System of “Analysis of Intersections Paths” for Signature Recognition

Farhad Shamsfakhr *farhad_sm@ymail.com*
Faculty of Engineering/Computer Engineering Department
Islamic Azad University/Hamedan Branch
Hamedan, 65138-734, Iran

Abstract

In today's world, the electronic city which is the offspring of the development of the information world, paves the way for a round-the-clock interaction among computers and networks. The planners of these electronic cities are mostly concerned about the accuracy and security of the exchanged information. In order to elevate security and raise speed and accuracy in the reviewing of network performance and the dependable identification of persons involved in electronic operations, recognizing the accuracy of the electronic signature is deemed absolutely essential. In this article, a system named "Analysis of Intersections" has been utilized for the accurate recognition of the electronic signature. Of important features of this system are the utilization of simple data structures such as array, stack, and list and determination of the sensitivity level for recognizing the accuracy of the signature by setting an error percentage for the size and recognition of the shape. An accuracy recognition test was performed on 15 samples of 150 types of signatures using "Analysis of Intersections". Findings indicated that this system showed an accurate recognition of 2,220 out of 2,250 signatures, indicating an applicability of 98.66 percent.

Keywords: Analysis of Intersections, Intersection, Threshold, Rotational Routing Algorithm, Intersection Recognition Samples, Adaptation of Paths.

1. INTRODUCTION

With the ever increasing advancement of the information technology over the recent years, electronic operations have gained momentum. Meanwhile, dissemination of personal and organizational information over the insecure worldwide web and the easy access of individuals and organizations to internet resources raise concerns regarding the unauthorized access of strangers to users personal information and the breaching of privacies. To the end of safeguarding the privacy of their users, planners of electronic cities have taken measures to define an identity for their users. This could happen in the form of defining a username and a password for the users or identifying unique characteristics of individuals such as fingerprint, face recognition, etc. Question is which method is the fastest, most precise, and most cost-effective of all? In fact, in order to increase the accuracy and precision of information on the one hand and the speed of electronic operation completion on the other, it is advisable that a sensible, optimum, intelligent, and easy-to-use method be created. A proper option for securing the entrance of users into the web is the accurate and intelligent identification of individuals signatures. Using intelligent systems, not only the speed of signature recognition but also its accuracy are augmented which is crucial when performing important monetary, information, and security operations. [2] The method introduced in this article concerns offline electronic signatures. In electronic signatures, there is no noise or halo of colors. In this study, we attempt to analyze the signature image as a collection of ways and introduce the main algorithms as follows: 1) way-finding algorithm shown as "Around _ perceive (x,y)" and "Way _ Finder (x,y,z)" functions; 2) intersection recognition algorithm shown as "Inter _ Section (x,y)" function; 3) end-checking algorithm shown as "sensing _ opr _ End _ checker"; and 4) a unique algorithm for comparing and adapting samples with other algorithms that are explained later. We refer to this system as "Analysis of Intersections". [3] By using electronic signatures, we do not need to take pre-processing operations such as elimination of noises. In some systems, however, the image of the signature has a halo of colors instead of the main signature color. In such a case, a pre-processing operation is needed to transform the halo of colors to only one color (e.g. black). After receiving the signature in the form of a bitmap file, we separate the signature from the background; change the color of the background to white and color of the signature to black.

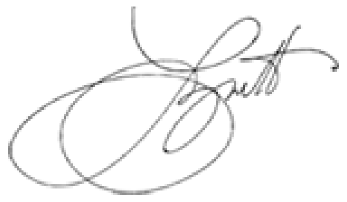


FIGURE 1: shows the main image of the signature.

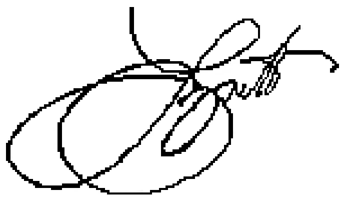


FIGURE 2: the pre-processed image.

The program starts with retrieving "Image _ Analysis ()" function. This function navigates the array of the image to find the first pixel of the signature image (black pixel). After finding the first pixel, "Around _ Perceive (x,y)" function is retrieved. The task of this function is to search the environment around the received pixel in order to find the path it is going to navigate. It finds the right path, navigates it, and continues this action until it reaches the first intersection. The design of this function does not lead to registration of repeated paths and being trapped in loops known as "wrong paths". We take the pixel signified in the following figure to be the first pixel of the signature image which has been found by "Image _ Analysis () "function.

