4756_svn:2668 _ wc: 5112_Submitted to AGILE 2010

Intersection of Nonconvex Polygons Using the Alternate Hierarchical Decomposition

Rizwan Bulbul, Andrew U. Frank

Vienna University of Technology
Department of Geoinformation and Cartography
Gusshausstrasse 27-29 E127, A-1040 Vienna, Austria
{bulbul, frank}@geoinfo.tuwien.ac.at

Abstract. Intersection computation is one of the fundamental operations of computational geometry. This paper presents an algorithm for intersection computation between two polygons (convex/nonconvex, with nonintersecting edges, and with or without holes). The approach is based on the decomposed representation of polygons, alternate hierarchical decomposition (AHD), that decomposes the nonconvex polygon into its convex components (convex hulls) arranged hierarchically in a tree data structure called convex hull tree (CHT). The overall approach involves three operations (1) intersection between two convex objects (2) intersection between a convex and a CHT (nonconvex object) and, (3) intersection between two CHTs (two nonconvex objects). This gives for (1) the basic operation of intersection computation between two convex hulls, for (2) the CHT traversal with basic operation in (1) and, for (3) the CHT traversal with operation in (2). Only the basic operation of intersection of two convex hulls is geometric (for which well known algorithms exist) and the other operations are repeated application of this by traversing tree structures.

1 Introduction

The intersection operation is of fundamental importance (Shamos and Hoey 1976) as it provides basis for computing other Boolean operations like union and difference etc. Also, it is the most expensive operation computationally, roughly taking 80 % of the running time (Greiner and Hormann 1998). The intersection operation has two problems, intersection detection and intersection computation. The intersection detection between two convex objects is a basic geometric operation (Chazelle and Dobkin 1987) and a great account of the topic can be found in (David 1997). We are not focusing on the issue of intersection detection and for simplicity we assume hereafter that the objects under consideration for intersection computation do intersect.

The Problem: Given two polygons (convex or nonconvex, with non-intersecting edges, and with or without holes), compute the intersection region that may be;

a) Empty (Figure 1a)

b) Convex (Figure 1b)

c) Nonconvex (Figure 1c)

d) A set of convex and/or nonconvex disjoint regions (Figure 1d)

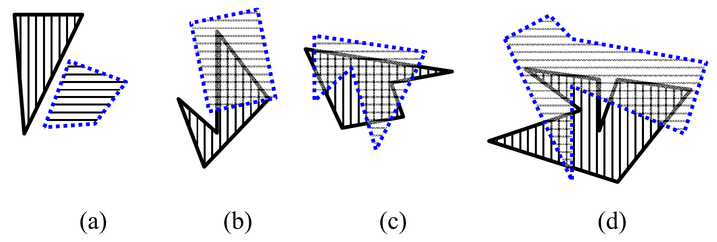


Fig. 1. Different polygon intersection scenarios

In GIS domain the set-theoretic Boolean operations (intersection, union and symmetric difference) are extensively used for extracting useful spatial information out of spatial data modeled as polygons (Margalit and Knott 1989). For example, polygon clipping is a frequent operation in GIS (Liu et al. 2007). Other Boolean operations in GIS include overlay, windowing, join and merge etc (Rigaux et al. 2001) (FranciscoMartınez et al. 2009). The map overlay operations are key operations in the GIS domain.

For convex polygons, optimal algorithms for intersection computation are known. Although, variety of solutions also exist for Boolean operations on complex polygons (nonconvex polygons with or without holes), these solutions are intricate having complex data structures leading to difficult implementations.

Our approach for the intersection computation is a simple algorithm based on the decomposed representation of polygons, AHD (Bulbul and Frank 2009), that decomposes the nonconvex polygon into its convex components (convex hulls) arranged hierarchically in a tree data structure called CHT (more in section 3). The intersection computation involves three operations;

1) Intersection between two convex objects.
2) Intersection between a convex and a CHT (nonconvex object).
3) Intersection between two CHTs (two nonconvex objects).

This gives for (1) the basic and simple operation of intersection computation between two convex hulls. For (2), the CHT traversal with basic operation in (1) and for (3), the CHT traversal with operation in (2). The approach is further discussed in detail in section 4. Only the basic operation of intersection of two convex hulls is geometric (for which well known algorithms exist) and the other operations are repeated application of this by traversing tree structures.

2 Previous Work

Many algorithms for Boolean operations on polygons have been reported in literature. The preliminary work by Shamos and Hoey (1976) provided a basis for geometric intersection problems. They have shown that the intersection of two simple plane *n*-gons can be detected in $O(n \log n)$. The intersection of two convex *n*-gons and two nonconvex *n*-gons can be computed in $O(n)$ *and* $O(n^2)$ respectively. Bentley and Ottmann (1979) gave the classical sweep line algorithm for counting and reporting all intersections by extending the work by Shamos and Hoey. They provided an algorithm for reporting all *k* intersections between two general planar objects in $O(n \log n + k \log n)$. The work by Bentley and Ottmann was further extended by Lauther (1981), and the reported algorithm has the expected time complexity of $O(n \log n)$. Another $O(n)$ time algorithm was presented by O'Rourke etal (1998). The algorithm is simple but is limited to convex polygons only.

The two plane sweep algorithms by Nievergelt and Preparata (1982) compute the geometric intersection of two nonconvex polygons in $O((n+k) \log n)$ and two convex polygons in $O(n \log n + k)$. The polygons can have self intersecting edges but degenerate cases are not tackled. The data structure is complex and implementation details are not given.

The work by Chazelle and Dobkin (1987) provides lower bounds on algorithms for intersection of convex objects in two and three dimensions. Their work is based on the assumption that the intersecting objects are available in random access memory, eliminating reliance on linear input reading time. The time bounds for two convex polygons in 2D and two convex polyhedra in 3D cases are $O(\log n)$ and $O(\log^3 n)$ respectively (in 3D case an additional multiplicative factor of $\log n$ for data structure preprocessing for standardization).

