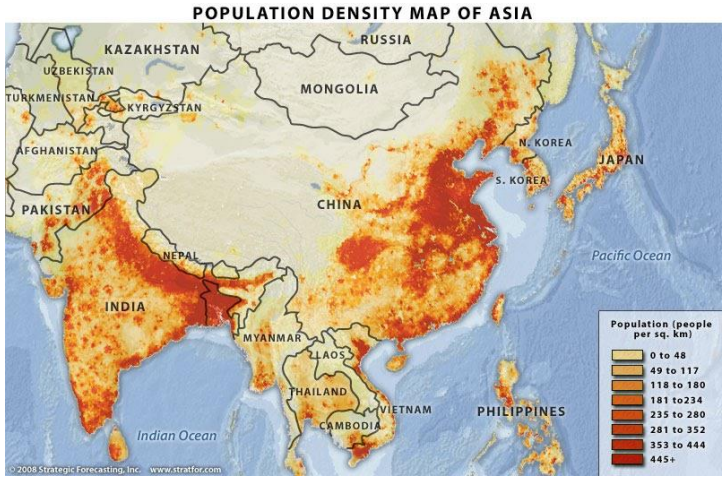
Effective Density Visualization of Multiple Overlapping Axis-aligned Objects

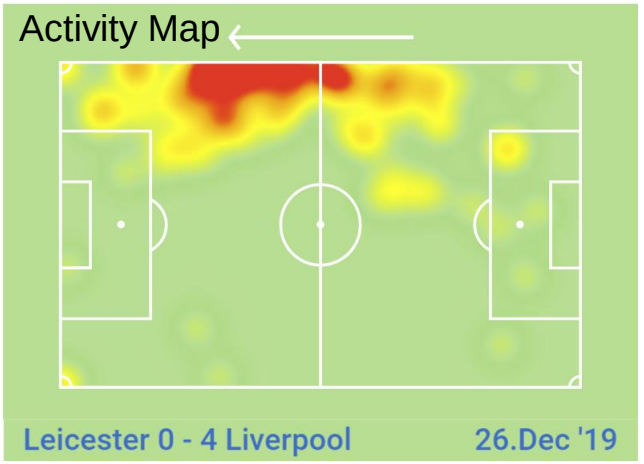
MSc. Thesis of Niloy Eric Costa
York University, Toronto, Canada

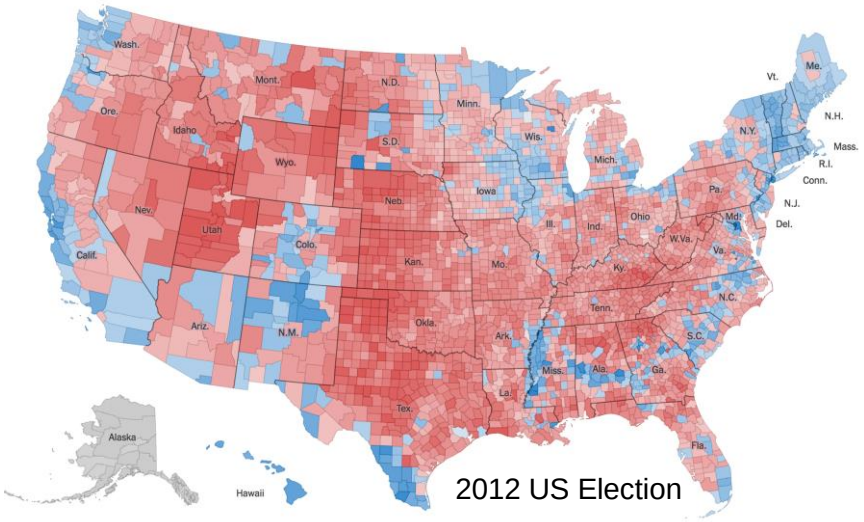
Background

YORK
UNIVERSITÉ
UNIVERSITY

Density-based visualization



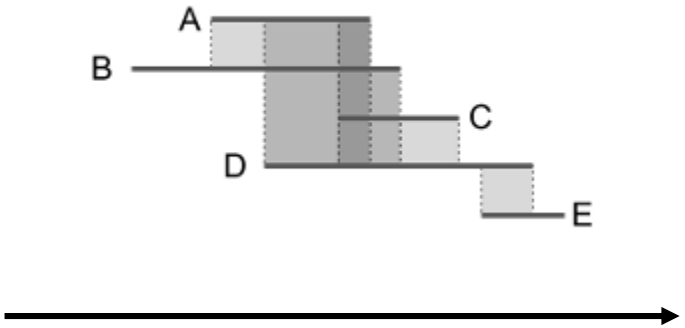




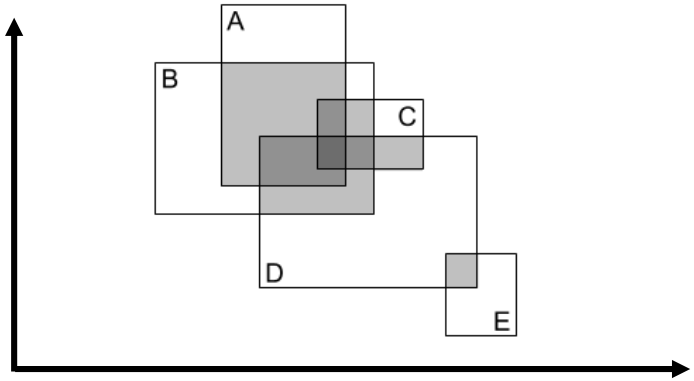
Observation

Many data analytics problems need to visualize the density of axis-aligned objects

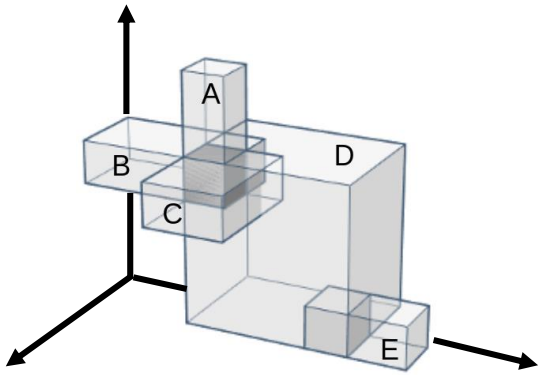
Axis-aligned geometric objects



1-D line segments/intervals



2-D rectangles

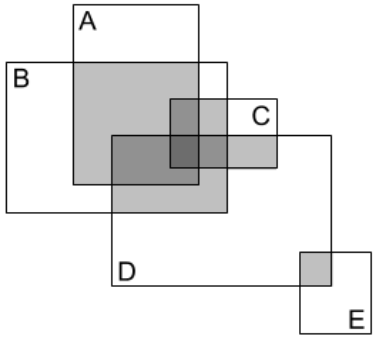


3-D boxes/cuboids

Need for effective density visualization of multiple overlapping axis-aligned objects

Research questions

1. How to detect multiple overlaps?
    i. How many overlapping elements?
    ii. Which rectangles are overlapping?
    iii. Size of the overlaps?



