A Novel Approach for Extraction of Polygon Regions

GJCST Computing Classification
I 5.1,I 5.4

Kemal Yüksek[1] Metin Turan[2]

***Abstract*-This paper presents a new algorithm to find out whether a polygon exists around a reference point given within the graphical domain. The algorithm is based on creating discrete line segments and then searching them using the orientations formed at segments intersections. The computational complexity of the searching algorithm has been determined as O(n^2)**

I. Introduction

One of the most important problems that must be solved when developing graphical based system is the determination of simple graphical objects (such as line, circle, etc.) within the drawing area. Although polygons are known as simple graphical elements, they are composed of more simple graphical objects as line segments. In order to understand all the properties of a polygon, one must know the each line segment of that object [1]. There are widely used methods finding line segments of each of the polygon inside the whole structure [2]. Polygonal objects can be classified as convex or concave in shape according to the connection of the points inside of them. Most of the algorithms have been developed up to now especially deals with convex polygons [3]. So there are a number of algorithms to create convex polygons from concave ones [4]. Most of the polygon algorithms depend on shapes.

Many studies on computer vision and robotics are used a ray of a single flash light to detect the edges of polygon regions. There have been such algorithms using the flash light approach [5,6,7]. The following studies deal with vector and/or raster graphics. They follow certain algorithms to detect polygon boundaries. Polygons created with vector graphics have been used on various applications of manufacturing industries to Geographical

Information System (GIS) [8, 9, 10]. Raster based algorithms have been used in areas such as remote sensing to extract land cover data from satellite images and pattern recognition [11, 12]. The studies on polygons concentrate on the techniques such as computing the centre of a polygon region, calculation the area size and finding the centre of gravity of the polygon, whether a certain point is located within the polygon or not, intersection points of two polygons, area size (Hidden regions) of two overlapping polygons, defining the location of polygons and the triangulation of a simple polygon [13, 14, 15]. The proposed method in this study is based on an algorithm designed to determine the polygon region characteristics[16].

The initial stage of the algorithm has been inspired of the well-known scan line filling algorithm [17]. The developed algorithm has already been applied for calculating the heat requirement of a building project.

II. The Problem Definition and Proposed Solution

Searching a polygon around a reference point within the drawing domain is an important issue in computer graphics. The problem may become more complex if the assumption about the type (convex or concave) of the polygon is ignored. The following part of the study has been organized to construct the requirements of the proposed method first, and then algorithm itself.

Polygon boundaries are constituted by line segments which are basic design elements. These line segments could be defined by a user at random, or the result of a straight line recognition algorithms on an image.

One instinctively may think that it is a very trivial problem. If all the cases are considered, the problem becomes more complex to solve. Especially the polygons in the drawing area may be concave or convex in shape. So, the algorithm must should solve for both of them.

The method in this article consists of two steps. The first step is to create individual discrete line elements which are separated at the intersection points. The second step is to distinguish those line segments that construct the polygon region around a given reference point.

In the process of creating discrete line segments, it would be necessary to use mathematical representation of line segments. Although there are two ways for the representation; analytical one and parametrical one. The first one suffers from the intersection testing [18]. In order to avoid this shortcoming, parametrical representation of line segments has been preferred. In this representation, the start and end points of two line elements P1, P2 and P3, P4 can be defined as follow

$P_i = (x_i , y_i)$, i=1,2,3,4;

$$P(s)=P_1+(P_2-P_1)s \quad 0 \le s \le 1 \qquad (3)$$

$$P(t)=P_3+(P_4-P_3)t \quad 0 \le t \le 1 \qquad (1)$$

As a result of this representation, the intersection of the two line elements Z = (x_z,y_z) is found from the following equation:

$$P(s)=P(t) \qquad (2)$$

About-[1] İstanbul Kültür University, Istanbul, Turkey
(e-mail-k.yuksek@iku.edu.tr)

In this method, the user can select any line element at random, and then, this line element is checked against all the others in the graphical database to test if it intersects any of them. If there are intersection points, then, the line elements at each intersection points are broken down as individual discrete line elements on the basis of this intersection point. Then the same process is repeated for the other intersection points. As a result of this process, all the line elements in the graphical database would be in the form of discrete line segments. Finally, the intersection points become vertices of the polygons in the system if there are any.