The work by Margalit and Knott (1989) presents an algorithm for set operations on polygon pairs having worst time complexity of $O(n^2)$. They give partial correctness proof of their solution and implementation is discussed but still complex and not easily understandable.

The work by Rappoport (1991), extended convex difference tree (*ECDT*) for representing two n-dimensional polygons (polytopes) and performing intersection and union operations, is similar to our approach. The differences between our approach and their approach both at data structure and operations level are given in Table 1.

Table 1. Difference between our and Rappoport's Approach

Our Approach	**Approach by Rappoport**
A single CHT data structure with build and process operations	Implementation details of how data structure is actually built and processed is not mentioned
Can easily handle holes	Nothing said about
Our approach is robust, no special topological handling	Needs special topological handling for robustness
Can handle multiple polygons with slight modifications	Two polygons
Simple structure as the children convex hulls of a parent node are contained within the convex region of that parent node.	Primitives in a right sub-tree of a difference node are contained within the primitive on the left side of that node.

The state of the art for finding all intersections among segments is given by (Bernard and Herbert 1992). The algorithm by Vatti (1992) is for clipping arbitrary polygons against arbitrary polygons. The polygons may be convex, concave or self intersecting. However, the self intersecting polygon is converted to a nonintersecting polygon by inserting the points of intersection during the clipping process. The algorithm also supports polygon decomposition by allowing the output in the form of trapezoids if required. The solution is complex and implementation is not easy. Its performance has not been proved asymptotically, rather a comparison with traditional clipping methods is provided.

The algorithm proposed by Greiner and Hormann (1998) also deals clipping arbitrary polygons like Vatti's algorithm. However, it is a simple algorithm based on the boundary segment manipulation and performs better than Vatti's algorithm over randomly generated general polygons. The data structure is a doubly link list or lists in case of multiple polygons. Only few degenerate cases are mentioned and it can not treat overlap degeneracies, and no complete complexity analysis provided. The solution is limited for 2D polygons and robustness issues related to the fixed precision floating point arithmetic are not catered.

The algorithm by Rivero and Feito (2000) to calculate Boolean operations for general planar polygons (manifold and non manifold, with and without holes) is based on simplicial chains and their operations. The strategy has been demonstrated for 2D and claimed to be valid for 3D polyhedra. The algorithm does not need special treatment of degenerate cases, and it is shown that its time is similar to Greiner's algorithm. The slight modifications in the work of Rivero and Feito by Peng, Yong et al. (2005) resulted in an algorithm which has been shown to be more efficient (execution time less than one third of that by Rivero and Feito).

CGAL (Fogel et al. 2006) provides Boolean operations for polytopes in 2-dimensional Euclidian space. Robustness is ensured through the use of exact arithmetic. The regularized operations are provided for two simple polygons with or without holes. The time complexity is $O(n^2)$ for simple polygons.

The algorithm by Liu, Wang et al. (2007) is for clipping arbitrary polygons with holes. The algorithm is based on segment manipulation and works by classification of intersection points into entry or exit points. Unlike solutions by Vatti and Greiner, this algorithm uses a single linked list data structure and performs better than Vatti's solution for smaller number of input points. The solution is limited to 2D polygons and modifications are needed for dealing multiple polygons and holes. The degenerate cases are specially treated and the methods are demanding having difficult implementation.

The algorithm by Martınez, Rueda et al. (2009) is based on classical plane sweep algorithm for computing intersections performing in time $O((n + k) \log(n))$. They claim the solution works for general polygons, although its working for polygons with holes is not demonstrated. Algorithmic details are given but the implementation issues are not discussed. The implementation seems difficult and edge overlaps are specially treated. Dimension independence and robustness issues are not discussed.

3 Preliminaries: Assumptions and Data Structure

Our approach for intersection computation between geometric objects is based on following assumptions and simplifications;

a) The objects are flat objects i.e. lines, polygons and polyhedra. For demonstration purposes we will confine ourselves to 2D polygons only.

b) The objects are simple polygons although the solution can handle self intersecting edges with few modifications (section 9).

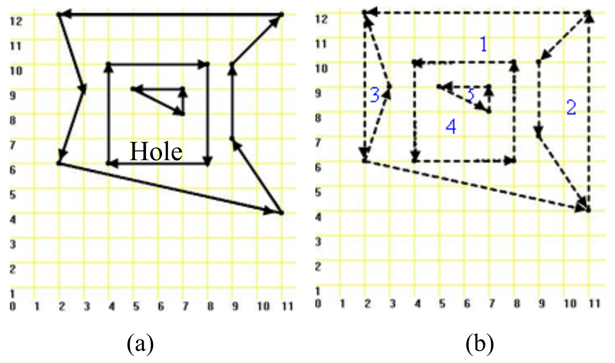
c) We are focusing on the intersection computation problem. The intersection detection problem is not considered that itself is a challenging problem and algorithms exist in literature (e.g. (Chazelle and Dobkin 1987; Shamos and Hoey 1976)).

d) The objects may contain holes or nested holes e.g. a hole within a hole in which case the two regions have opposite orientation of edges. In Figure 2a, region 5 is within region 4 and both have opposite orientation.

e) The edge representations are based on "Left-handed" rule, the boundary edges are oriented anticlockwise while the hole edges are oriented clockwise.

f) A region is represented by a single or multiple polygons. We have demonstrated our solution for intersection computation on a pair of polygons. Our solution is extendible for intersection computation involving more than two polygons.

The data structure, CHT, is an arbitrary tree in which every node represents a convex hull. The AHD process decomposes any nonconvex polygon with or without holes into convex components which are arranged hierarchically in a CHT. Two basic functions *build* and *process* are used for populating the data structure and processing the data structure to retrieve the original object respectively. For implementation details of the AHD, CHT and, the associated functions the reader is referred to (Bulbul and Frank 2009). The example in Figure 2 shows the AHD process and the resulting CHT. Given a nonconvex polygon with nested holes (see Figure 2a), it is decomposed into its convex components using AHD process as shown in Figure 2b, and the resulting data structure, CHT, is shown in Figure 2c.



