

---

**ABSTRACT**

Given a set  $S$  of disjoint simple polygons on a bounded 2D-Plane, we studied the problem of finding a set of polygon  $P$ , such that stabbing  $S$  by segment stabbing region in optimal way. We present a novel approach for solving the convex traversing method in PTIME. Convex traversing is a mechanism of layering continues convex hull extracted from points  $P$ . Whenever adding a point in the convex traversing plane, it is necessary to compute the whole convexation again, which contradict the better runtime. We introduce the new notion of dynamic algorithm to reduce the runtime of this problem into LOG-SPACE bounded. Hence output produced at every dynamic modification results only a LOG-SPACE bounded reconstruction.

**KEYWORDS:** Polygon, Convex hull, Convexation, PTIME, Segment Stabbing, LOG-SPACE bounded.

---

**INTRODUCTION**

Computational geometry is a research forum in theoretical and applied aspects of geometry, emerged from the field of algorithms design and analysis. Much of its computations are performed on geometrical objects known as polygons. Polygons are convenient representation for many real world objects and have its scope in application areas like Computer Graphics (CG), Computer-Aided Design (CAD), Geographic Information System (GIS), and Robotics and so on. One of the first problems studied in computational geometry was based on computation of efficient convex hull and segment stabbing problem on given polygon in either 2D or 3D. This problem attained more popularity due to its usage in widespread applications. In this study a new novel approach is developed for the above problem by proving it in LOG-SPACE bounded time complexity which also solves the open problem whether a logarithmic bound can be achieved in the worst case and whether logarithmic bounds can be amortized for line stabbing queries.

**A. Basics**

A polygon is the region of a plane bounded by a finite collection of line segments forming a closed shape. Let  $p_0, p_1, p_2, \dots, p_n$  be  $n$  points on the plane where a cyclic ordering of the points from  $p_0$  following  $p_{n-1}$  is implied. If  $e_0 = p_0 p_1, e_1 = p_1 p_2, \dots$  then  $e_i = p_i p_{i+1}, \dots, e_{n-1} = p_{n-1} p_0$  be  $n$  segments connecting the points. The points  $p_i$  are called the vertices of the polygon and the segments  $e_i$  are called its edges. A polygon is convex if its interior angle is lesser than  $\pi$ . Convex polygon construction and reconstruction play a major role in computational geometry. Here a simple polygon is generated randomly without any constraint and used for segment stabbing and convex traversing problem to prove the reduced time complexity of the algorithm.

**B. Overview**

In this study a simple polygon is constructed for segment stabbing problem to produce a disjoint set of polygon in reduced runtime complexity. Obtaining this output from stabbing method, convex traversing is performed on the set of polygon to reconstruct it into a convex polygon in LOG-SPACE bounded running time which is considered to be efficient method. Therefore, the main goal focus on convex traversing of multiple polygons on a plane and asymptotic runtime analysis for dynamic convex traversing region reconstruction in LOG-SPACE bounded runtime. The rest of the section is organized as follows. The next section discusses the existing system methods based on segment stabbing and convex traversal. The remaining section describes the working of the segment stabbing and convex traversing and also presents the results of runtime complexity of the methods and algorithms.

