# Implementing efficient geo queries for the SPARQL engine QLever

Jonathan Zeller

Albert Ludwigs Universität Freiburg

March 17, 2025

# Introduction

# Introduction

Geo Query example: Search for nearby restaurants

- at most 1 km to the current position
- at most 300 m to a tram station

# Introduction - Query for nearby restaurants

```
PREFIX osmkey:<https://www.openstreetmap.org/wiki/Key:>
PREFIX geo:<http://www.opengis.net/ont/geosparql#>
PREFIX spatialSearch:<https://qlever.cs.uni-freiburg.de/spatialSearch/>
SELECT ?nameRes
(MIN(?dist) AS ?distanceToRestaurant)
(MIN(?distTramRes) AS ?distanceTramToRestaurant) WHERE {
  # current position
  ?uniBib osmkey:name "Universitätsbibliothek" .
  ?uniBib osmkey:addr:postcode "79098" .
  ?uniBib osmkey:addr:street "Platz der Universität" .
  ?uniBib geo:hasGeometry ?geoUniBib .
  ?geoUniBib geo:asWKT ?wktUniBib .
  # search for restaurants
  ?restaurant osmkey:amenity "restaurant" .
  ?restaurant osmkey:name ?nameRes .
  ?restaurant geo:hasGeometry ?geoRes .
  ?geoRes geo:asWKT ?wktRes .
  # search for tram stations
  ?tramStation osmkey:railway "tram_stop" .
  ?tramStation osmkey:name ?nameTram .
  ?tramStation geo:hasGeometry ?geoTram .
  ?geoTram geo:asWKT ?wktTram .
  # restaurants near the current position
  Service spatialSearch: {
  _:config spatialSearch:algorithm spatialSearch:boundingBox;
  spatialSearch:left ?wktUniBib;
  spatialSearch:right ?wktRes;
  spatialSearch:maxDistance 1000;
  spatialSearch:bindDistance ?dist . }
  # restaurants near tram stations
  Service spatialSearch: {
  _:config spatialSearch:algorithm spatialSearch:boundingBox;
  spatialSearch:left ?wktRes;
  spatialSearch:right ?wktTram;
  spatialSearch:maxDistance 300;
  spatialSearch:bindDistance ?distTramRes . }
} GROUP BY ?nameRes
ORDER BY ASC(?distanceToRestaurant) ?distanceTramToRestaurant
```

# Intoduction - Query results

Query results:

140 lines found | 2,769ms in total | 2,768ms for computation | 1ms for resolving and sending

Limited to 100 results; show all 140 results

| | ?nameRes | ?distanceToRestaurant | ?distanceTramToRestaurant |
|---|---|---|---|
| 1 | UNI Galerie | 0.0435082 | 0.168826 |
| 2 | Ristorante Pizzeria Bürgerstube | 0.120112 | 0.139919 |
| 3 | La Culinaria im Theater | 0.123929 | 0.0271464 |
| 4 | La Pepa | 0.133394 | 0.208734 |
| 5 | Greencity | 0.147808 | 0.00966177 |
| 6 | Olivia | 0.197508 | 0.0702376 |
| 7 | Mensa Rempartstraße | 0.203003 | 0.0706977 |
| 8 | Muho | 0.204744 | 0.0822748 |
| 9 | Mai Wok | 0.209249 | 0.0904129 |
| 10 | Schwarzer Kater | 0.211607 | 0.140932 |

# Introduction - Subquery for Restaurants

OSM-ZELLER | Index Information | Backend Information | Shortcuts / Help

```
1 PREFIX osmkey:<https://www.openstreetmap.org/wiki/Key:>
2 PREFIX geo:<http://www.opengis.net/ont/geosparql#>
3 SELECT ?nameRes ?wktRes WHERE {
4   ?restaurant osmkey:amenity "restaurant" .
5   ?restaurant osmkey:name ?nameRes .
6   ?restaurant geo:hasGeometry ?geoRes .
7   ?geoRes geo:asWKT ?wktRes .
8 }
```

Execute | Download | Share | Reset | Clear cache | Analysis | Examples | 3. Context sensitive suggestions

Automatically add names to result

## Query results:

100,824 lines found | 2,110ms in total | 1,034ms for computation | 1,076ms for resolving and sending

Limited to 100 results; show all 100,824 results

| | ?nameRes | ?wktRes |
|---|---|---|
| 1 | Culinario | POINT(6.837417 51.094770) |
| 2 | M. Rhein Grill & Pizzeria | POINT(6.889274 51.089559) |
| 3 | Fiume | POINT(6.885111 51.100551) |
| 4 | Landhaus Krombach | POINT(7.887200 50.725832) |
| 5 | Imperiale | POINT(7.650335 49.783001) |

# Introduction - Subquery for Tram Stations

OSM-ZELLER | Index Information | Backend Information | Shortcuts / Help

```
1 PREFIX osmkey:<https://www.openstreetmap.org/wiki/Key:>
2 PREFIX geo:<http://www.opengis.net/ont/geosparql#>
3 SELECT ?nameTram ?wktTram WHERE {
4   ?tramStation osmkey:railway "tram_stop" .
5   ?tramStation osmkey:name ?nameTram .
6   ?tramStation geo:hasGeometry ?geoTram .
7   ?geoTram geo:asWKT ?wktTram .
8 }
9
```

Execute | Download | Share | Reset | Clear cache | Analysis | Examples | 3. Context sensitive suggestions

Automatically add names to result

## Query results:

10,304 lines found | 425ms in total | 200ms for computation | 225ms for resolving and sending

Limited to 100 results; show all 10,304 results

| | ?nameTram | ?wktTram |
|---|---|---|
| 1 | Alfred-Kowalke-Straße | POINT(13.520201 52.504716) |
| 2 | Rothenbachstraße | POINT(13.437708 52.570223) |
| 3 | Stadionbrücke | POINT(9.722838 52.357666) |
| 4 | Geschwister-Scholl-Straße | POINT(11.962701 51.507604) |

# Introduction - spatial join

Table: First spatial join input

| ?nameRes | ?wktRes |
|---|---|
| UNI Galerie | POINT(7.844367 47.994072) |
| Ristorante Pizzeria Bürgerstube | POINT(7.843553 47.994797) |
| Restaurant in Hamburg | POINT(53.500000 10.000000) |

Table: Second spatial join input

| ?stationName | ?wktTramStation |
|---|---|
| Stadttheater | POINT(7.846133 47.995821) |
| Erbprinzenstraße | POINT(7.845330 47.992305) |
| Tram Station in Berlin | POINT(52.500000 13.500000) |

# Introduction - spatial join

| ?nameRes | ?wktRes |
|---|---|
| UNI Galerie | POINT(7.844367 47.994072) |
| Ristorante Pizzeria Bürgerstube | POINT(7.843553 47.994797) |
| Restaurant in Hamburg | POINT(53.500000 10.000000) |

| ?stationName | ?wktTramStation |
|---|---|
| Stadttheater | POINT(7.846133 47.995821) |
| Erbprinzenstraße | POINT(7.845330 47.992305) |
| Tram Station in Berlin | POINT(52.500000 13.500000) |

Table: Result of the spatial join operation

| ?nameRes | ?wktRes | ?stationName | ?wktTramStation | ?dist |
|---|---|---|---|---|
| UNI Galerie | POINT(7.844367 47.994072) | Stadttheater | POINT(7.846133 47.995821) | 0.23 |
| UNI Galerie | POINT(7.844367 47.994072) | Erbprinzenstraße | POINT(7.845330 47.992305) | 0.21 |
| Ristorante Pizzeria Bürgerstube | POINT(7.843553 47.994797) | Stadttheater | POINT(7.846133 47.995821) | 0.14 |
| Ristorante Pizzeria Bürgerstube | POINT(7.843553 47.994797) | Erbprinzenstraße | POINT(7.845330 47.992305) | 0.29 |