2. How to evaluate the efficiency of the methods?

3. What are the real-world use cases for these methods?

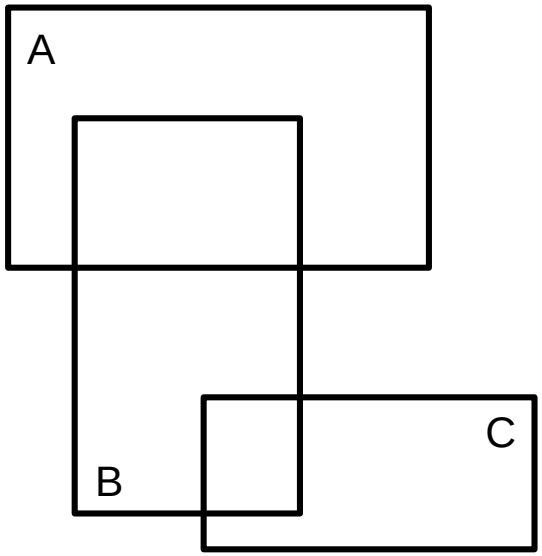
Object intersection problem

Input

a set of axis-aligned geometric objects

Output

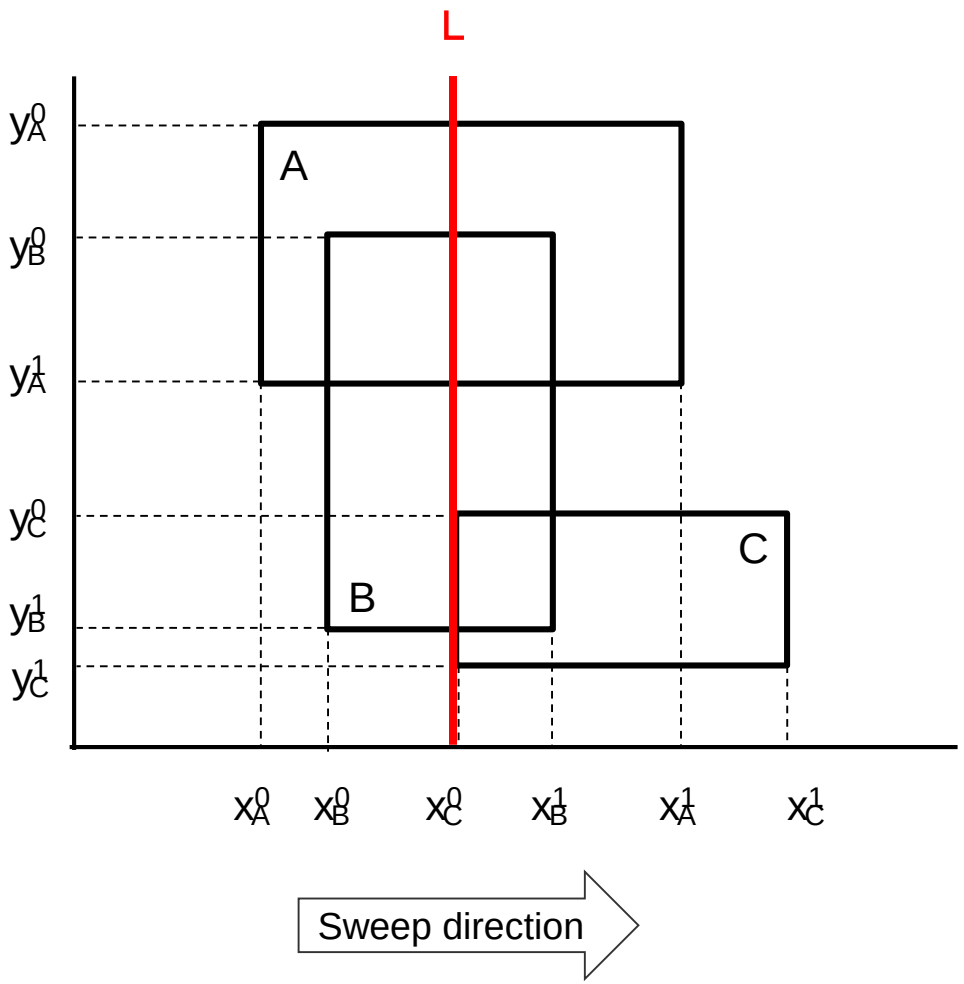
pairs of intersecting objects
size of overlap



(A,B)
(B,C)

how can we address this problem?

Sweep-line algorithm



an efficient one pass computational geometry algorithm

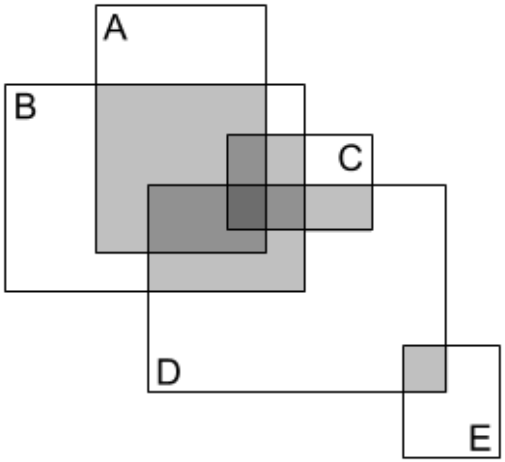
Multiple Object Intersection Problem

YORK
UNIVERSITÉ
UNIVERSITY

The problem

Input

a set of regions in R^2

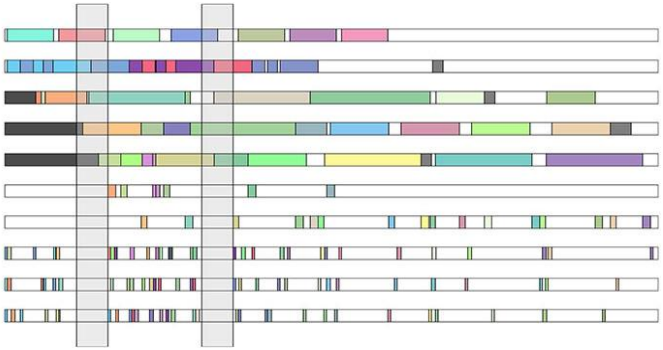


Output

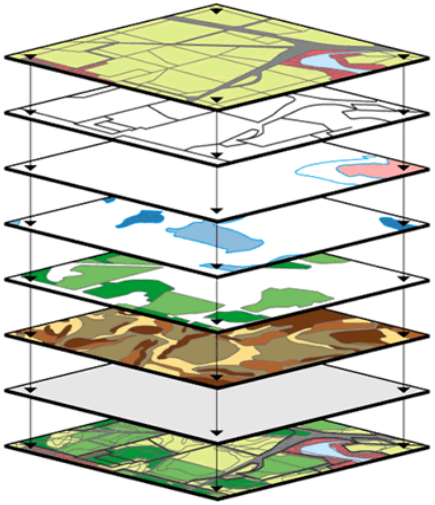
enumeration of all intersecting regions
size of each common region
position of each common region

(A,B)
(A,C)
(A,D)
(B,C)
(B,D)
(C,D)
(D,E)
(A,B,C)
(A,B,D)
(A,C,D)
(B,C,D)
(A,B,C,D)

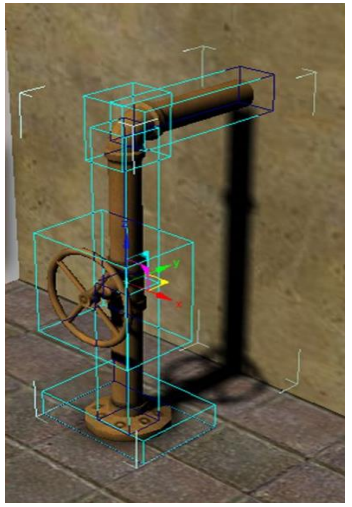
Many applications



task scheduling



spatial databases



simulations

Baseline Methods

YORK
UNIVERSITÉ
UNIVERSITY

Sensible baseline algorithms

Baseline 1: naive algorithm

iteratively check all possible ways that *n* objects can intersect

(-) limitation

there are 2^n ways, so exponential computational cost

Baseline 2: grid-based approach

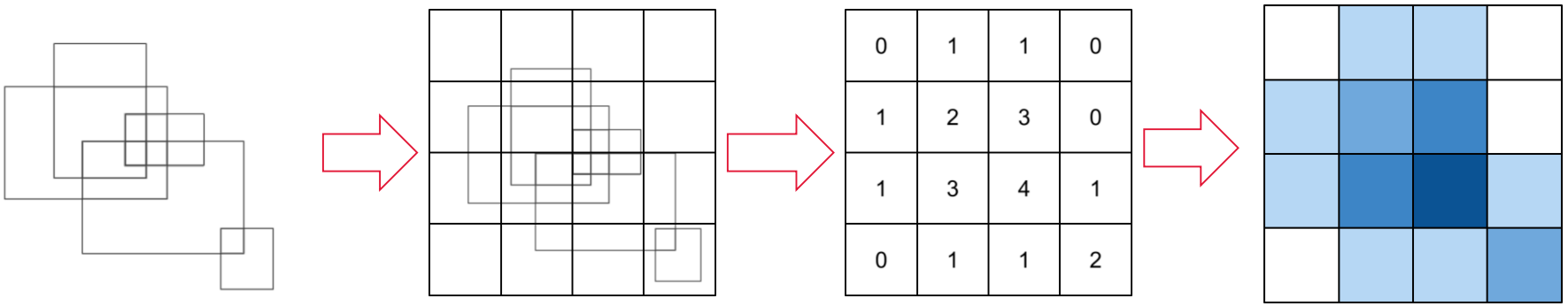
create a grid, perform orthogonal queries to find objects intersecting with each grid cells, assign value based on intersections

(-) limitation

trade-off between accuracy and time-performance based on grid-cell sizes

Grid-based approach

1. Use R-tree* to create a grid
2. Search in the tree for finding z-index scores
3. Color each grid-cells based on the corresponding z-index scores



