procedure named tracer (listing 3). If tracer recognizes the given point *S* as an isolated point, then it ends the external contour tracing. Otherwise tracer returns contour point (say *T*) following *S*. This algorithm repeatedly calls tracer which returns contour point following current contour point (for example contour point following *T* and so on) until it meets both of the following conditions.

1. The tracer outputs *S* (starting point of the contour)
2. The contour point following *S* is *T* (second point of the contour).

As shown in figure 5 (refer [1]), stating point of the contour is *S* and the contour point following *S* is *T*, then the path traced by the tracer is *STUTSVWVS*.

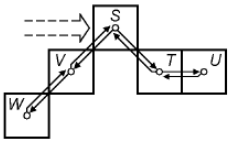

Figure 5: Tracing a contour of a stripe-shaped component (reference [1])

The procedure tracer is responsible for finding the contour point following a given point *P*. As explained in Chang's algorithm, each in eight neighbors of a given point is assigned an index as shown in figure 6. Then searching for the next neighbor contour point is done in clockwise direction through these indexed neighbor points.

5	6	7
4	*P*	0
3	2	1

Figure 6: Indices assigned to neighbor points of P

When the tracer procedure is called by external contour tracing procedure, tracer should first determine an initial point to start searching the next adjacent contour point. This algorithm uses two conditions to determine the initial point. If the given point *P* is a starting point of an external contour, then the initial searching point is set to 7. *P* being the starting point of an external contour, it is already known that the pixel above it, is a pixel that does not belong to the current component (belongs to background with respect to the current component is being traced), and hence the next pixel in clockwise direction is at index position 7. When *P* is not a starting point of an external contour, then its initial searching position is set to (d + 2) mod 8 where d is the index of previous contour point.

After determining the initial point, tracer starts searching for the next contour point in clockwise direction. If no contour point following *P* is found throughout the whole circle of neighbor pixels, *P* is considered to be as an isolated point and then it returns status of the point as isolated to the external contour tracing procedure which then ends contour tracing for the given point. If it finds a contour point among eight neighbors then it is the first contour point following *P*.

```
externalContourTracing procedure {

firstPixel = currentPixel;

call tracer procedure

if (current pixel is an isolated (NOT isIsolated) point)
      return;

secondPixel = currentPixel;

do {
      Assign label value of firstPixel to current pixel position in
      label image L;
      previousPixel = currentPixel;
      call tracer procedure;

} while (NOT (firstPixel==previousPixel AND secondPixel==currentPixel))

call fillContour procedure;
}
```

Listing 2: Algorithm for *externalContourTracing* procedure

```
tracer procedure {

initialize isIsolated = true;

if (the currentPixel is a starting point of an external contour) {
      initialize currentIndex = 7;
} else {
      previousIndex = (currentIndex + 4) MOD 8;
      currentIndex = (previousIndex + 2) MOD 2;
}


while (NOT completely search all 8 neighbour points AND isIsolated) {
      if (pixel value of currentPixel == pixel value of currentIndex
          position) {
            previousPixel = currentPixel;
            currentPixel = currentIndex position;

            call setNextQueuePoint procedure;
            if (pixel value of nextQueuePoint == pixel value of
                currentPixel position) {
                  PUSH nextQueuePoint into queuePoint (a data strucute);
                  Assign label value of firstPixel to pixel position of
                  nextQueuePoint in label image L;
                  isIsolated = false;
            }
      }
      else {
            currentIndex = (currentIndex + 1) MOD 8;
      }

}
}
```

Listing 3: Algorithm for tracer procedure

Once tracer found a contour point following *P*, then it also searches for a possible pixel belonging to the current component on the outermost line of pixels inside the contour line (figure 7) described as follows. We refer to this pixel as internal pixel.