# The Spatial Join Algorithm(s)

# The Baseline Algorithm

**The Baseline Algorithm:**

```
subResult1 = getSubResultLeft()
subResult2 = getSubResultRight()
result = createEmptyResultTable()
for row1 in subResult1:
    for row2 in subResult2:
        d = distance(row1.getGeometry(), row2.getGeometry())
        if d < maxDistance:
            result.add(row1, row2, d)
```

# The Bounding Box Algorithm - basic idea

Basic idea of the Bounding Box Algorithm

- Avoid checking each pair of entries, by using a filter
- Put the points of one sub-result in an efficient data structure
- Query the data structure with the points of the other sub-result
- Check the distance only for the filtered query results
- Use a filtering, that guarantees that no results are missed

# The Bounding Box Algorithm - basic idea

Basic idea applied to the example from the introduction

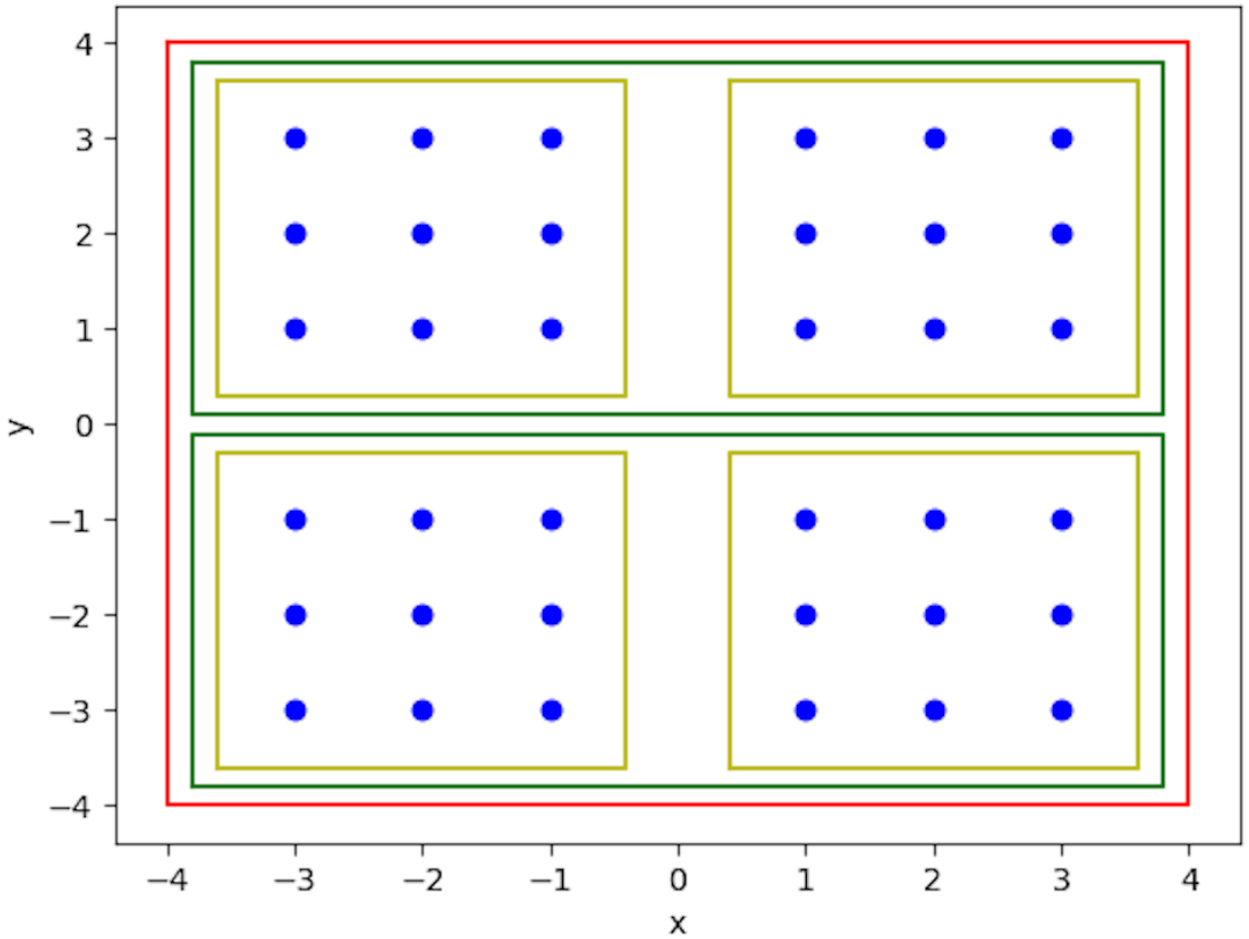
- Build the data structure for the tram station
- For each restaurant, check only the tram stations, that are in the same city as the restaurant
- Avoid calculating all distances between the restaurant and the tram stations in other cities

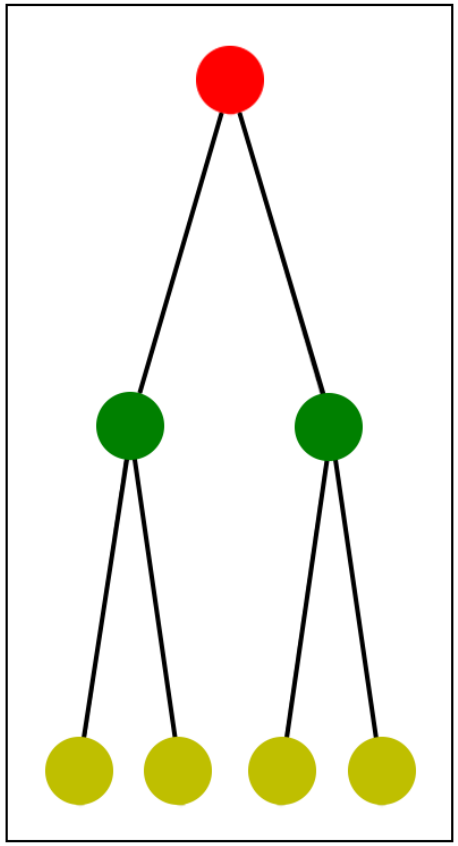
# R-trees

R-trees

- Hierarchical tree-like structure
- Stores nearby points in a node
- The children of a node partition the points of the node
- Can be queried using a rectangle, which returns all points, that are inside the rectangle

# R-trees



# Query box

- Given a maximum distance $m$ and a start point
- Construct a box large enough to contain all points within $m$ metres of the start point
- Consider the spherical geometry of the Earth