Figure 1 illustrates the process of how to create line segments. In case of a discrete line drawn at random intersects with other discrete lines in the graphical database. Figure 2 shows the possible segmentations.

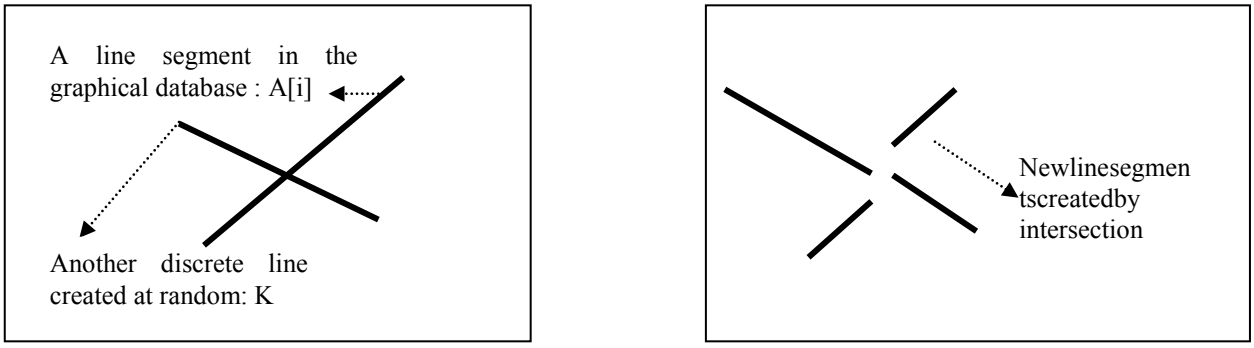


Fig. 1. The process of creating discrete line elements

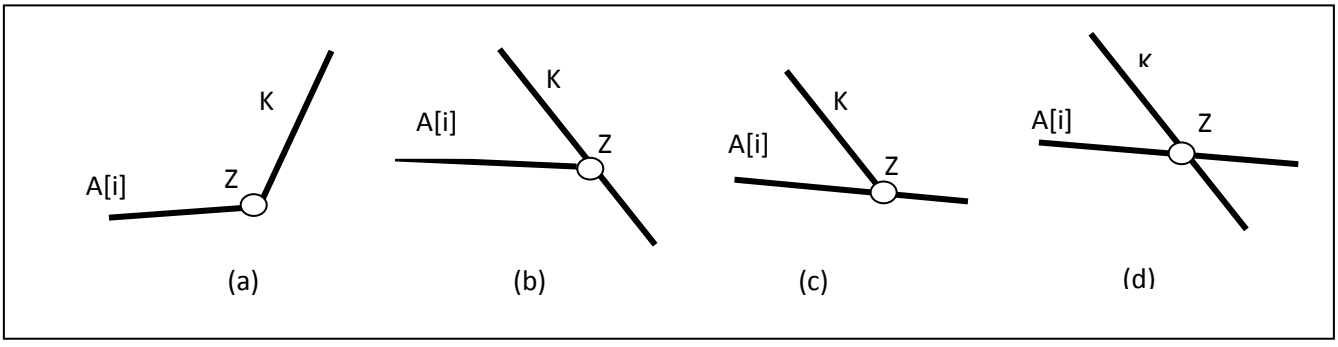
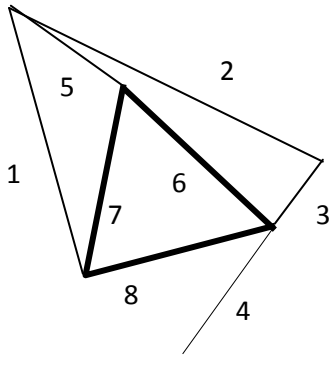


