International Journal of Current Research and Modern Education (IJCRME)
Impact Factor: 6.925, ISSN (Online): 2455 - 5428
(www.rdmodernresearch.com) Volume 3, Issue 1, 2018

A STUDY ON PRE AND POST PROCESSING OF FINGERPRINT THINNED IMAGE TO REMOVE SPURIOUS MINUTIAE FROM MINUTIAE TABLE

K. Krishna Prasad* & P. S. Aithal**

* Research Scholar, College of Computer and Information Science, Srinivas University, Mangalore, Karnataka

** Vice Chancellor, College of Computer and Information Science, Srinivas University, Mangalore, Karnataka

Cite This Article: K. Krishna Prasad & P. S. Aithal, "A Study on Pre and Post Processing of Fingerprint Thinned Image to Remove Spurious Minutiae from Minutiae Table", International Journal of Current Research and Modern Education, Volume 3, Issue 1, Page Number 197-212, 2018.

Abstract:

In Fingerprint recognition, after the initial preprocessing, the feature is extracted from the Fingerprint thinned image. Extraction of crucial and beneficial capabilities or features of interest from a fingerprint image is an essential venture during recognition. Feature extraction algorithms pick handiest or only applicable features important for enhancing the performance of matching and recognition rate and outcomes with the feature vector. The feature extraction algorithms or techniques require only relevant features like minutiae details and do not require any background details or domain-specific details. They need to be smooth or easy to compute with a purpose to gain a viable or practicable technique for a huge image series. Minutiae details or fingerprint ridge ending or bifurcation details using skeletonized or thinning approach is a very popular method for feature extraction. The preprocessed thinned image is further post-processed to remove some false minutiae from minutiae table and which is generated through crossing number theory. One more purpose of post-processing is to reduce the number of minutiae points by removing false minutiae structures like spurs, ride breaks, short ridge, holes or islands, bridges, and ladders. In this paper w × w window neighborhood is considered for each minutia in Minutiae Table. Minutiae Table contains Ridge ending or bifurcation code as 1 or 3 with its location details means x and y position in two columns and the sum of these details as its fourth column. These Minutiae tables are used for generating Fingerprint Hash code, which can be used as index-or identity key in order to uniquely identify an individual person or as one factor in Multifactor Authentication Model.

Key Words: Post Processing, Thinning, Ridge Ending, Ridge Bifurcation & Minutiae Table.

1. Introduction:

Automatic Fingerprint Identification System (AFIS) consists of numerous techniques and process before matching and identification, which are Contrast Adjustment or Image Enhancement, Image Segmentation, Skeletonization or Thinning, Preprocessing the thinned image, Orientation, Post processing, and Minutiae Extraction [1-12]. Minutiae details or fingerprint ridge ending or bifurcation details using skeletonized or thinning approach is a very popular method for feature extraction. Initially, the fingerprint image is preprocessed and the last stage of preprocessing is thinning. The preprocessing is usually consists of series of a process like filtering, image enhancement, binarization, segmentation and thinning. The binarized image after segmentation is then thinned using a set of policies that eliminate pixels from ridges till the ridges are one-pixel period or length [13]. There are numerous strategies available in the literature for Skeletonization or thinning method [14-16]. After extracting the minutiae from the thinned image a few post-processing is carried to cast off any spurious minutiae and final features of the fingerprint image are obtained. The strategies on this elegance are of types–crossing number based and morphology-based.

However, techniques based totally on thinning are sometimes sensitive to noise and the skeleton shape does no longer conform to intuitive expectation. Nonskeletonized feature extraction uses a binary image based totally technique. The principle problem within the minutiae extraction technique the use of thinning processes comes from the reality that minutiae within the skeleton image do not usually correspond to true minutiae inside the fingerprint image. In fact, quite a few spurious minutiae are determined because of undesired spikes, breaks, and holes. Consequently, put up processing is usually followed to keep away from spurious minutiae, which are based on each statistical and structural fact after characteristic or feature detection. After skeleton formation or on thinned image some preprocessing operation is done in order to remove spurious minutiae or ridge patterns. One of the morphological operations called erosion. The morphological establishing operation is blended with the morphological erosion and the dilation operations. Where in erosion operation is implemented to shrink or thin an object and dilation operation is utilized to make bigger or thicken an object. In a Skeletonised, fingerprint image, white regions encompass background and some styles of noises. For obtaining an amazing and easier skeleton or minutiae features the provided set of rules adapts the morphological erosion operation to delete the white areas occupied via noise or to identify non-minutiae points.

In this research, we use either skeletonized or thinned image minutiae feature extraction based on crossing number theory. In order to implement pre and post-processing algorithm, we use MATLAB2015a. The

remaining part of the paper is organized as follows. Section 2 describes Preprocessing of Thinned Fingerprint Image with its theory and algorithm. Section 3 explains about Feature extraction. Section 4 describes Minutiae Table. Section 5 describes post processing. Section 6 explains the application of Minutiae Table. Section 7 concludes the paper.

2. Preprocessing of Thinned Fingerprint Image:

Fingerprint image can be identified or recognized with or without the use of skeletonized or thinned process. But fingerprint thinning is one of the most generally used pre-processing techniques before feature extraction. After skeleton formation or on thinned image some preprocessing operation is done in order to remove spurious minutiae or ridge patterns. The morphological establishing operation is blended with the morphological erosion and the dilation operations. An eight connected pixel mask is moved across the thinned image in order to remove white spaces or non-minutiae points. The 8-connected pixel mask is also called window. The window size is 3 × 3. The eight windows are obtained by keeping each pixel of the thinned image in a central position with value 1. If the 8-connected pixel mask with respect to this central position contains any edge means similar value (i.e. 1) in any one of the eight directions then central pixel is deleted. The 8 windows or pixel masks are shown below in Figure 1.

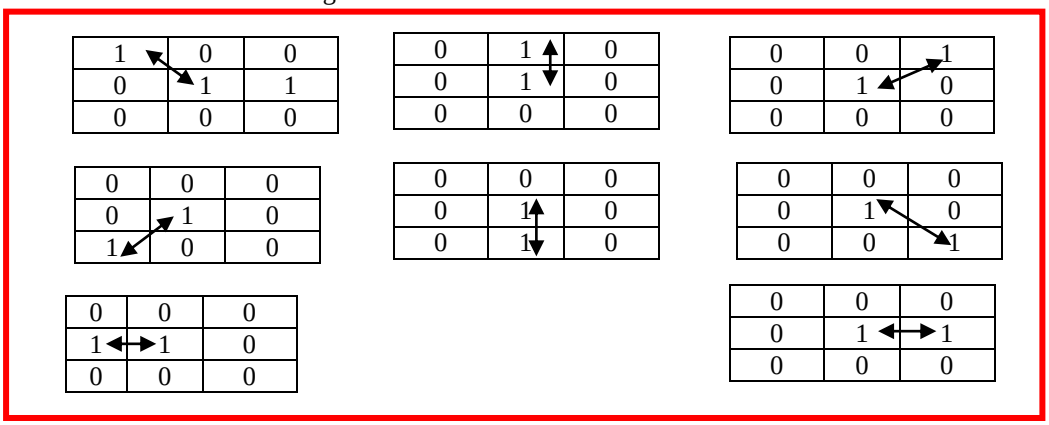


Figure 1: Eight windows of size 3 × 3 (pixel mask) used in skeleton preprocessing

The eight-pixel masks are in eight directions, North-West, South-West, South, North, South, South-East, West, and East from central pixel. The preprocessing function is called twice with an intention to remove more non-minutiae points or simply to improve the efficiency. Here erosion is thinned the white spaces or non-minutiae points. The result of the first window is taken as input for the second window and so on till the last or 8th window.

2.1 Algorithm for Preprocessing of Thinned Image:

This algorithm considers thinned image, $I_{skeleton}$ as input image and $I_{preprocessed}$ as output image.