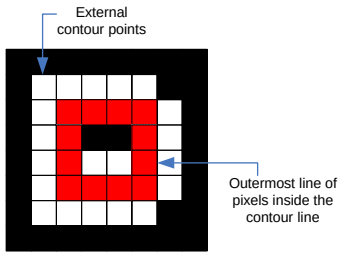

Figure 7: Outer-most line of pixels inside the contour line.

The tracer procedure examines for an internal pixel *N* on the outer-most line of contour which is a neighbor point of the current contour point (hereafter referred as inner layer). The position of the point *N* to be examined is determined based on the position of the first contour point following *P* found by tracer. If the next contour point following the current contour point *P* lies either on index 0 or on index 1, then the internal pixel *N* should be on index 2 (figure 8 (a)). If the next contour point following *P* lies either on index 2 or index 3, then the internal pixel *N* should be on index 4 (figure 8 (b)). Likewise if next contour point is either on index 4 or index 5, then internal pixel *N* should be on index 6 (figure 8 (c)) and if next contour point is either on index 6 or index 7, then the internal pixel *N* should be on index 0 (figure 8 (d)). This algorithm is explained clearly in listing 4.

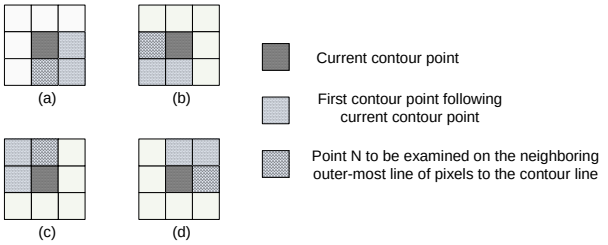

Figure 8: Searching positions of point N based on next contour point following the current contour point P.

After determining the position of internal pixel *N* and the relative index of the current pixel *P* with respect to the internal pixel *N*, if its grayscale value equals to grayscale value of the current pixel *P* and it has not already been assigned a label value, then tracer inserts this information into the *queuePoin*t data structure. Also it sets the label value of N equals to the label value of *P*.

External contour tracing procedure, calls the procedure tracer repeatedly until it returns to the starting pixel of the contour (S) and contour point following S is the second contour point (T).

At this point algorithm completely marks the contour that belongs to the current component and all the internal pixels belonging to inner layer of the component. Also the *queuePoint* holds position details of these internal pixels and the relative index of the previous contour points which identified these internal pixels. This *queuePoint* holds pixel details in the order traced by the algorithm.

```
setNextQueuePoint procedure {

switch (currentIndex) {
      case 0:
      case 1:
            nextQueuePoint.row = previousPixel.row+1;
            nextQueuePoint.col = previousPixel.col;
            nextQueuePoint.linkIndex = 6;
            break;
      case 2:
      case 3:
            nextQueuePoint.row = previousPixel.row;
            nextQueuePoint.col = previousPixel.col-1;
            nextQueuePoint.linkIndex = 0;
            break;
      case 4:
      case 5:
            nextQueuePoint.row = previousPixel.row-1;
            nextQueuePoint.col = previousPixel.col;
            nextQueuePoint.linkIndex = 2;
            break;
      case 6:
      case 7:
            nextQueuePoint.row = previousPixel.row;
            nextQueuePoint.col = previousPixel.col+1;
            nextQueuePoint.linkIndex = 4;
            break;
}
}
```

Lisiting 4: Algorithm for determining the position of *internal pixel*

4.2 Step two

Step two of this algorithm (lisiting 5) searches for all *internal pixels* of the component surrounded by the contour. For this, it uses the *queuePoint*. Algorithm retrieves first pixel information (say *N*) stored in the q*ueuePoint*. Then algorithm searches for possible pixels that belong to the component among neighbor pixels of *N*. But, the number of pixels searched can be reduced from 8 neighbor pixels to 5 neighbor pixels. When inserting information of neighbor pixels into the q*ueuePoint* we included their *x* and *y* coordinates and also the index value of the current pixel *P* with respect to the newly recognized pixel *N*. This information can be used to reduce the number of searching positions from 8 to 5. As illustrated in figure 9, there are four pixels common to each other among eight neighbors of any two given neighbor pixels including them selves. Each time when searching neighbor pixels which belong to the same component, we examine these five pixels. Hence when searching pixels belonging to the same component among eight neighbors of pixel *N* we can reduce redundant checking of pixels common to eight neighbors of both pixels *N* and *P*. So, in this case there are only five pixels to be examined as to whether they are belonging to the same component or not. This can be done by searching index positions using the equation (d + offset) mod 8; where d is the index position of neighbor pixel *P* and offset is an integer value among 2, 3, 4, 5, and 6 which represents corresponding offset from d.