## RELATED WORK

The previous methods on segment stabbing and convex traversing are NP-Hard for some cases like the convex stabber method proposed by Esther M. Arkin et al proves that the existence of convex stabber from a given set of segment is NP-Hard and finds a convex traversal of pair wise disjoint segments in 2D. It [4] works on both scaled copies of polygon and on regular polygons. The polynomial time algorithm is better when used on regular polygons but it remains NP-Hard for the scaled copies of polygon. A.Gheibi et al proposed a polygonal shape reconstruction in a plane using a simple shape algorithm. [6] Focuses on producing polygonal output by handling two types of input. One is given in a dotted sample and other is based on boundary sample. Simple shape algorithm reconstructs the given input using convex traversing approach to produce a polygonal shape. The proposed works in  $O(n \log n)$  time for both inputs but it faces problem when finding holes in the shape and reporting a bunch of simple polygons instead of one. [8] Describes a convex segmentation in footstep planning for a walking robot in a difficult terrain such that the robot moves to the specified location without colliding with any obstacles through the environment safely using mixed integer convex optimization method. In this method user is required to seed input each position for the robot to move and the positions are taken as points to perform convex traversing ingenerating a safe path for the robot. It takes long time to footstep planning and improves its algorithm to work for dynamic footstep planning. [3] Proposal for unfolding and overlapping the polyhedral by edge unfolding method and reconstructs the polyhedral by edge orientation and edge length. It focuses on analyzing the computational complexity for convex and non-convex cases in unfolding. It solves in NP-Complete and more study is required to improve its runtime. [7] Developed an algorithm for detecting the texture of the objects by using convex hull in direct line detection method and clustering method. The objects surface is partitioned into two regions to identify the texture or color of the surface. The pixel values are considered to detect the non-uniform texture. Based on the analysis, the non-uniform texture is detected and removed to provide rich uniform texture by computing convex hull on each pixel values. This algorithm faces problem when detecting the objects with shadows or reflections. [5] Performed a recursive computation on a minimum length polygon using time optimal algorithm to identify minimum perimeter and minimum length of the polygon. The given polygon is taken in a tree structure and cavity of the tree is computed by convex traversing the nodes of the cavity tree. This algorithm is limited to a constant when finding a height of the cavity tree. It produces a polynomial time complexity in finding the minimum perimeter and length of the polygon. Asish Mukhopadhyay proposes an algorithm that gives partial solution to Tamir's open problem and identifies the minimum area of the given convex polygon that weakly intersects the set of  $n$  vertical line segments by using bipartite stabbing and dualization method. It [2] reduces the shortest path computation problem but is impractical for polygon triangulation when it is based on chazelle's linear time algorithm. Adrian Dumitrescu et al generate a polynomial time approximation scheme to compute minimum perimeter for the set of intersecting convex polygons a plane. If a polygon  $P$  intersects another polygon of a set of segments  $S$  if every segment of  $S$  intersects the interior or boundary of  $P$ . [1] The minimum distance of points are computed by considering the minimum and maximum coordinates of each point in  $x$ . it produce NP-Hard in finding minimum length of intersecting polygon.

## SEGMENT STABBING IN POLYGON

A simple polygon is constructed by the user of any dimensions. This polygon is considered as input for segment stabbing method which is performed by the following steps.

### Point detection

The given polygon  $P$  is bounded on a 2D plane where the area inside the bounded region is called interior and outside bounded region on the plane is called exterior to polygon. The query points are inserted in the polygon to find whether the points lie interior of the polygon or exterior. This is identified by using an algorithm called point detection algorithm where  $p[i]$  represents polygon points and  $q$  denotes query points. Three cases are considered to find whether the points lie inside or outside the polygon by assuming each point as a ray with one head arrow passing through the edges infinitely.

### Algorithm: point detection

**Input:** Set  $P$  of  $n$  points

Query point  $q$

**Output:** Yes if  $q$  lies inside the polygon

No otherwise

CheckPolygon( $P$ [],  $q$ ):

Intersect = 0

For  $i$  in range( $0 \dots n - 1$ ) do:

/\* Case 1 \*/

```

If p[i] == q Then
return TRUE
/* Case 2 */
Else If q lies between p[i] and p[i + 1] Then
return TRUE
/* Case 3 */
Else
slope = (p[i].y - p[i + 1].y)/(p[i].x - p[i + 1].x)
cons = p[i].y - p[i].x * slope
Xinter = (q.y - cons)/slope
Pin = {Xinter, q.y}
/* Case 3. a */
If Xinter == q[i + 1].x Then
If both p[i] and p[i + 2] not lies above or below p[i + 1]
Then
Intersect + +
End If
/* Case 3. b */
Else If Xinter lies between p[i].x and p[i + 1].x Then
Intersect + +
End If
End If
End For
Return (InterSect%2 == 1)
End

```

**Lemma and Proof:**

**Notations:**

$P_1, P_2, P_3 \dots P_n$  - (n polygon points)  
 $R_1, R_2, R_3 \dots R_k$  - (k intersection points)  
 $Q$  - Query Point