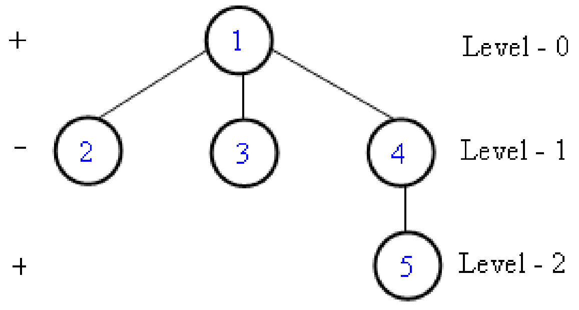


Fig. 2. Input polygon (a) decomposed input (b) convex hull tree data structure (c)

The nodes (convex hulls) in CHT at different levels are given positive and negative signs alternately as shown in Figure 2. The positively signed convex hulls represent the regions to be included while the negatively signed convex hulls represent regions to be excluded from their parent region represented by a positive convex hull. For example in Figure 2, the input nonconvex polygon is decomposed into five convex components, its convex hull node 1, two delta regions or concavities or notches (node 2, and node 3) and two nested holes (node 4, and node 5).

4 Our Approach

The closure of the basic intersection operation (Convex-convex intersection, CCINT) over convex sets is the most important property. Thus our approach exploits this property for computing convex-nonconvex Intersection, CNINT, by recursively applying CCINT between convex hull of the convex object and the component convex hulls of the nonconvex object. Similarly, the intersection between two nonconvex polygons (Nonconvex-nonconvex intersection, NNINT) is computed by recursively traversing CHT of one of the nonconvex objects with CNINT.

The NNINT is the generic intersection operation, GINT, because it deals all the three cases as it is based on CNINT, which in turn is based on CCINT as shown in Figure 3.

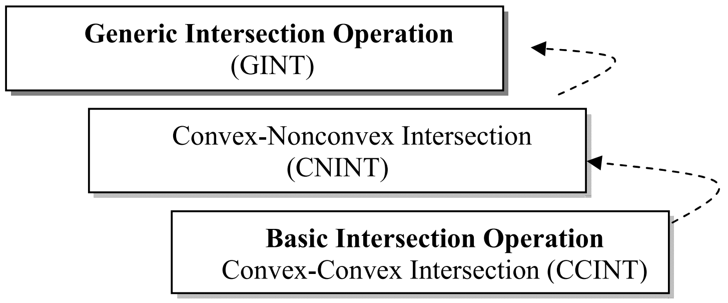

Generic Intersection Operation
(GINT)
Convex-Nonconvex Intersection
(CNINT)
Basic Intersection Operation
Convex-Convex Intersection (CCINT)

Fig. 3. Our approach for intersection computation

4.1 Basic Intersection Operation: Convex-Convex Intersection

The basic operation is closed under intersection and the result is always a convex region. The intersection computation between two convex polygons is the basic operation (for which known solutions exist). Any of the existing solutions can be used but we have provided an algorithm for intersection computation between two convex polygons using convex hulls.

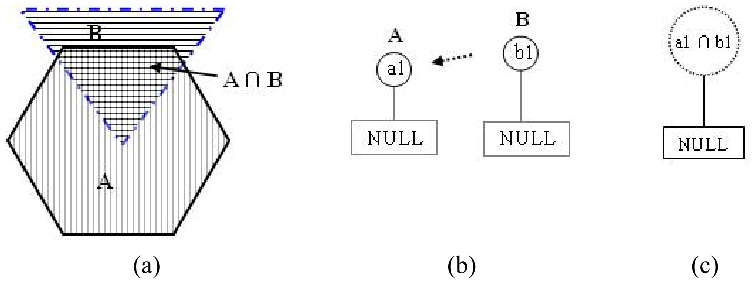


Fig. 4. Two convex polygons (a) CHTs of intersecting convex polygons (b) resultant CHT of A∩B

We have provided an algorithm for convex-convex intersection, the pseudocode of which is provided in section 5.1. Figure 4 shows two convex polygons and demonstrates their intersection at data structure level.

4.2 Convex- Nonconvex Intersection

The process of intersection computation between a convex polygon and a nonconvex polygon involves the repeated application of basic operation between the convex hull of the convex polygon and the convex hull components of the nonconvex polygon that are traversed recursively in CHT. The resulting convex hull of each basic operation is then placed in the resultant CHT at position same as the position of component convex hull of the nonconvex polygon involved in the basic operation. The further recursive processing of children trees of a component convex hull is stopped if the basic operation involving that component convex hull is null. This avoids unnecessary computations reducing the overall number of operations.

For example, Figure 5a shows a convex polygon *A* and a nonconvex polygon *B* and Figure 5b shows the decomposed inputs with their components numbered. Figure 5c shows the data structure representation of the input polygons *A* and *B*. The process starts by computation of basic operation between convex hull of convex polygon *A* (that is *a1*) and the convex hull of the nonconvex polygon *B* (that is the root *b1*). Since *b1*, the component convex hull of the nonconvex polygon is at root the result of basic operation between *a1* and *b1* will form the root of the resultant CHT as shown in Figure 5d. The process is then recursively applied to the children trees of *b1*. Since the intersection between convex hull *a1* and *b3* is null, the child tree of *b3* containing convex hull *b4* will not be processed further

(Figure 5e). The resultant intersection is a nonconvex region that is represented by two convex hull components as shown in Figure 5f. The processing (merging) of the resultant CHT, containing two component convex hulls results in the resultant intersection region as shown in Figure 5g.

