Polygon/Datacube Clipping in Database Queries

Peter Baumann[1,2], Dimitar Misev[1,2], and Bang Pham Huu[1,2]

[1] Constructor University, Bremen, Germany
[2] rasdaman GmbH, Bremen, Germany
{pbaumann,dmisev,bphamhuu}@constructor.university

Abstract. Multi-dimensional gridded data, also known as datacubes, appear not only in business and statistics OLAP datacubes, but in practically all science and engineering domains, too. Examples encompass in particular spatio-temporal datacubes in geo, genetics, brain imaging, cosmology, and many more sciences. As these typically constitute "Big Data" it is natural to consider database management techniques for their storage and analysis. In case of datacubes, this suggests Array Database Systems with corresponding array query languages such as the SQL/MDA standard.

However, typical datacube extraction operations focus on axis-parallel sub-cubes whereas in practice arbitrarily shaped regions are of interest, such as national boundaries or brain organs. It is desirable to consider polygonally bounded areas in queries, requiring polygon/raster clipping methods as long known in computer graphics.

In this contribution we describe a family of polygonal extraction operations on multi-dimensional arrays, named *polygon/datacube clipping*, which extends classical polygon/raster clipping from 2-D to n-D and adds novel functionality such as curtain and corridor extraction. From a user perspective, functionality is wrapped in only one additional function.

We present theory, query language embedding, implementation, and representative practical applications using the pioneer Array DBMS, rasdaman, which shows a very satisfying clipping performance. This opens the door for substantially enhanced datacube analytics in many application domains.

Keywords: polygon/datacube clipping · datacubes · curtain · corridor · rasdaman · SQL · WCPS.

1 Introduction

Multi-dimensional datacubes have gained substantial attention over the last years in many communities beyond classical OLAP business data analysis. It is acknowledged that datacubes represent a unifying paradigm in particular for spatio-temporal data as they appear in natural sciences and technology, such as 1-D timeseries, 2-D imagery, 3-D x/y/t image timeseries and x/y/z geophysical

and human brain data, 4-D x/y/z/t ocean, atmospheric, and cosmological data, to name but a few.

When it comes to Web services on massive arrays - so too large to download and process locally - region specification usually is restricted to rectangular shapes with axis-parallel boundaries. In many situations, however, it is desirable to offer such more general region descriptions, such as analysis of municipalities, countries, water body boundaries, etc. in geographical information systems (GIS) or human brain organs in Health research. Common methods include polygonal and polyhedral boundaries and bitmasks. Based on the seminal work done in Computer Graphics [24, 16, 14] scientific data analysis, GIS, and others since long supports polygon-based raster extraction in both clients [20] and servers [19]. Sometimes even polygon/polygon clipping resorts to first rasterizing and then intersecting polygon-bounded regions [11]. This is well established in 2-D, and efficient algorithms exist; challenges are different, though, on massive multi-dimensional raster data. At the same time, clipping can lead to massive data reductions over bounding boxes, as the example of a flight route from Bristol/UK to Geneva/CH illustrates in 3-D, even disregarding the time dimension. To the best of our knowledge, specifically on server-side no generalization of clipping to n-D cases nor any query language embedding has been published.

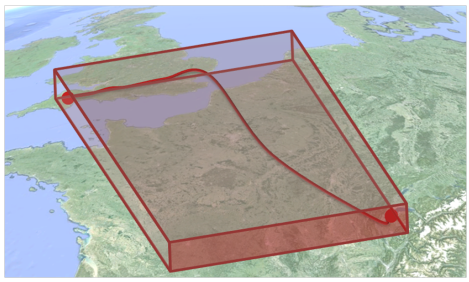

Fig. 1. Polygon line versus bounding box cutout from weather datacube (source: M. Dixson).