**Conditions for intersection:**

- 1) There exist a  $P_i$  in  $P$  such that  $P_i = Q$  or  $Q$  lies on any edge of polygon then  $Q$  lies on the polygon.
- 2) If  $Q$  intersect polygon edge (Except start point and end point of edge) then counted as intersection
- 3) If there exist a  $P_i$  in  $P$  and  $R_i$  in  $R$  such that  $R_i = P_i$  and  $Q \triangleleft P_i$  then check the following property
- 4) If any adjacent point of  $P_i$  lies above and another lies below to  $P_i$  then  $R_i$  is counted as intersection, otherwise not.

**Proof:**

Assume  $R (R_1, R_2, R_3 \dots R_k)$  are k intersection points of  $Q$  on polygon edges, when  $Q$ 's x-coordinate incremented up to rightmost point of polygon.  $Q$  changes its locality at every intersection. For every two intersection  $Q$  retain its locality.

$$slope = (p[i].y - p[i + 1].y)/(p[i].x - p[i + 1].x) \quad (1)$$

$$cons = p[i].y - p[i].x * slope \quad (2)$$

$$Xinter = (q.y - cons)/slope \quad (3)$$

$$Pin = \{Xinter, q.y\} \quad (4)$$

“Equation (1) used for finding slope of the edge”, “(2)” use the calculated value of slope to find its locality, “(3)” determines the intersection point from the computed values and finally stored in “(4)”. Using the above equations the intersection points are identified on every edge of the polygon. It is performed until it detects all the points that are given by the user as query points. If  $k(\text{intersection})$  is even then  $Q$  remain in same locality and flipped the locality if it is odd. We conclude that  $Q$  lies outside if intersection is even, otherwise  $Q$  lies inside. This algorithm is extended for finding intersecting line segments and stabbing points in the polygon.

**Edge stabbing**

The next process involves identifying the stabbing points (intersecting points) when a query line is given across the polygon. This line cuts the polygon along its edges and the point that intersects the edge is called as stabbing points. The process of stabbing the edges of the polygon is termed as edge stabbing. It identifies stabbing point by visiting each points along the edges and add those points to the list which is needed for the segmentation process. Each point  $p$  in polygon  $P$  is considered to find the query point  $q$  that intersects the edge of the polygon  $e$ . The edge stabbing algorithm is given below with its proof.

**Algorithm: Edge stabbing**

**Input:** polygon  $p$ , edge  $e$   
**Output:** set of intersecting points  $L$   
 $L = \phi$   
 For each point  $p \in P$ , do  
 $q = p \rightarrow next$   
 if (pointstabbing( $\{p, q\}, e$ )) then  
 $L = getintersecting(\{p, q\}, e)$ ;  
 End if  
 End for;  
 For each point  $p \in L$  do  
 Line  $L = getPointInEdge(p)$ ;  
 If  $L$  is not visited then,  
 Add stabbingpoint( $L$ );  
 End if  
 End for;  
 End algorithm

**Lemma and Proof:**

**Notations:**  $P$  – polygon,  $p(p_1, p_2 \dots p_n)$  -  $n$  polygon points,  $q$  – query point,  $e$  – edge of polygon,  $L$ - line

**Conditions for intersection:**

- 1) There exists a line  $L$  which is initially defined as null.
- 2) Consider each point  $p$  such that  $p \in P$  and  $q$  refers to the next point from  $p$ .
- 3) If point  $p$  and  $q$  lies on edge  $e$  of polygon then it is stored in line  $L$ . These points are stabbing points.
- 4) Consider  $p \in L$  such that  $L$  gets all stabbing points by visiting from one point to another point and unvisited points are added to  $L$ .

**Proof:**

Assume  $p(p_1, p_2, p_3 \dots p_k)$  are polygon points of  $P$  on edge  $e$  and  $q$  is incremented to next point of polygon.  $q$  changes its locality at every intersection. If  $\{p, q\}$  lie in  $e$  then it is considered as stabbing point and stored in  $L$ . Each point in edge is visited and unvisited point is added in  $L$ .