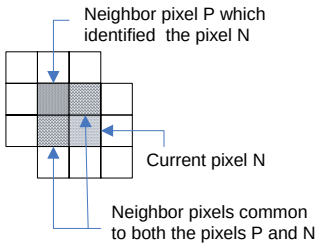

Figure 9: There are four pixels common to each other in the eight neighborhoods of two pixels.

While searching for neighbor pixels of *N*, if the algorithm finds that they belong to the same component and if they are not labeled earlier, then it sets the label of corresponding pixel as label of pixel *P*. It also inserts relevant pixel information related to this newly encountered pixel into the *queuePoint*. This process is repeated until there are no more data in the *queuePoint*.

At the end of these two steps, we have completely encountered all the pixels that belongs to a component for a given starting point of a contour.

```
fillContour procedure {

define currentPoint, neighborPoint of data type Point;
define currentIndex, offset of data type int;

while (NOT queuePoint is empty) {
      currentPoint = POP first element of queuePoint;

      for (offset=2 to offset<=6) {
            set currentIndex = (currentPoint.linkIndex + offset) MOD 8;
            set neighborPoint = pixel position of CurrentIndex (row and
                col);
            if (pixel value of neighborPoint == pixel value of
                firstPixel) AND (neighborPoint position in image L is
                unlabeled) {
                  Assign label value of firstPixel to pixel position of
                  neighborPoint in label image L;

                  neighborPoint.linkIndex = (currentIndex + 4) MOD 8;
                  PUSH currentIndex position into queuePoint;
            }
      }
}
}
```

Listing 5: Algorithm for searching internal pixels, belonging to the same component

5. EXPERIMENTAL RESULTS

An experiment was done to compare the performance of processing time of this new connected component algorithm with Kesheng Wu's [10] algorithm. In an earlier study [16], we

compared processing time of three fastest connected component labeling algorithms [9, 10,15] using 30 mammogram images taken from the MIAS database. From this comparison we concluded that our version of Kesheng Wu's algorithm which was modified for labeling components in grayscale images outperformed other algorithms [9,10,15] with respect to processing time. Hence in this experiment we compared performance of this new algorithm with Kesheng Wu's algorithm. For this experiment, we used the same set of images which was used in our previous experiment. Those 30 mammogram images were digitized at 200 micron/pixel and clipped or padded into size 1024x1024 pixels. In this experiment, these algorithms implemented using C++ and tested on Pentium 4 machine with 2.8 GHz speed and 512MB memory.

Graph 1 shows the results of performance comparison between the new algorithm and Kesheng Wu's algorithm. But according to this comparison, this algorithm takes longer time duration comparatively to Kesheng Wu's algorithm. When comparing processing time, Kesheng Wu's algorithm is about five times faster than the new algorithm. Total processing time for the new algorithm and Kesheng Wu's algorithm were 30.249 seconds and 7.967 seconds respectively. So, on average the new algorithm takes 1.0083 seconds and Kesheng Wu's algorithm takes 0.2655 seconds per mammogram image. Further analysis shows (figure 10) that processing time of the new algorithm is also directly proportional to the total number of pixels in connected components.