We present a novel operator, *polygon/datacube clipping*, for server-side polygon-based clipping on massive multi-dimensional datacubes. It extends classical 2-D polygon/raster clipping into n-D datacubes and additionally allows simplified extraction of non-planar regions ("curtains") and extraction of data along an arbitrary path ("corridors"). The algorithms are optimized for the processing of partitioned ("tiled") arrays. Over existing approaches it has several distinct advantages:

The operation proposed has a very simple signature: $clip(a, p)$ takes an array A and a polygon p and an array A and delivers an array commensurate with the polygon's overlap area, with all inside cells retaining their original values and all outside cells set to null (or inverse). Polygons can be planar, non-planar, and oblique in n-D space, and polygons and arrays can have different dimensions.

Two new operations, induced by the new multi-dimensional situation, are added under the same signature: curtains and corridors.

This approach is implemented in the pioneer Array DBMS, rasdaman [5], and available in open-source [28]. The operator is integrated in two high-level datacube query language standards, domain-neutral ISO SQL/MDA [18] and spatio-temporal OGC/ISO WCPS [7, 17]. Hence, aside from the spatio-temporal geo datacube candidates like satellite image timeseries, climate analysis and the like, clipping becomes available for further datacube domains like astrophysics, brain imaging, and genetics [3]. Sample real-world scenarios would include:

- 3-D x/y/t satellite datacube: How many wildfires have there been in Greece last year?
- 3-D x/y/z human brain datacube: How much electrical activity was measured in the brain in the Hippocampus region?
- 4-D x/y/z/t weather datacube: What is the weather forecast like along my flight track, in space and time?

We present the concepts and the array query operator and show that the operator embeds smoothly in standardized datacube query languages. The implementation, done in the rasdaman Array DBMS [3], is benchmarked considering several relevant situations: lines from 2-D and 3-D, polygons from 2-D, codimension 1 affine subspaces from 3-D, and curtains from 3-D data with both linear and polygonal cross sections. Performance observations are discussed based on the OGC datacube reference implementation, rasdaman.

The remainder is organised as follows. In the next section we recapitulate the state of the art. In Section 3, a polygon/datacube clipping query operator is introduced, with its special cases of line, polygon, curtain, and corridor queries. Subsequently, we present an exemplary implementation of these operators. Section 5 evaluates the concept through practically motivated examples and performance assessments. Section 6 concludes the paper.

2 State of the Art

Polygon clipping was first algorithmically tackled by Sutherland and Hodgman [24], Weiler and Atherton [30], and Vatti [29]. Rasterization and scanline clipping was added, e.g., by Newman and Sproull [16]. A hierarchical approach was contributed by Greene [13]. Modern fast clipping relies on methods like the Greiner-Hormann algorithm [14] and Tiled Raster Clipping [15].

In the GIS field, 2-D polygon clipping on rasters is widely used and available. With APRIL (Approximating Polygons as Raster Interval Lists), a technique is proposed which rasterizes polygons on a fine grid, represents raster cells as intervals, and encodes them for efficient cell/polygon overlap computation [11]. This appears highly efficient, yet specialized - for a generic query service we find it important to support a spectrum of functionalities and use cases, and balanced efficiency over all of them. Specifically for the common "filter-and-refine" pipeline in clipping, Georgiadis et al drive this further by distinguishing two sets

of intervals in the cells [12]. Again, the method mainly addresses polygon/polygon clipping and uses some fine raster (where "fine" is driven by the accuracy aimed at, not some given raster set) only for computational efficiency. We feel that there is no common granularity applicable to all sorts of problems, and in polygon/raster clipping the grid resolution anyway is given by the raster parameter. Hence, this approach is not helpful for our situation. Skala provides an overview of clipping and intersection algorithms for 2-D and 3-D cases [22]. In summary, no n-D generalization nor a query language embedding is published.

Most Array Databases do not support advanced operations like polygon clipping. For example, the SciDB documentation [21] does not contain such a function. Clipping and other more advanced spatial operations have to be handled through client-side code, complicating geospatial analysis for users and reducing efficiency, performance, and scalability. One exception is PostGIS Raster, an extension to PostgreSQL implemented through its object-relational extensibility mechanism [19]. This is known to have severe scalability and performance issues. From a user perspective, it is less convenient that the raster functionality is partly in SQL, and partly passed as query strings using a separate, different microsyntax. Also, tile handling is up to the user in PostGIS which complicates queries significantly. Only 2-D arrays are supported. A deep analysis on 19 array systems is given in [4].