Input data-set

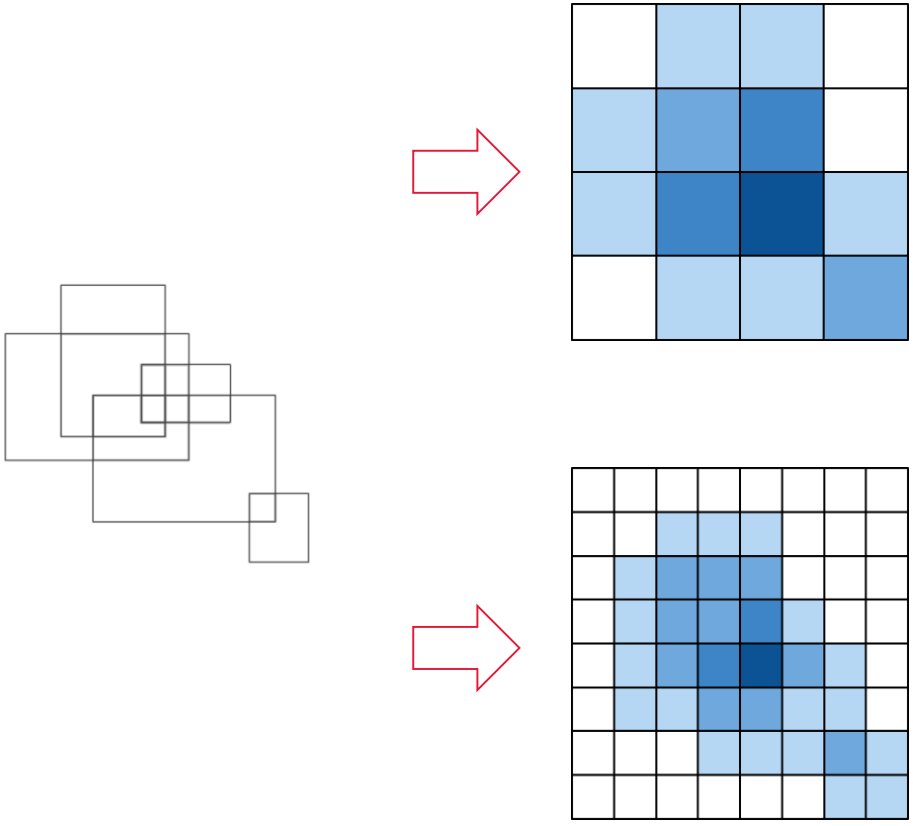
1. 4 X 4 grid

2. z-index scores of cells

3. 4 X 4 grid heat-map

*R-tree is a depth balanced tree, provides aid in faster spatial queries

Grid-based approach trade-off



Trade-off

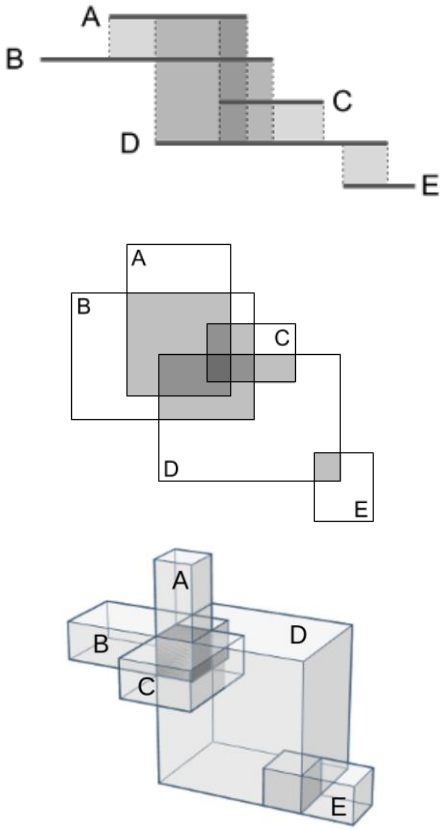
- 4 X 4 grid is less accurate, but z-indexes calculated quickly
- 8 X 8 grid is more accurate, took longer to calculate each z-index score

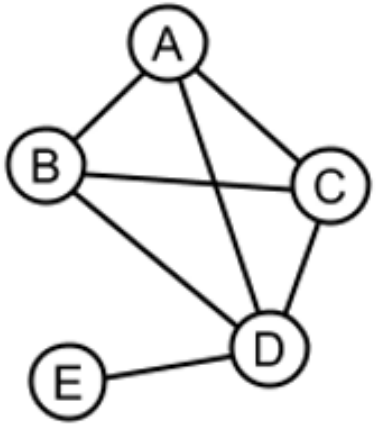
Our Approach (OverLap-HeatMap)

YORK
UNIVERSITÉ
UNIVERSITY

Observation 1

intersections of *n*-dimensional objects (1-D, 2-D, 3-D, …) can be universally modeled as an intersection graph



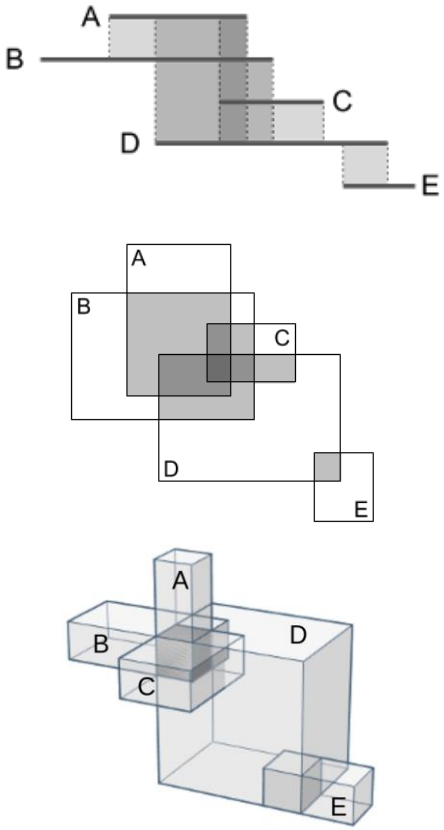


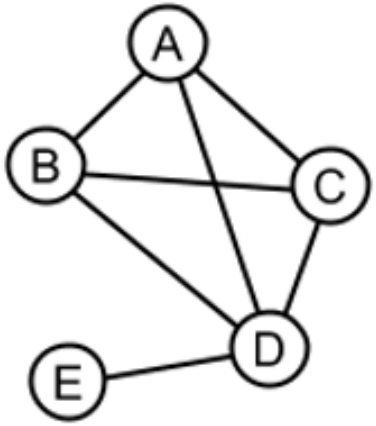
intersection graph:

- **vertex**: represents an object
- **edge**: represents that two objects intersect

Observation 2

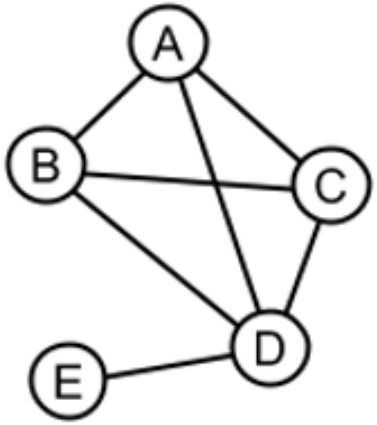
a *k*-clique in the intersection graph, corresponds to *k* objects that are simultaneously intersecting and share a common region





k-clique

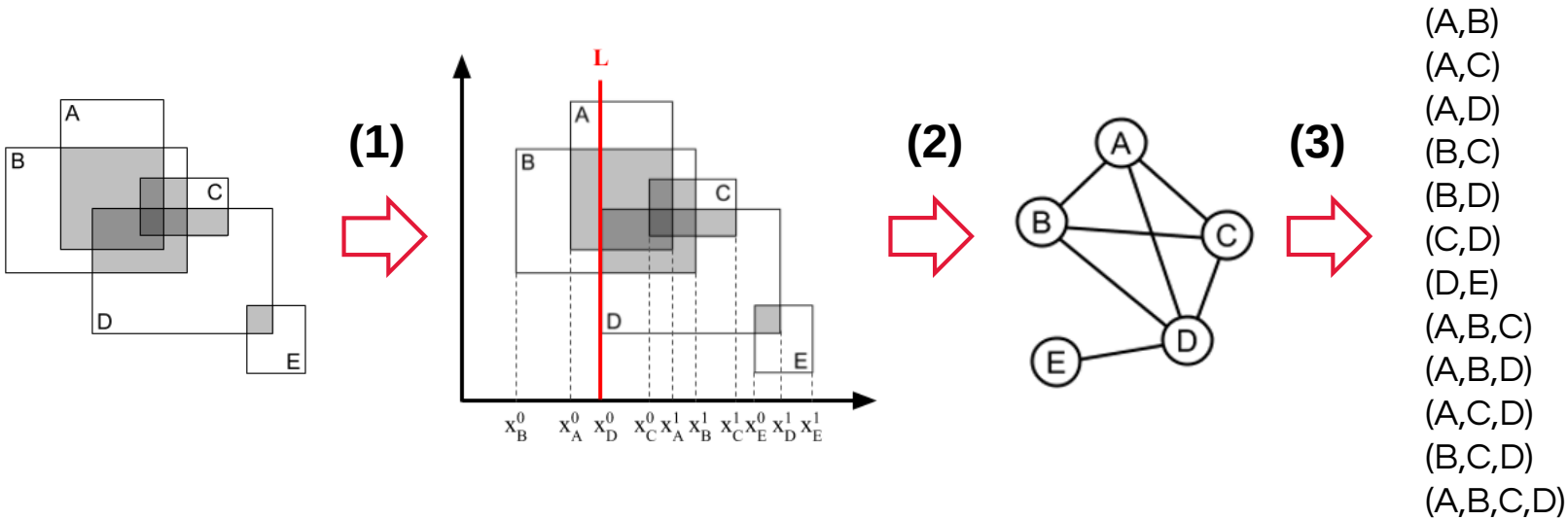
a *k*-clique is a complete subgraph of size *k* (i.e., a subset of *k* vertices that are all connected to each other)



2-cliques: all edges
3-cliques: ABC, ABD, ACD, BCA
4-cliques: ABCD (**maximal clique**)

OL-HeatMap* algorithm (sketch)

1. Apply sweep-line to find intersecting pairs
2. Construct the rectangle intersection graph (RIG)
3. Apply a clique enumeration algorithm on graph

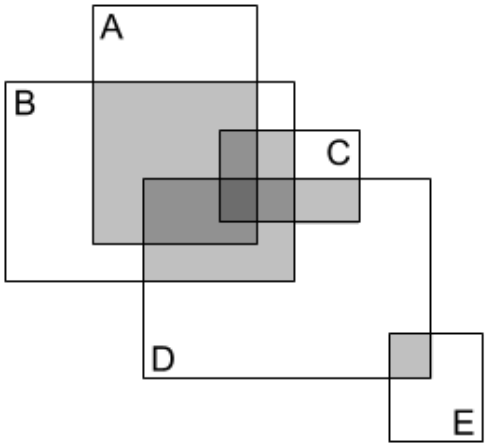


*OL-HeatMap is an extended version of SLIG - **S**weep-**L**ine (with an auxiliary) **I**ntersection **G**raph
By Tilemachos et al.