# Query box

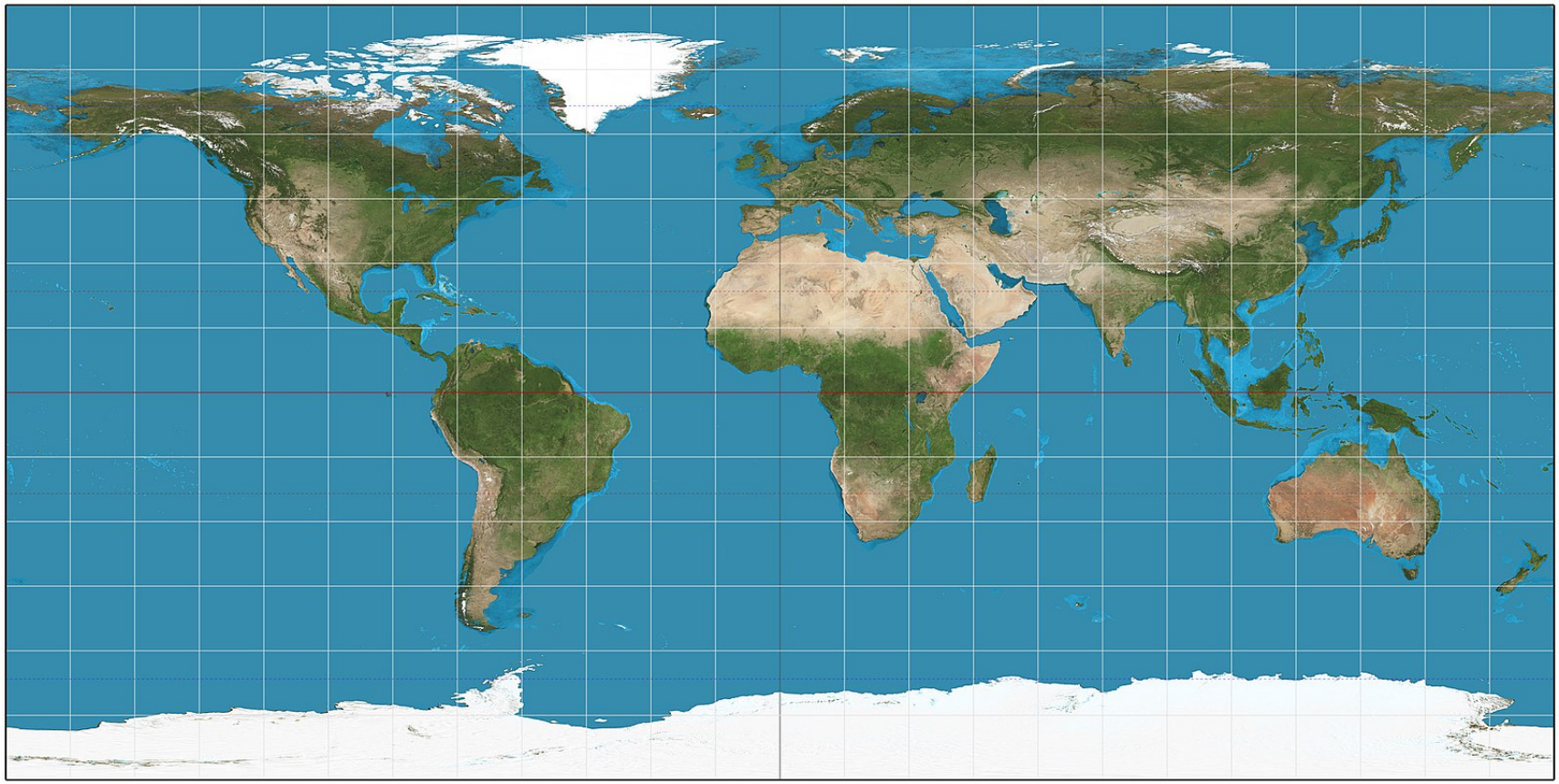
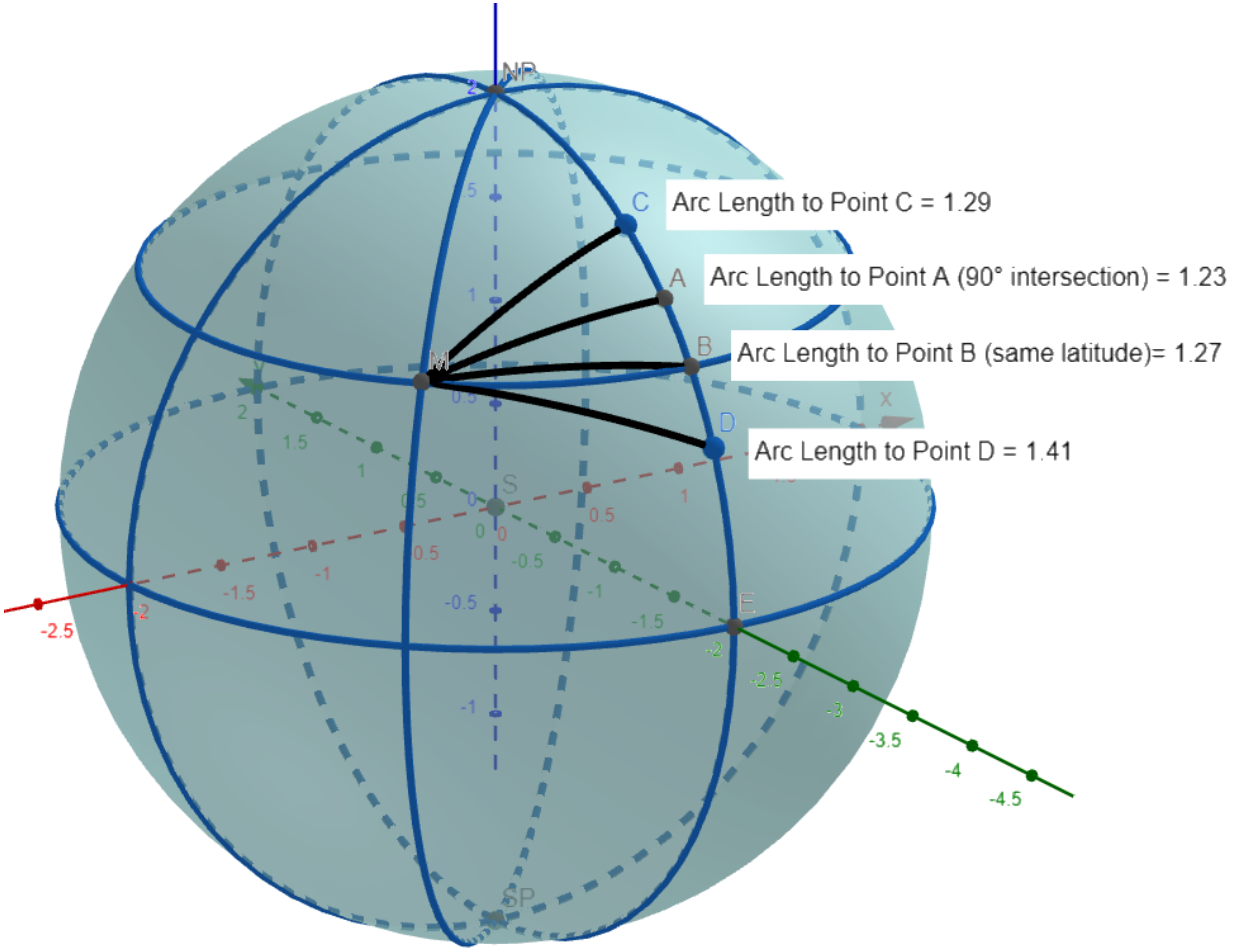


Figure: Equirectangular mapping of the earth, source: https://en.wikipedia.org/wiki/File:Equirectangular_projection_SW.jpg

# Query box


NP
C
Arc Length to Point C = 1.29
A
Arc Length to Point A (90° intersection) = 1.23
B
Arc Length to Point B (same latitude)= 1.27
M
D
Arc Length to Point D = 1.41
S
E
SP

# Query box

The query box on the sphere

- Furthest apart longitude and latitude boundaries intersect line from start point at 90 degrees
- The line length is given by the maximum distance
- Calculate furthest apart longitude and latitude
- Use the boundaries to create a Cartesian box for the R-tree

# The Bounding Box Algorithm

**The Bounding Box Algorithm:**

```
subResult1 = getSubResultLeft()
subResult2 = getSubResultRight()
result = createEmptyResultTable()
smallerRes, largerRes = compareSize(subResult1, subResult2)
rtree = buildRtree(smallerRes)
for row in largerRes:
    queryBox = calculateQueryBox(row.getGeometry())
    candidates = rtree.query(queryBox)
    for candidate in candidates:
        d = distance(candidate, row.getGeometry())
        if d < maxDistance:
            result.add(row, candidate.getRow(), d)
```