1. Initialize 8 windows which are shown below

$temp_1 = erosion(I_{skeleton}, W_1)$ \\ w1= [1 0 0; 0 1 0; 0 0 0]
$temp_2 = erosion(temp_1, W_2)$ \\ w2= [0 1 0; 0 1 0; 0 0 0]
$temp_3 = erosion(temp_2, W_3)$ \\ w3= [0 0 1; 0 1 0; 0 0 0]
$temp_4 = erosion(temp_3, W_4)$ \\ w4= [0 0 0; 0 1 0; 1 0 0]
$temp_5 = erosion(temp_4, W_5)$ \\ w5= [0 0 0; 0 1 0; 0 1 0]
$temp_6 = erosion(temp_5, W_6)$ \\ w6= [0 0 0; 0 1 0; 0 0 1]
$temp_7 = erosion(temp_6, W_7)$ \\ w7= [0 0 0; 1 1 0; 0 0 0]
$I_{erosion} = erosion(temp_7, W_8)$ \\ w8= [0 0 0; 0 1 1; 0 0 0]
\\ $I_{erosion}$ →Erosion Applied image

2. Find the size of Erosion Applied image as $[R_{erosion}\ C_{erosion}] = size(I_{erosion})$
3. for each pixel of the $I_{erosion}$ image except first and last pixel do
   Check for $I_{erosion}$ image pixel value, if $I_{erosion} = 1$
   Assign to a temporary images of 3 × 3 size, as
   $temp1 = I_{erosion}(i-1:i+1, j-1:j+1)$
   temp2=[temp1(1,1);temp1(1,2);temp(1,3);temp(2,1),
   temp(2,2),temp(2,3);temp(3,1);temp(3,2);temp(3,3);]
   Initialize a counter as, counter=0;
   for each pixel of temporary image do
   Check, if (temp2 (1, k) = temp1 (1, k))
   Increment counter, counter = counter +1
   end if
   end for
   Check, if (counter = 9)

$I_{preprocessed\ 1}$ (i, j) = 0 or $I_{preprocessed\ 2}$ (i, j) =0

end if

end for

2.2 Workflow and Flowchart for Preprocessing of Thinned image:

The above algorithm for Preprocessing of Thinned image is explained using workflow and flowchart. The input for this algorithm is skeletonized, represented as, $I_{skeleton}$. The final output is preprocessed image denoted as, $I_{preprocessed}$. The Workflow and Flowchart of the preprocessing of the thinned image are shown in Figure 2 and Figure 3 respectively.

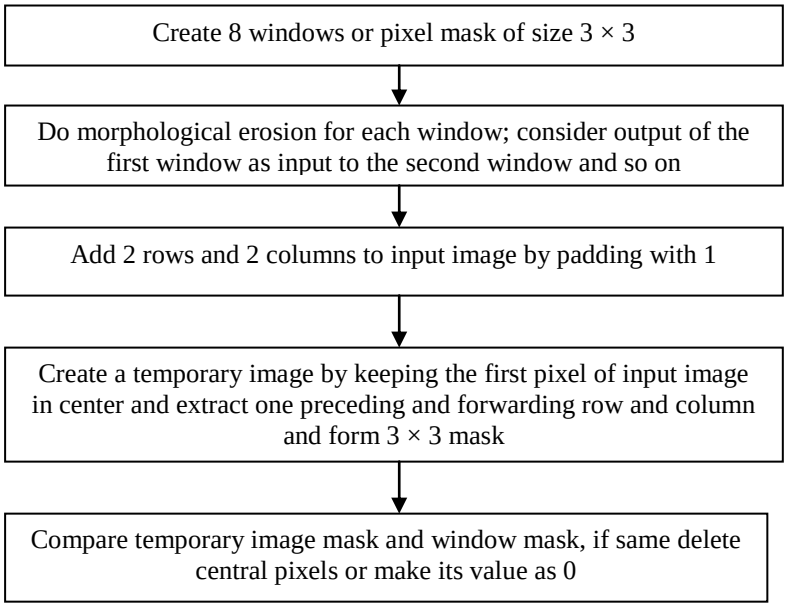


Figure 2: Workflow of the preprocessing of thinned image

2.3 Analysis of Preprocessing of Thinned Image:

Preprocessing for the thinned image is performed with an ultimate intention remove white spaces or non-minutiae points. Minutiae include ridge ending and ridge bifurcation, crossing, and isolated pixel and many more. But we concentrate only on ridge ending and bifurcation. Preprocessing removes white edges in all 8 directions.

The Table 1 shows thinned image pixel position removed for a 101_1.tif image taken from FVC ongoing DB1_B dataset. The datasets used for this study is from Fingerprint Verification Competition (FVC) ongoing 2002 benchmark datasets DB1_B, DB2_B, DB3_B, and DB4_B. Each dataset consists of 10 different fingerprint images and 8 impressions for each fingerprint labeled from 1 to 8. These datasets consist, a total of 3520 (880×4) fingerprints, but out of which only 40 fingerprints are available as a free resource for research testing purposes under the name of four datasets as DB1_B, DB2_B, DB3_B, and DB4_B. These datasets consist of image sizes of 388 pixels by 374 pixels (142 Kpixels) with resolution of 500 dpi, 296 pixels by 560 pixels (162 Kpixels) with resolution of 569 dpi, 300 pixels by 300 pixels (88 Kpixels) with resolution of 500 dpi, and 288 pixels by 384 pixels (108 Kpixels) with resolution of about 500 dpi for DB1_B, DB2_B, DB3_B, and DB4_B respectively. First two datasets are captured through optical sensor and DB3 and DB4 are captured through a capacitive sensor and SFinGe v2.51 sensor respectively. We use only DB1_B dataset for algorithm test purpose.

Table 1: Thinned image pixel position removed during preprocessing

S.No	Pixel Position of Input image	Window Value	Window Name	Function Call Name
1	(101, 46)	[[100]; [010]; [000]]	window1	preprocessing
2	(118, 103)	[[000]; [010]; [001]]	window6	preprocessing
3	(85, 45)	[[000]; [110]; [000]]	window7	preprocessing
4	(102, 09)	[[000]; [011]; [000]]	window8	preprocessing

In Table 1, column name, 'Function call name' values preprocessing1and preprocessing 2 indicates, preprocessing function called during first and second call respectively. We call preprocessing with an intention to improve the efficiency of filtering process and thereby enhanced the efficiency of the matching process based on extracted features. The time complexity of pre-processing is Big-Oh (n^2).

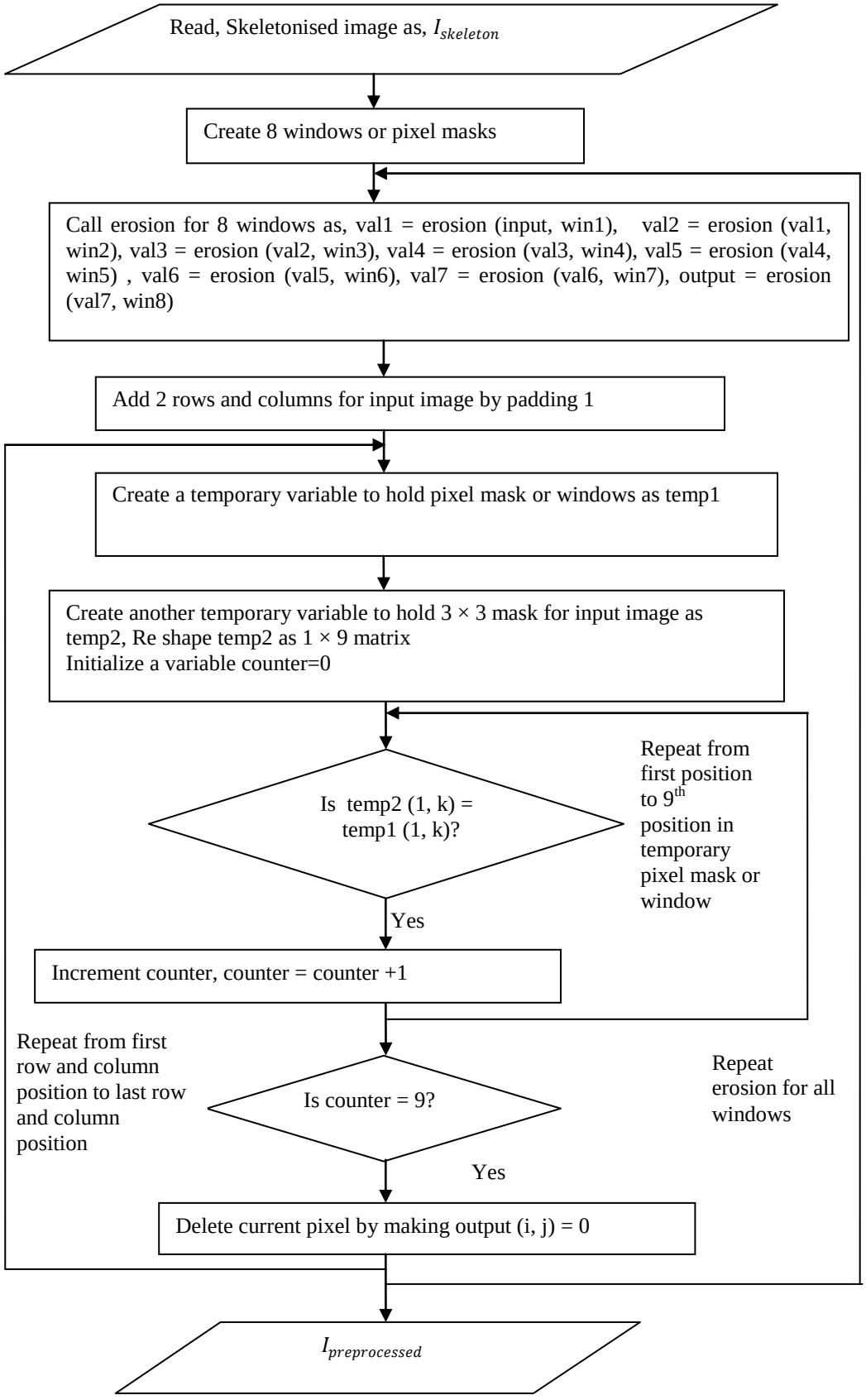


Figure 3: Flow chart of preprocessing state of Thinned image

3. Feature Extraction:

The individuality characteristic of the fingerprint is determined by the local ridge characteristics known as minutiae, which can be one of the most important standards utilized in fingerprint identification systems [17-

18]. There are more than one hundred fifty minutiae characteristics are diagnosed in literature. These local ridge characteristics aren't similarly distributed. Minutiae are labeled in two sorts primarily based on minutiae factors as ridge ending and bifurcation. We concentrate only ridge ending and bifurcation. From a preprocessed thinned image, we can able to classify pixel positions into one of the possible 8-connected neighbors. A ridge pixel is called an isolated pixel if it does not contain any 8 connected neighbors. The ending is referred based on 8-connected neighbor having value 1. When 8-connected neighbor having value 3, then it's referred as bifurcation. If 8-connected neighbor having value exactly 4 then that is called as crossing.