OL-HeatMap: Other metrics computed

z-index

The number of simultaneously overlapping objects in a set



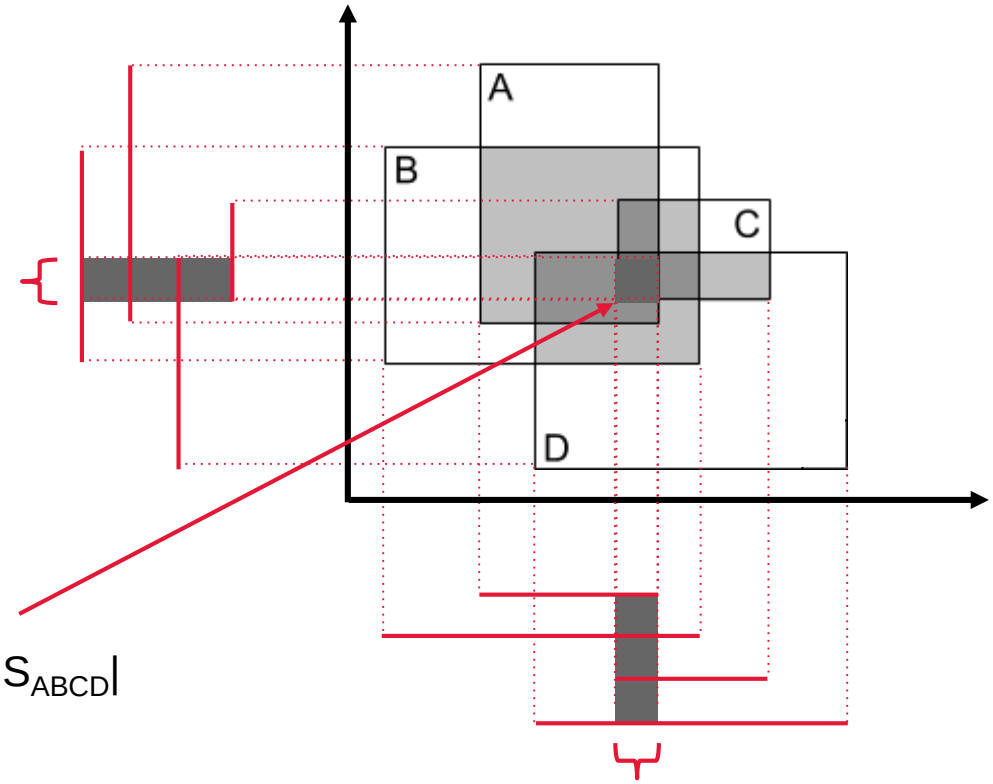
$z_{ABCD} = 4$

$z_{DE} = 2$

...

size of overlap(|S|)

For more dimensions, $|S|$ is the **product** of the common region **lengths** in **each dimension** $|S_o|$

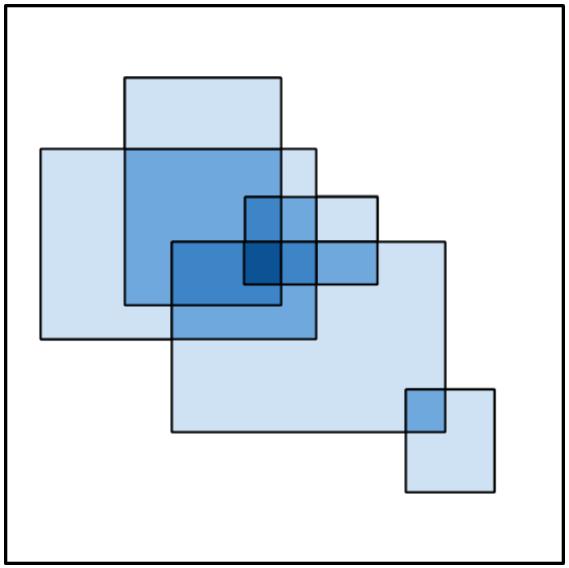


OL-HeatMap: Final visualization

Coloring the boxes

Each common region **S** should be colored only once based on their intersection cardinality. We skip drawing of rectangles which are completely covered by another.

Currently ~30% less overlaps are colored



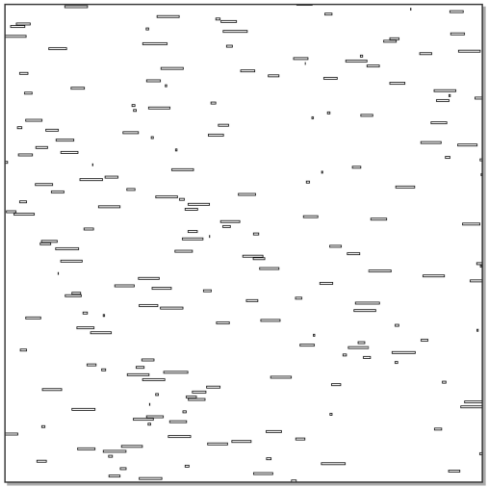
Experimental Evaluation

YORK
UNIVERSITÉ
UNIVERSITY

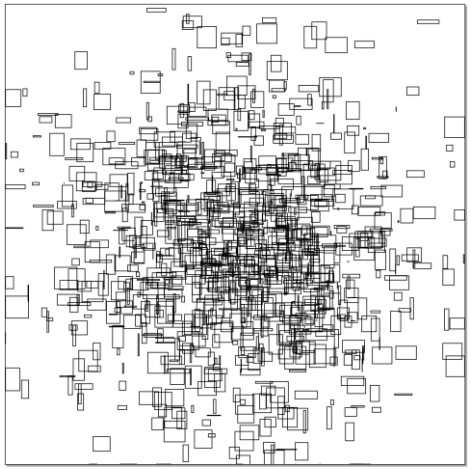
Experiment overview

- Accuracy performance
- Runtime performance
- OL-HeatMap versatility (extension to 1D objects)
- OL-HeatMap flexibility (real world use-cases)
- OL-HeatMap scalability