In summary, to the best of our knowledge there is no service definition or implementation with n-D polygon/datacube clipping support which is both general and scalable, similar to what we introduce below.

3 Clipping on Datacubes

In this Section, we present polygon clipping on arrays conceptually. We address, step by step, line selection, polygons, curtains, and corridors. They all share a Big Data requirement arising from the partitioned ("tiled") arrangement of array storage in the server: a clipping algorithm must respect any given partitioning (regular or irregular), and only the minimum of tiles needed should be loaded into main memory. Ideally, any partition inspected should be loaded from persistent memory only once.

By way of terminology, we prefer *cell* over *pixel* or *voxel* to emphasize the n-D generality.

3.1 Line Selection

We start with single straight lines and the classical Bresenham Line Algorithm, which finds pixels on a 2-D raster along a line given by its end points, and generalize it to n-D.

Selecting cells along a line in a multidimensional datacube can be performed as follows. First, determine the tile containing the starting point and load it; proceed with the Bresenham line drawing algorithm until an entry is found outside the tile; load the correesponding, adjacent tile; continue iteratively until

the end point is reached. This method extends in a straightforward way to lines in n-D space. The output, a set of points in the cube, can conveniently be represented by a coordinate list or as a bit mask.

This algorithm obviously visits only relevant tiles, and each of them only once, so the key requirements are respected.

3.2 Polygon Selection

The extension from one line segment to a polygon is straightforward; however, additionally now the interior of closed polygon rings is to be determined.

Let a polygon ring P be defined by a set of p vertices $P = \{p_i\}_{i=0}^{p}$ where $p_0 = p_k$ and each p_i has n dimensions. Note that P does not have to be complanar, but can have arbitrary orientation and warps in n-D space. We randomly take the first line segment and perform the line algorithm from above, yielding the cells along that line. This is repeated for all line segments in turn. The output is a list of cells, which can be represented convenientla as either as a coordinate list or as a bit mask.

To determine the area covered by a polygon ring we first triangulate the polygon ring so as to obtain a set of planar regions where the interior cells can be found through common polygon filling algorithms, e.g., [1]. Special care has to be taken to avoid multiple tile loading caused by adjacent triangles.

The result, basically again a set of points in the cube, just significantly larger, can be represented by a coordinate list or, given the sizings, possibly more conveniently as a bit mask.

In practice, often not only rasterized polygons of thickness 1 are needed, but with some user-specified extent "left" and "right" along each axis. In GIS, such an extension of a line into an area of given extent is called a *buffer*. This is relevant, for example, in a flight plan where the flight path corridor is extended to inform the pilot also about what the situation is if there is a slightly different altitude, geographic offset, or earlier/later passing (Figure 2).

3.3 Curtains

We call a 2-D, not necessarily planar, region in an n-D cube a *curtain* if, for some given polygon line, the cells on that polygon are retrieved plus additionally along one axis direction all cells from lower to upper boundary. Figure 3 shows a curtain example along a baseline (left) and a trajectory along time walking through the cube, such as a flightroute might do.

Algorithmically, each line segment can be replaced by a rectangle where the "curtain coordinate" is extended up and down to the boundaries of the array. A triangulation could be performed, but is unnecessary as each resulting rectangle is planar. With this preparation, evaluation can continue as described above.

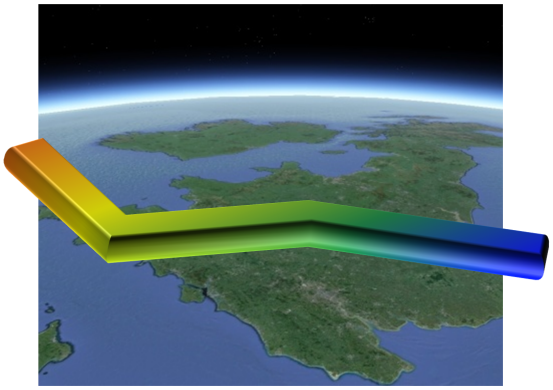

Fig. 2. Schematic 3-D corridor (source: M. Dixson, UK MetOffice).