The minutiae extraction processed defined in [19], used a 3 × 3-pixel mask to find or search ridge ending and ridge bifurcations. This method caused some problems or flaws due to the ridge ending repository at borders and spurious bifurcation or false minutiae inside the fingerprint. To remove these false minutiae, a series of rules are used [20]. In this regard, usually, fingerprints are less corrupt. In this all the fingerprint ridge patterns located at the border of the image are referred as invalid, this is due to the fact that, while capturing fingerprint image through sensors or any other capturing device only finite or countable number of points are only in contact. In this research, we make use of crossing number based theory to extract minutiae details-ridge ending and bifurcations.

3.1 Crossing Number:

The preprocessed, thinned fingerprint image's ridge pixel usually contains only single pixel with value 1 or 0. Consider that (x, y) denote a pixel on a thinned ridge, and, p0, p1..... p8, denotes its 8 neighborhood pixels. Because the number of minutiae detected is more, the possibility of correct result increases. The concept of the crossing number (CN) is initially used by Kasaei et al., (1997) [21], for the purpose of extracting the minutiae from thinned or skeleton image. The nearby pixel of every ridge pixel in the image has scanned the usage of a 3×3 window from which the minutiae are extracted as shown in Figure 4. The crossing number may be used to categorize a ridge pixel as a finishing, bifurcation or non-minutiae point. As an example, a ridge pixel with a crossing-number of zero will correspond to an isolated factor and a crossing number of 4 correspond to a crossing factor. The Rutovitz's, crossing number for a ridge pixel is given in (1).

$$CN_p = \frac{1}{2}\sum_{i=1}^{8}|p_i - p_{i+1}| \qquad (1)$$

In (1) pi is a pixel value in the neighborhood of pixel p which is a central pixel with pi value is 1 or 0 and also, p1= p9. The crossing number CN_p at a point p is expressed as half of the cumulative total between pairs of adjacent pixels belonging to the eight-neighborhood of p and is shown in Figure 4.

P1	P2	P3
P8	P	P4
P7	P6	P5

Figure 4: Neighborhood of crossing number based feature extraction for 3 × 3 size

3.2 Minutiae Extraction Algorithm based on Crossing Number:

In this study, minutiae are extracted from the preprocessed-thinned image using the crossing number theory. This algorithm takes $I_{preprocessed}$ as input and produces output in the form of Minutiae table. The algorithm for feature extraction is as follows.

1. Find the row and column size of $I_{preprocessed\ 2}$ and assign to m and n
2. Declare a variable to hold number of neighborhood as, count=0
3. for each pixel of $I_{preprocessed\ 2}$ except first and last pixel, do

Check, if $I_{preprocessed\ 2}(i, j) = 1$

Create a temporary image of size 3 × 3 neighborhood

tempimg = $I_{preprocessed\ 2}$ (i-1 to i+1 , j-1 to j+1)

Reshape temping and assign to temping1 as

tempimg1 = [tempimg(1,1) ; tempimg(1,2) ; tempimg(1,3) ; tempimg(2,3) ; tempimg(3,3) ; tempimg(3,2) ; tempimg(3,1) ; tempimg(2,1) ; tempimg(1,1)]

Declare a variable to hold crossing number count and initialize with value 0

$N_c = 0$

repeat for 8-connected neighbor

$N_c = N_c + |temp1(k) - temp1(k+1)|$

end for

Divide value of N_c by 2, $N_c = 0.5 * N_c$

Check, if ($N_c = 1$) or ($N_c = 3$)

Increment the count as count = count+1

Assign to M_{table} as, $M_{table}\,(Count, :) = [i, j, N_c, (i + j + N_c)]$

\\M_{table} → Minutiae Table

end if

end for

3.3 Workflow and Flowchart for Minutiae Extraction Based on Crossing Number:

The above algorithm for Minutiae extraction based on crossing number is explained using a flowchart. The input for this algorithm is preprocessed thinned image, $I_{preprocessed}$. The final output is Minutiae Table, represented as, M_{table} . The flowchart of the crossing number based minutiae extraction is shown using Figure 5. The workflow of this is shown in Figure 6.