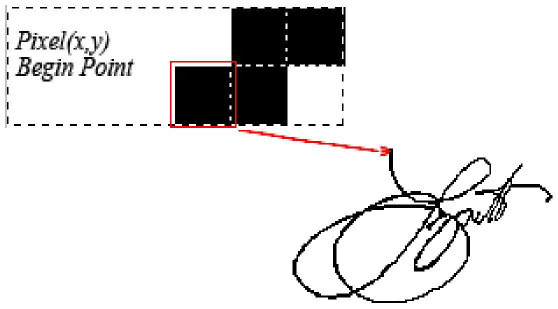


FIGURE 3: shows the magnified image of the signature.

After receiving x and y of the first pixel as the input parameters of "Around _ Perceive (x,y)" function, this function reviews 8 pixels around the current pixel in order to find one pixel to continue the path. This way the path is navigated and this continues until the first intersection in the signature image is reached. Further explanations in this regard are given in the Appendix. By this function, after finding one pixel as the found path, we must study the position of the pixel in every stage in order to reach the intersection. To this end, we retrieve "Inter _ Section" function. This function tries to receive the path pixel and decide whether it is in the intersection or not. If yes, the intersection is found and "Around _ Perceive" function has completed its job successfully. If not, "Around _ Perceive" is retrieved through the same pixel.

2. INTERSECTION

[4] Intersection is defined as the pixel where at least 4 different paths meet. In Inter _ Section function, various figures of the intersection, known as "intersection recognition samples" exist. [1] Comparing the position of the path pixel (the input pixel of Inter _ Section function) with the pixels around indicates whether it can be candidate as the intersection center or not.

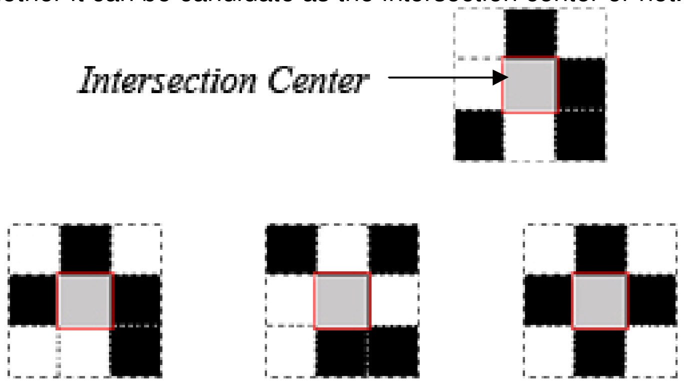


FIGURE 4: shows some of the intersection recognition samples.

2.1 Intersection Recognition

[5] After a pixel is candidate as the centre of intersection, the function decides whether this pixel is a definite center or just candidate as one. Pixel A in Figure 5 is just a candidate center of intersection as it is not the intersection of at least 4 distinct paths and is just a path pixel. In Figure 6, however, pixel A is a definite center as it is the intersection of at least 4 distinct paths. If a pixel is the intersection of 3 distinct paths, that pixel is known as the center of a three-way junction which after being added a virtual path, may become a definite intersection center. To decide whether an intersection center is definite or not, "Inter _ Section" function uses an algorithm known as Rotational Routing algorithm described fully in the Appendix.

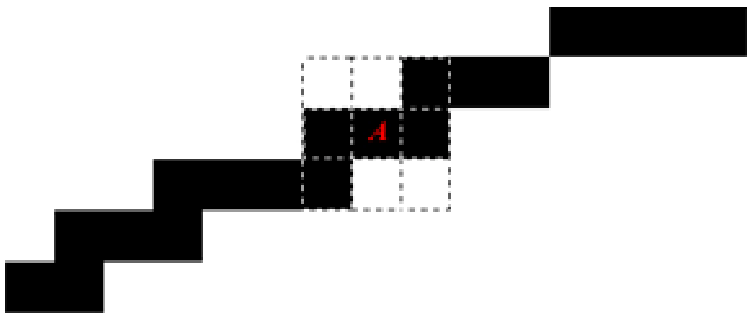


FIGURE 5: shows the candidate intersection center.