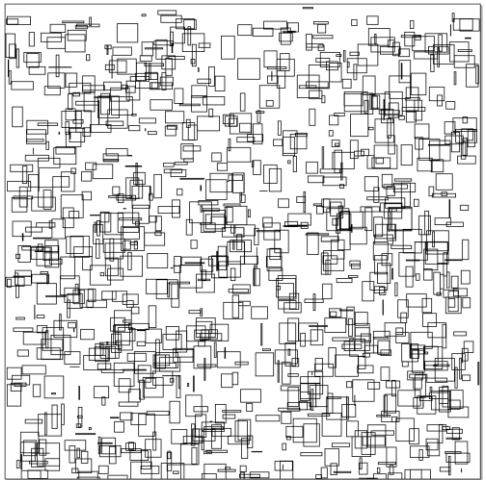
Randomly generated objects



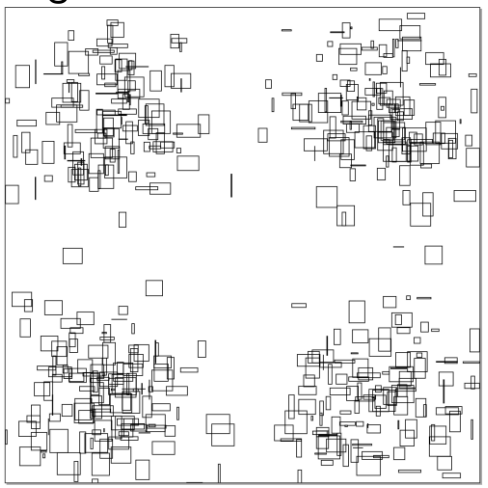
1-D intervals



2-D rectangles – gaussian distribution

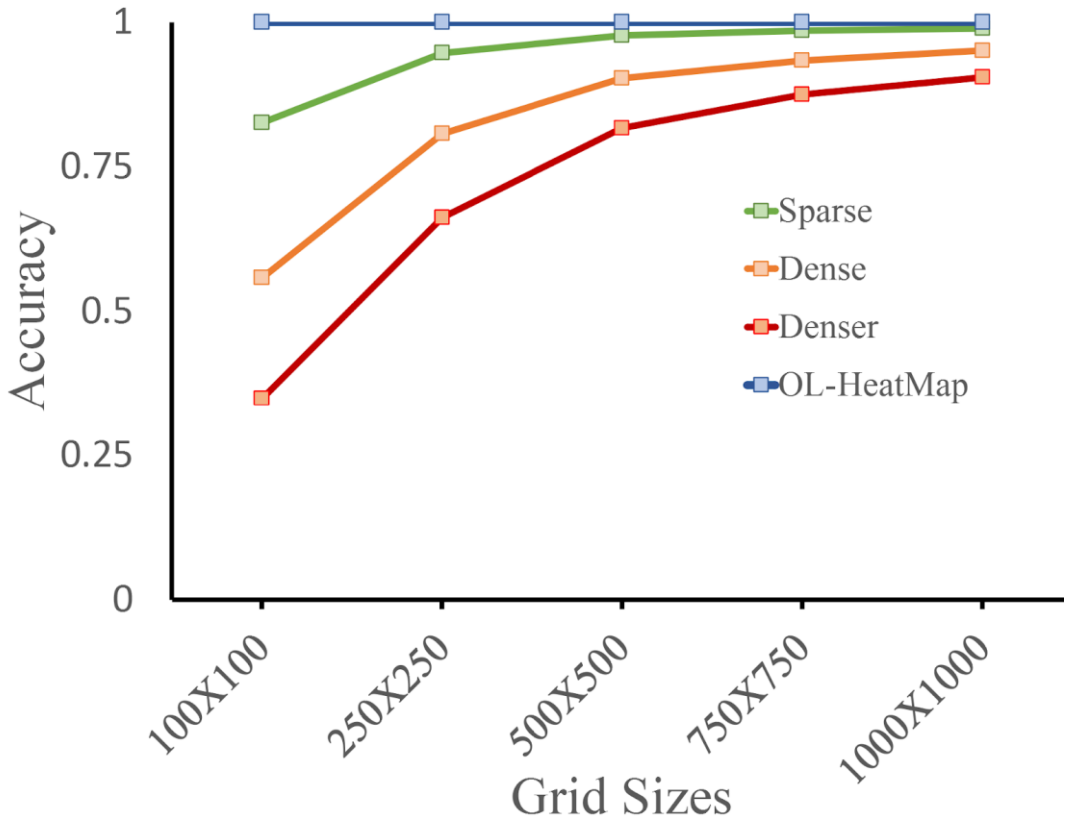


2-D rectangles – uniform distribution



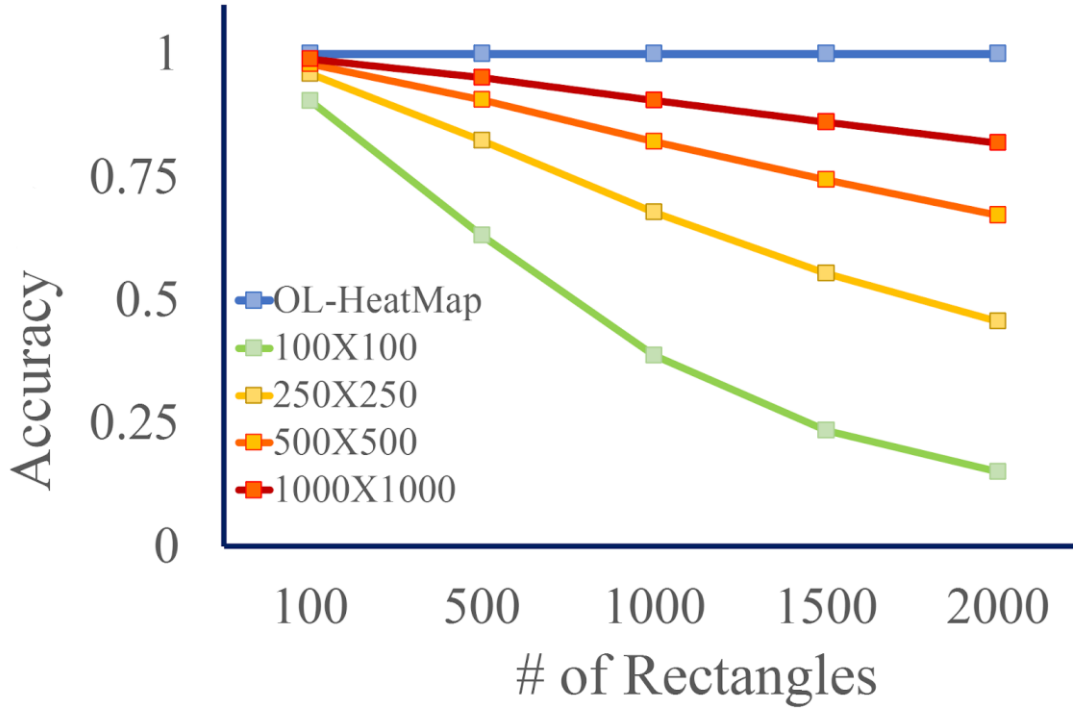
2-D rectangles – bi-modal distribution

Accuracy



Measurement of accuracy for different grid sizes

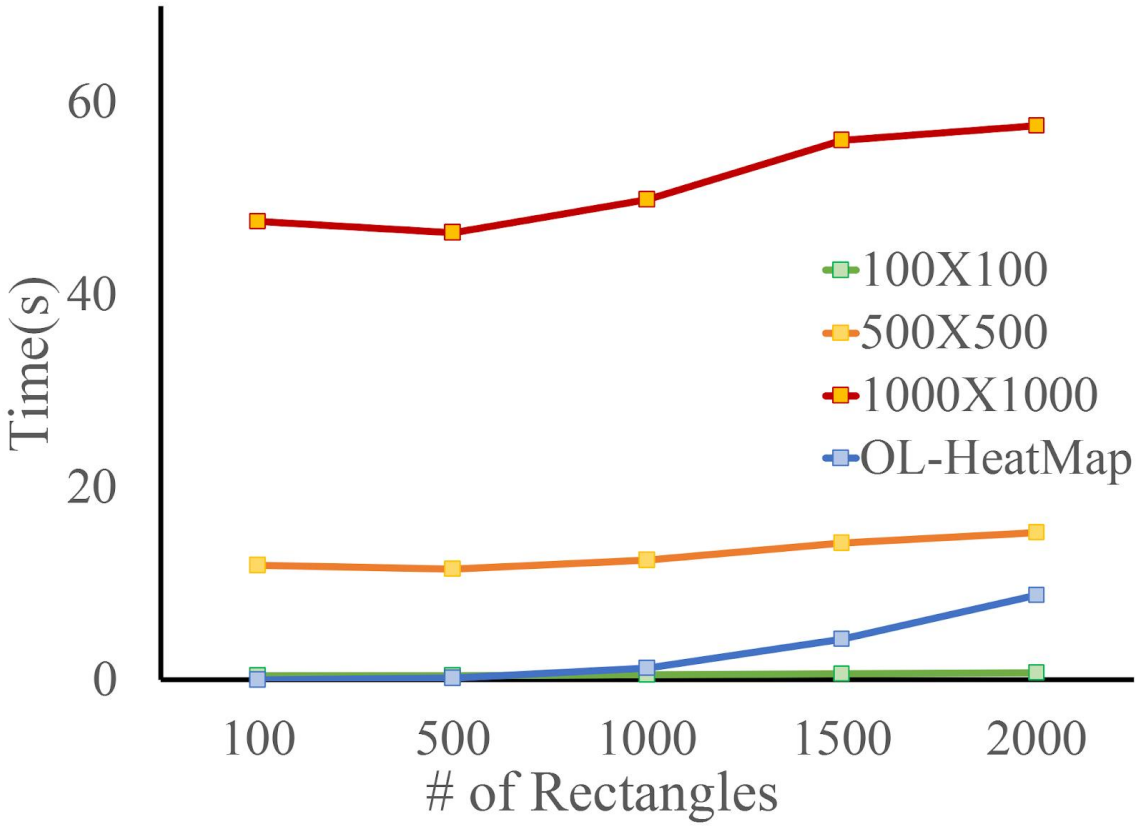
Accuracy



Accuracy performance of OL-HeatMap vs. grid-based

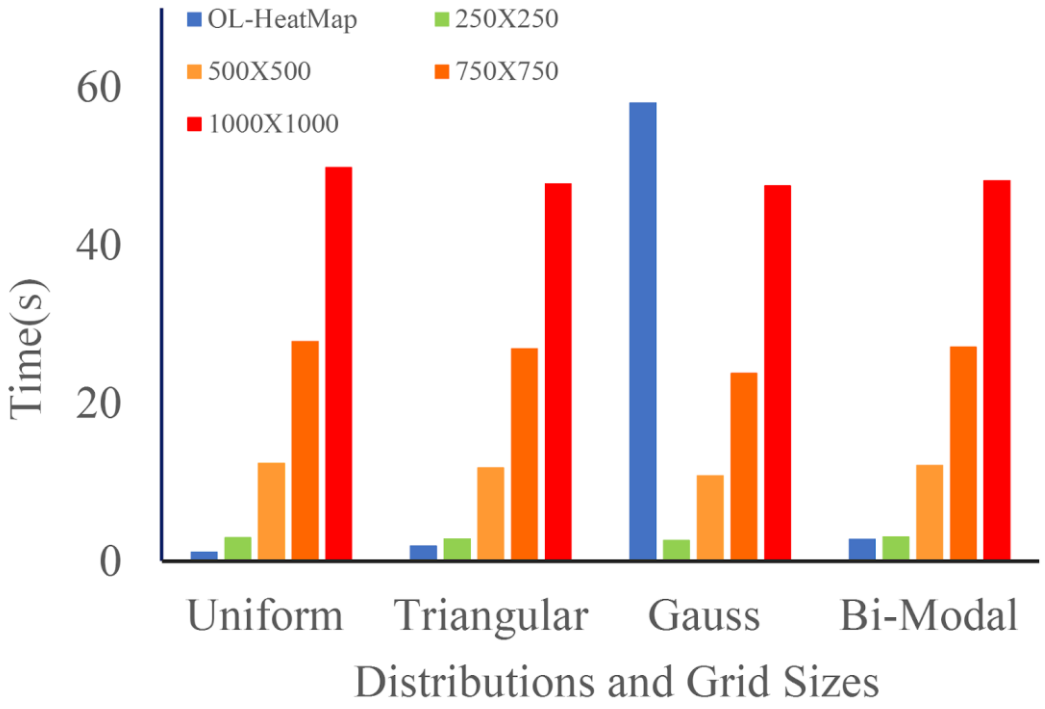
OL-HeatMap is 100% accurate. However, a finer grid can achieve similar accuracy