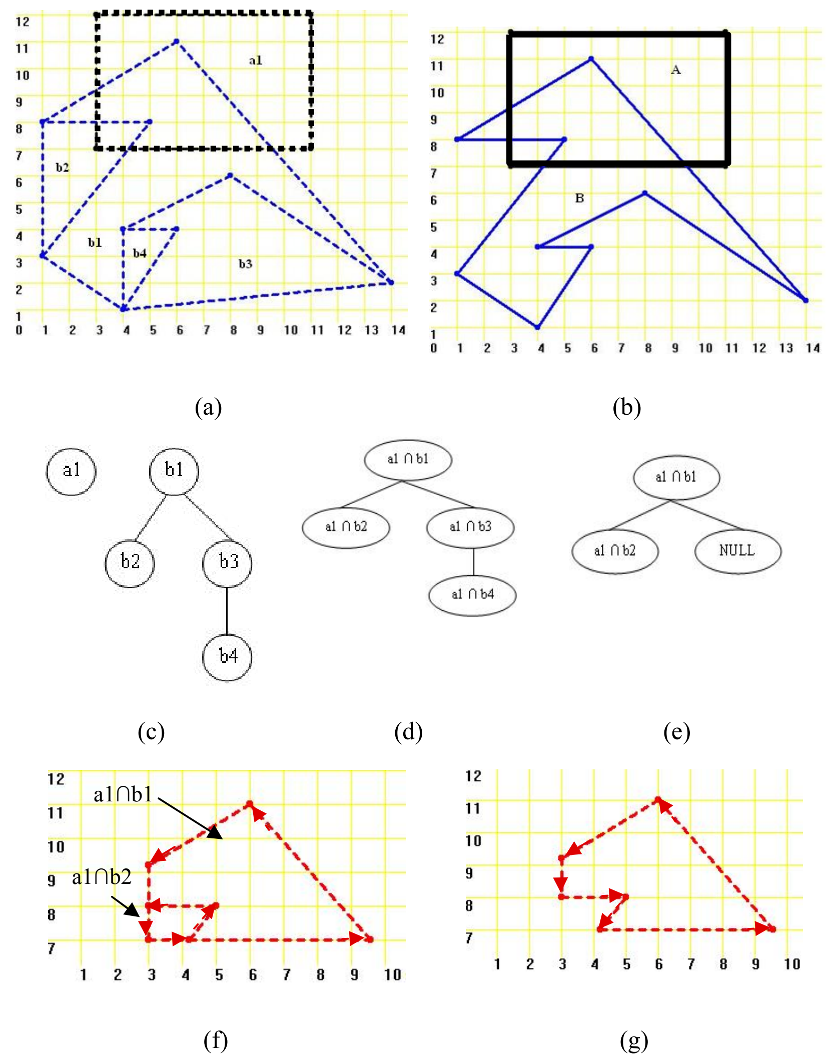


Fig. 5. Input polygons (a) decomposed inputs (b) CHTs of input polygons (c) intersection of convex hull of A, a1 and the CHT of B (d) CHT of intersection (e) intersection tree component hulls (f) intersection region A∩B (g)

Since the intersection of a convex and a nonconvex polygon may be a set of disjoint convex and/or nonconvex regions (e.g. see Figure 1d), the resultant CHT (if it does not represent a single convex region) is further processed for simplification, that results in multiple CHTs each representing the disjoint intersection region.

4.3 Generic Intersection Operation

The intersection operation for a convex polygon and a nonconvex polygon is used to compute the intersection between two nonconvex polygons represented by CHTs. Suppose two intersecting nonconvex polygons *A* and *B* as shown in Figure 6a. Figure 6b shows the decomposed convex hull components of both nonconvex polygons and their CHTs are shown in Figure 6c. The intersection process starts by taking the convex-nonconvex intersection between convex hull of nonconvex polygon *B* (*b1*) and the CHT of the nonconvex polygon *A* (Figure 6d).

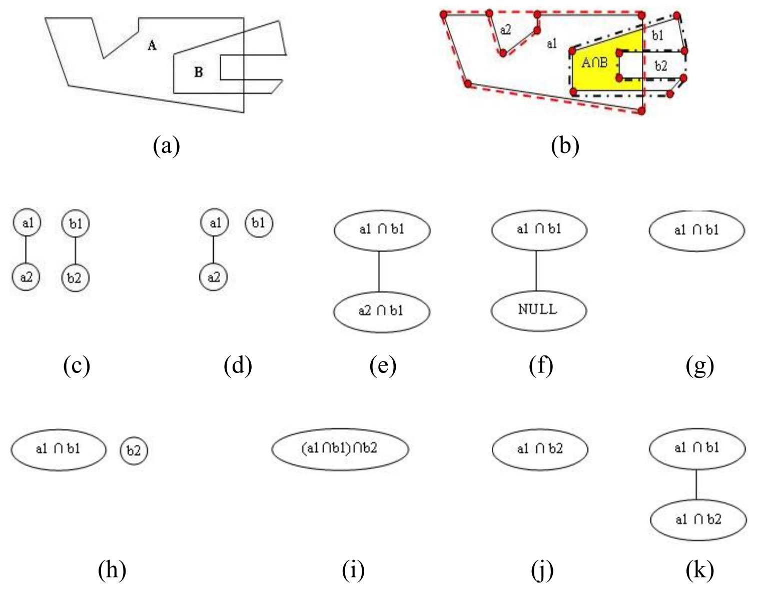


Fig. 6. Generic intersection operation

Since the intersection between *b1* and *a2* convex components is null (Figure 6e and Figure 6f), the result is a single node CHT (Figure 6g) having the convex hull (*a1*∩ *b1*) which is then intersected recursively with all the children trees of convex hull *b1* of the nonconvex polygon *B* . Since, *b1* has only one child (which is also a single node representing a convex hull) the intersection of ($a1 \cap b1$) and child tree (Figure 6h) results in a convex hull ($a1 \cap b2$) represented in a single node CHT as shown in Figure 6i and Figure 6j. The resultant CHT containing ($a1 \cap b2$) is then added (grafted) back to the parent intersection CHT containing single node ($a1 \cap b1$). Thus, the result is a nonconvex region shown as shaded in Figure 6 b and the resultant CHT shown in Figure 6k.