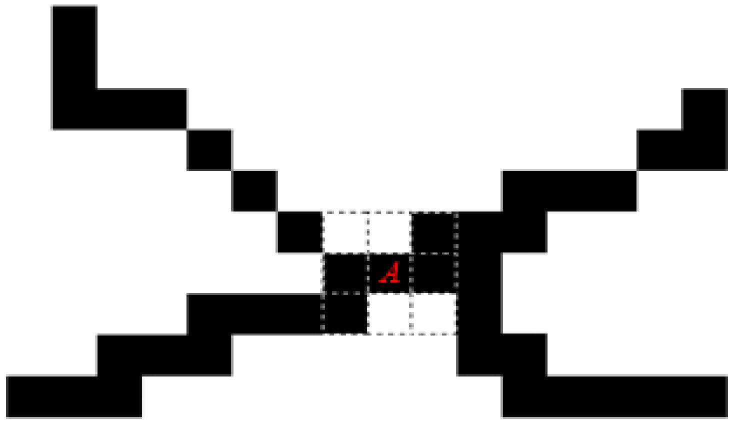


FIGURE 6: depicts the definite one.

After recognizing the intersection, all paths ending to the intersection are registered in the array temporarily, with the direction of every path and then the paths are navigated. Navigating these paths is done by "Way _ Finder (x,y,z)" function. As said before, every path has a direction. In order to optimize searches, the method of navigating each path is based on the direction of that path. Therefore, for navigating two paths with two different directions, we use two different methods, described fully in the Appendix. Navigating the path continues to the end of the path or until meeting an intersection. If the path is not repeated, it is registered in the array. "Way _ Finder (x,y,z)" function reviews the position of the pixel in every stage of navigation in order to reach the end of the path. To find out whether this path is terminated or not, "sensing _ opr _ End _ checker" function is used. This function receives a pixel from "Way _ Finder" function and decides whether the current path has ended or not. If the path has not ended, the said function will retrieve "Way _ Finder" function again. Therefore, all the paths ending to the first intersection in the signature are registered in the array. Then, all the paths of this intersection are eliminated from the signature image and all the stages of the operation are repeated until no non-navigated path remains in the signature image. If a path does not end to an intersection, it will be called a separated path. Such paths are eliminated without being registered in the array. Figure 8 shows the signature analysis stages of Figure 7. In every stage of signature processing, navigated paths of an intersection are shown. The signature image after one stage of processing is also depicted. The red path is a virtual path that we add to three-way junctions so that they will be turned to intersections.

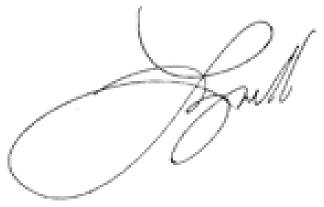


FIGURE 7: shows a signature sample.

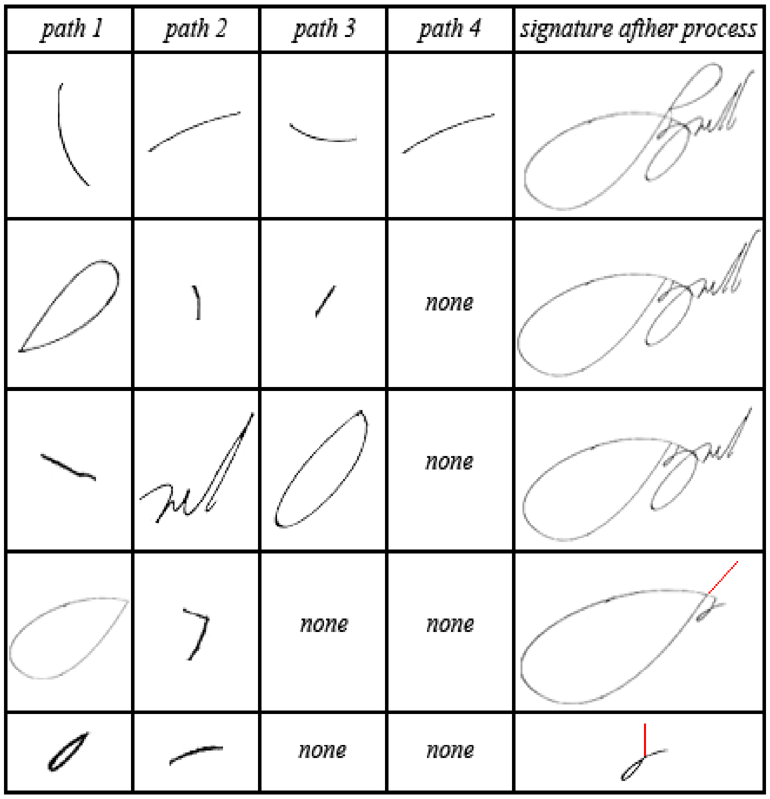


FIGURE 8: shows the stages of processing the paths of that signature along with processed paths in every stage and the image of the signature after being processed in each stage.