**Polygon segmentation**

The final step of segmentation is performed by computing the visibility polygon for line segment using edge stabbing method. The polygon will be segmented based on the stabbing coordinates (points) and forms island polygon. Island polygon is a kind of segment made in the polygon where the entire segment is contained inside the polygon. This stabbing method is executed repeatedly to find all the line segments or edges that are intersected. The process is iteratively performed based on visibility of each line segment and ends when all the intersected line segments are visited. Now the polygon is segmented separately by the above process and each segments of polygons are highlighted. This process is called as segment stabbing method.

**Algorithm: Segment stabbing**

**Input:** polygon  $p$ , edge  $e$   
**Output:** set of intersecting points  $L$   
 Segment Stabbing ( Polygon  $P$ , Segment[]  $q$ )  
 For each line segment  $lq \in q$  do  
 For each line segment  $lp \in p$  do  
 If  $lq$  intersects  $lp$  then  
 Add intersecting points to  $p$  and  $q$   
 Push all points in  $p$  to stack  
 While stack is not empty  
 $S = stack.pop()$ ;  
 While true:  
 If  $S$  outside  $q$  then  
 $S = stack.pop()$ ;  
 Else if  $S$  inside  $q$  then  
 $S = S \cup select\ next\ point\ lies\ in\ p\ from\ q$   
 If  $S$  is contained new point  
 Break;  
 Make  $S$  a separate Polygon Segment

End

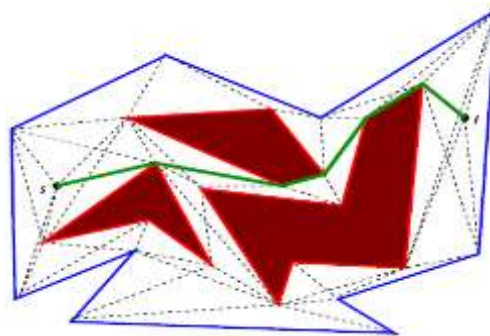
**Lemma and Proof:**

**Notations:**  $P$  – polygon,  $l_q$  – query line segment (user),  $l_p$  – line segment (polygon)

**Proof:**

If line segment  $l_q$  intersects  $l_p$  then add intersecting points to  $p$  and  $q$ . Push all points that belong to polygon in a stack. Pop the points that lie outside the polygon and store stabbing points in  $S$  that lies inside the polygon. Report all segments that are not contained in segment stabbing. Now the polygon is segmented and those segments that are inside the polygon is called island polygon. Only the segment that stabs the polygon are considered and used for segmenting the polygon.

The fig 1. Shows the segmented disjoint set of polygons and those polygons are highlighted. This implementation is enhanced by studying [9] and its drawback is overcome by executing in PTIME (polynomial time) which is considered to be efficient runtime when compared to the previous methods and algorithms. Our stabbing algorithm is proved efficient by comparing to one of the previous results and its result analysis is given in performance evaluation section.



**Figure1: Highlighted segments of polygon**

**PERFORMANCE EVALUATION**

Comparing this with our stabbing problem where a simple polygon  $P$  is constructed and to verify a point lies inside or outside the polygon is computed using the point detection algorithm which gives  $O(n)$  time complexity. After finding the point, query line intersection at the edge of polygon is calculated using edge stabbing algorithm that gives  $O(n + k)$  time complexity. Obtaining above results polygon stabbing is performed using segment stabbing algorithm computed in time  $O((n + k)m)$ . Finally the polygon is segmented giving a reconstructed polygon with constant stabbing of the time  $O(nm + n \log n)$ , which shows the reduced time complexity of our algorithm when compared with Rappaport algorithm. It is depicted in the following table and a graph that shows the performance evaluation of the algorithm.