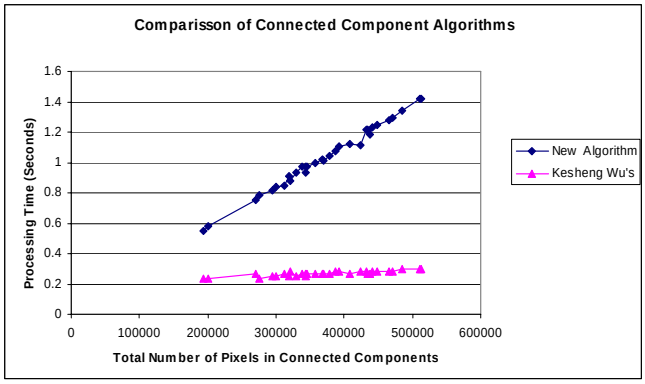

Figure 10: Comparison of processing time of two connected component labeling algorithms

Even though the new algorithm performs relatively slower but in a justifiable speed, it outperforms in several other aspects which will be discussed in the section six.

6. DISCUSSION

While scanning a given image in raster scan order if this algorithm encounters a starting point of a new contour, it first completely traces and labels all contour points and then labels all *internal pixels* that belongs to the same component. One major advantage of this algorithm over other similar algorithms is that it is capable of identifying component by component completely, as scanning in raster direction. Hence, this algorithm can be considered as a sequential component searching algorithm. This is a novel approach for connected component labeling algorithms. This approach can be useful in several areas such as image matching, image registration and content based information retrieval etc.

Similarly as Chang's algorithm is done for binary images, this new algorithm not only labels components but also extracts component contours and sequential orders of contour points, thus can be used in many image processing applications.

This algorithm needs only a single pass over an image and it completely labels all the components in the image. Hence this does not need re-labeling as required by most of the other component labeling algorithms. Also this algorithm assigns the same label for all the internal pixels belonging to the same component and assigns different labels for different components. Further this algorithm ensures consecutive label values for each new component and hence requires minimum number of label values.

But one disadvantage of this algorithm is the comparatively long processing time. When we concern grayscale images, especially medical images such as mammograms, most components consist of either single pixel or very few pixels. Hence most of the time this algorithm searches all 8 neighbor points while tracing contour points. This is the main cause for the long processing time. But it would be possible to improve the speed of the algorithm by considering neighbor pixels having closer gray values to each other are as belonging to the same component. Here, we need to use some clustering techniques to group neighbor pixels into components. In that case, it would be possible to reduce number of pixels to be searched among neighbor pixels.

Further, we applied this algorithm on grayscale digital mammograms and tried to assess the possibility of using it as a pre-processing tool for diagnosing tumors and possible abnormalities in breast. Multi-scale image analysis technique in scale-space proposed by Tony Lindeberg [11] also used in this application. Blobs in mammograms were identified by using component information such as adjacent regions and gray values collected during the labeling algorithm. These blobs would represent possible suspicious areas and hence further processing will be needed in these areas.

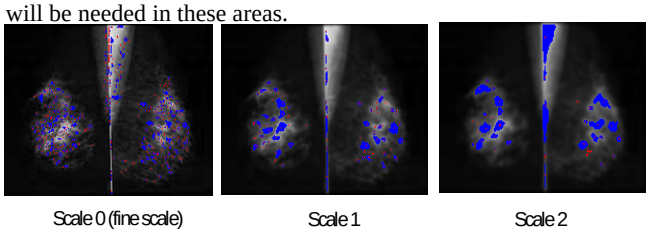

Figure 11: Application of connected component labeling algorithms on grayscale mammograms for detecting blobs in different scales in scale-space.(blue – blob area, red – local maxima points)

Figure 11 represents blobs detected in a mammogram in three different scales on scale-space. A large number of blobs are shown in fine scale (scale 0) due to noise and texture. But as we increase the scale parameter (smoothing) in scale space, smaller blobs are mearged into larger blobs; these information can be

stored in data structures and can be used for further analysis. This shows the possibility of using connected component labeling algorithms directly on gray scale images for diagnosing tumors and other abnormalities.