3. Adaptation of Paths

After storing all pixels in every path in the array, we have to use these paths to identify the signature. In order to identify the signature, first "Path Information Table" must be formed for each of the paths of the signature. This Table includes: 1) starting point: containing the features of the first pixel of every path; 2) ending point: containing the features of the last pixel of every path; 3) x range: containing the range of x's in every path; 4) y range Dx=Max(x)-Min(x): containing the range of y's in every path; and 5) the whole path Dy=Max(y)-Min(y): containing the number of pixels in every path. "Path _ Receipt _ Data _ Process" function creates the Path Information Table, fully described in the Appendix. After such table is created for all paths, we have to decide on the shape, direction, size & the slope measurement of every path based on the information contained in this table as follows: first a Measure is considered for every signature. Measure is a unit that measures the length of the path. For instance, if the length of a path is 30 pixels and its measure is 3, the length of this path will be 10 measures.

Measure = Path Length / n

By using Measure, size will no more be a limiting factor; therefore, any size signature can easily be recognized. We consider that Harry Hilton signed his signature in two different sizes.

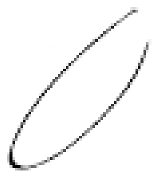


FIGURE 9: shows a path of his first signature.

3.1) length path = 180 pixel , Measure = 180 \ 10=18 , length path=180\18=10 unite and

FIGURE 10: a path of his second signature.

3.2) length path = 78 pixel , Measure = 78 \ 10=7.8 , length path=78\7.8=10 unite The lengths of both paths are 10 units.

Threshold

Threshold is the amount we use as the error limit. If in the rectangular in Figure 11 the threshold is 4, the rectangular will be complete if we ignore 4 pixels (error limit=4).

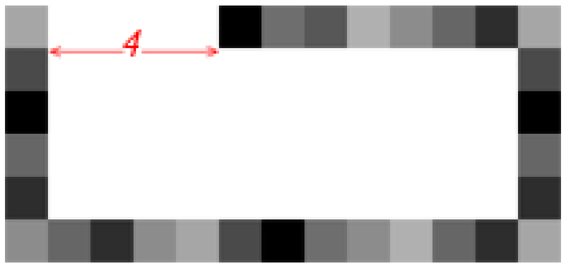


FIGURE 11: the threshold is 4, the rectangular will be complete if we ignore 4 pixels (error limit=4).

In this article, we calculate Span _ Limit = Path Length /7. A different calculation method can also be used. The smaller the Span _ Limit, we will identify shapes with smaller error compared with real shapes. The bigger the Span _ Limit, bigger adaptation errors are ignored. The slope of each path are calculated as follows:
Y=mx+b , m = dy/dx = (y2-y1)/(x2-x1)
We apply our own definition for the shape, direction, and size of images. For example, Figure 12 in the signature is called a big quasi-rectangular which is shown in Figure 12.

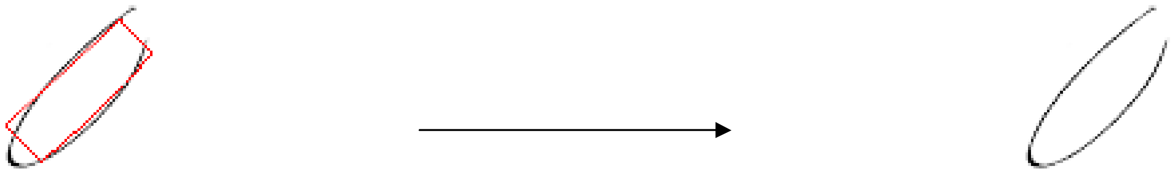


FIGURE 12: big quasi-rectangular.