Runtime cost



Comparison of time for different data-set sizes

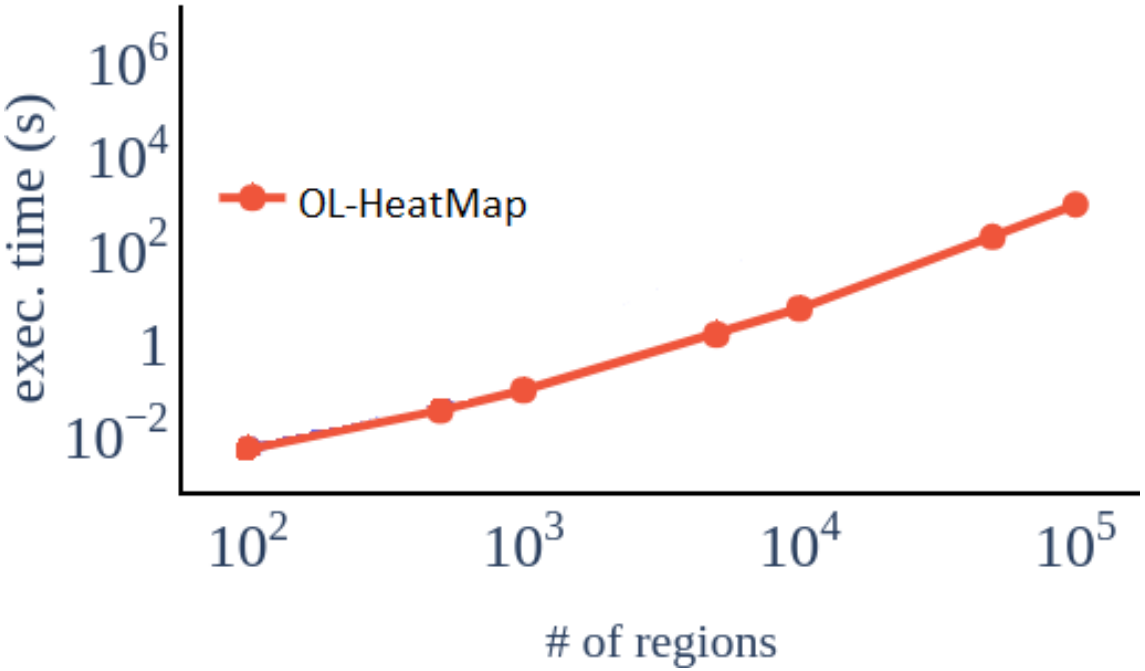
Runtime cost



Comparison of time for different data distributions

Finer grid sizes takes a lot of time to compute in order to achieve similar accuracy that of OL-HeatMap

Scalability



Execution Time vs OL-HeatMap Scalability

OL-HeatMap can scale up-to a million regions

Real World Use Cases

YORK
UNIVERSITÉ
UNIVERSITY

Real-world use cases (1D)

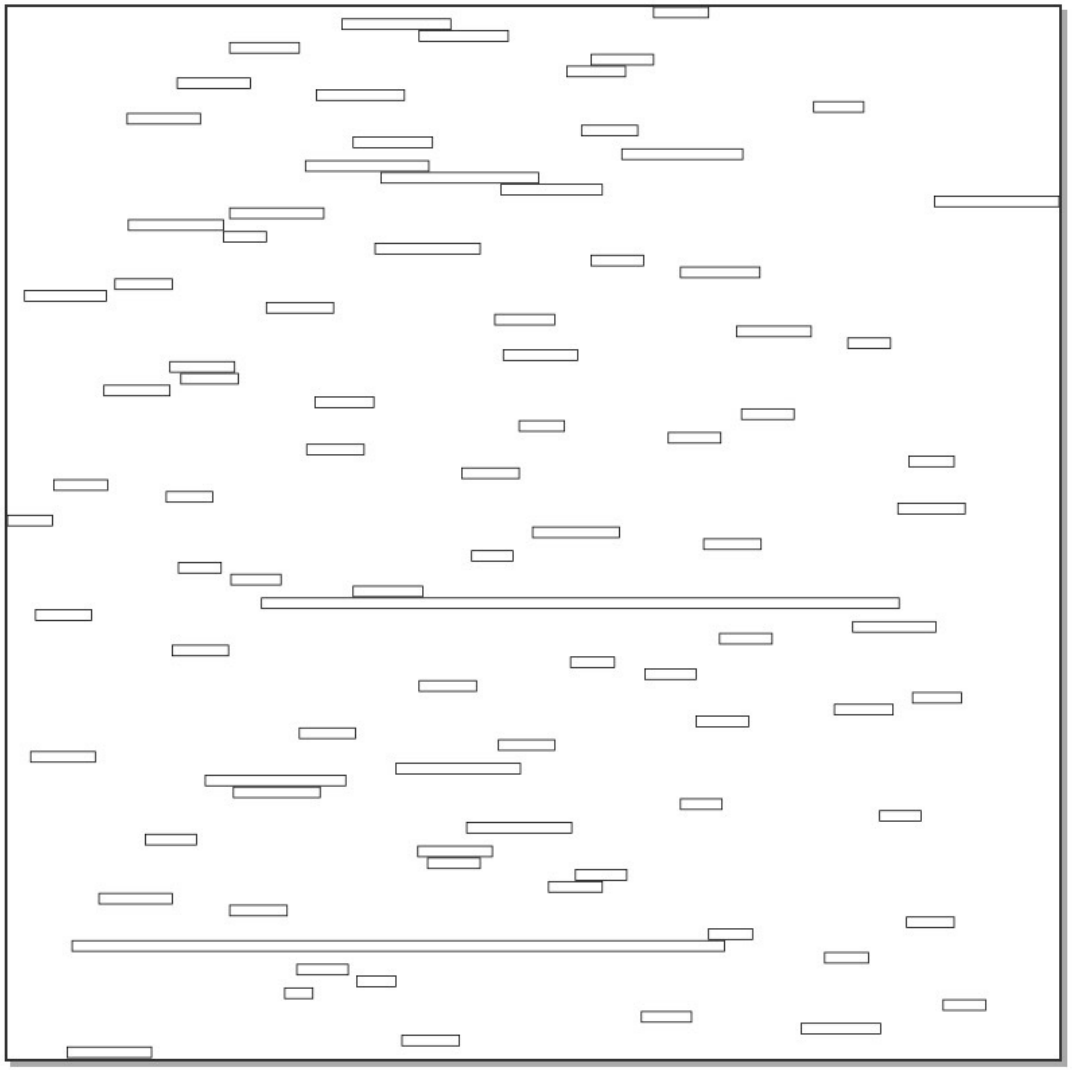
The Data

- US Airline Carrier Data (1987-present)
- We used John Wayne Airport, Orange County, California
- 1D intervals created by time aircraft spent on runway

Visualization Goal

- Find highest density of runway traffic
- Finding least used time slot for a runway
- Overview of airport usage in a single day (February 1st, 2019)
- Providing aid in Air Traffic Management

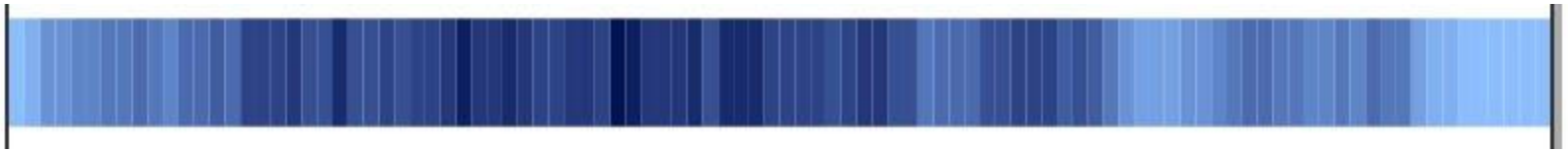
Airline carrier data



Overview of the February 1st, 2019

Time Left to Right – 0000 – 2359 Hours

Airline carrier data



100 Grid. Time - 0000-2359 Hours [24 Minute Intervals]

50 Grid. Time - 0000-2359 Hours [48 Minute Intervals]

OL-HeatMap. Time - 0000-2359 Hours

Real-world use cases (2D)