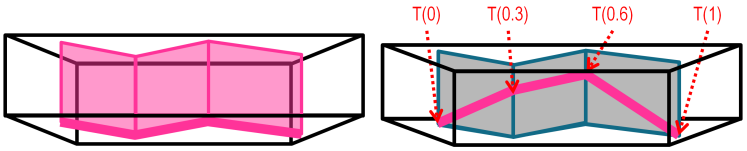

Fig. 3. Schematic 3-D curtains.

4 Implementation

Polygon/datacube clipping is implemented [28] in the rasdaman Array DBMS [27], a full-stack database system specialized on the management and analytics of massive multi-dimensional arrays. Effectively, rasdaman has coined the field of Array Databases and datacube services [2, 5]. The rasdaman query language, *rasql*, has been adopted into the SQL standard as Part 15, Multi-Dimensional Arrays (MDA) [18]. Datacube query semantics relies on Array Algebra [6] for processing and optimization, as well as parallelization and location-transparent distributed processing.

Storage in rasdaman relies on tiling of large arrays into sub-arrays forming the unit of access in the server [10]. Arbitrary tiling patterns can be specified per datacube by the administrator through an integrated Storage Layout Language allowing administrators to define arbitrary regular and irregular tiling patterns. Therefore, it is important that clipping does not make any assumptions about the array's tiling.

A geo semantics layer on top of the core rasdaman server supports the Open Geospatial Consortium (OGC) Web Coverage Processing Service (WCPS) geo datacube query language OGC [7] and ISO [17]. In rasdaman, WCPS queries internally get translated into rasql queries. Internally, clip queries in WCPS get mapped to corresponding clipping in rasql.

The rasdaman DBMS is operational on 7+ Petabyte of Earth data, such as in the global EarthServer datacube federation[26] and with single queries parallelized across more than 1,000 AWS cloud nodes [9].

Polygons can self-intersect, and multi-polygons are possible (such as for areas with many lakes). Syntax relies on Well-Known-Text (WKT) where a sequence of closed rings $R_1, ..., R_n$ is written as

```
polygon( (R1), ..., (Rn) )
```

The first polygon R_1 defines the exterior border of the polygon and the optional remaining rings R_i define holes in the R_1 area. Aside from polygon rings, it is possible to indicate multi-polygons, i.e., lists of polygons, again following WKT syntax:

```
multipolygon( ((R_1)), ..., ((R_m)) )
```

Function $clip(a, p)$ accepts an n-D array expression a and an n-D polygon or multi-polygon p of same dimension (except for curtains). The array cell type must be Boolean for an easy decision of the clop algorithm; notably, this can always be achieved through some Boolean array expression such as a threshold:

```
clip( T.a > threshold, polygon((0 0 0, 0 10 0, 0 10 10)) )
```

The output is an array of the same size and dimension as a, with all points selected by the (multi-) polygon retaining the values of a and all others set to *null*.

5 Application Examples

Below we present some real-life examples; see the public demo [25] for interactive clipping.

Example 1: 2-D polygon-bounded analytics. The WCPS query below clips a Digital Terrain Model (DTM) of Germany, extracting the state of Thuringia (Figure 4). The inside pixels get colored, outside pixels are set to null and appear transparent against the background map.

```
for $c in (Germany_DTM)
return encode( clip( $c, polygon(( ... )) ), "png", "<coloring>" )
```

Example 2: 1-D polyline extraction. From the same DTM, the next query extracts a route (Figure 5 left) - say, to determine how steep a bicycle tour through Germany would be (Figure 5 right).

```
for $c in (Germany_DTM)
return encode( clip( $c, linestring( ... ) ), "csv")
```