If the result of convex-nonconvex intersection is a set of disjoint regions (a set of CHTs, each representing a region) then for each CHT of the disjoint region, the same process is repeated independently of the other CHTs representing the disjoint intersection regions.

5 Algorithms

In this section we provide the pseudocode of algorithms for each of the three operations discussed in previous section. An example is shown with each algorithm to describe algorithm.

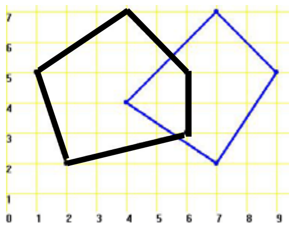
5.1 Pseudocode: Convex- Convex Intersection

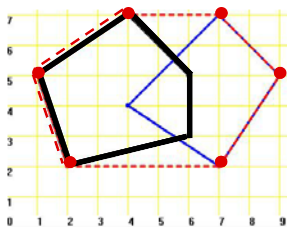
The pseudocode of the basic intersection operation between two convex polygons is shown in Figure 7. The algorithm uses the QuickHull algorithm (O'Rourke 1998) for convex hull computation and *compIntPoints* routine computes the intersection points by pairing the intersecting delta edges.

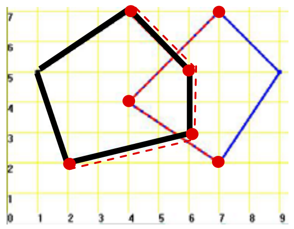
```
Input: Two convex polytopes (convex hulls)
1:   Initialize set of intersection region points $I$ as an empty set
2:   $ch \leftarrow$ convexHull ($poly1 + poly2$)
3:   $de \leftarrow$ ( ($poly1$ edges) + ($poly2$ edges)) – ($ch$ edges)
4:   $ip \leftarrow$ compIntPoints ($de$)
5:   $irp \leftarrow (poly1 - poly2) + (poly2 - poly1) + ip$
6:   $I \leftarrow$ convexHull ($irp$)
Output: Set of intersection region points $I$ (always a convex hull)
```

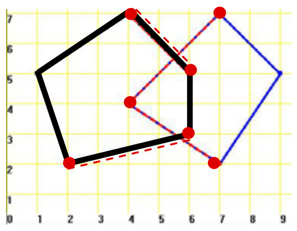
Fig. 7. Pseudocode of convex-convex intersection

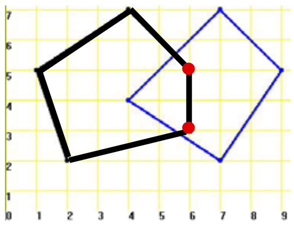
The algorithmic steps of basic intersection are shown in the example in Figure 8.

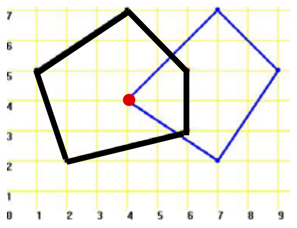

(a) Input convex polygons

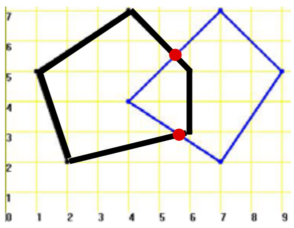

(b) Convex hull of (poly1+ poly2)


(c) Delta edges


(d) Intersecting edges


(e) Poly1 – poly2


(f) Poly2 – poly1


(g) Intersection points

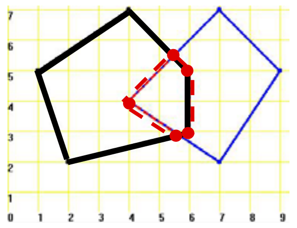

(h) Intersection region

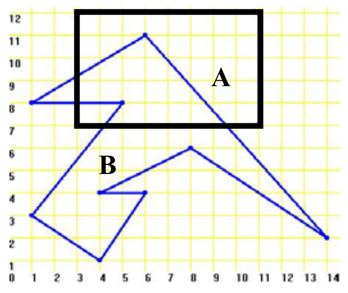
Fig. 8. Convex-convex intersection example

5.2 Pseudocode: Convex –Nonconvex Intersection

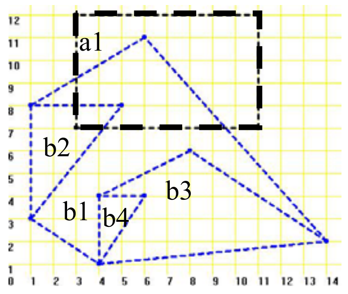
As mentioned earlier, the intersection operation between a convex and a nonconvex polygon is not closed under intersection. The pseudocode of the CNINT is shown in Figure 9 and the example is shown in Figure 10.

```
Input: A convex polytope (convex hull) poly1 and a nonconvex polytope poly2
(CHT). poly1 = ch and poly2 = Node x xts where x is the convex hull of poly2
and xts is the set of children CHTs
1:    Initialize I as an empty tree
2:    if xts is empty
3:      then
4:        t ← Node (intersection of ch and x) [ ]
5:        I ← set of trees containing only t
6:    else
7:        ct ← map (CNINT ch ) xts // set of trees by recursively traversing xts with (CNINT ch )
8:        it ← Node (CCINT of ch and x) ct
9:        eir ← process it // set of edges of intersection region obtained by processing it
10:       ir ← splitRegions eir // set of disjoint regions obtained by separating closed regions
11:       I ← build ir //set of trees obtained by building each region in ir
Output: Set of disjoint intersection regions I represented in CHT
```

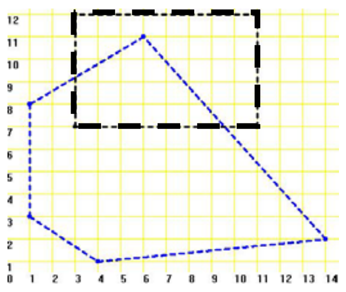
Fig. 9. Pseudocode of convex-nonconvex intersection algorithm