Fig. 2. The cases of intersection of two discrete lines in the graphical database

A new entry is inserted in adjacent matrix for each new line segment created by intersection. Adjacent line segments are stored in this bit matrix. Figure 3 shows an example of adjacent matrix. It is used to search the possible candidates for the next coming line segment on the polygon which accelerates the SearchPoly algorithm



	1	2	3	4	5	6	7	8
1	0	1	0	0	1	0	1	1
2	1	0	1	0	1	0	0	0
3	0	1	0	1	0	1	0	1
4	0	0	1	0	0	1	0	1
5	1	1	0	0	0	1	1	0
6	0	0	1	1	1	0	1	1
7	1	0	0	0	1	1	0	1
8	1	0	1	1	0	1	1	0

Fig. 3. A simple polygonal region and related adjacent matrix

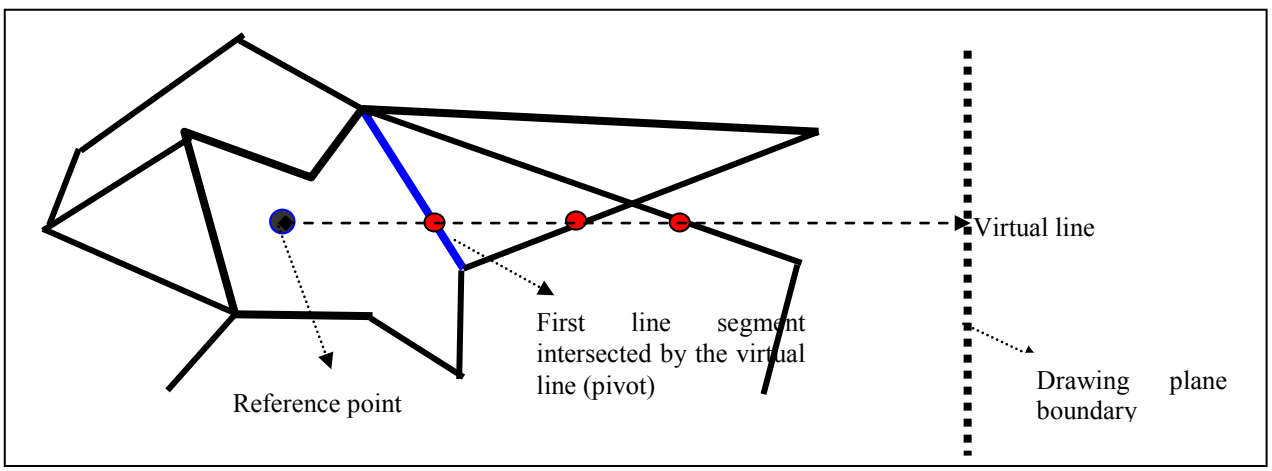
The pseudo-code for creating line segments is given below:

n : the number of the line segments in the graphical database
A[i]: the set of the line segments in the system (i=1,2,...n)
A[i].first :one end of the line segment A[i]
A[i].second :the other end of the line segment A[i]
K: current line which is just drawn by the user (selected line)
K.first: one end of the K
K.second: the other end of the K
Z: intersection point

```
Procedure createSegments( K )
{
  For each line segment A[i], i=1,..,n
  {
     If A[i] line segment intersects with discrete K line at a point Z
          If Z is start or end point of A[i] {
               If Z is start or end point of K then save K as line segment
               Else {divide K into two separate line segments
                         A[n+1]=(K.first,Z), A[n+2]=(Z,K.second)
                         update adjacent matrix}
          }
          Else {
               If Z is start or end point of K then
               {divide A[i] into two separate line segments
                A[i]=(A[i].first,Z), A[n+1]=(Z,A[i].second)
                update adjacent matrix}
               Else {divide K and A[i] into two separate line segments
                         A[i]=(A[i].first,Z), A[n+1]=(Z,A[i].second)
                         A[n+2]=(K.first,Z), A[n+3]=(Z,K.second)
                         update adjacent matrix}
```

Finally, the model is ready to check a given reference point if there is a polygon region around it or not. The second step is to take a reference point. Once the reference point has been selected, a vertical virtual line from the reference point is drawn to the right boundary of the drawing plane. The virtual line intersects some line segments and results a set of intersection points along the same line. The line segment, which has an intersection point that has the minimum distance to the reference point, has been selected. This is the first line segment of the polygon region.