**Table 1. Result analysis**

n	Stabbing Algorithm	Speed up	Rappaport Algorithm
10	66.4385619	6.28818332	417.777857
20	172.877128	13.8467712	2393.79007
30	294.413437	22.8534947	6728.37586
40	425.75426	33.0166664	14056.9855
50	564.38569	44.1714703	24929.7421

Time complexity of Rappaport algorithm –  $O(3^m n + n \log n)$  (5)

Time complexity of stabbing algorithm –  $O(nm + n \log n)$  (6)

Speed up =  $(O(3^m n + n \log n)) / (O(nm + n \log n))$  (7)

Where,  $m$  is the number of different segment directions

$n$  is the number of points in polygon

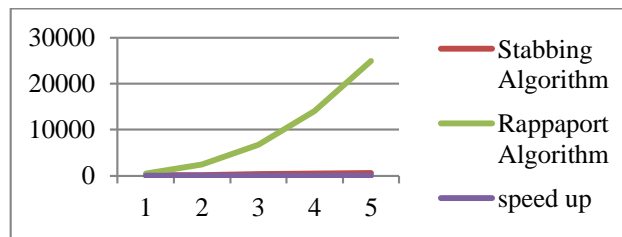


Figure2: Time chart comparison of algorithm speedup

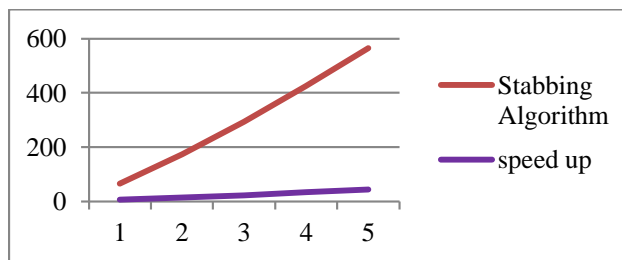


Figure3: Linear speedup of stabbing problem

The analysis clearly shows that stabbing problem gives reduced time complexity when compared to Rappaport algorithm that was used for segment stabbing of polygon. Therefore, segment stabbing is solved in polynomial time.

## CONVEX TRAVERSAL

### Convex hull

**Definition:** A set  $S$  is convex if any point  $p \in S$  and  $q \in S$  implies that the segment  $pq \subseteq S$ . convex hull of set of points in  $S$  is the set of all the convex combinations of points of  $S$ . Graham's scan algorithm is one of the optimal way to construct a convex hull for a given set of points.

**Theorem 1:** The convex hull of a set of  $n$  points in the plane can be computed in  $O(n \log n)$  time by Graham's scan algorithm.

**Proof:** Consider  $N$  is a set of points in the plane and  $H$  be the outer layer of points called convex hull point of  $N$ , such that  $N/H$  be the remaining points on the plane lies inside continues line segments connect  $H$  as a convex layer. Convex traversing is a mechanism of layering continues convex hull extracted from points  $N$ .  $H \subseteq N$  where  $|N| = n$  and  $|H| = h$  such that the convex hull of a set of  $n$  points in the plane can be computed in  $O(n \log n)$  time by proving the computation of upper hull and lower hull using induction on the number of points treated.

## DYNAMIC CONVEX HULL

**Definition:** Let  $P$  be a set of disjoint closed polygon (obtained from the output of segment stabbing method) on a plane and  $H$  be the outer layer points called convex hull point of  $P$ , such that  $P/H$  be the remaining points on the plane lies inside continues line segments connect  $H$  as a convex layer. Convex traversing is a mechanism of layering continues convex hull extracted from points  $P$ . Whenever adding a point in the convex traversing plane, it is necessary to compute the whole convexation again, which contradict the better runtime. We introduce the new notion of dynamic algorithm to reduce the runtime of this problem into LOG-SPACE bounded. Hence output produced at every dynamic modification results only a LOG-SPACE bounded reconstruction.

**Theorem 2:** Runtime analysis of convex traversing and quick hull approximately leads to  $O(n \log n)$  time.

**Proof:** Let  $P$  be a set of  $n$  points on a plane. Quick hull outputs  $h$  points which are covered by the convex hull of  $P$ . Runtime of quick hull is  $O(n \cdot h)$  [given by asymptotic runtime analysis of computational geometry, Joseph O'Rourke], which is stated as output sensitive analysis. Three cases are considered on comparison namely,

- 1)  $h = \log n$  ( $\log n$  points in hull)
- 2)  $h > \log n$  (more than  $\log n$  points in hull)
- 3)  $h < \log n$  (less than  $\log n$  points in hull)