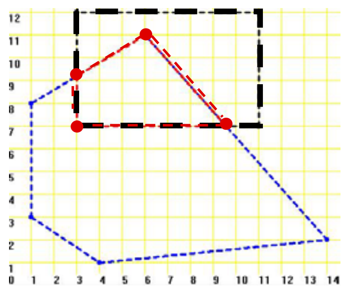
(a) Input polygons



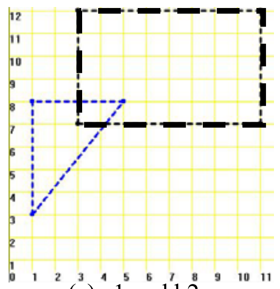
(b) Decomposed input



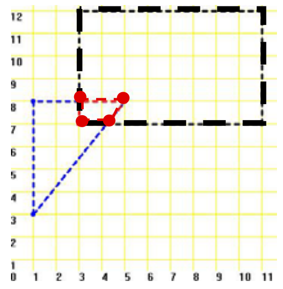
(c) a1 and b1



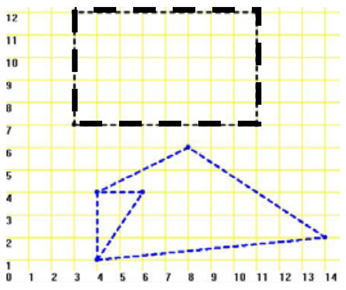
(d) a1∩ b1



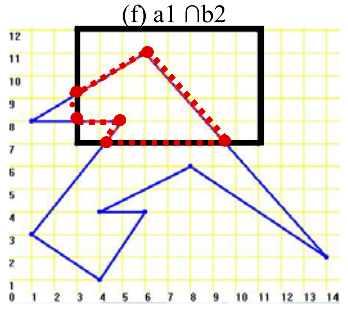
(e) a1 and b2



(f) a1 ∩b2



(g) a1∩b3 =NULL



(h) A∩B

Fig. 10. Convex-nonconvex intersection example

5.3 Pseudocode: Generic Intersection Operation

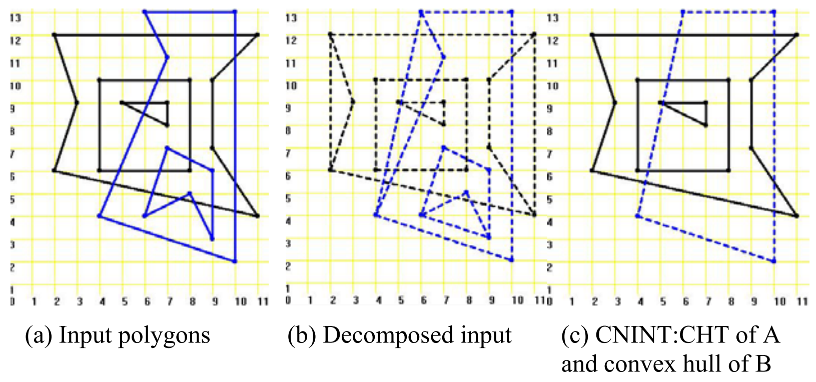
The algorithm for generic intersection operation incorporates the previous two intersection operations. It is generic in the sense that it computes the intersection of two polygons (convex or nonconvex and with or without holes). The pseudocode of the GINT is shown in Figure 11 and an example is shown in Figure 12.

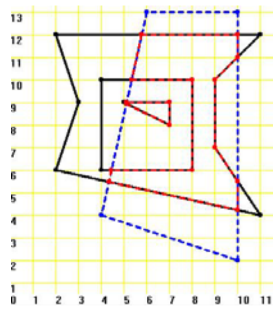
Input: Two polygonal regions (convex/nonconvex, simple or nonsimple) in *CHT* notation. Poly1 = Node *x xts* and poly2 = Node *y yts* where *x* and *y* are the convex hulls and *xts* and *yts* are the set of children trees of polygon 1 and polygon2 respectively

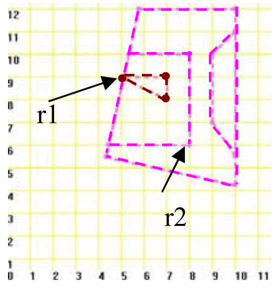
```
1:   Initialize I as an empty tree
2:   if xts is empty   //poly1 is convex
3:       then return CNINT x poly2
4:   else if yts is empty   //poly2 is convex
5:       then return CNINT y poly1
6:   else
7:       oi ← CNINT y poly1
8:       if oi is empty
9:           then return I
10:      else for every tree $t_x$ in oi
11:              for every tree $t_y$ in yts
12:                  ct ← compute GINT $t_x$ $t_y$
13:                  nt ← addtrees ct in tree $t_x$
14:                  I ← addtree nt into I
15:      return I
```

Output: Set of disjoint intersection regions *I* in *CHT*

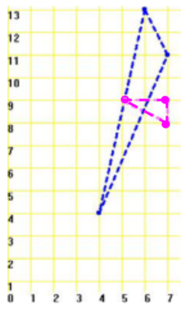
Fig. 11. Pseudocode of generic intersection algorithm

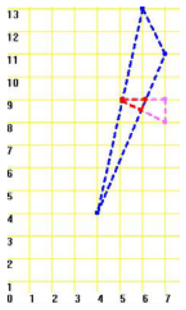

(a) Input polygons (b) Decomposed input (c) CNINT:CHT of A and convex hull of B

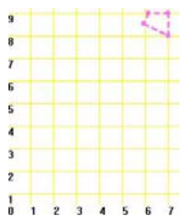

(d) Result of (c)

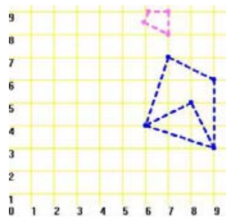


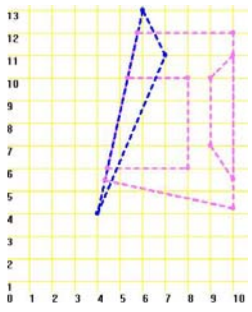
(e) result in (c) is simplified and two disjoint regions result