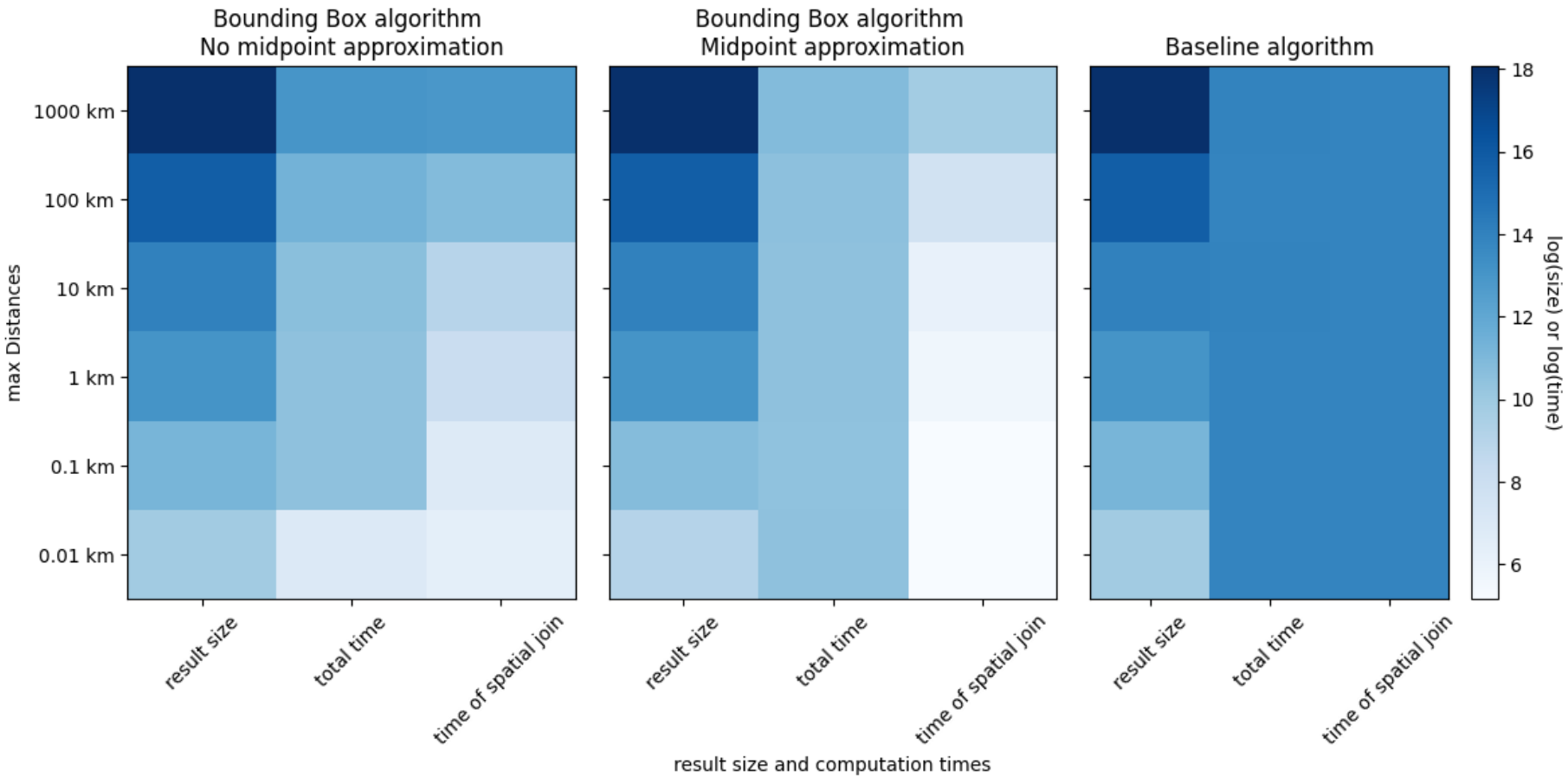
# Evaluation

# Evaluation - Queries on the OpenStreetMap of Germany

Evaluation of the Spatial Join algorithms on the OpenStreetMap dataset of Germany

- Query for university building pairs, that are at most *maxDistance* meters apart from each other
- Comparison of the baseline and bounding box algorithms
- Comparison of the times for the area midpoint approximation and the true distance calculation
- Comparison of total time and time for the spatial join operation only

# Evaluation - Queries on the OpenStreetMap of Germany



# Conclusion

- Efficient spatial join algorithm
- Integration into the search engine Qlever (infrastructure, tests)
- Theoretical analysis, including a proof of correctness
- Practical analysis, showing very good performance

# Any Questions?

Thank You!

# Evaluation - Query Box Efficiency

Evaluation of the efficiency of the query box

- Synthetic data set with uniform density
- Varied max distance, number of points in the R-tree and latitude
- Analysis of the ratio $\frac{nrPointsInQueryBox}{nrPointsInRtree}$
- Analysis of the ratio $\frac{nrPointsInResult}{nrPointsInQueryBox}$