Example 3: Line string extraction from 4-D datacube. In this aviation scenario, the pilot needs weather information for flight planning [8] from 4-D weather forecast datacubes (Figure 6). An actual flight route obtained from flightradar24 is used for clipping; note the 4-D space/time coordinates in the LineString. The resulting diagrams exemplarily show wind speed and lighting,

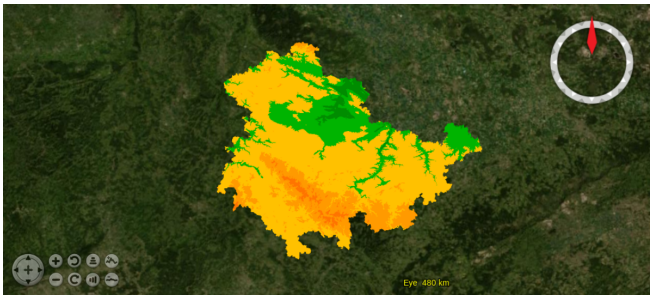

Fig. 4. Example 1: Standard GIS map clipping.

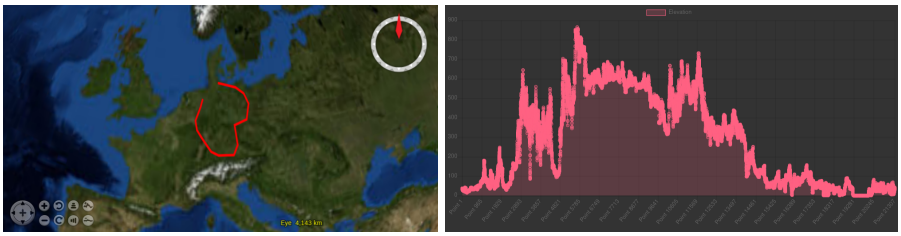

Fig. 5. Example 2: Line extraction on map (left) and resulting values (right).

at the location and altitude of the aircraft and at the time it will pass by the points. As can be seen, towards the end of the flight severe weather is to be expected. Notably, the datacubes encompass about 31 TB of data whereas the flight preview has about one Kilobyte, so can be transferred to the cockpit (and updated) easily.

```
for $c in (Wind)
return
  encode(
    clip(
      sqrt(pow($c.u, 2.0) + pow($c.v, 2.0)),
      LineString(
        "2024-08-01T02:00" 5000  53.3899 -6.2183,
        ...
        "2024-08-01T05:00" 35990 48.2137 6.3696
      )
    ),
    "json"
  )
```

6 Evaluation

In this Section, we analyze the performance behavior of various types of clipping queries. Measurements were performed using rasdaman v10 on Ubuntu 22.04 running on an Intel core i7 processor and 8GB of RAM. Benchmarks were run using `/usr/bin/time` with five runs and averaging measurements obtained over the runs. All arrays were fully initialized with random float values, thereby disabling rasdaman's capability of partially materialized arrays. Tiling was chosen