Figure 3 illustrates the selection of the reference point, the virtual line and determining the first line segment of the polygon region. One end of the first line segment is taken as a basis (in other words base point) for executing the algorithm further. As there are two ends of the first line segment, obviously there are two alternative directions to go forward. Selection of the end point also defines the direction of the algorithm. Figure 4 shows decision of one end of the first line segment (in other words the direction of algorithm).



Once the algorithm direction has been defined, it remains same through the whole algorithm. After deciding the base point of line segment, the other line segments starting from or ending at the same point are selected from the graphical database. The angles created between the based line segment and the other selected line segments are calculated using opposite direction of the algorithm direction. For example, if the algorithm direction is clock-wise, the angles are evaluated in anti-clock-wise direction.

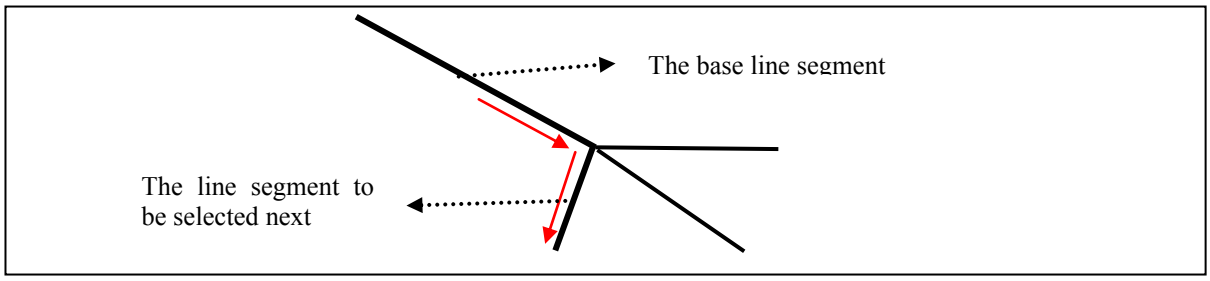


Fig. 4. Selection of algorithm direction

Figure 5 shows the direction of computation and the angles created by the line segments joining together at a base point. The line segment which gives the smallest angle at a base point is the new base line segment of this polygon region. And the same operations are performed repeatedly.

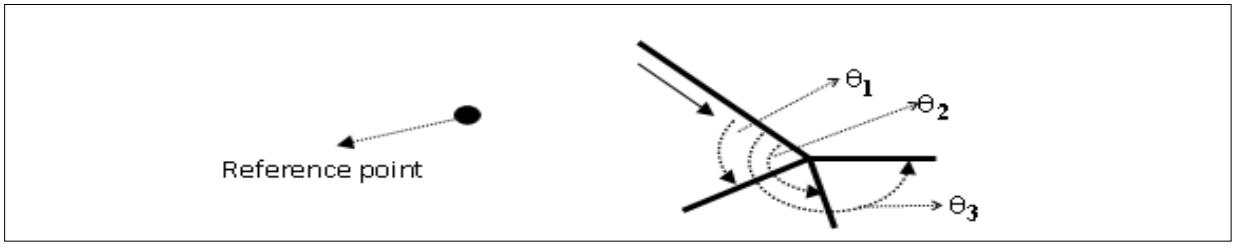


Fig. 5. Angular calculations at the base point (opposite to the algorithm direction)

The algorithm ends when the repeated operation reaches the first line segment or there isn't any line segment to go forward (selection of new base line segment).. If it reaches the first line segment, it means that all the line segments traced up to now construct the desired polygon. If it fails to find a next line segments, ,it shows that there is no closed polygon containing the starting point or there is an open door in the selected area. Both of these cases might have significant meanings depending on the application.