# Evaluation - Query Box Efficiency

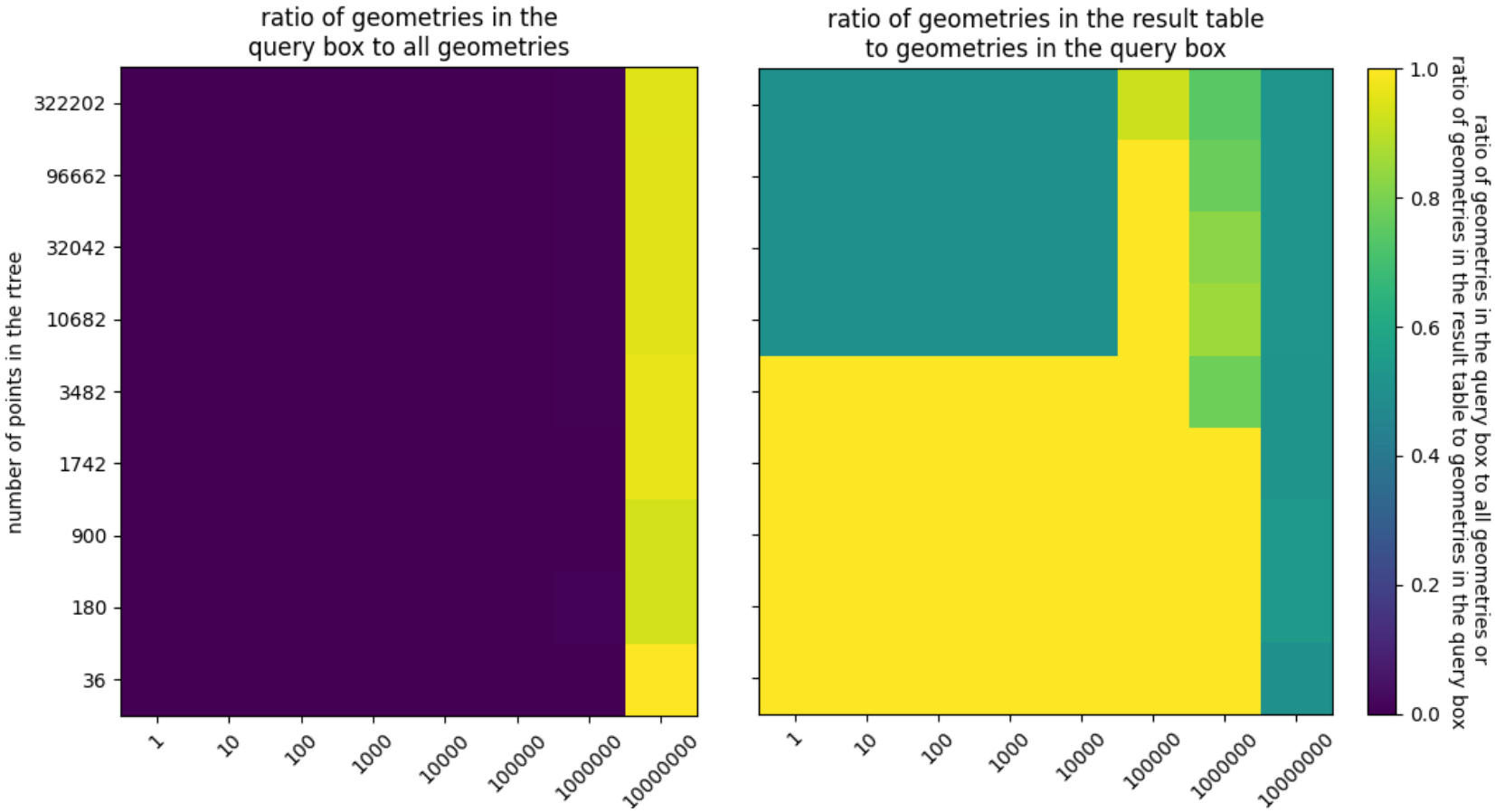



Figure: Ratios $\frac{nrPointsInQueryBox}{nrPointsInRtree}$ and $\frac{nrPointsInResult}{nrPointsInQueryBox}$ for a latitude of 10

# Evaluation - Query Box Efficiency

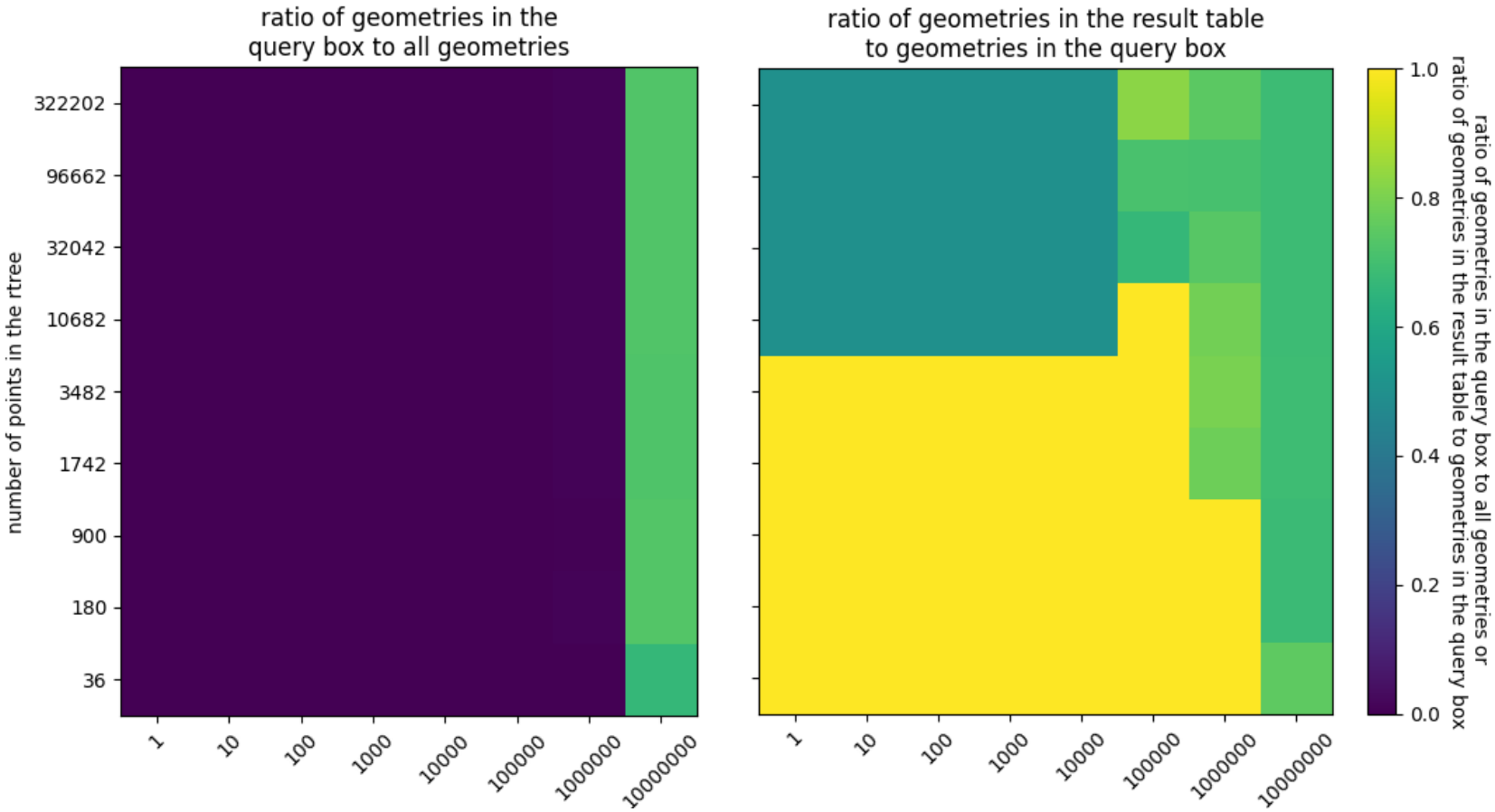



Figure: Ratios $\frac{nrPointsInQueryBox}{nrPointsInRtree}$ and $\frac{nrPointsInResult}{nrPointsInQueryBox}$ for a latitude of 50