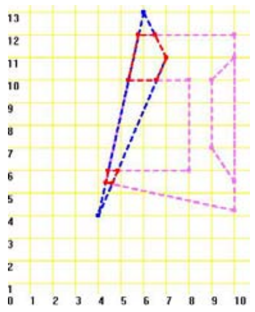

(f) r1 ∩ first child of B

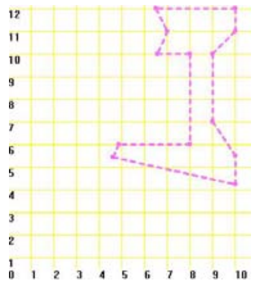

(g) Result of (f)

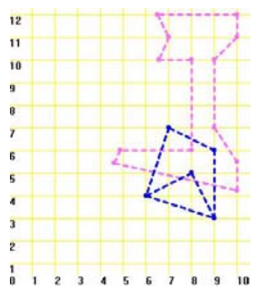

(h) Remove (g) from r1

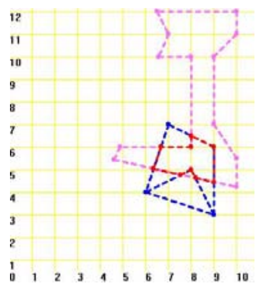

(i) Result in (h) ∩ second child tree of B

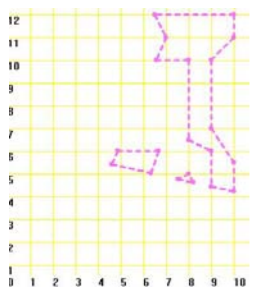

(j) r2 ∩ first child of B


(k) Result of (j)


(l) Remove result in (k) from r2


(m) Result in (l) ∩ second child tree of B


(n) Result of (m)


(o) Remove (n) from (l)

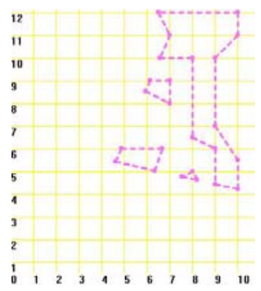

(p) Disjoint intersection regions

Fig. 12. Nonconvex-nonconvex intersection example

6 Implementation

The algorithms introduced in previous section are implemented in Haskell (Jones 2003). It is a functional programming language that supports lazy evaluation, higher order functions, and big numbers (big integers, big rational etc.). In Haskell, the CHT is defined recursively as;

data Tree = Node CHull [Tree]

The CCINT and CNINT operations are computed by recursively traversing the CHT of nonconvex polygon using *map* higher order function. Also, all the points are represented by their homogenous big integer coordinates. Big integers in Haskell are represented as “Integer” data type and they allow arbitrary precision arithmetic limited by the size of the main memory. Thus, the use of big integers allows robust intersection computations operations avoiding rounding errors.

7 Special Cases

The intersection of two polygons may be a point (Figure 13c), a line (Figure 13b), polygon or a combination consisting of the three (Figure 13f). In cases where the intersection region is not a polygon or a set of polygons, special treatment is needed. A variety of special cases for the basic intersection have been shown in (O'Rourke 1998) and few are shown in Figure 13.

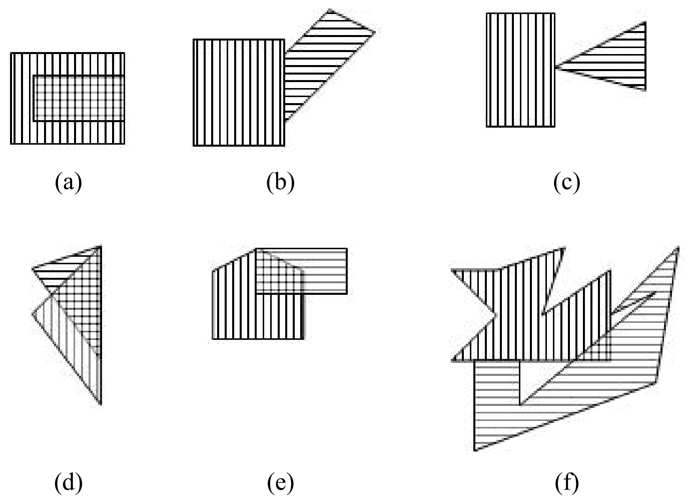


Fig. 13. Some special cases for intersection computation

Since our approach is based on basic intersection operation, we need not to tackle the special cases for convex-nonconvex or nonconvex-nonconvex intersection operations. If the special cases are tackled for basic operation, other operations based on it will also tackle the special cases.

For special cases like complete or partial overlap our approach do not need any special treatment. However, for cases when the intersection is a point or a line we need special treatment which is achieved by making changes not in the main intersection algorithms but in the *QuickHull* module and the *build* function of the AHD process. Table 2 lists special cases and whether these cases are specially treated in our algorithm or not.

Table 2. Special case treatment

Case	Specially treated?
Complete overlap	No
One is inside other	No
Edges overlap	No
Points overlap	No
Intersection is a line	Yes
Intersection is a point	Yes

8 Solution Characteristics

Our generic solution to perform Boolean intersection operations on general polygons has following properties;

1. Our approach is based on the basic intersection operation between two convex polygons. Thus it exploits the benefits associated with convexity.
2. It uses a single hierarchical tree data structure called *convex hull tree* which is an arbitrary tree of convex hulls. The data structure has two methods *build* and *process* for populating and accessing the stored data in the data structure.
3. The tree data structure allows reduced number of computations. We process the children trees of a node only if the intersection is not null. We proceed in a branch in CHT if the intersection is not null. Rule is "*an object A can not intersect with the component hulls of object B, if object A is not intersecting with the convex hull of object B*".
4. Our approach is robust as we are using homogenous big integer coordinates for representing points. Robustness is an important issue in geometric computations (David 1997). Most of the algorithms, as discussed in previous work section, do not cater the robustness issue. Only few address the issue (David 1997; Smith and Dodgson 2007).
5. In some applications, the result of Boolean operation may be needed to be decomposed e.g. Vatti (1992) mentions a case when

decomposed output is useful. So our approach is useful for applications where the decomposed output is needed in the form of convex components.