The pseudo-code for searching polygon region is given below. Moreover, Figure 6 illustrate execution of searchPoly(B) algorithm

BLS: base line segment
ALS: adjacent line segment
CLS: current line segment
BP : base point

boolean **Function searchPoly**(BLS , BP)
{
  Set BLS to CLS
  while (TRUE)
  {
    If (bit sum of the CLS row in the adjacent matrix is zero) return false
    /* One may eliminate (put zero) the columns in the adjacent matrix whose line segments searched before */
    Calculate the angles between the CLS and its adjacent line segments (use adjacent matrix)
    Choose the smallest ALS measured in the opposite direction to the algorithm direction
    Find new BP for selected ALS
    Save selected ALS into the line segments list of possible polygon
    If selected ALS joins with other end of BLS return true
    Set ALS to CLS
  }

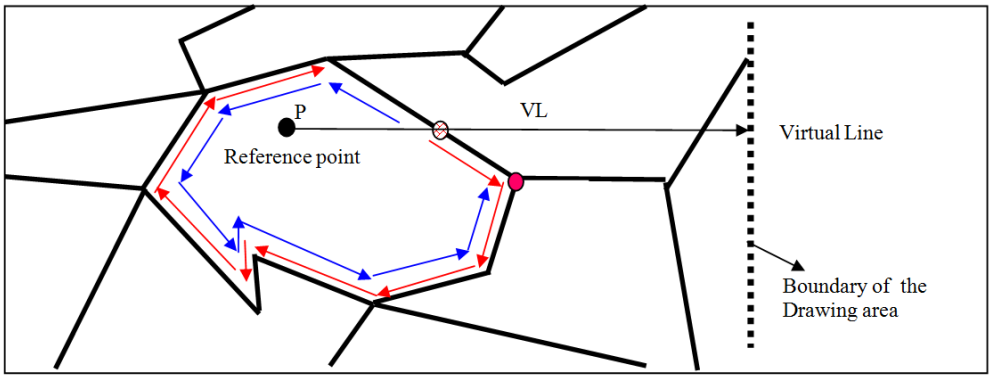


Fig. 6. Execution of the searchPoly() function

III. Analysis of the Algorithms

The total cost of constructing line segments must be examined for worst and best cases. For each case, It is assumed that there n line segments are drawn. In best case, it is assumed that there isn't any intersection causes fragmentation of discrete lines. The size of the Line segment database determines the complexity as O(n). For the scan operation on the Adjacent matrix causes O(n2). As a result of this, the complexity of the searchpoly method is O(n2). In worst case, it is assumed that each line drawn i, i=2,3,..n intersects all previously drawn lines. Then the new line i will cause to generate at most 2i-1 new line segments (where (i-1) of them generated by segmenting previous lines, and i due to new line drawn). Therefore the total number of line segments is

$$1+ \sum_{i=2}^{n} 2i-1 \qquad (4)$$

which is $O(n^2)$. By adding adjacency matrix calculation the total complexity will be $O(n^2)$.

Composing algorithm must be applied during the drawing or defining the discrete lines. So, complexity of composing algorithm does not have any effect on performance of searching polygon region.

The complexity of searchPoly(BLS,BP) algorithm can be decreased to n*n by using the adjacency matrix. It is clear that, if search space can be limited to some line segments, the algorithm works faster. However, the cost of the search algorithm is still acceptable for the application to the complex cases.

IV. Implementation

The algorithm has been implemented within a CAD based heat requirement calculation of a building project. It has been developed using Visual Basic programming language and its graphics tools. Figure 7.a and Figure 7.b show examples of composing segments while Figure 8 demonstrates how to select a specific polygonal region.