# Evaluation - Query Box Efficiency

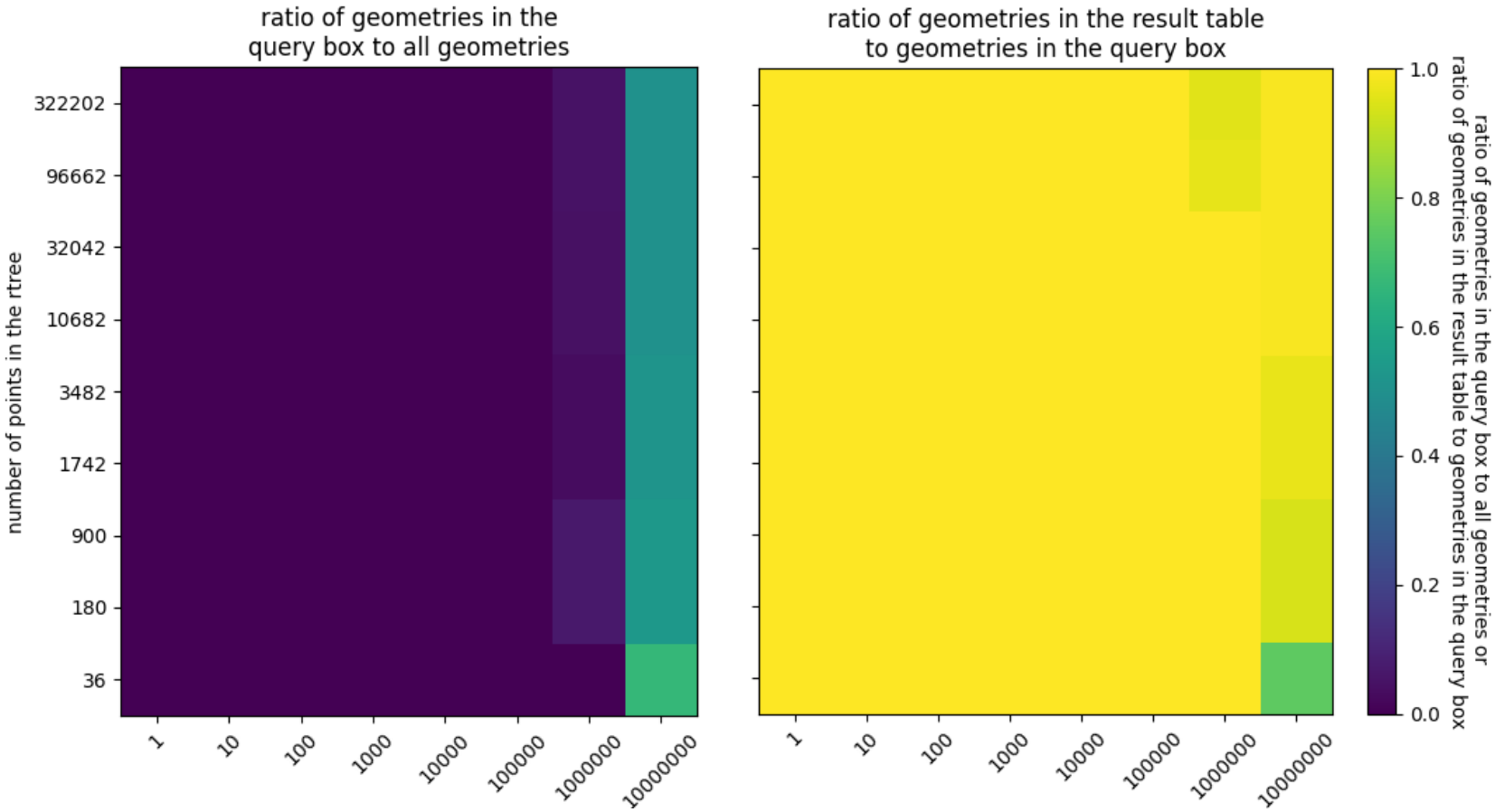
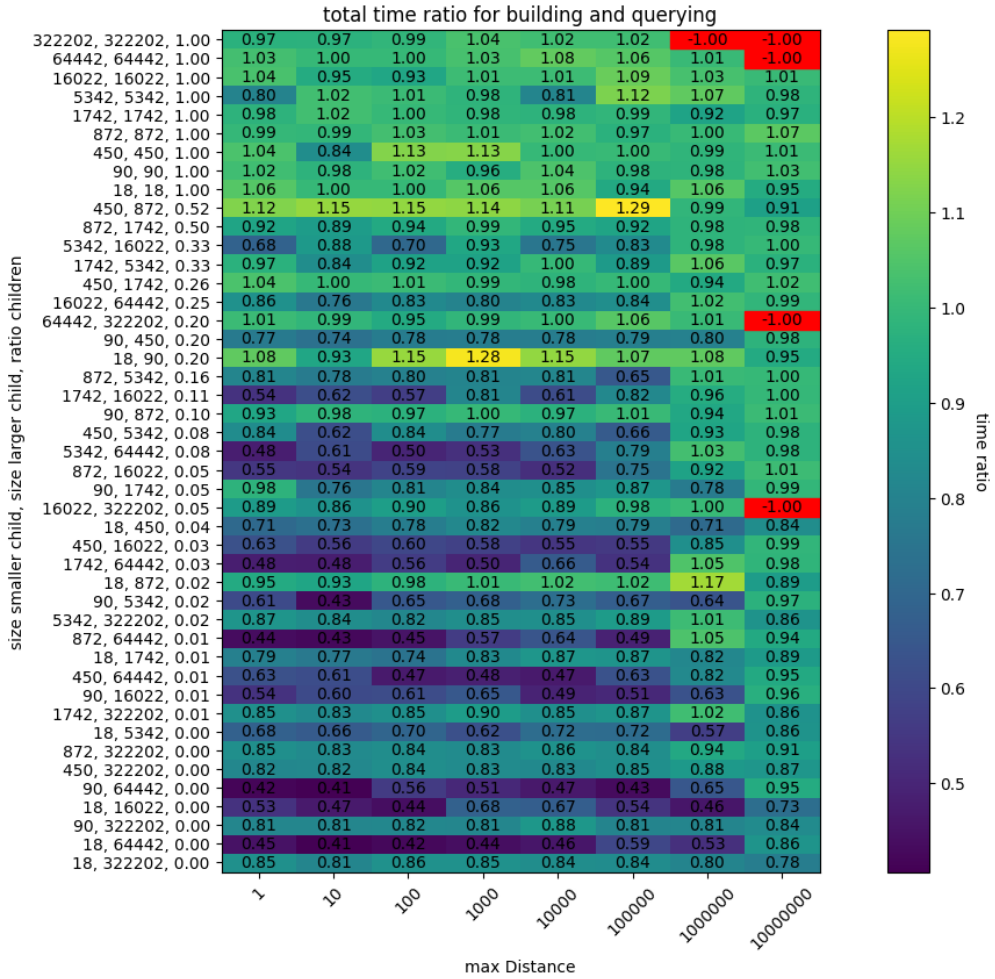



Figure: Ratios $\frac{nrPointsInQueryBox}{nrPointsInRtree}$ and $\frac{nrPointsInResult}{nrPointsInQueryBox}$ for a latitude of 90