6. Our approach is easily implementable. The approach is implemented in Haskell and the code is compact having only 137 lines of code. Less code means less errors and ease of maintenance.
7. Most of the special cases need no special treatment like overlapping edges etc. Some require few modifications like when the result has dangling edges or points etc.
8. It is easily extendible for *n*-dimensions.

9 Suggestions for Improvement and Future Work

The approach can be improved by slight modifications. For example;

a) For cases where region is represented by multiple polygons we have to make the representation of a polygonal region as a list of convex hull trees rather than a tree.

b) For the self intersection cases, some preprocessing can be done involving the computation of intersection points of intersecting edges, updating the intersecting edges and then ultimately segregating the closed regions formed, so as to represent it with multiple simple polygons. The idea is to convert/decompose a nonsimple polygon with multiple simple polygons or use any established method to deal nonsimplicity first and then continue with our solution.

The future goal is to devise a solution which has following properties;

1) Supports other Boolean operations union and symmetric difference etc. We have implemented the union and symmetric difference operations based on AHD and the work will be presented soon.
2) Dimension independent having single implementation. For implementing the dimension independent Boolean operations on polytopes, the dimension independent AHD (Bulbul et al. 2009) will be used.

References

Bentley, J. L. and T. A. Ottmann. 1979. "Algorithms for Reporting and Counting Geometric Intersections." IEEE Computer Society.

Bernard, Chazelle and Edelsbrunner Herbert. 1992. "An optimal algorithm for intersecting line segments in the plane." Journal of the ACM (JACM) 39(1):1-54.

Bulbul, Rizwan and Andrew U. Frank. 2009. "AHD: The Alternate Hierarchical Decomposition of Nonconvex Polytopes (Generalization of a Convex Polytope Based Spatial Data Model)." In 17th International Conference on Geoinformatics. Fairfax, USA.

Bulbul, Rizwan, Farid Karimipour and Andrew Frank. 2009. "A Simplex based Dimension Independent Approach for Convex Decomposition of Nonconvex polytopes." In 10th lnternational Conference on GeoComputation (GeoComputation 2009). UNSW, Sydney, Australia.

Chazelle, B. and D. P. Dobkin. 1987. "Intersection of convex objects in two and three dimensions." Journal of the ACM (JACM) 34(1):1-27.

David, M. Mount. 1997. "Geometric intersection." In Handbook of discrete and computational geometry: CRC Press, Inc.

Fogel, Efi, Ron Wein, Baruch Zukerman and Dan Halperin. 2006. "2D Regularized Boolean Set-Operations." In In Cgal-3.2 User and Reference Manual, Cgal Editorial Board, Ed., http://www.cgal.org/Manual/3.2/doc_html/cgal_manual/Boolean_set_operations_2/Chapter_main.html.

FranciscoMartınez, Antonio Jesus Rueda and Francisco Ramo´n Feito. 2009. "A new algorithm for computing Boolean operations on polygons." Computers&Geosciences.

Greiner, Gunther and Kai Hormann. 1998. "Efficient Clipping of Arbitrary Polygons." ACM Transactions on Graphics (TOG) 17(2):71 - 83

Jones, Simon. 2003. Haskell 98 Language and Libraries: The Revised Report: {Cambridge University Press}.

Lauther, Ulrich. 1981. "An O (N log N) algorithm for Boolean mask operations." In Proceedings of the 18th conference on Design automation. Nashville, Tennessee, United States: IEEE Press.

Liu, Young Kui, Xiao Qiang Wang, Shu Zhe Bao, Matej Gambosi and Borut Zalik. 2007. "An algorithm for polygon clipping, and for determining polygon intersections and unions." Computers & Geosciences 33(5):589-598.

Margalit, Avraham and Gary D. Knott. 1989. "An Algorithm for Computing the Union, Intersection or Difference of two Polygons." Computers & Graphics 13:167-183.

Martınez, Francisco, Antonio Jesus Rueda and Francisco Ramo´n Feito. 2009. "A new algorithm for computing Boolean operations on polygons." Computers & Geosciences.

Nievergelt, J. and F. P. Preparata. 1982. "Plane-sweep algorithms for intersecting geometric figures." ACM.

O'Rourke, Joseph. 1998. Computational Geometry in C (Cambridge Tracts in Theoretical Computer Science): Cambridge University Press.

Peng, Yu, Jun-Hai Yong, Wei-Ming Dong, Hui Zhang and Jia-Guang Sun. 2005. "A new algorithm for Boolean operations on general polygons." Computers & Graphics 29(1):57-70.

Rappoport, Ari. 1991. "The n-dimensional extended convex differences tree (ECDT) for representing polyhedra." In Proceedings of the first ACM symposium on Solid modeling foundations and CAD/CAM applications. Austin, Texas, United States: ACM.

Rigaux, Philippe, Michel Scholl and Agnes voisard. 2001. Spatial Databases: With Applications to GIS: Morgan Kaufmann Publishers Inc. San Francisco, CA, USA.

Rivero, M. and F. R. Feito. 2000. "Boolean operations on general planar polygons." Computers & Graphics 24(6):881-896.

Shamos, Michael Ian and Dan Hoey. 1976. "Geometric intersection problems." In Proceedings of the 17th Annual Symposium on Foundations of Computer Science: IEEE Computer Society.

Smith, J. M. and N. A. Dodgson. 2007. "A topologically robust algorithm for Boolean operations on polyhedral shapes using approximate arithmetic." Butterworth-Heinemann.

Vatti, Bala R. 1992. "A Generic Solution to Polygon Clipping." Communications of the ACM 35(7):57-63.