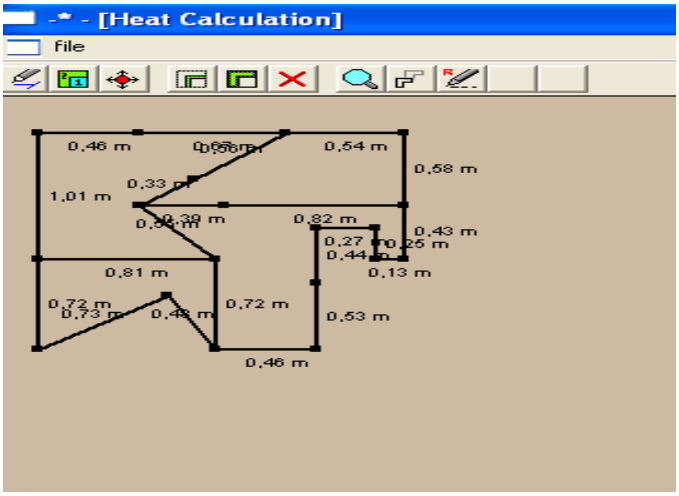


Fig. 7.a. Composing segments within the CAD based Heat Evaluation Program (before the intersection)

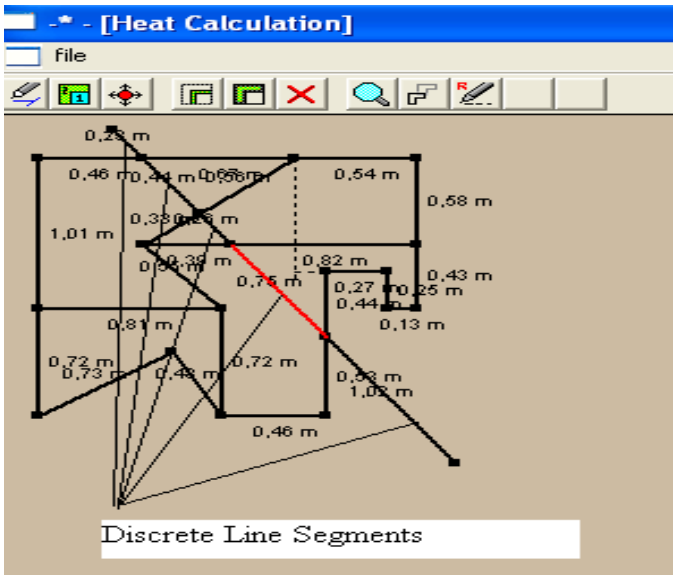
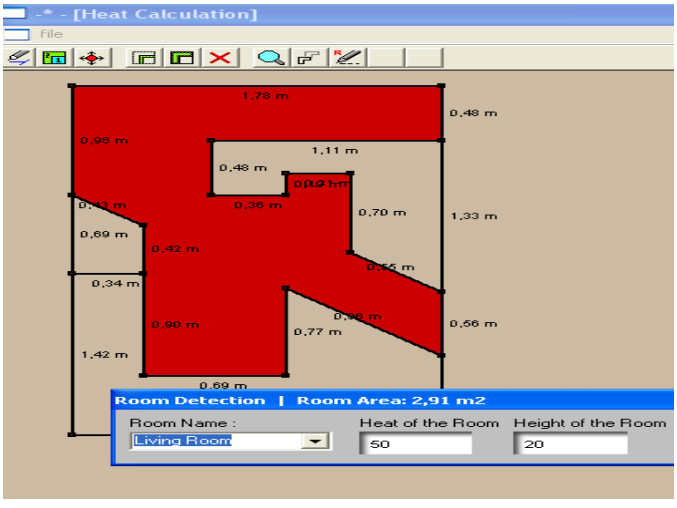


Figure 7.b. Composing segments within the CAD based Heat Evaluation Program (after the intersection)



V. Conclusion

In the light of the literature review has been made in this particular field, this method and algorithms seem to be a unique solution for the definition of the polygon region characteristics. The analysis of the algorithms indicates that they have got a fast and robust structure. This method can be used a wide variety of application areas. These applications may spread to the region definition, finding orientations and graphical object recognition. Such applications may be implemented in the Computer Graphics Industries, particularly, Computer Aided Design (CAD), Computer Aided Manufacturing (CAM), Robotics and Geographical Information System (GIS).