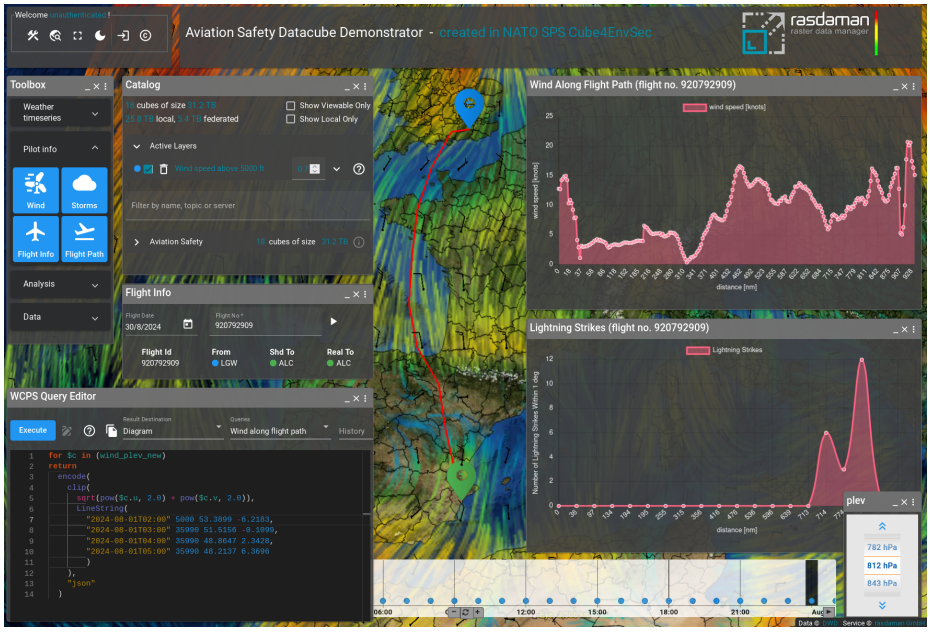

Fig. 6. Wind along aircraft route.

symmetric in dimensions with a tile size of about 1 MB as earlier benchmarks have shown that this is a suitable size on standard hardware architectures [23]. Also, any compression was switched off so that the footprint is 1:1 the size one would expect from the array extents. All times indicated in the tables are in seconds, except if otherwise indicated.

We conduct a series of experiments with increasing clipping complexity and capture performance as well as RAM and CPU use. Emphasis is placed not on the complexity of the polygons - such research has been done elsewhere [24][1][29] - but on volumetric scalability aspects, in particular: polygons spanning multiple tiles.

Experiment 1: Line extraction. First, we extract cells along a line from a 2D array ("line selection") and observe the efficiency of oblique selections with varying angle. The dataset consists of a 10,000 x 10,000 array of floats of size 400 MB, with tile size 1MB. Hence, the array is composed of 400 tiles altogether.

The line start is anchored at the arrays origin at (0,0) while the end point initially is at the top-left (so axis-parallel) and walks to the top-right corner (so diagonal) in increments of 1000 pixels (Table 6). The first line reflects a "flat", axis-parallel clipping - the classic datacube extraction. Incrementally, the selection line gets an increasing angle, ultimately cutting diagonally through the plane. A factor of 3.24 performance penalty is observed between axis-parallel and diagonal case.

Experiment 2: Flat polygon clipping. A polygon ring is used to designate the area to be extracted from the 2-D test array. Test polygons are star-shaped

Start point	End point	Avg (ms)
(0, 0)	(0, 9999)	55.3
(0, 0)	(999, 9999)	60.1
(0, 0)	(1999, 9999)	70.6
(0, 0)	(2999, 9999)	85.2
(0, 0)	(3999, 9999)	97.6
(0, 0)	(4999, 9999)	109.3
(0, 0)	(5999, 9999)	125.4
(0, 0)	(6999, 9999)	140.2
(0, 0)	(7999, 9999)	156.8
(0, 0)	(8999, 9999)	168.9
(0, 0)	(9999, 9999)	179.7

Table 1. Runtime of 2-D line clipping

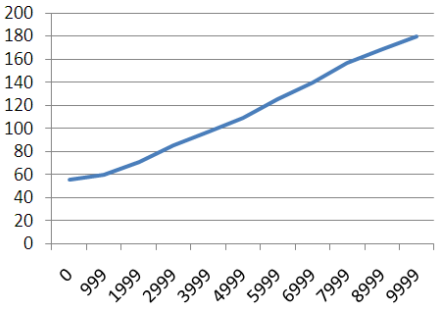

Fig. 7. Line clipping runtime [sec].

with n corners and $2n$ edges, reaching out to the extent limits of the test array. Table 6 lists the cases tested and results obtained.

Experiment 3: Oblique polygon clipping. A 3-D array of extent 250 x 1000 x 1000 was used, forming a 1 GB datacube partitioned into about 1,000 3-D tiles of size 100 x 100 x 100 each. Test polygons are triangles with two coordinates matching the same array plane while shifting the third coordinate up along the third array dimension. In Table 6, the shift offset of this third coordinate is indicated in column 1, giving a measure of the obliqueness of the clip triangle.