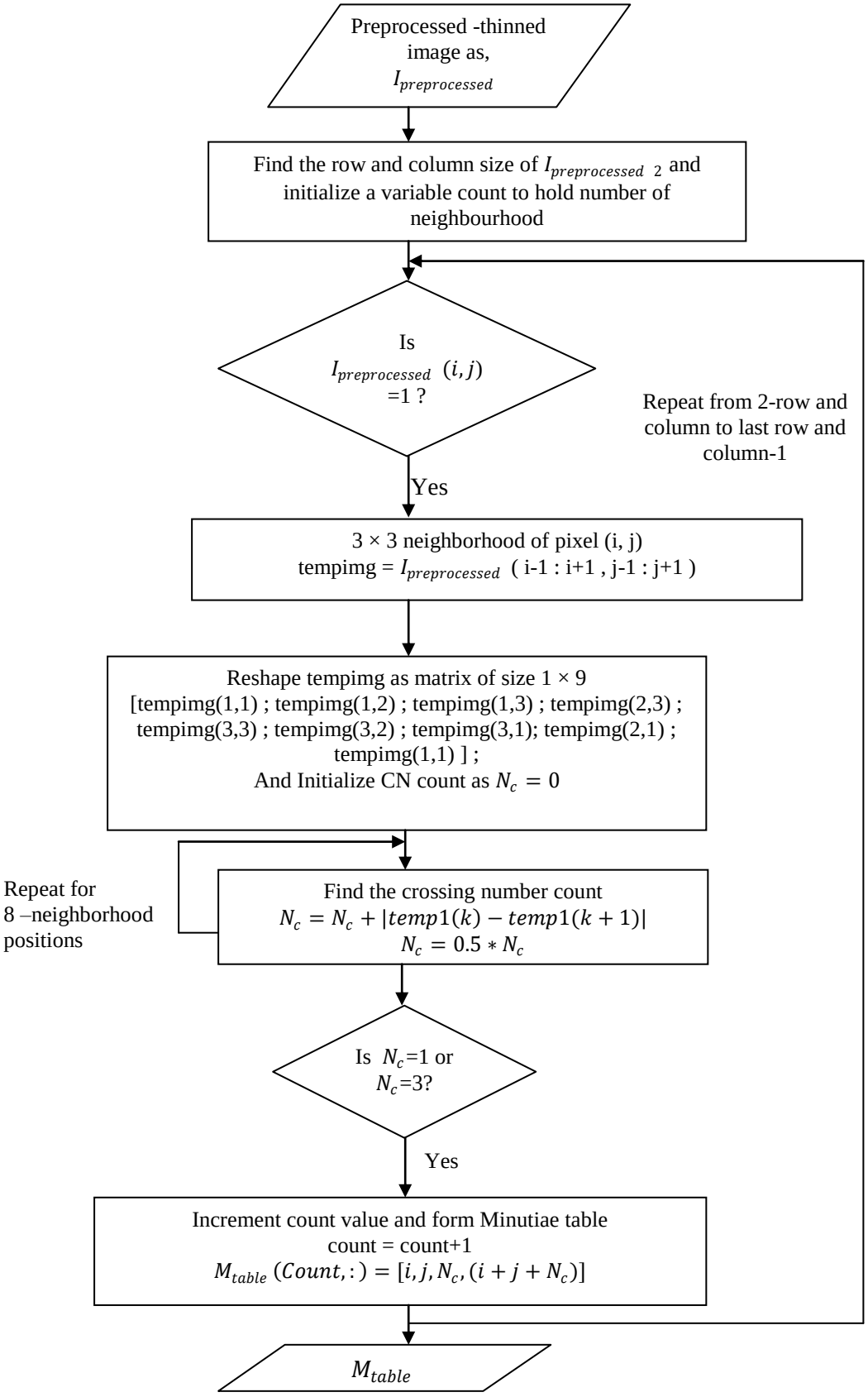


Figure 5: Flowchart of the Minutiae extraction based on crossing number

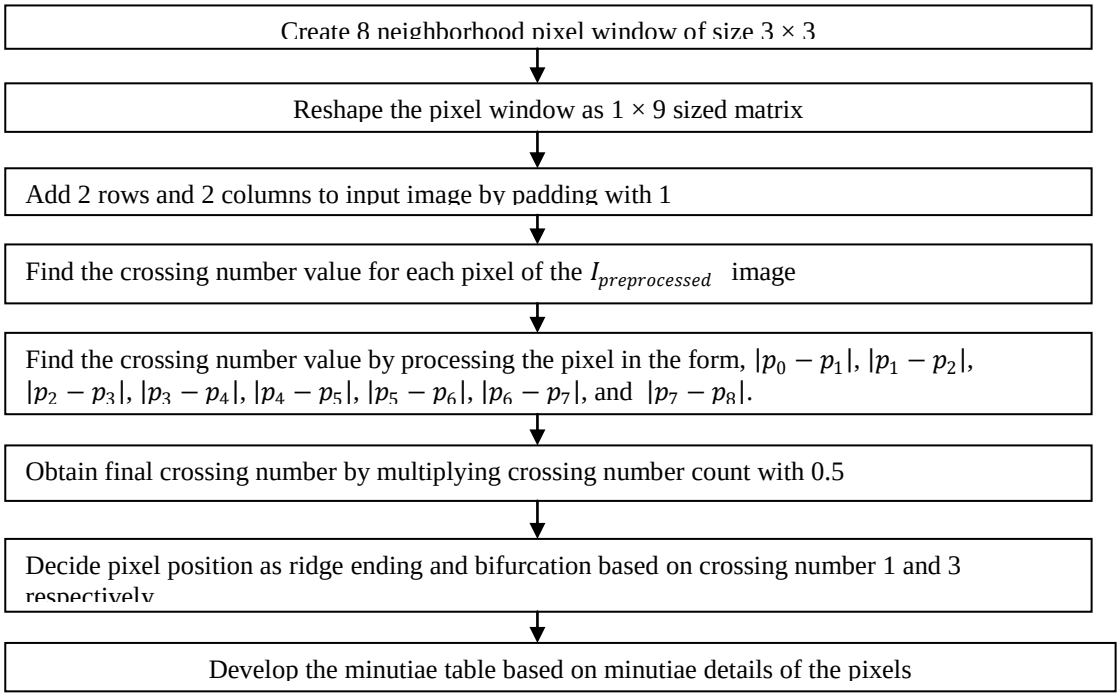


Figure 6: Workflow of the minutiae extraction using crossing number theory

4. Minutiae Table:

After extracting the features from, $I_{preprocessed}$ image minutiae details are stored in Minutiae Table. This Minutiae Table contains five columns. The first column contains a serial number. The second column contains minutiae x position, third column minutiae y position and fourth ridge ending or bifurcation as code 1 or 3 and the fifth column contains the sum of a second, third and fourth column. The structure of Minutiae table is shown in Table 2.

Table 2: Structure of M_{table}

S.No	Minutiae pixel 'x' position	Minutiae pixel 'x' position	Crossing Number	Sum of 2nd, 3rd and 4th column
1	5	86	3	94
2	15	76	3	94
3	17	106	3	126
4	18	85	3	106
5	21	49	3	73
6	21	83	3	107
7	25	57	3	85
8	25	61	3	89
9	25	99	3	127
10	46	105	1	152

This table only shows a sample value for four columns of M_{table} . Actually, an image may consist of hundreds of minutiae pixels. But in this table, only first 10 minutiae pixel details are shown.

5. Post Processing - Processing Minutiae Table:

After applying fingerprint image preprocessing on raw fingerprint, which include filtering, enhancement, binarisation, and segmentation thinned image or skeleton is formed. Further skeleton or thinned image is preprocessed to remove white areas occupied by the noise. Again preprocessed thinned image is further post processed to remove some false minutiae from minutiae table and which is generated through crossing number theory. One greater reason of post processing is to reduce wide variety of minutiae points by disposing of false minutiae structures [22]. The post processing algorithm used in this study is based on [23], here w × w window neighbourhood is considered for each minutiae in minutiae table. The size of the w is calculated on the basis of following statement w = 2d + 1, where d is considered as local ridge distance.

In this study we consider d as 1 unit. So w = 2×1+1 = 3. Average ridge distance in each region or from every pixel is usually referred as local ridge distance and it is an integer value due to round off calculation. The fingerprint image minutiae ridge ending and bifurcation is first analyzed from thinned image and in this study which is thinned preprocessed image. The minutiae extracted through crossing number based theory may include some spurious minutiae structure, which should be eliminated to maximum extent or full with the aid of

post processing the preprocessed thinned image. If 3×3 window is considered, then m along the branch of the window in all 8-directions and test whether any other ending in terms of pixel value 0→1 is not found then consider that pixel as true ridge.

5.1 Post processing Algorithm-Description:

The post-processing algorithm takes three arguments as Skeleton image after preprocessing, $I_{preproc\ essed}$, Minutiae table, M_{table} , and window size represented as param in our study. The output of this algorithm is final Minutiae table represented as final_M_{table} . Initially, some variables are declared and initialized with some values as follows. This algorithm is explained based on window size 3 × 3.

window = param / 2

If window variable value is not an integer then round-off it. For example in this study param value is 3 means, after round off window value will be 1. Create two matrixes, Rw and Clmn, of size 3 × 1 for holding indices of nonzero elements in the matrix.

Rw = zeros (3, 1), Clmn = zeros (3, 1)

Add two rows and columns for the $I_{preprocessed}$ image by padding value 0. Next, find the total number of values or rows in M_{table} . Use a variable, count, to hold index position for final_M_{table} maxval = size (M_{table}), count =0

I_1=$I_{preprocessed}$ after padding 2 row and 2 column with value 0

Move along the minutiae table, M_{table} from 1 row or position to last row or position. Use a variable part1 of size 3 × 3 to hold pixel values of the I_1 image, corresponding to M_{table} minutiae pixel position.

part1= I_1(M_{table} (ith row) to M_{table} (ith row) + 2, M_{table} (ith column) to M_{table} (ith column) + 2) Here i, represents M_{table} index from first position to till last position. Because we have padded 2 row and 2 column on both sides of the I_1 the minutiae pixel position occupies central position of the window. All those pixel positions which is having value 1 around the neighborhood of central pixel are referred as connected with the candidate or central pixel. Initially part1 is multiplied by -2 so that in the window all elements which are having value 1 become, -2. Next we initialize central or candidate pixel with value -1.

partI = -2 * partI ;

partI (Window+1 , Window+1) = -1 ; // Here window value is 1 because the param value is 3. The window size is 3 × 3. So it refers in this context, candidate pixel with value, PartI (2, 2). Next again another variable is created as temppart1 with size w × w with initial value zero.

temppart1 = w × w sized matrix with initial value 0.

This temppart1 will be copied with a value of part1 and candidate pixel will be assigned with value zero.

Next find the number of connected braches of candidate pixel by identifying non-zero window element pixel position and store it on Rw and Clmn as mentioned above.

[Rw, Clmn] = find (non-zero index position of temppart1 in row and column matrix)

The above statement simply returns no of connected branches of candidate minutiae pixel.

Next, we check whether candidate pixel is minutiae ending or bifurcation. If M_{table} 3, column value 1 means its ridge ending, and 3 means ridge bifurcation, which is based on crossing number based theory.

If there exists, only one edge out of the all edges of the windows of candidate pixel, with the transition as, 0→1 and count=1, then it's considered as true ridge ending. If the candidate pixel is ridge bifurcation then check for transition 0→1, 1→2, and 2→3, then and the count is 1 for each transition then its valid ridge bifurcation, while evaluating all edges of the candidate minutiae pixel.

