Polygon Clipping and Polygon Reconstruction

Leonardo Guerreiro Azevedo
Ralf Hartmut Güting
(azevedo@cos.ufrj.br, rhg@fernuni-hagen.de)
Polygon clipping is an important operation that computers execute all the time.

Polygon clipping is the process of removing those parts of a polygon that lie outside a clipping window (Liang and Barsky, 1983).
An algorithm that clips a polygon is rather complex.

New edges may be added, and existing edges may be discarded, retained, or divided. Multiple polygons may result from clipping a single polygon.
The Cohen-Sutherland algorithm is based on region binary code (*outcode*) related to the window. The bits in the outcode represent: Top, Bottom, Right, Left.

For example, the outcode 1010 represents a point that is top-right of the window:
- top: 1
- bottom: 0
- right: 1
- left: 0

*Based on the outcode, we can perform trivial accept or reject.*
Both ending points are in the window
- (bitwise OR of ending points outcodes == 0): trivial accept

Both ending points are in the same part, which is not visible
- (bitwise AND of ending points outcodes != 0): trivial reject

Both ending points are in different parts (non trivial situation )
- the intersection of the outpoint and window border is then calculated (i.e. with the parametric equation for the line)
- this new point replaces the outpoint.
- the algorithm repeats until a trivial accept or reject occurs.
Both ending points are in the window
  ➢ (bitwise OR of ending points outcodes == 0): trivial accept

Both ending points are in the same part, which is not visible
  ➢ (bitwise AND of ending points outcodes != 0): trivial reject

Both ending points are in different parts (non trivial situation )
  ➢ the intersection of the outpoint and window border is then calculated (i.e. with the parametric equation for the line)
  ➢ this new point replaces the outpoint.
  ➢ the algorithm repeats until a trivial accept or reject occurs.
There are several well-known polygon clipping algorithms, each having its strengths and weaknesses. Sutherland-Hodgman algorithm is the oldest one (from 1974). It performs a clipping of a polygon against each window edge in turn. It accepts an ordered sequence of vertices $v_1, v_2, v_3, ..., v_n$ and puts out a set of vertices defining the clipped polygon.
Polygon clipping algorithms

- Some problems of Sutherland-Hodgman algorithm:
  - It does not work if the clip window is not convex.
  - If the polygon is not also convex, there may be some dangling edges.

- Liang-Barsky algorithm
  - more complicated,
  - but in certain cases fewer intersections need to be calculated than for Sutherland-Hodgman algorithm

- Weiler and Atherton algorithm
  - more complicated.
  - it allows clipping of a subject polygon by an arbitrarily shaped clip polygon.
  - applicable only in 2D.

- Even more ways to clip a polygon exist. None of them is totally perfect
  - vertices may disappear
  - a ghost vertex may be created.

- Therefore, the hunt for the perfect clipping algorithm is still open
Polygon reconstruction

- Polygon reconstruction is the process of reconstructing a polygon from a set of segments those are not in any specific order.
- For example, polygon’s segments may be stored in a list where one segment that follows another one does not have a common point.
In this work we propose new algorithms

- an algorithm for polygon clipping by a rectangle window
- an algorithm for polygon reconstruction from a set of “disconnected” segments

The algorithms

- do not assume any specific orientation of polygon’s segments
- do not rely on the computation of parity or wrap numbers of a reference point.
- each segment can be processed independent from the others
- handle polygons that have multiple boundaries and polygons with holes.

The algorithms were implemented in SECONDO a platform for implementing and experimenting with various kinds of data models.

It is developed as a research prototype at Fernuniversität in Hagen (Germany)
A halfsegment is just a segment for which the left or the right end point is considered as significant.

A halfsegment $h = (s, d)$ consists of an segment $s$ and a flag $d$ emphasizing one of the segment’s end points which is called the dominating point of $h$.

If $d = \text{left}$ then the left (smaller) end point of $s$ is the dominating point of $h$, and $h$ is called left halfsegment. Otherwise, the right end point of $s$ is the dominating point of $h$, and $h$ is called right halfsegment.

In SECONDO, each segment $s$ is mapped to two halfsegments $(s, \text{left})$ and $(s, \text{right})$. 
The polygon structure employed in this work was proposed by Güting et al. (1995) and Güting and Schneider (1995) and used in SECONDO.

Polygon are composed by faces. A face have a cycle and may have one or more holes.

Example: Polygon composed by three faces (f, f', and f'').
- Face f is composed by the cycle c and the hole h.
- Face f' is composed by the cycle c' and the holes h₁' and h₂'.
- Face f'' is composed by the cycle c'' and it has no hole.
In practice, a polygon is represented essentially as an ordered list (array) of halfsegments. The order used is the one suitable to support plane-sweep algorithms.