# Evaluation - build R-tree analysis



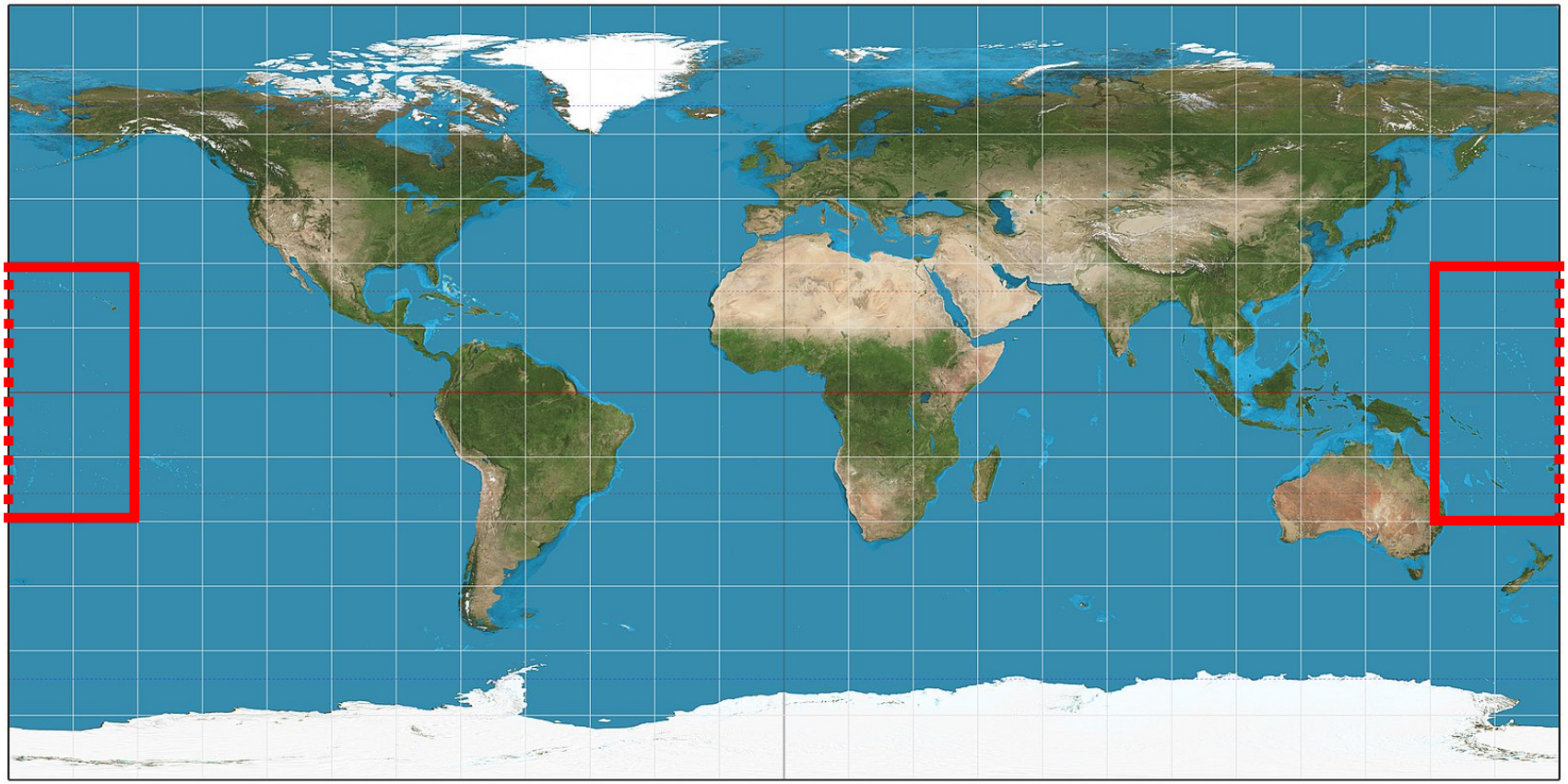


Figure: adapted from
https://en.wikipedia.org/wiki/File:Equirectangular_projection_SW.jpg