Our definition of shapes and directions is given in the Appendix.
The shape, direction, slope and size of a path are referred to as the features of a path. In order to identify the accuracy of the signature, the following operation is taken: The number of paths in signature A is compared with the number of paths in signature B. If they are equal, the features of every path in one signature are compared with those of the corresponding path in the other signature. In case features of the paths in both signatures are identical, the accuracy of the signature is vindicated. In order to improve the identification performance, a certain percentage of size differences must be ignored. For instance, if the features of every single path in one signature is exactly identical with those of the corresponding path in the other signature, but there is discrepancy in the size or the length of paths by 1 or 2 units, we may consider the length of

paths as identical after ignoring 2 units of discrepancy. An accuracy recognition test was performed on 15 samples of 150 types of signatures using "Analysis of Intersections". Findings indicated that this system showed an accurate recognition of 2,220 out of total 2,250 signatures, indicating an applicability of 98.66 percent.

Authors	Method	Results
Ammar, M. 1991	Distance Threshold	85.94%
Ammar et al., 1990	Distance Statistics	88.15%
Quek & Zhou, 2002	Neuro-Fuzzy Network	96%

TABLE 2: Comparative Analysis of different off-line signature verification systems.

4. REFERENCES

[1] S. E. Umbaugh, Computer Imaging Digital Image Analysis and Processing, CRC Press, 2005

[2] M.Savov, G.Gluhchev, Automated Signature Detection from Hand Movement, "International Conference on Computer Systems and Technologies _ CompSysTech" 2004

[3] A.Zimmer, L.Luan Ling, A Hybrid On/Off Line Handwritten Signature Verification System, "Proceedings of the Seventh International Conference on Document Analysis and Recognition
(ICDAR 2003) " 0-7695-1960-1/03 17.00 C 2003 IEEE

[4] A recognition algorithm for the intersection graphs of paths in trees . Computer Science Division, Department of Mathematical Sciences, Tel-Aviv University, Ramat-Aviv, Tel-Aviv, Israel Received 10 March 1977;

[5] A recognition algorithm for the intersection graphs of directed paths in directed trees . Department of Computer Science, University of Illinois, Urbana, Ill. 61801, USA

Annex

1. "Around Perceive (X,Y)" function

The task of this function is to search around the pixel received by itself to find the right route at its own front and to keep sensing the same until it comes to the first intersection . this effect the eight pixels around the current pixel are surveyed and the first black pixel so found shall be considered as the conducted pixel of continuation of the route. The adjacent pixels are surveyed line by line beginning from the first line and the first column to the last line and the last column in the following way.

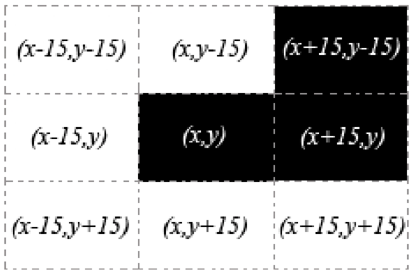


FIGURE 1: pixels around the current pixel(* Now suppose that each Pixel measures 15x15 in dimensions.)

```
For  j = y - 15    To    y + 15    Step 15
For   i = x - 15    To x + 15    Step 15
    If    i <> x    Or    j <> y    Then
        color = pic3.Point(i, j)
        If color = 0 Then
            {
            .
            .
            .
            }
```

The current pixel that is the input of the around perceive (X,Y) function with in each pause is refereed to as parent pixel and the candidate pixels of the same phase are culled child pixels. In case a candidate pixel is taken as a route, its parent pixel shall be reserved in an array named pixels Memory. After the candidate pixel has been discovered within each please, it is this surveyed from the viewpoint of being a repeated pixel where we may face two alternative circumstances; 1st: there is a minimum of one (1) unrepeated candidate pixel. Absence of candidate pixel in the array of pixels _Memory would indicate that the route has been discovered and that it is the route of candidate pixel. Then, it is considered if its parent pixel exists in the array parent pixel shall be reserved in the array, it is reserved in the array 2nd. All candidate pixels are repeated pixels. In case all candidate pixels are present in pixels Memory array, this means that the route has not been discovered in which case a stack shall be used. With this state also the parent pixel shall be reserved in the array provided that it is not a repeated pixel. Lack of discovery of a route indicates that the route we have covered to this point is wrong and that we should go back to take another way. Pointer points at the end of the array pixels Memory. Therefore, we are to reduce the pointer by one unit per phase and consider as new candidate pixels the child pixels pointed at by the pointer continuing this to the point where a route appears. We convert the array of pixels Memory to stack in fact by using the pointer and retrieve the function by the values available in the stack in the following method.