Corners	Avg (s)	RAM (GB)	CPU (%)
6	3.41	1.6	75.4
12	3.40	1.6	74.8
24	3.46	1.7	74.6
48	3.75	1.8	74.4
96	3.48	2.0	75.2

Table 2. 2-D polygon clipping runtime.

Corners	Avg (s)	RAM (MB)	CPU (%)
0	0.60	152.6	95.2
249	1.84	387.02	85.2
499	3.01	623.3	89.4
749	4.44	977.05	90.2
999	6.46	1213.48	79.6

Table 3. 3-D polygon clipping runtime.

Experiment 4: Curtain clipping. This test relies on the same triangular polygons as above, with height offsets of 0, 249, 499, 749, and 999.

7 Conclusion

We have presented a novel query operator for generalized polygon clipping in Array Databases,. Polygons can contain holes and can be self-intersecting. The expressive power includes n-D flat and oblique clipping, with curtains and corridors as special cases, as well as subspace clipping (topic of a separate paper). This operator naturally integrates with the SQL standard datacube extension,

Height	Avg runtime (s)	RAM (MB)	CPU (%)
0	0.61	152.9	94.8
249	3.50	406.8	93.1
499	7.01	658.0	88.9
749	10.29	906.1	87.0
999	13.77	1159.3	74.5

Table 4. 3-D curtain query runtime.

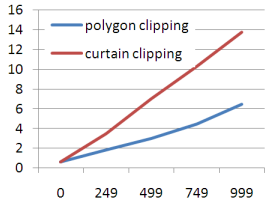

Fig. 8. 3-D polygon and curtain clipping runtime.

MDA, as well as the geo datacube query language standard, OGC/ISO WCPS. The resulting easy-to-use $clip(a, p)$ operator is implemented in open-source *rasdaman community* [27] and documented in detail [28].

While the task of polygon/raster clipping is well-known and solved since decades, our approach brings along several novelties:

- Designed to work on n-D datacubes.
- Additional practically relevant functionality, such as curtains and corridors.
- Easy to use: same syntax for all cases - server dispatches automatically.
- Scalable, amenable to existing server optimizations and parallelization.
- Embedded in high-level query languages, allowing expressions of arbitrary complexity.

Future work includes proposing polygon/datacube clipping for inclusion in the WCPS standard. Further, a generalization to polyhedrally bounded regions is under development.

Altogether, we believe that this functionality can significantly enhance the quality of service of datacube engines in general and Array Databases specifically.

Acknowledgements In this work, Peter Baumann has contributed the initial idea and first concept as well as writing, funding acquisition, and project administration; Dimitar Misev has strongly contributed to concepts development and implemented clipping in rasql; Bang Pham Huu has created WCPS clipping implementation and demos; the authors gratefully acknowledge the mathematical background and trial code contribution by Brennan Bell.

Data and Software Availability The source code is available in the *rasdaman community* repository, with DOI 10.5281/zenodo.1040170. The demonstration examples are publicly available at https://standards.rasdaman.com and https://cube4envsec.org. No AI has generated any of the contents of this paper.

Disclosure of Interests The authors have no competing interests to declare that are relevant to the content of this article, a polygon/datacube query operator in open-source rasdaman community.