7. CONCLUSIONS AND FURTHER RESEARCH

Here we presented a new algorithm for connected component labeling. In terms of component by component labeling and extracting contour information, this would be very useful in many applications including medical imaging. As our main research is on the filed of mammograms; we applied this algorithm on mammogram images downloaded from MIAS database. In our further research, we will be investigating the possibility of using this algorithm for preprocessing stage of mammogram image analysis for diagnosing abnormalities by using the scale space theory [11].

8. REFERENCES

[1] Chang F., Chun-Jen C., and Chi-Jen L., A Linear-time Component -labeling Algorithm using Contour Tracing Technique, *Computer Vision and Image Understanding 93(2)*, p 206-220, 2004, Elsevier Science Inc.

[2] Christoyianni I., Dermatas E., Kokkinakis G., Automatic Detection of Abnormal Tissue in Mammography, *In Proc. International Conference on Image Processing, Volume 2*, p 877-880, 2001.

[3] David R., Arnau O., Joan M., Marta P., and Joan E., Brest Segmentation with Pectoral Muscle Suppresion on Digital Mammograms, *In Proc. Iberian Conference on Pattern Recognition and Image Analysis(2)*, p 471-478, 2005.

[4] Fiorio C., and Gustedt J., Two Linear Time Union-Find Strategies for Image Processing, *Theoretical Computer Science, Vol 154(2)*, p 165-181, 1996.

[5] Gonzalez R.C., and Woods R.E., *Digital Image Processing (Second Ed)*, Prentice Hall, 2002.

[6] Haralick R.M., Some Neighborhood Operations, Real Time/Parallel Computing Image Analysis, *Plenum Press, New York*, p 11-35, 1981.

[7] Hashizume A., Suzuki R., Yolouchi H., An Algorithm of Automated RBC Classification and its Evaluation, *Bio Medical Engineering, vol. 28(1)*, p. 25-32, 1990.

[8] Jasjit S., Suri, Rangaraj M., Rangayyan, Recent Advances in Breast Imaging, Mammography, and Computer-Aided Diagnosis of Breast Cancer, *SPIE Publication*, 2006.

[9] Jung-Me P., Carl G. Looney, Hui-Chuan C., Fast Connected Component Labeling Algorithm Using a Divide and Conqure Technique, *CATA 2000 Conference on Computers and Their Applications*, pp 373-376, Dec. 2000.

[10] Kesheng W., Ekow O., and Arie S., Optimizing Connected Component Labeling Algorithms, *In Proceedings of SPIE Medical Imaging Conference*, p. 1965-1976, Apr 2005.

[11] Lindeberg T., *Scale-Space Theory in Computer Vision*, Kluwer Academic Publishers, 1994.

[12] Lumia R., Shapiro L., Zuniga O., A New Connected-Components Algorithm for Virtual Memory Computers, *Computer Vision Graphics and Image Processing, Vol 23*, p 207-217, 1983.

[13] Michael A., Wirth, *A Non-rigid Approach to Medical Image Registration: Matching Images of the Breast*, Ph.D. Thesis, 2000, RMIT University, Melbourne, Australia

[14] Rosenfeld A., Pfaltz J.L., Sequential Operations in Digital Processing, *Journal of ACM, Vol. 13*, 471-494, 1966

[15] Suzuki K., Isao H., and Noboru S., Linear-Time Connected-Component Labeling Based on Sequencial Local Operations, *Computer Vision and Image Understanding 89(1)*, p 1-23, Acedemic Press, 2003

[16] Yapa R.D., Harada K., A performance Evaluation of Connected Component Labeling Algorithms on Grayscale Digital Mammograms, *In Proc. 7th International Information Technology Conference, Colombo, Sri Lanka*, 2006 Sept.

[17] Digital Mammography Database Ver 1.2, The Mammographic Image Analysis Society, (*http://www.wiau.man.ac.uk/services/MIAS/MIASweb.html*)