5.2 Post Processing of Minutiae Table- Algorithm:

Input: $I_{preprocessed\ 2}$, M_{table}

Output: final_M_{table}

Parameters: $I_{preprocessed\ 2}$, M_{table} , Window size (param)

1. Initialize a variable window as, window = round (param / 2) //take integer value
2. Declare a variable Rw and Clmn to holds indices of nonzero elements in the matrix, Rw(3,1) with initial value 0, Clmn (3,1) with initial value 0
3. Initialize a new image as I_1=$I_{preprocessed\ 2}$ after padding 2 row and 2 column with value 0
4. Find the size of M_{table} as, size(M_{table}) // M_{table} -Minutiae Table
5. Initialize a variable Count to hold total number of rows of M_{table} , Count = 0
6. for all values of M_{table} do

Extract w × w sized window from I_1

Initialize a temporary variable part1 as part1= I_1(M_{table} (ith row) to M_{table} (ith row) + 2, M_{table} (ith column) to M_{table} (ith column) + 2)

Multiply part1 by -2, Part1 = -2 * Part1

Initialize part1 candidate pixel value as -1, part1 (window+1, window+1) = -1

Create a temporary variable as temppart1 with size w × w with initial value zero

to hold window value and copy part1 to temppartI

temppart1 (param , param) dimension with value 0.

tempPart1(Window to Window+2 , Window to Window+2) = part1(Window to Window+2 , Window to Window+2)

Initialize temppart candidate pixel value to zero.

temppart1 (Window+1 , Window+1) = 0 ;

find (non-zero index position of temppart1 in row and column matrix) and assign to Rw and Clmn

Check if candidate pixel is ridge ending or bifurcation, if it is 1 then ridge ending, if, 3 then ridge bifurcation

if (Table(i, 3)= =1)

Create a window Test with value zero of size w× w Test (param, param) with value 0

Find the maximum value of non-zero index position of Rw or simply

Size of Rw as, Max= size (Rw)

Traverse from first position of Rw to Max or last position for each value of Rw do

Check the connected branch of Part1 with candidate minutiae pixel

or simply check for value, -2.if (PartI(Rw(z) , Clmn(z)) = = -2)

Reassign Test window with value 1 Test(Rw(z) , Clmn(z)) = 1 ;

end if

end for

Check whether candidate pixel has only one connected edge or border by

calling extract_ring method

borders = extract_ring (Test)

Initialize a variable to hold count for ridge ending to ensure that it has only one connected edge, T01 = 0

Traverse from first position of the border to last position minus one

for each value of border-1 do //all the w × w size-1 borders

Check, if (Borders(p)= =0) & (Borders(p+1)= 1))

increment count T01 → T01 = T01 + 1

end if

end for

Ensure that T01 has only one connected edge

Check, if (T01 = = 1)

Increment final_M_{table} index by one

Count = Count + 1 ;

Load candidate minutiae pixel in final_M_{table}

final_M_{table} (Count, 1) = M_{table} (i , 1) // row index of candidate pixel

final_M_{table} (Count, 2) = M_{table} (i , 2) //column index of candidate pixel

final_M_{table} (Count, 3) = M_{table} (i , 3) //type of ridge (ending or bifurcation)

final_M_{table} (Count, 4) = M_{table} (i , 4) //sum of row, column, and type of ridge

end if

else part of candidate ridge type, means bifurcation

check that candidate minutiae pixel has at least three connected branches

if size(Rw(1))>=3 && size(Clmn(1))>=3

Name first 3 connected branch with candidate minutiae pixel as 1, 2, and

3 edges or branches (Minimum 3 branches are required if its bifurcation)

Part1(Rw(1) , Clmn(1)) = 1

Part1 (Rw(2) , Clmn(2)) = 2

Part1 (Rw(3) , Clmn(3)) = 3

Assign Part1 to temporary variable Test1

Test1 = Part1

Check whether candidate pixel has three connected edge or border by

calling for all three marked edges

Borders = extract_ring (Test1)

Initialize a variable to hold count for ridge bifurcation to ensure

that it has exactly three connected ridges means 6 points while

traversing along all connected points

T01 = 0

Traverse from first position of the border to last position minus

one for marked edge-1, edge-2 and edge-3

for each value of border-1 do //all the w × w size-1 borders

if (Borders(p)=0) & (Borders(p+1) =1) (Borders(p) =0) &

(Borders(p+1)=2) (Borders(p) =0) & (Borders(p+1) =3))

increment count T01 → T01 = T01 + 1

end if

end for

Ensure that T01 has only three connected edge

if (T01 = = 3)

Increment final_M_{table} index by one count = count + 1

Load candidate minutiae pixel in final_M_{table}

final_M_{table} (Count, 1) = M_{table} (i , 1) // row index of //candidate pixel

final_M_{table} (Count, 2) = M_{table} (i , 2) //column index of //candidate pixel

final_M_{table} (Count, 3) = M_{table} (i , 3) //type of ridge (ending //or bifurcation)

final_M_{table} (Count, 4) = M_{table} (i , 4) //sum of row, column, //and type of ridge

else part of if (T01 = = 3)

if size(Rw(1))<3 && size(Clmn(1))<3

continue with next iteration

end if

end if

end for

5.3 Post Processing of Minutiae Table- Workflow and Flowchart:

The workflow and flowchart of Post processing of Minutiae Table are shown in Figure 7 and Figure 8.