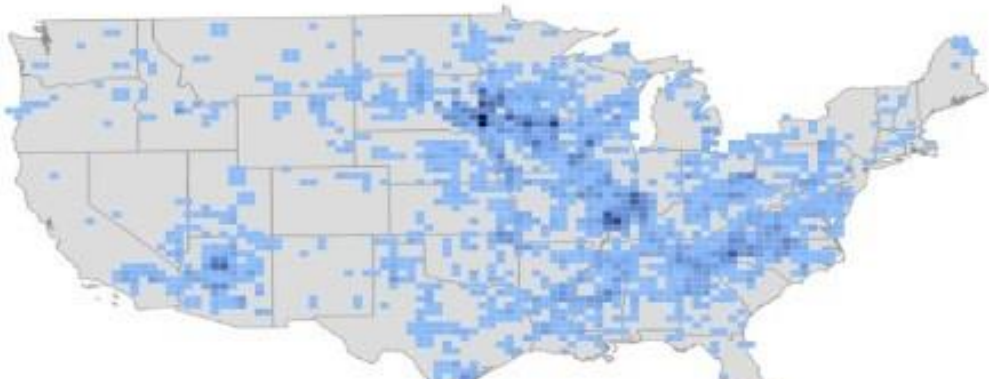
The Data

- US Storm Events Database, NOAA (1953-present).
- Relevant information regarding significant weather event.
- Begin Long., Lat., and End Long., Lat. Used to create bounding boxes

Visualization Goal

- Determining storm hot-spots in US during 2017-2019
- Finding states with less severe weather incidents
- Finding the borders of “Tornado Alley”
- Visualize using OL-HeatMap to show the sizes, density and severity of these events
- Finding all hurricanes in Florida from 1953-2018 {Using a subset of the entire dataset}

US storm events database

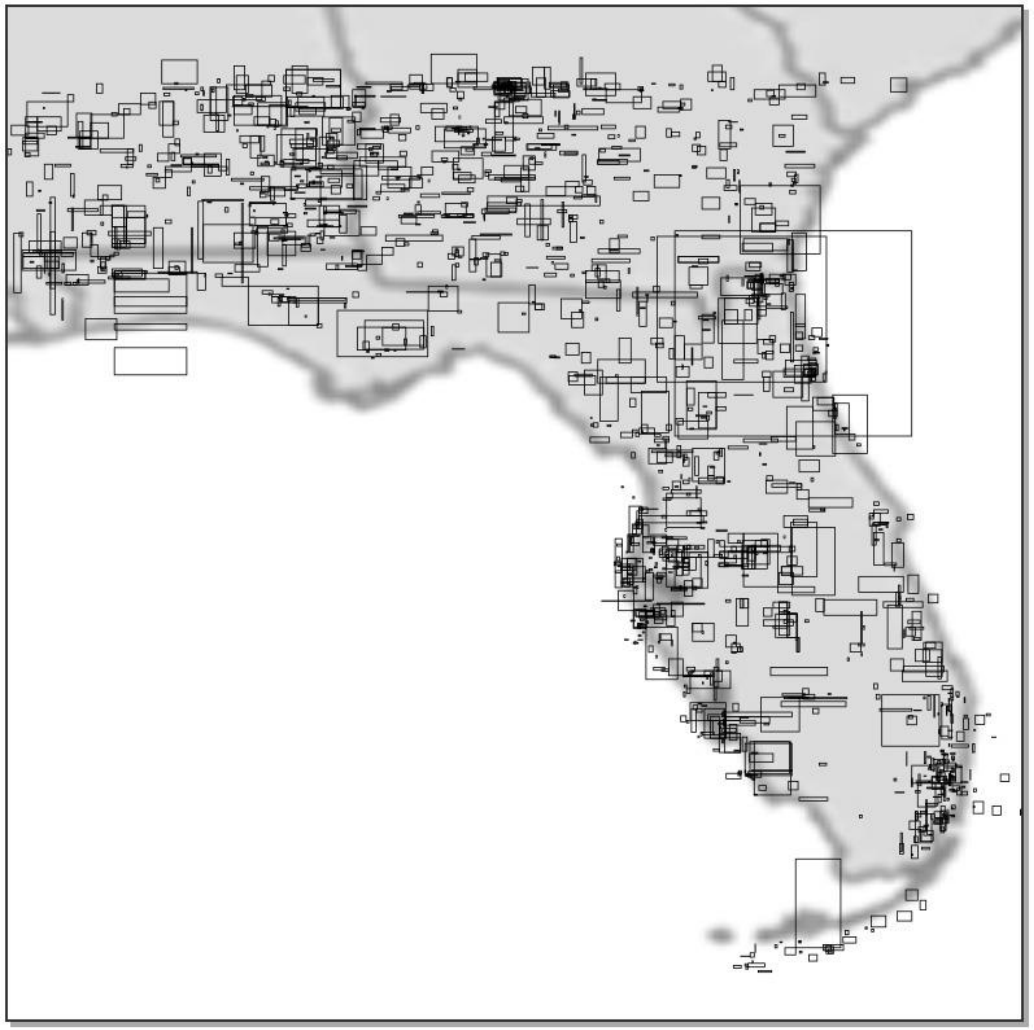


Grid-based visualization

OL-HeatMap

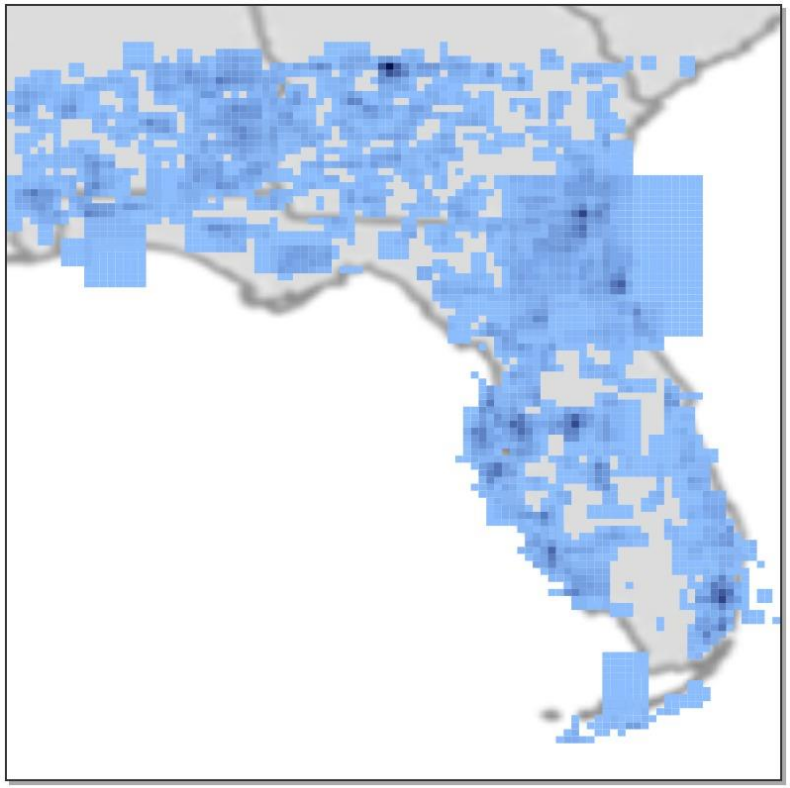
Storms in US [2017-2019]

US storm events database

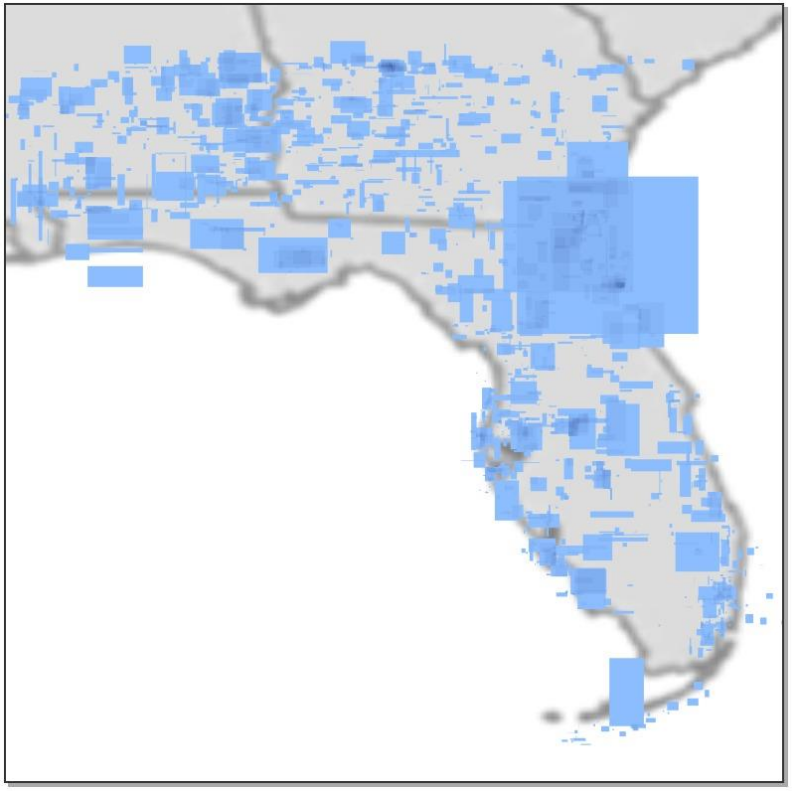


Overview of Florida [1953-2018]