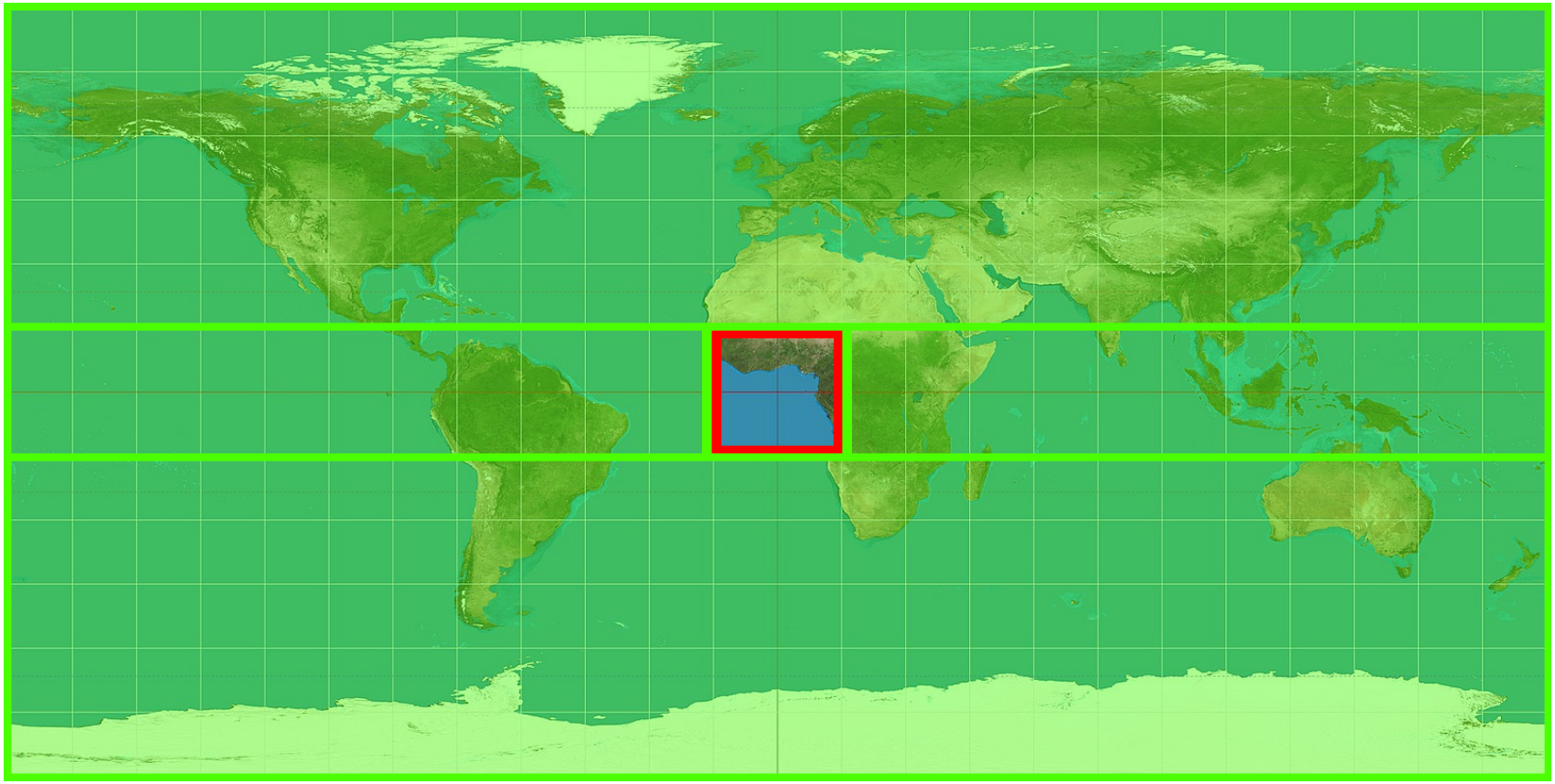
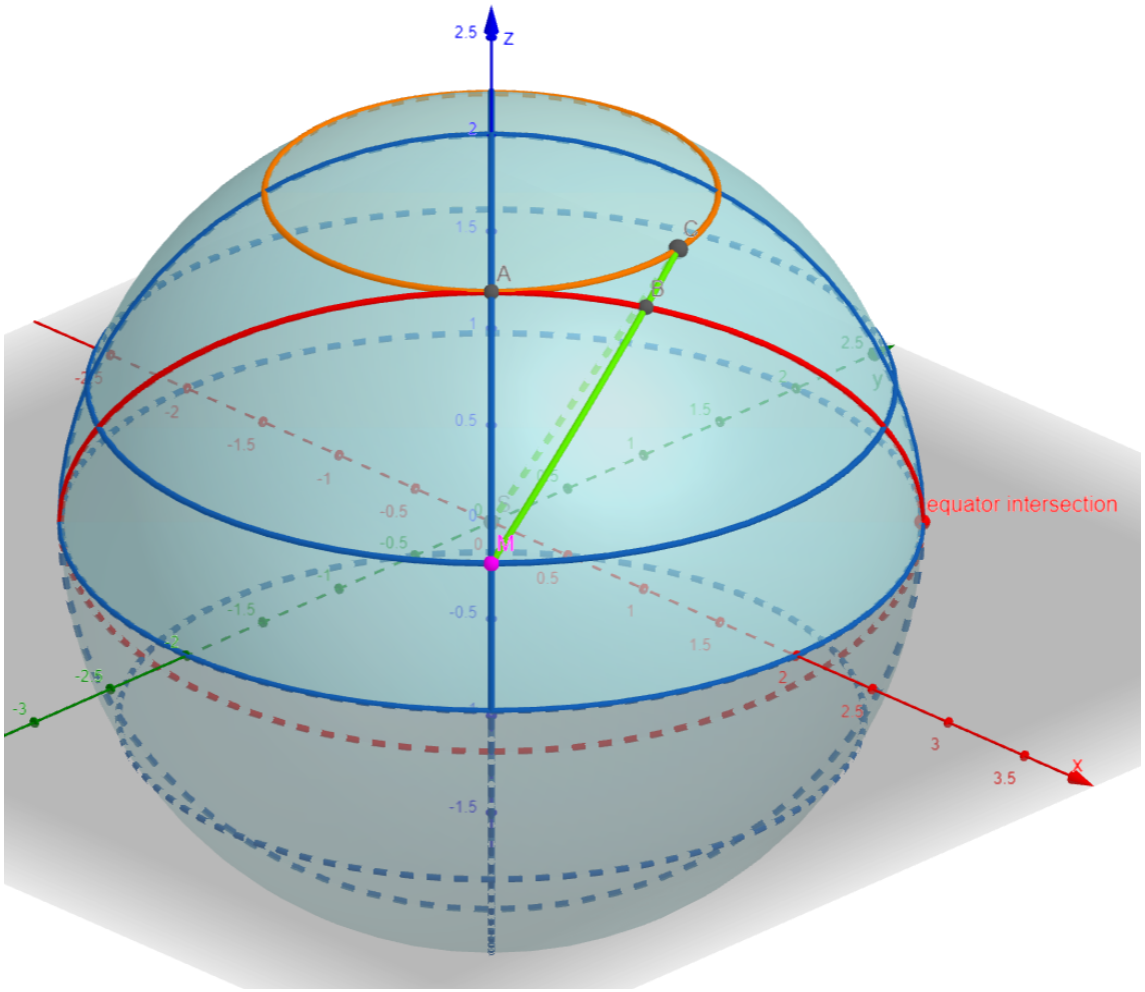
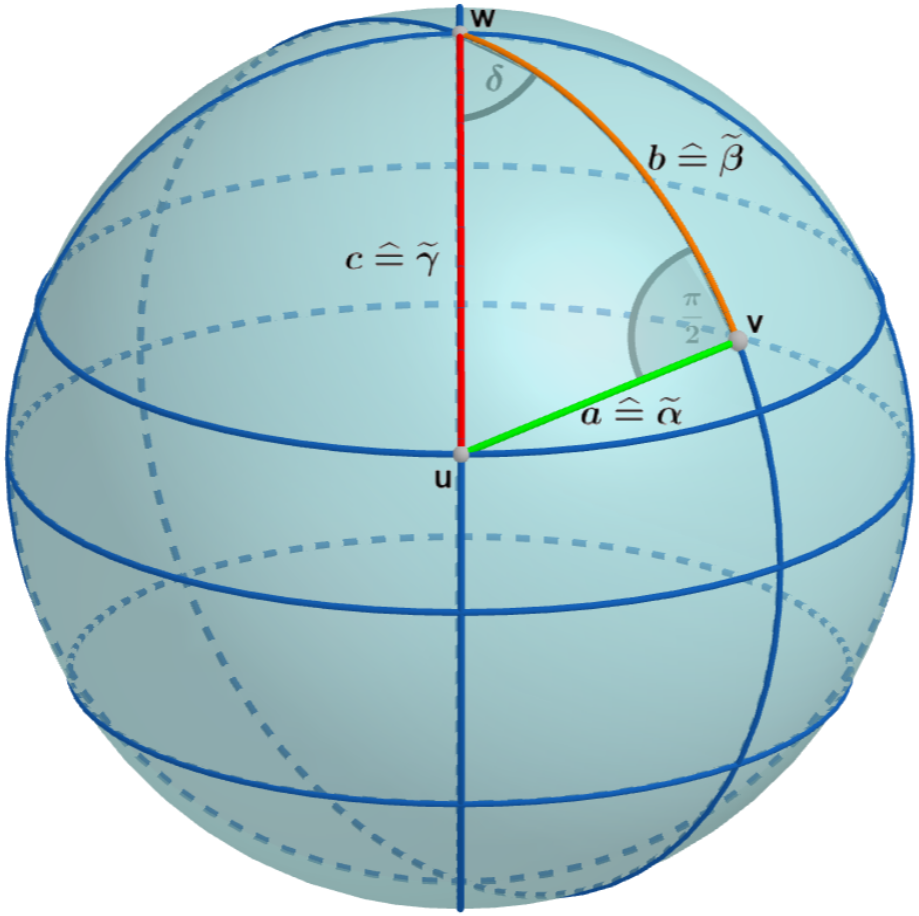
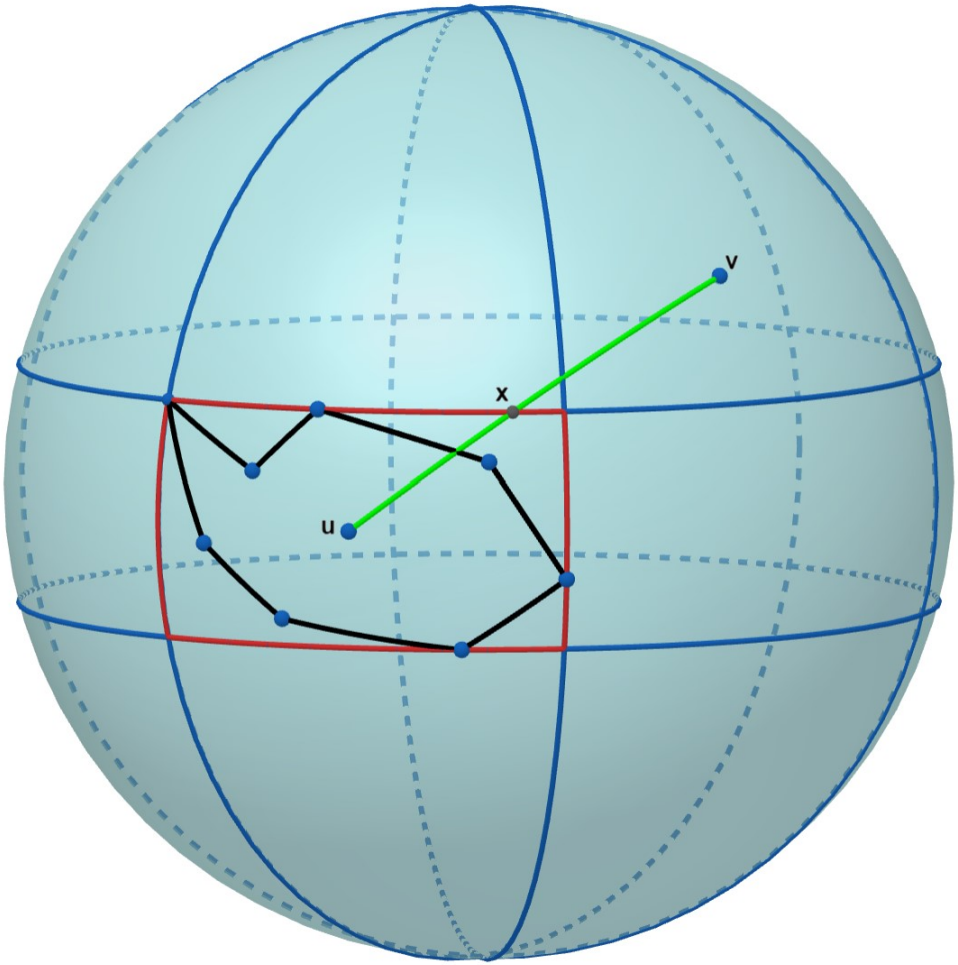


Figure: adapted from
https://en.wikipedia.org/wiki/File:Equirectangular_projection_SW.jpg


z
A
B
C
equator intersection
x
y


w
δ
b ≙ β̃
c ≙ γ̃
π/2
v
a ≙ α̃
u


v
x
u