References

1. Anitha, S., Evangeline, D.: An efficient fence fill algorithm using inside-outside test **3** (November 2013)
2. Baumann, P.: Language support for raster image manipulation in databases (1992), intl. Workshop on Graphics Modeling, Visualization in Science & Technology
3. Baumann, P.: The rasdaman array dbms: Concepts, architecture, and what people do with it (invited keynote). In: 34th Intl. Conf. on Scientific and Statistical Database Management (SSDBM) (July 2022). https://doi.org/10.1145/3538712.3543825
4. Baumann, P., Misev, D., Merticariu, V., Pham, B.: Array databases: concepts, standards, implementations. Springer Journal Big Data **28i**(8) (2021). https://doi.org/10.1186/s40537-020-00399-2
5. Baumann, P.: Management of multidimensional discrete data. VLDB Journal **3**(4), 401–444 (1994), https://dl.acm.org/citation.cfm?id=615204.615207
6. Baumann, P.: A database array algebra for spatio-temporal data and beyond (1999), https://dl.acm.org/citation.cfm?id=646411.692530, proc. 4th Intl. Workshop on Next Generation Information Technologies and Systems
7. Baumann, P.: Web coverage processing service (wcps) language interface standard (2021), https://docs.ogc.org/is/08-068r3/08-068r3.html
8. Cube4EnvSec: Cube4envsec: Geo insight when you need it, where you need it., https://cube4envsec.org/
9. Dumitru, A., Merticariu, V., Baumann, P.: Exploring cloud opportunities from an array database perspective (2014), aCM SIGMOD Workshop on Data Analytics in the Cloud (DanaC)
10. Furtado, P., Baumann, P.: Storage of multidimensional arrays based on arbitrary tiling (1999), intl. Conf. on Data Engineering (ICDE)
11. Georgiadis, T., Mamoulis, N.: Raster intervals: An approximation technique for polygon intersection joins. Proc. ACM Manag. Data **1**(1) (May 2023). https://doi.org/10.1145/3588716
12. Georgiadis, T., Tzirita Zacharatou, E., Mamoulis, N.: Raster interval object approximations for spatial intersection joins. The VLDB Journal **34**(1) (Dec 2024). https://doi.org/10.1007/s00778-024-00887-4
13. Greene, N.: Hierarchical polygon tiling with coverage masks (1996), proc. SIGGRAPH
14. Greiner, G., Hormann, K.: Efficient clipping of arbitrary polygons. ACM Trans. Graph. **17**(2), 71–83 (Apr 1998). https://doi.org/10.1145/274363.274364
15. Hasselgren, J., Akenine-Möller, T.: Efficient depth buffer compression. In: Proceedings of the 21st ACM SIGGRAPH/EUROGRAPHICS Symposium on Graphics Hardware. p. 103–110. GH '06, Association for Computing Machinery, New York, NY, USA (2006). https://doi.org/10.1145/1283900.1283917
16. Newman, W.M., Sproull, R.F.: Principles of Interactive Computer Graphics. McGraw-Hill (1973)
17. n.n.: Geographic information - schema for coverage geometry and functions - part 3: Processing fundamentals (2023), https://committee.iso.org/sites/tc211/home/projects/projects—complete-list/iso-19123-3.html
18. n.n.: Iso/iec 9075-15:2023: Information technology database languages — sql — part 15: Multi-dimensional arrays (sql/mda) (2023), https://www.iso.org/standard/84807.html

19. PostGIS: Postgis st_clip, https://postgis.net/docs/RT_ST_Clip.html
20. QGIS: Qgis, https://qgis.org/download
21. SciDB: Scidb documentation, https://paradigm4.atlassian.net/wiki/spaces/scidb/overview
22. Skala, V.: A brief survey of clipping and intersection algorithms with a list of references (including triangle-triangle intersections). Informatica **34**(1), 169–198 (2023). https://doi.org/10.15388/23-INFOR508
23. Stancu-Mara, S., Baumann, P.: A comparative benchmark of large objects in relational databases. In: Intl. Symp. on Database Engineered Applications (IDEAS). Coimbra, Portugal (2008)
24. Sutherland, I.E., Hodgman, G.W.: Reentrant polygon clipping. Communications of the ACM **17**(1), 32 – 42 (1974)
25. rasdaman team: Earth datacube playground, https://standards.rasdaman.org/demo_clipping.html
26. earthserver team: The earthserver datacube federation, https://earthserver.world
27. rasdaman team: rasdaman, https://rasdaman.org
28. rasdaman team: rasdaman documentation, https://doc.rasdaman.com
29. Vatti, B.: A generic solution to polygon clipping. Communications of the ACM **35**(7), 56 – 63 (1992)
30. Weiler, K., Atherton, P.: Hidden surface removal using polygon area sorting. Computer Graphics **11**(2), 214 – 222 (1977)