Around_Perceive(Pixels_memory(pointer).x, Pixels_memory(pointer).y)

2. Intersection Identification Function

How many patterns are required to identify the intersection? Is it possible for us to identify intersection candidate in the form of signature by using these patterns? It may appear initially that to improve intersection identification performance we are to create all combinations of N black pixels in M-1 locations and in other words we should have 2520 different intersection identification patterns:

$$C(8,4)=\frac{8!}{4!(8-4)!}=2520$$

It is evident that it shall be extremely time- consuming to create this number of patterns and it is a useless effort. We don't need to do this; there is no requirement of taking as a pattern any four of eight combination. We take as a model only the states more frequently occurring in the intersection of a signature and those states do not exceed 40 in number. Whereas one compares to intersection patterns any singles pixel adjacent to the input pixel of the function inter section, then any pixel conforming to none of our patterns shall be left and shall give its place to an adjacent pixel in an order of comparison. As an instance let us suppose that the pattern 1 has not been defined in the function; that pattern 2 has been defined in the functional and that we find the following form in the signature. It is evident that we cannot identity the position of pixel A in the intersection be cause we have not defined pattern 1 in the function. However, me may identity the position of pixel B in the intersection as pattern 2 has been defined in the function.

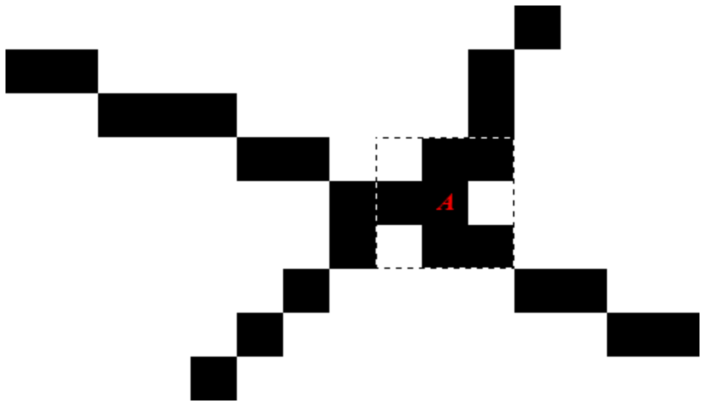


FIGURE 2: position of pixel A in the intersection

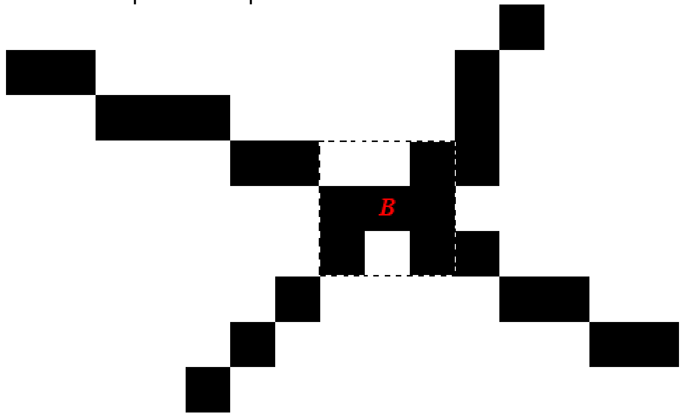


FIGURE 3: position of pixel B in the intersection

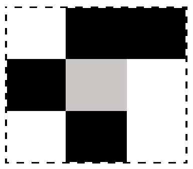


FIGURE 4: pattern1

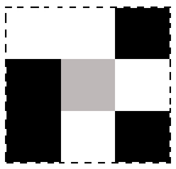


FIGURE 5: pattern2

As observed, the intersection was finally identified with the definition of one pattern.

3. ROTATIONAL ROUTING

To determine if a pixel can be the center of a definite intersection, the function inter section takes use of a method which we call Rotational routing. Because of congeries of pixels at the intersections,

For recognizing definite intersection, the distincgtion action of the directions would be perform in the place in which, the route is released of the pixel congestion.

all the possible routes on the eight sides of the intersection are counted clockwise after the intersection center candidate has been identified within the intentional routing method. Keep in mind that the starting point of the aforesaid routes shall be considered to be at a clearance by three (3) pixels from the intersection center so that a separation in made from the point of congeries of pixels at the intersection center. The rotational routing method at a depth of 3 is considered here. However, a rotational routing at a depth of 4 may be used as an alternative with the advantages of bringing greater flexibility and preventing errors.