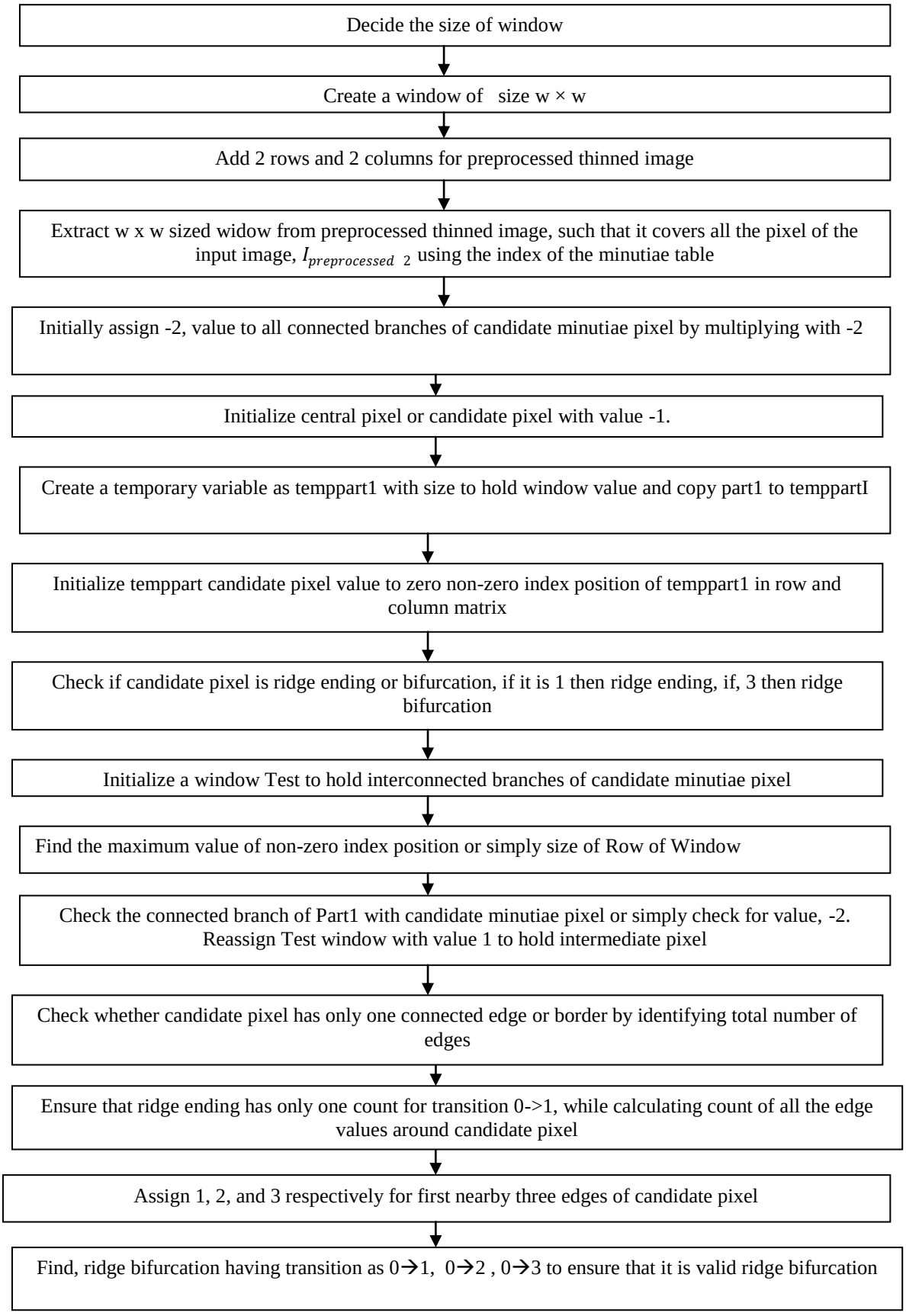
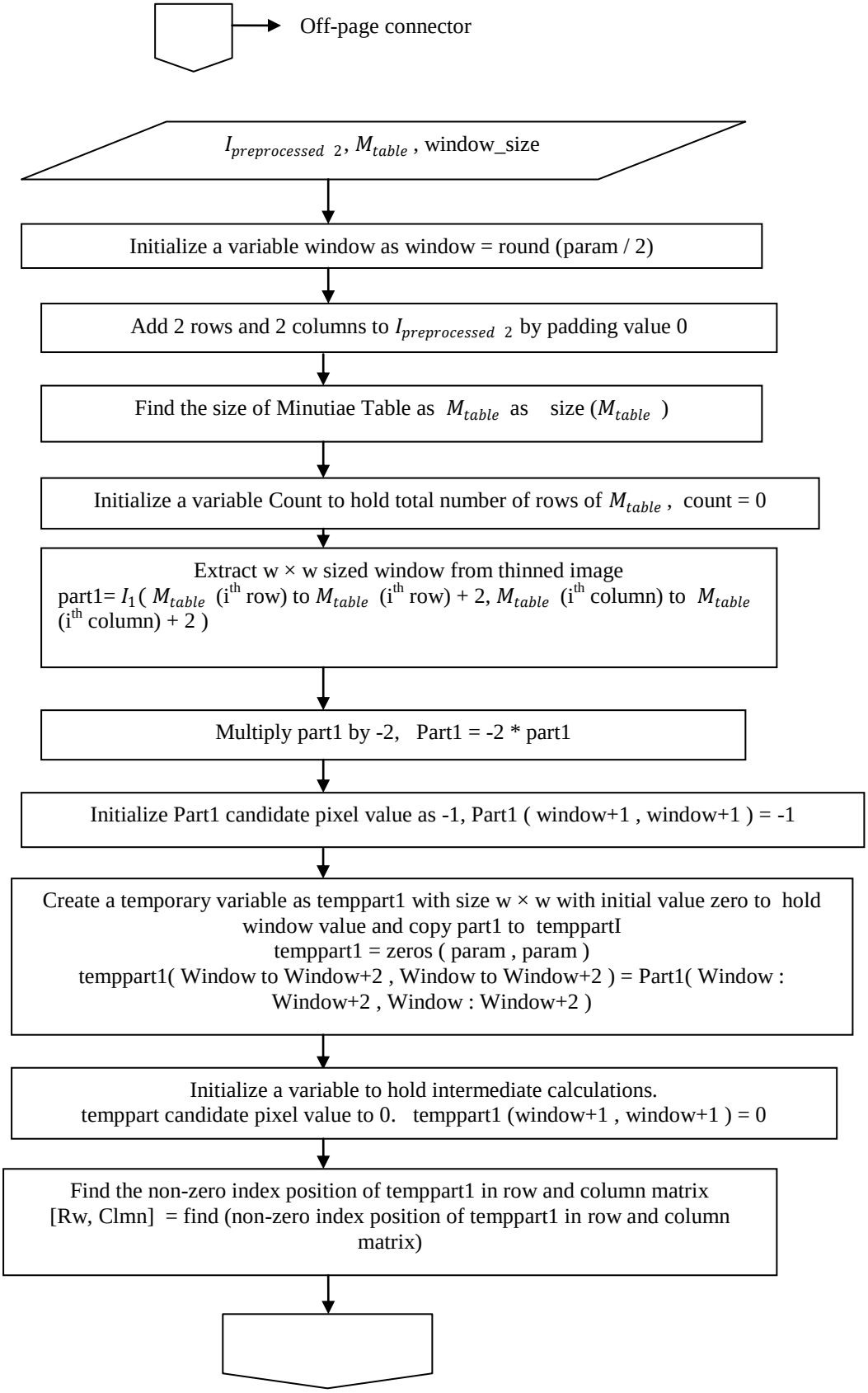
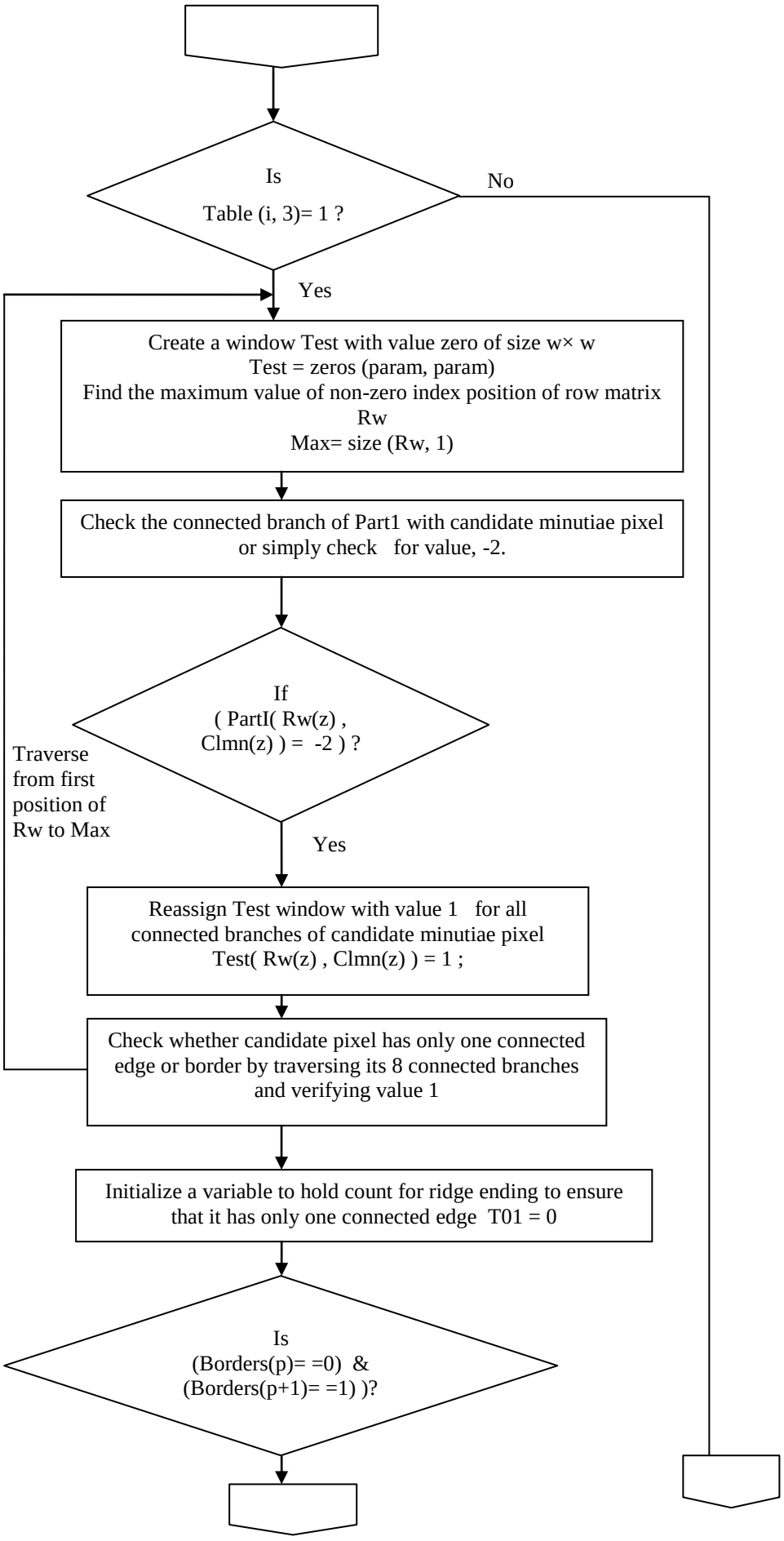
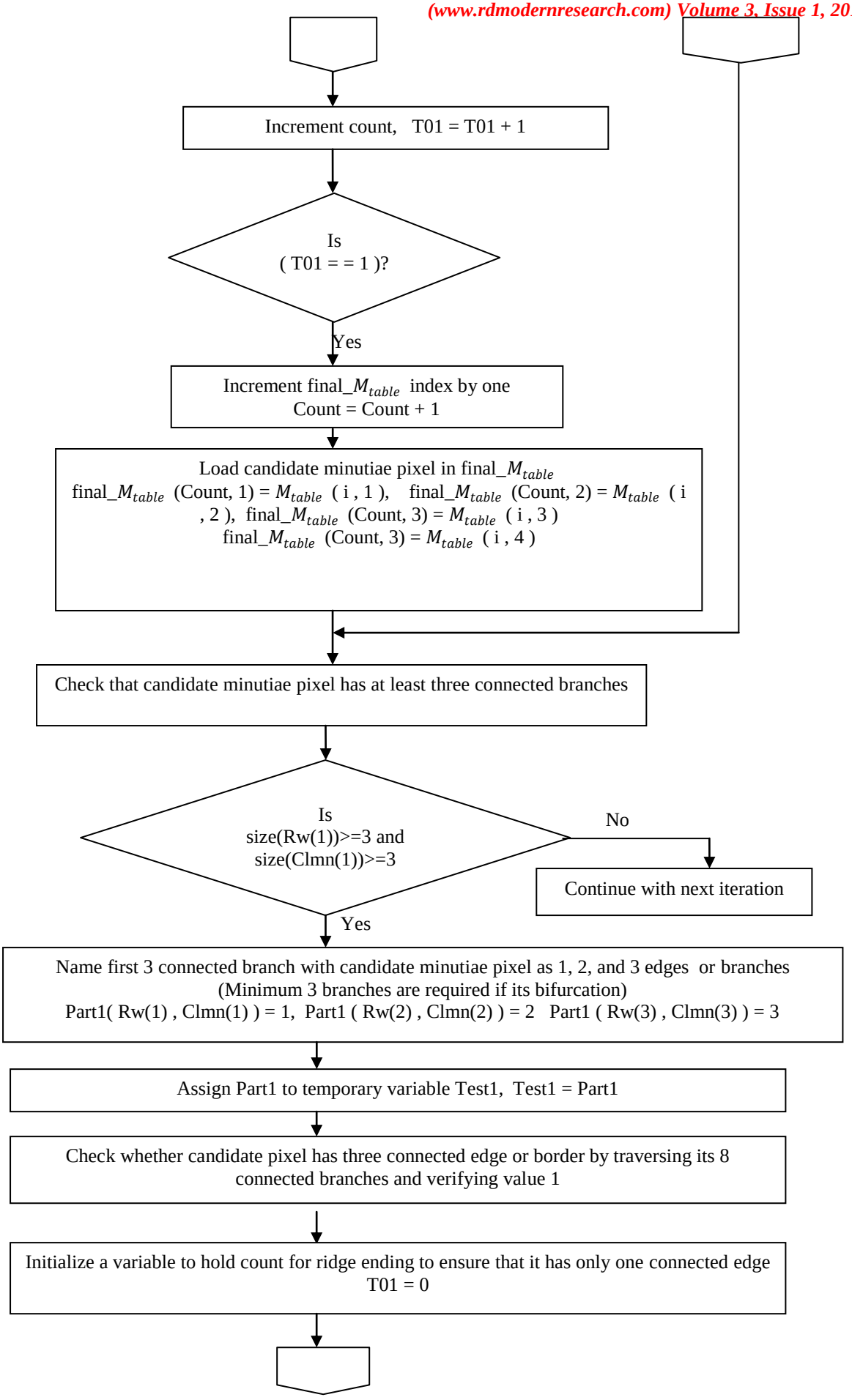