The probability of the three cases can be given as

$$P(h = \log n) = 1/n \quad (8)$$

$$P(h > \log n) = 1 - 1/\log n \quad (9)$$

$$P(h < \log n) = 1/\log n \quad (10)$$

Summing up “(8)”, “(9)” and “(10)”

$$= (n + 1) \log(n + 1)$$

$$\approx n \log n$$

## CONCLUSION

Therefore, the proposed method of dynamic convex hull is computed for a given disjoint set of polygons obtained from segment stabbing method gives a reduced runtime into LOG-SPACE bounded for every dynamic modification by adding a point in the convex traversing plane, the whole convexation is computed again in better runtime i.e. in PTIME (polynomial time). Hence output produced using dynamic approach results only in LOG-SPACE bounded reconstruction.



## ACKNOWLEDGEMENTS

This work was guided by Professor Elango S. from Kingston Engineering College, Vellore. I wish to thank for his exemplary guidance and valuable feedbacks for helpful discussions. I would also like to extend my sincere gratitude to my parents for their constant support and encouragement in completing this work.

## REFERENCES

- [1] Adrian Dumitrescu and Minghui Jiang, “Minimum-Perimeter Intersecting Polygons”, *Algorithmica*(2012)63.pp. 602-615, Apr 2011.
- [2] Asish Mukhopadhyay, Chanchal Kumar, Eugene Greene, Binay Bhattacharya, “On intersecting a set of parallel line segments with a convex polygon of minimum area”, *Information Processing Letters* 105 pp.58-64, Sep 2007.
- [3] Brendan Lucier, “Unfolding and Reconstructing a polyhedral”, Thesis, University of Waterloo, pp.1-107,2006.
- [4] Esther.M. Arkin, Claudia Dieckmann, Christian Knauer, Joseph S.B. Mitchell, Valentin Polishchuk, Lena Schlipf, Shang Yang, “Convex transversals”, *Computational Geometry* 47, pp. 224-239, Nov 2012.
- [5] Gisela Klette, Recursive computation of minimum-length polygons, *Computer Vision and Image Understanding* 117, pp. 386-392,2013.
- [6] A.Gheibi, M.Davoodi, A.Javad, F.Panahi, M.M.Aghdam, M.Asgaripour and A.Mohades, “polygonal shape reconstruction in the plane”, *IET Comput.Vis.*,2011, Vol.5, Iss.2, doi: 10.1049/iet-cvi.2009.0079, pp.97-106, Aug 2010.
- [7] Kefei Lu and Theo Pavlidis, “Detecting textured objects using convex hull”, Credit Suisse group, stony Brook University, pp.1-12, 2006.
- [8] Robin L. H. Deits, “Convex Segmentation and Mixed- Integer Footstep Planning for a Walking Robot”, the Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, pp.1-82, September 2014.
- [9] Schlipf L., “New results on convex stabbers”, in: Abstracts of the 26<sup>th</sup> European Workshop on Computational Geometry, *EuroCG*, 3(3)., pp. 57-60, Jun 2014.

**AUTHOR BIBLIOGRAPHY**

	<p><b>Ms. B. USHA RANI</b> was born in Vellore, India, in 1992. She received the B.E. degree in computer science and engineering from the Sri Venkateswara College of Engineering and Technology, Chennai, India, in 2014, and pursuing her M.E. degree in computer science and engineering from the Kingston Engineering College, Vellore, India. She has attended many workshops and seminars on recent trends and technologies. She was selected as NNSC Zonal Winner in Cloud Computing conducted by Indian Institute of Technology(IIT) Bombay, Bombay, India, in 2015.</p>
	<p><b>Mr. S. ELANGO</b> have more than 4 years of teaching experience. He did his Under Graduate in B.E.-Computer Science and Engineering at Anna University and Post Graduate, M.Tech - Computer Science and Engineering at Dr. M. G. R University, Chennai. He is occupied as Assistant Professor, in Kingston Engineering College. He had published more than 10 research papers in International Journal, National and International conferences. He is guiding number of research scholars in the area of cloud computing.</p>