For the future studies, this approach can be extended for the 3D model in order to obtain the edges or surfaces of a polyhedral structure.

VI. References

1) J. Pach, E. R. Campo, On circumscribing polygons for line segments, Computational Geometry 10 (1998) 121-124
2) [X.Li, T.W.Woon, T.S.Tan, Zhuang, Decomposing Polygon Meshes For interactive Applications, Proceedings of the 2001 symposium on Interactive 3D graphics,2001,35-42
3) M.A. Lopez, S. Reisner, Efficient approximation of convex polygons, International Journal of Computational Geometry & Applications Vol. 10, No. 5, (2000), pp 445-452
4) J. Fernandez, L. Canovas, B. Pelegrin, Algorithms for the decomposition of a polygon into convex polygons, European Journal of Operational Research 121 (2000) 330-342
5) [5] S. Y. Shin, T. J. Woo, An Optimal Algorithm for Finding all Visible Edges in a Simple Polygon, IEEE Transactions on robotics and automation. Vol 5. Num. 2, April ,1989
6) S. M. Lavalle,B.H.Simov, G.Slutzki, An algorithm for searching a polygonal region with a flashlight, International Journal of Computational Geometry & Applications Vol. 12, Nos. 1 & 2 (2002) 87-113.
7) M.Keil, J. Snoeying, On the time bound for convex decomposition of simple polygons Int. Journal of Computational Geometry & Applications Vol. 12, No:3. (2002).
8) P. K. Agarwal,N. Amenta, M. Sharir, Placement of one convex polygon inside another, Discrete Computational Geometry, Vol 19, 1999,pp 95-104
9) [9] Y. Ishigami, How Many Diagonal Rectangles Are Needed to Cover an Orthogonal Polygon?, Discrete Computational Geometry, Vol 24,2000,pp117-140
10) K.Jha Manoj, C. McCall, P. Schonfeld, Using GIS, Generic Algorithms, and Visualization in Highway Development, Computer-Aided Civil and Infrastructure Engineering 16(6), (2001), 399-414.
11) V.Sridhar,M.A. Nascimento, ,Xiaobo Li, Region-based image retrival using multiple-features, in "5th International Conference VISUAL 2002", Vol 2314, pp 61-75.
12) C. Huggel, A. Kääb, W. Haeberli, B. Krummenacher, Regional-scale GIS- models for assessment of hazards from glacier lake outbursts: evaluation and application in the Swiss Alps, Natural Hazards and Earth Sciences (2003), 647-662.
13) J. L. A reliable test for inclusion of a point in a polygon, ACM SIGCSE Bulletin Volume 34 , Issue 4 (December 2002),81 - 84
14) D. M. Mount, Intersection Detection and Seperators for simple polygons, Annual Symposium on Computational Geometry Proceedings of the eighth annual symposium on Computational geometry Berlin, Germany, (1992), 303 - 311
15) W. Lenhart, R. Pollackt, J. Sack, R. Seide1, M. Sharir, S. Suri ,G. Toussaint, S. Whitesides, C. Yap, Computing the Link Center of a Simple Polygon, Annual Symposium on Computational Geometry Proceedings of the eighth annual symposium on Computational geometry, Waterloo, Ontario, Canada,(1987),1-10
16) D. P. Luebke, A Developer's Survey of Polygonal Simplification Algorithms, IEEE Computer Graphics and Applications, May 2001, pp. 24-35
17) T. Whitted, A scan line algorithm for computer display of curved surfaces, ACM SIGGRAPH Computer Graphics,vol. 12,issue 3, 1978,pp. 26-31
18) D.F.Rogers, Lines intersections Algorithms, Procedural Elements for Computer Graphics, 3rd printing,1988,pp 172-183