Figure 7: Workflow of Post Processing of Minutiae Table

The flowchart of post-processing spans across multiple pages, so flowchart symbol off-page connector are used to connect the flowchart symbols. The off-page connector is shown below.




Is
Table (i, 3)= 1 ?
No
Yes
Create a window Test with value zero of size w× w
Test = zeros (param, param)
Find the maximum value of non-zero index position of row matrix Rw
Max= size (Rw, 1)
Check the connected branch of Part1 with candidate minutiae pixel or simply check for value, -2.
If
(PartI(Rw(z) , Clmn(z)) = -2) ?
Traverse from first position of Rw to Max
Yes
Reassign Test window with value 1 for all connected branches of candidate minutiae pixel
Test(Rw(z) , Clmn(z)) = 1 ;
Check whether candidate pixel has only one connected edge or border by traversing its 8 connected branches and verifying value 1
Initialize a variable to hold count for ridge ending to ensure that it has only one connected edge T01 = 0
Is
(Borders(p)= =0) &
(Borders(p+1)= =1))?


Increment count, T01 = T01 + 1
Is
(T01 = = 1)?
Yes
Increment final_M_{table} index by one
Count = Count + 1
Load candidate minutiae pixel in final_M_{table}
final_M_{table} (Count, 1) = M_{table} (i , 1), final_M_{table} (Count, 2) = M_{table} (i , 2), final_M_{table} (Count, 3) = M_{table} (i , 3)
final_M_{table} (Count, 3) = M_{table} (i , 4)
Check that candidate minutiae pixel has at least three connected branches
Is
size(Rw(1))>=3 and
size(Clmn(1))>=3
No
Continue with next iteration
Yes
Name first 3 connected branch with candidate minutiae pixel as 1, 2, and 3 edges or branches
(Minimum 3 branches are required if its bifurcation)
Part1(Rw(1) , Clmn(1)) = 1, Part1 (Rw(2) , Clmn(2)) = 2 Part1 (Rw(3) , Clmn(3)) = 3
Assign Part1 to temporary variable Test1, Test1 = Part1
Check whether candidate pixel has three connected edge or border by traversing its 8 connected branches and verifying value 1
Initialize a variable to hold count for ridge ending to ensure that it has only one connected edge
T01 = 0

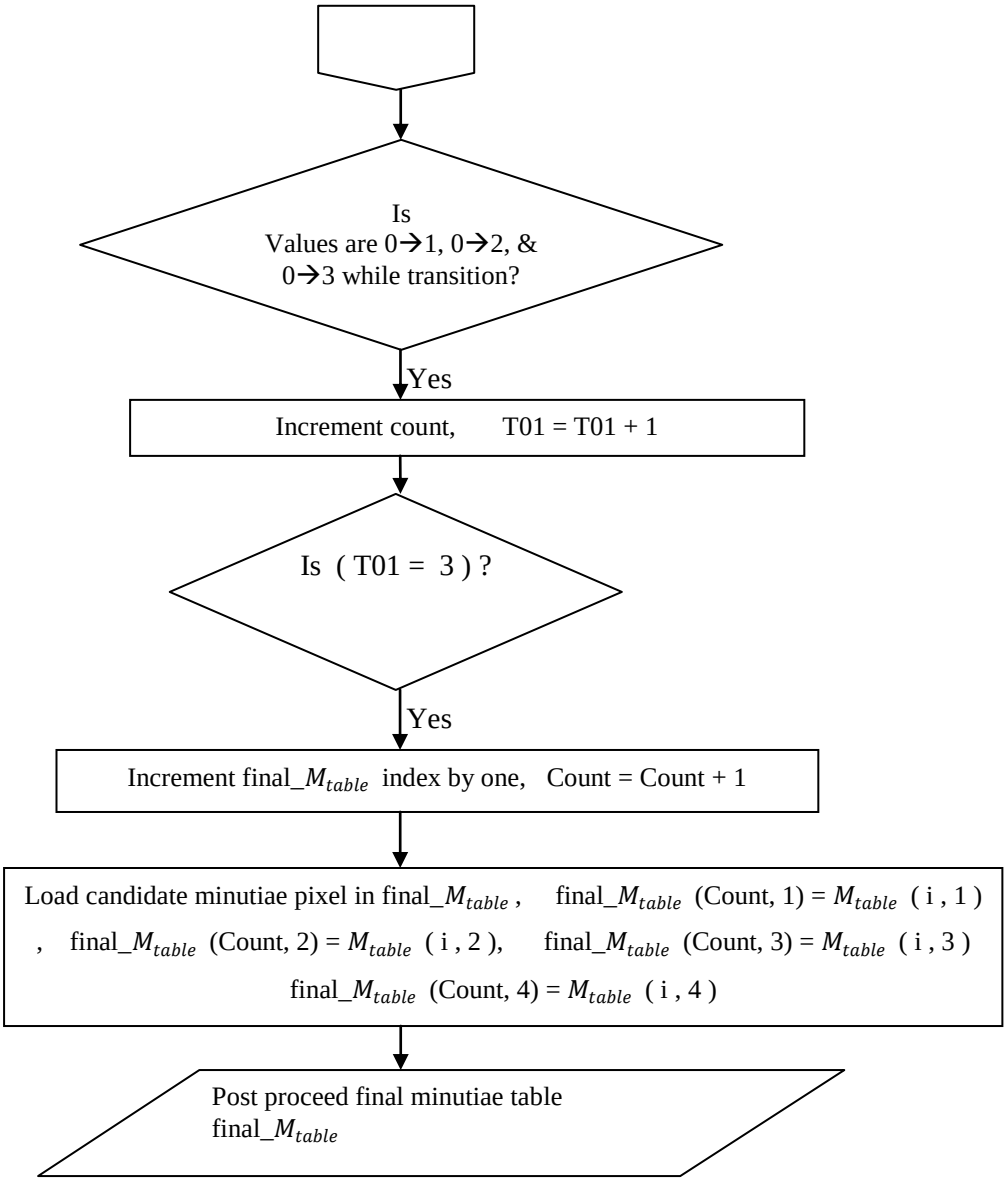


Figure 8: Flow chart for post processing of Minutiae Table

5.3 Analysis of Post processing Minutiae Table

Post processing of minutiae is used to eliminate false minutiae structures occurred due to spurs, ride breaks, short ridge, holes or islands, bridges, and ladders. The postprocessing Minutiae Table algorithm stores minutiae table pixels on Final minutiae table if it is only valid ending or bifurcation, after verifying against all spurious minutiae. The process of elimination or deletions of spurious minutiae are explained below [24-25]. The gray color cell represents connected edge with respect to candidate minutiae pixel, which is in the centre of the window represented in red color. Table 3 shows total number of ridge ending and ridge bifurcation pixels identified before and after post processing operation for sample images of FVC ongoing 2002 DB1_B benchmark dataset. From the Table 3, it is understood that post processing operation drastically reduces total number of pixels in final Minutiae Table. The total number of ridge bifurcation and ending pixel is depending on the structure of thinned fingerprint image. The time complexity of post processing minutiae table is Big-Oh (n).