Each halfsegment has a set of attributes storing the cycle (or hole), the face that it belongs and the position of its halfsegment partner.
Sutherland-Cohen line clipping algorithm is used to clipping halfsegments against the window

- It is probably the most efficient method for trivial acceptance and rejection cases
- Clipping of one halfsegment is completely independent of other halfsegments clipping
- It is possible to employ a parallel implementation

Sutherland-Cohen line clipping algorithm may returns a clipped halfsegment or an intersection point

Intersection points between segments and edges must be handled, new halfsegments may be produced

Turning points
FOR each halfsegment $s$ in a set of segments DO
    SutherlandCohenLineClipping($s$, window, clippedHS, intersectionPoint, isIntersectionPoint));
    IF (isIntersectionPoint)
        evaluateTurningPoint(intersectionPoint);
    ELSE
        add clippedHS to the halfsegment output list;
        evaluateTurningPoint(clippedHS.leftPoint);
        evaluateTurningPoint(clippedHS.rightPoint);
    create new halfsegments from turning points;

Reconstructe the polygon;
Evaluate turning point

- **InsideAbove flag**
  - **True**: the area inside the polygon lies above the halfsegment; or, if the halfsegment is a vertical line, it means that the area inside the polygon is on the left of the half segment.
  - **False**: otherwise.

Halfsegments where *InsideAbove* is **true**

Halfsegments where *InsideAbove* is **false**
Evaluate turning point

- For each window edge the inside above flag is evaluated, and a direction for the turning point is set. This flag will be used to create new edges.
  - LEFT and RIGHT edges: up or down directions
  - TOP and BOTTOM edges: left or right directions
Evaluate turning point

- For each window edge the inside above flag is evaluated, and a direction for the turning point is set. This flag will be used to create new edges.
  - LEFT and RIGHT edges:
    - UP: IF insideAbove of the half segment the turning point belongs is true
    - DOWN: Otherwise

![Diagram of turning points with UP and DOWN flags](image)
Evaluate turning point

- For each window edge the inside above flag is evaluated, and a direction for the turning point is set. This flag will be used to create new edges.
  - TOP and BOTTOM edges: the inside above flag is not enough to set a direction
  - We also evaluate the position of halfsegment point that is above the turning point.
Evaluate turning point

- For each window edge the inside above flag is evaluated, and a direction for the turning point is set. This flag will be used to create new edges.
  - TOP and BOTTOM edges: the inside above flag is not enough to set a direction
  
- Since halfsegment’s point $P$ is above the turning point $T$.
  - IF halfsegment.insideAbove is TRUE
    - IF $T.x > P.x \rightarrow$ RIGHT
    - ELSE $\rightarrow$ LEFT
  
- ELSE
    - IF $T.x > P.x \rightarrow$ LEFT
    - ELSE $\rightarrow$ RIGHT
Create new Halfsegments from Turning Points

- At the end of turning point evaluation we will have:
  - a set of turning points for each edge
  - each turning point has its direction set
- So, we can create new halfsegments from those turning points
At the end of the clipping we have a set of halfsegments which do not have any information about which polygon’s part they belong to (face, cycle and cycle’s edge).

Polygon reconstruction algorithm sort polygon halfsegments in plane sweep order, and cross them adjusting properly the face number, cycle number, and edge number (polygon’s halfsegments attributes).
SECONDO: a platform for prototyping and teaching

- SECONDO is a generic environment supporting the implementation of database systems for a wide range of data models and query languages.
- It is extensible by algebra modules, using a well-defined interface.
  - Algebras group data types and operations.
- It supports new data models and data structures.
- It may be used by groups at universities to pursue and prototype their research ideas for database systems.
- Some examples of Algebras:
  - Standard Algebra
  - Relational Algebra
  - Spatial Algebra
  - Temporal Algebra
  - RTree Algebra
  - Midi Algebra
  - etc
Cooperation of SECONDO components

- GUI
- Optimizer
- Kernel
SECONDOPKernel architecture

GUI

Optimizer

Kernel architecture

Command Manager

Query Processor & Catalog

Alg$_1$, Alg$_2$, Alg$_n$

Storage Manager & Tools
Polygon clipping algorithms in SECONDO

Name: \texttt{windowclippingin}
Signature: (line x rect) $\rightarrow$ line
\hspace{1cm} (region x rect) $\rightarrow$ region
Syntax: \texttt{windowclippingin( _, _ )}
Meaning: computes the part of the object that is \textbf{inside} the window.
Example: query windowclippingin(trajectory(train7), bbox(thecenter))

Name: \texttt{windowclippingout}
Signature: (line x rect) $\rightarrow$ line
\hspace{1cm} (region x rect) $\rightarrow$ region
Syntax: \texttt{windowclippingout( _, _ )}
Meaning: computes the part of the object that is \textbf{outside} the window.
Example: query windowclippingout(trajectory(train7), bbox(thecenter))

\textbf{Polygon reconstruction} algorithm
\begin{itemize}
  \item It was implemented as a method of class Region in Spatial Algebra
  \item It is used by the clipping algorithms to compute the clipped polygons.
\end{itemize}
In this work, we proposed two new algorithms:
- polygon clipping by a rectangle window; and,
- polygon reconstruction.

The polygon reconstruction algorithm may be used in any case where it is needed to compute a polygon from an unordered set of segments. An example of application is polygon clipping.

Related to polygon clipping algorithm, we proposed and implemented two proposals:
- an algorithm to return the portion of the polygon that is inside a rectangle window; and,
- an algorithm to return the portion of the polygon that is outside the window.

Although our algorithm implementations in SECONDO has a good performance, we do not execute an experimental evaluation against other similar algorithms, which we plan to do as future work.
Polygon Clipping and Polygon Reconstruction

Leonardo Guerreiro Azevedo
Ralf Hartmut Güting
(azevedo@cos.ufrj.br, rhg@fernuni-hagen.de)