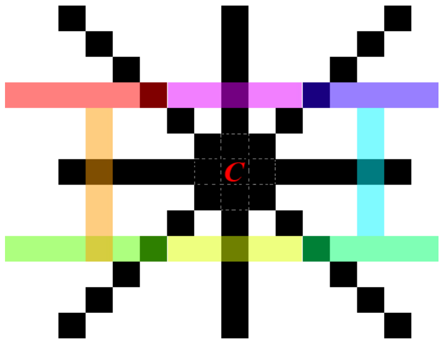


FIGURE 6: Rotational Routing at a depth of 3.

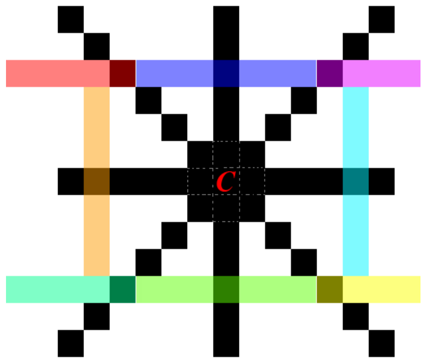


FIGURE 7: Rotational Routing at a depth of 4.

The following figure illustrates method of functioning of rotational routing at a depth of 3.

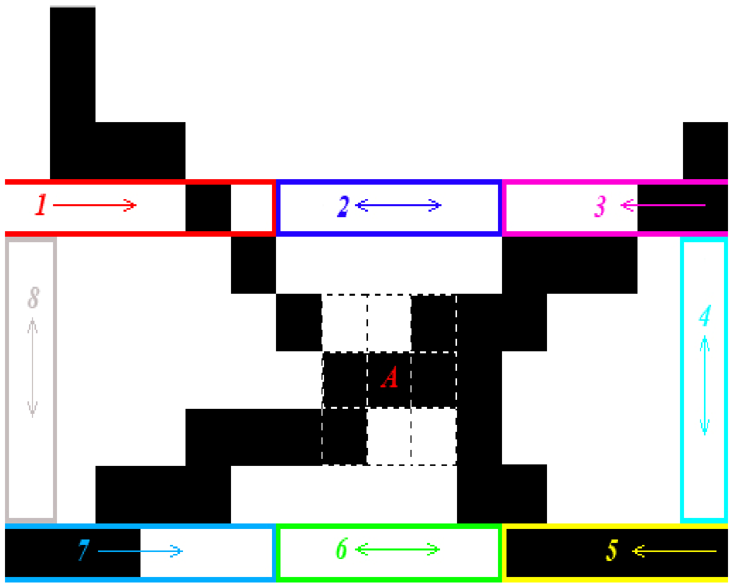


FIGURE 8: functioning of rotational routing at a depth of 3.

(X,Y) is intersection center coordinate of the pixel dimensions are 15x5 in this example). A is the center of intersection. The searches 1,2,3,5,6 and 7 are made at 3 pixels from the intersection center (x-45) or (y-45). The searches 4 and 8 are made at an optional clearance from intersection center. However, the clearance shouldn't be so little that it is included within the area of congeries of pixels and not so large that make the length of the route so discovered excessively small. The search is in every case made at the beginning of the search area and continued to the end. The search is terminated upon reaching the first pixel. The pixel so encountered is recorded in the array as the first pixel of the discovered route. Each intersection has a maximum of eight routes and hence eight searches shall be required to discover the eight cutes. The search no 1 is dedicated to the discovery of the route No.1 (northwestern route). The length of search route includes the distance between an optional clearance, say X-225 as an instances to specified clearance e.g, X-45. The search No. 2 is dedicated to the discovery of the route No.2 (northern route). The length of the search route includes to distance between X-30 to X+30. search No.3 is dedicated to the route No.3 (northeastern route). The length of search route is the distance between an optional clearance, say X+225 to the specified clearance X+45. the search no 4 is dedicated to the discovery of route No.4 (eastern route). The length of search route includes to distance between Y-30 and Y+30 The search no 5 is dedicated to the route No.5 (southeastern route). The length of the search route is the distance between an optional charana say X+225 to the known clearance X+45. The search No. 6 is dedicated to the route No.6 (southern route). The length of search route covers the distance from X-30 to x+30. The search No. 7 is dedicated to discovery of route No. 7 (southwestern route) the length of search route is the distance from an optional clearance, say X-225 to the specified clearance X-45. Search No. 8 is dedicated to the discovery of route No. 8 (western route). The length of search route covers the distance from Y-30 to Y+30. Therefore, all the routes, their respective numbers and their directions are reserved in an array namely Inter Section Ways Array by using rotational routing algorithm. In case number of the routes is equal to exceeds 4, then the intersection candidate pixel is an absolute intersection and otherwise the function Around_ Perceive is retrieved by using a candidate pixel and the remaining part of the route is measured until the intersection appears again.