Table 3: Total number of ridge ending and bifurcation pixels before and after post processing operation

S.No	Image Name	Total number of ridge ending and bifurcation pixels before post processing operation	Total number of ridge ending and bifurcation pixels after post processing operation
1	101_1	419	11
2	101_5	324	77
3	102_2	1015	55

4	103_3	623	32
5	104_4	450	52
6	104_7	811	582
7	104_8	488	02
8	105_8	829	67
9	106_6	693	54
10	109_3	819	18
11	109_8	879	26
12	110_3	666	18
13	110_8	900	23

6. Applications of Minutiae Table:

After obtaining Final Minutiae Table, through post-processing phase, next to these minutiae details are further converted into a form which is suitable or compatible to convert into hash code. In order to convert Hash code, we can use MD5 or SHA-256 Hash algorithm. These Hash code can uniquely identify a person or can act as index key or identity key. One of the application of Minutiae Table for Hash code generation and how that Hash code can act as one of the factors in Multifactor Authentication Model is explained below [10].

Initially on the client side using an interface user loads fingerprint image into the system. At first Finger image, foreground feature is extracted from the background using segmentation Later, using Gabor filtering fingerprint image features are extracted. These features are encrypted and sent to the server. As soon as these features arrive at the server in encrypted form, the server receives that and request for One Time Password from OTP generator. OTP generator is a module or function, which is located at server machine. Time synchronized OTP is sent to the registered mobile phone user. Client system prompts a message to enter OTP, which is received to the registered mobile phone of the user. The user enters that OTP through the client interface and this OTP is compared with server generated OTP at the server side. If OTP is verified, server requests for the password, the user enters the password through the client-side interface and entered password reaches to the server. The server verifies the user entered a password with the already stored password in its database. Since database password is stored in encrypted format. The password which is stored in the database in encrypted form and finger user-id hash code is encrypted one again to enhance security. So if an intruder gets stored hash codes from the database, still authentication cannot become successful. If both password and Fingerprint Hash code match then a user is considered as an authenticated user. In other words authentication process successfully completes when OTP, Password, and Fingerprint Hash code matches. If anyone out of Fingerprint Hash code or Password does not matches user is considered as a un-authorized user. If OTP not matches then the user is blocked from further steps in the authentication process. In this research study, this is not implemented as server and client in different machines.

7. Conclusion:

In Automatic Fingerprint Identification System (AFIS) preprocessing of a fingerprint play crucial role in enhancing the performance of matching and identification accuracy. After applying fingerprint image preprocessing on raw fingerprint, which includes filtering, enhancement, binarization, and segmentation thinned image or Skeletonization processes or techniques. Further skeleton or thinned image is preprocessed to remove white areas occupied by the noise. Again preprocessed thinned image is further post-processed to remove some false minutiae from minutiae table and which is generated through crossing number theory. In this paper, we have discussed preprocessing, feature extraction, and post-processing with its theory, algorithm, workflow diagram, and flowchart. The final minutiae table obtained after post-processing can be effectively used for generating Fingerprint Hash code, which can be used as index-or identity key in order to uniquely identify an individual person or as one factor in Multifactor Authentication Model.

8. References:

1. K. Krishna Prasad, & Aithal, P.S., “A Critical Study on Fingerprint Image Sensing and Acquisition Technology,” International Journal of Case Studies in Business, IT and Education (IJCSBE), 1(2), pp. 86-92, 2017. DOI: http://dx.doi.org/10.5281/zenodo.1130581
2. K. Krishna Prasad, & Aithal, P.S., “A Conceptual Study on Image Enhancement Techniques for Fingerprint Images,” International Journal of Applied Engineering and Management Letters (IJAEML), 1(1), pp. 63-72, 2017. DOI: http://dx.doi.org/10.5281/zenodo.831678
3. K. Krishna Prasad, & Aithal, P.S., “Literature Review on Fingerprint Level 1 and Level 2 Features Enhancement to Improve Quality of Image,” International Journal of Management, Technology, and Social Sciences (IJMTS), 2(2), pp. 8-19, 2017. DOI: http://dx.doi.org/10.5281/zenodo.835608
4. K. Krishna Prasad, & Aithal, P.S., “Fingerprint Image Segmentation: A Review of State of the Art Techniques,” International Journal of Management, Technology, and Social Sciences (IJMTS), 2(2), pp. 28-39, 2017. DOI: http://dx.doi.org/10.5281/zenodo.848191
5. K. Krishna Prasad, & Aithal, P.S., “A Novel Method to Contrast Dominating Gray Levels during Image contrast Adjustment using Modified Histogram Equalization,” International Journal of Applied

Engineering and Management Letters (IJAEML), 1(2), pp. 27-39, 2017. DOI: http://dx.doi.org/10.5281/zenodo.896653

6. K. Krishna Prasad, & Aithal, P.S., "Two Dimensional Clipping Based Segmentation Algorithm for Grayscale Fingerprint Images," International Journal of Applied Engineering and Management Letters (IJAEML), 1(2), pp. 51-65, 2017. DOI: http://dx.doi.org/10.5281/zenodo.1037627.
7. K. Krishna Prasad, & Aithal, P.S., "A conceptual Study on Fingerprint Thinning Process based on Edge Prediction," International Journal of Applied Engineering and Management Letters (IJAEML), 1(2), pp. 98-111, 2017. DOI: http://dx.doi.org/10.5281/zenodo.1067110
8. K. Krishna Prasad, & Aithal, P.S., "A Study on Fingerprint Hash Code Generation using Euclidean Distance for Identifying a User," International Journal of Management, Technology, and Social Sciences (IJMTS), 2(2), pp. 116-126, 2017. DOI: http://doi.org/10.5281/zenodo.1133545
9. K. Krishna Prasad, & Aithal, P.S., "An Alternative Approach to Fingerprint Hash Code Generation based on Modified Filtering Techniques," International Journal of Innovative Research in Management, Engineering and Technology, 2(12), pp. 1-13, 2017. DOI: IJIRMET1602012001.
10. K. Krishna Prasad, & Aithal, P.S., "A Study on Multifactor Authentication Model Using Fingerprint Hash Code, Password and OTP," International Journal of Advanced Trends in Engineering and Technology, 3(1), pp. 1-11, 2018. DOI: http://doi.org/10.5281/zenodo.1135255.
11. K. Krishna Prasad, & Aithal, P.S., "A Study on Fingerprint Hash Code Generation Based on MD5 Algorithm and Freeman Chain Code," International Journal of Computational Research and Development, 3(1), pp. 13-22, 2018. DOI : http://doi.org/10.5281/zenodo.1144555.
12. K. Krishna Prasad, & Aithal, P.S., "A Comparative Study on Fingerprint Hash Code, OTP, and Password based Multifactor Authentication Model with an Ideal System and Existing Systems," International Journal and Advanced Scientific Research, 3(1), pp. 18-32, 2018. DOI: http://doi.org/10.5281/zenodo.1149587.
13. V. Espinosa-Duro, "Mathematical Morphology approaches for fingerprint Thinning," In Proceedings of the IEEE 36th Annual 2002 International Carnahan Conference on Security Technology, pp. 43-45, 2002.
14. M. Ahmed, & R. Ward, "A rotation invariant rule-based thinning algorithm for character recognition," In Proceedings of the IEEE Transactions on Pattern Analysis and Machine Intelligence, 24(12), pp.1672-1678, 2002.
15. P. M Patil, S. R. Suralkar, & F. B. Sheikh, "Rotation invariant thinning algorithm to detect ridge bifurcations for fingerprint identification," In Proceedings of the IEEE 17th International Conference on In Tools with Artificial Intelligence, pp. 8, 2005.
16. X. You, B. Fang, V. Y. Y. Tang, and J. Huang, "Multiscale approach for thinning ridges of fingerprint", in Proc. Second Iberian Conference on Pattern Recognition and Image Analysis, volume LNCS 3523, pp. 505–512, 2005.
17. E. Newham, "The biometric report," SJB services, 733, 1995.
18. A. A. Moenssens, Fingerprint techniques. Chilton, 1975.
19. T. Y. Zhang, & C. Y. Suen, "A fast parallel algorithm for thinning digital patterns," Communications of the ACM, 27(3), pp. 236-239. 1985.
20. C. Arcelli, & G. S. Di Baja, "A width-independent fast thinning algorithm," In Proceedings of the IEEE Transactions on Pattern Analysis and Machine Intelligence, (4), pp. 463-474, 1985.
21. S. Kasaei, M. Deriche, & B. Boashash, "Fingerprint feature extraction using block-direction on reconstructed images". In Proceedings of the IEEE TENCON'97 Region 10 Annual Conference on Speech and Image Technologies for Computing and Telecommunications, 1, pp. 303-306, 1997.
22. D. Maltoni, D. Maio, A. K., Jain, & S. Prabhakar, "Handbook of Fingerprint Recognition," Annals of Physics. 54, 2003.
23. M. U. Akram, A. Tariq, S. A. Khan, & S. Nasir, "Fingerprint image: pre-and post-processing," International Journal of Biometrics, 1(1), pp. 63-80, 2008.
24. S. Tabbone, & L. Wendling, "Multi-scale binarization of images," Pattern Recognition Letters, 24(1-3), pp. 403-411, 2003.
25. J. Yang, L. Liu, and Y. Fan, "A modified Gabor filter design method for fingerprint image enhancement", Pattern Recognition Letter, 24(12), pp.1805–1817, 2003.