4. OPTIMIZATION OF SEARCHES

Any pixel to be searched by Way_ Finder function for the continuation of the route, should have a specific search method of its own. Uniform order of searches dedicated to all roles shall result in an increase in number of comparisons, emergence of errors and a drop in processing rate of the program. Suppose that current pixel is the pixel A and the purpose is to find pixel B for the continuation of the route.

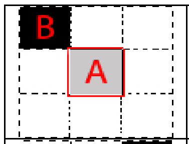
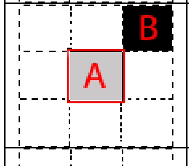
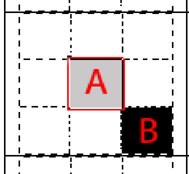
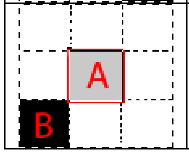
	For j = y – 15 To y + 15 Step 15 For i = x - 15 To x + 15 Step 15
	For j = y – 15 To y + 15 Step 15 For i = x + 15 To x - 15 Step -15
	For j = y + 15 To y - 15 Step -15 For i = x - 15 To x + 15 Step 15
	For j = y + 15 To y - 15 Step -15 For i = x + 15 To x - 15 Step -15

FIGURE 9: every route along with the proper search orders (dimensions of each pixel measure 15x15)

5. RULES OF FORM & ROUTE DIRECTION IDENTIFICATION

The following rules are those of form and route direction identification. As observed , form and route identification rule , name the name of route form, an example of the same and the direction of the respective route have all been given in each line.

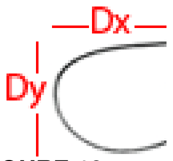


FIGURE 10: sample

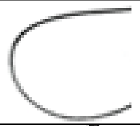
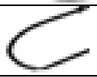
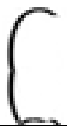
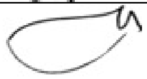
Law	Shape	Sample	Direction
1.1) If Abs(DX - DY) < (Span_Limit)	Half square		horizontal
1.2) If ((DX - DY) >= (Span_Limit)) & (DX <= 2 * DY)	Half rectangular		horizontal
1.3) If (DX > Span_Limit) & ((DX + Span_Limit) <= (DY))	small Half square		horizontal
1.4) If (DX > 2 * DY)	big Half rectangular		horizontal

TABLE 1: Horizontal forms Identification rules. (1.) If Abs(yn - y1)> Span_Limit & Abs(xn - x1)<= Span_Limit)

In case name of the aforesaid rules applies to a form, then the following rules shall be surveyed:

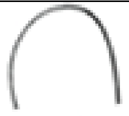
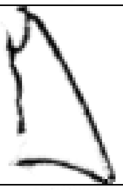
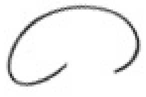
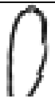
Law	Shape	Sample	Direction
2.1) If Abs(DX - DY) < (Span_Limit)	Half square		vertical
2.2) If ((DY - DX) >= (Span_Limit)) & (DY <= 2 * DX)	Half rectangular		vertical
2.3) If (DY>Span_Limit) & ((DY + Span_Limit) <= (DX))	small Half square		vertical
2.4) If (DY > 2 * DX)	big Half rectangular		vertical

TABLE 2: Vertical forms identification rules .(2.) If Abs(yn - y1)<= Span_Limit & Abs(xn - x1)> Span_Limit)

In case none of the afore rules applies to a form, then the following rules shall be considered.

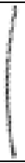
Law	shape	The Sample	Direction
3) If (DY <= Span_Limit)	Horizontal line		horizontal
4) If (DX <= Span_Limit)	Vertical line		vertical

TABLE 3: Secondary rules.