US storm events database



Grid-based visualization

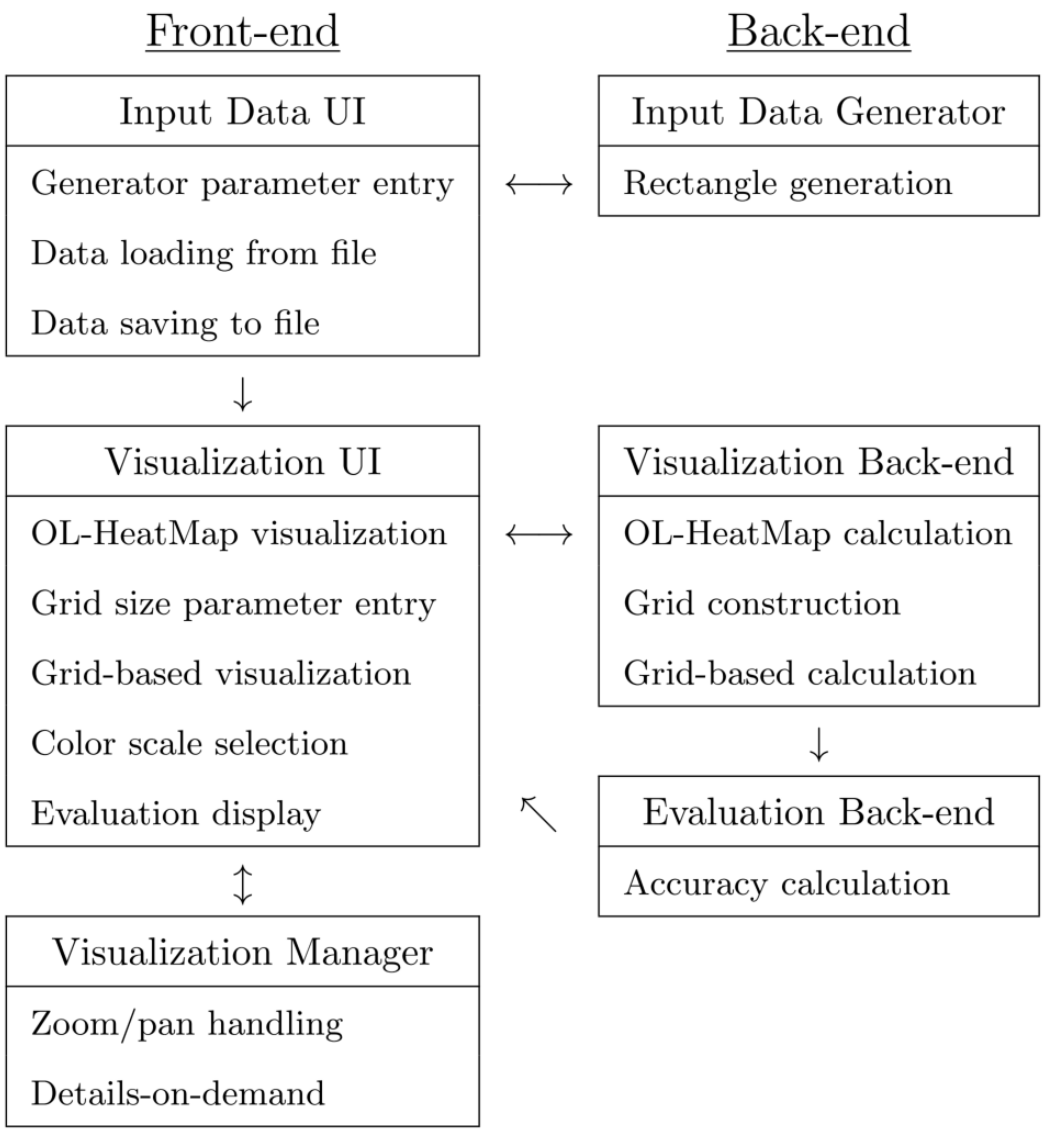


OL-HeatMap

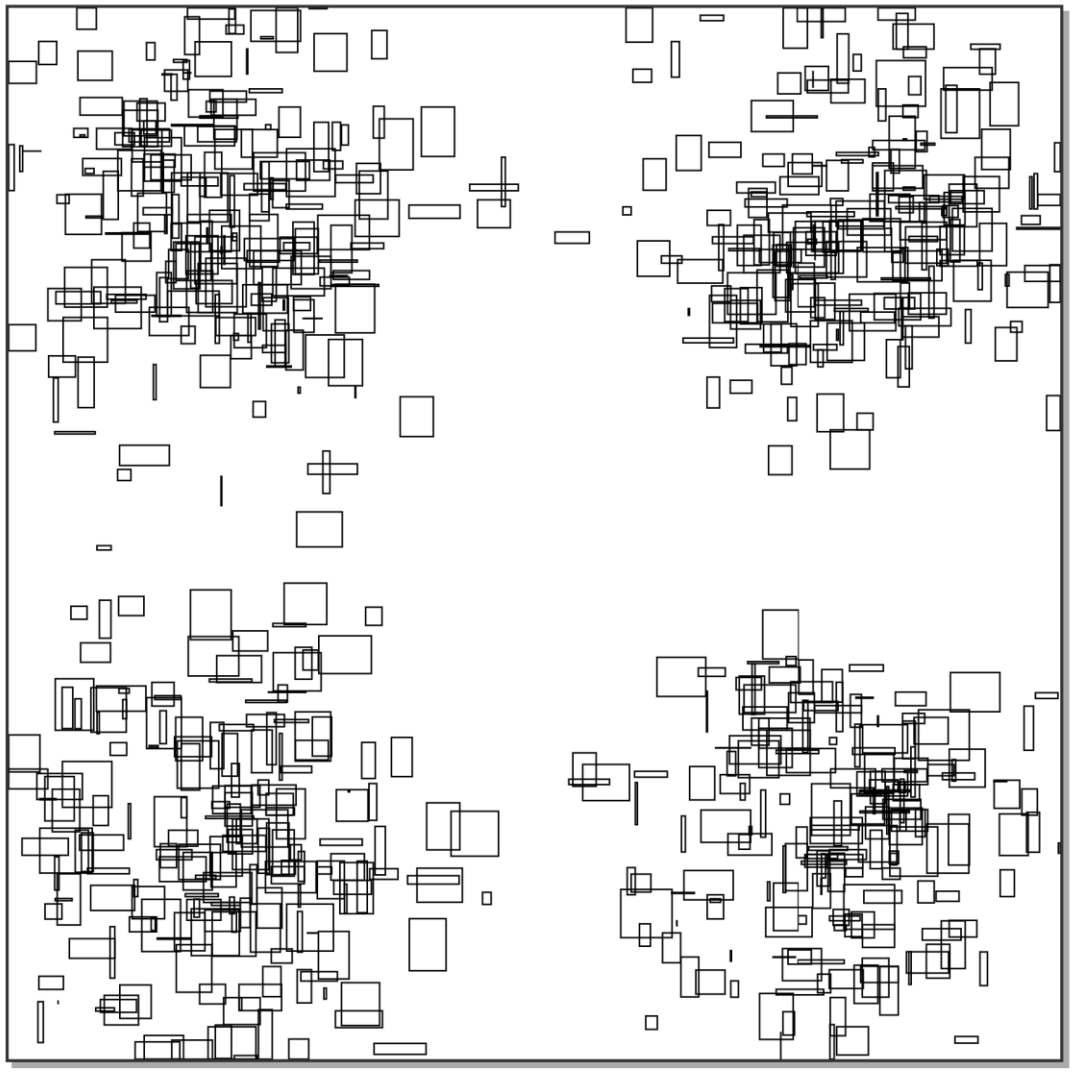
Proof-of-Concept Demo System

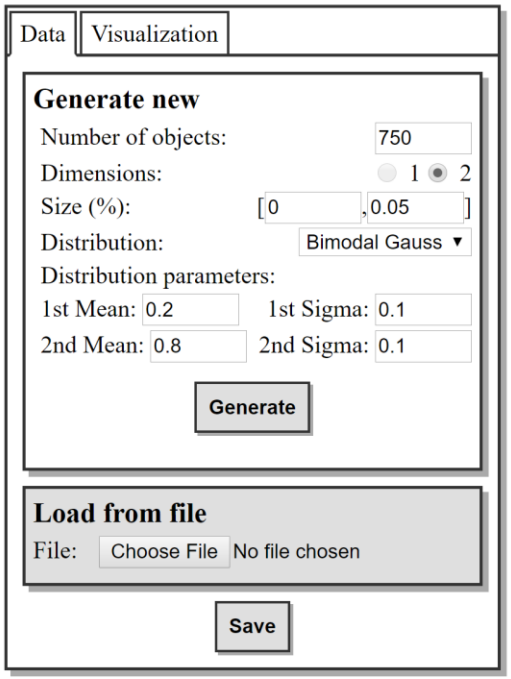
YORK
UNIVERSITÉ
UNIVERSITY

System overview



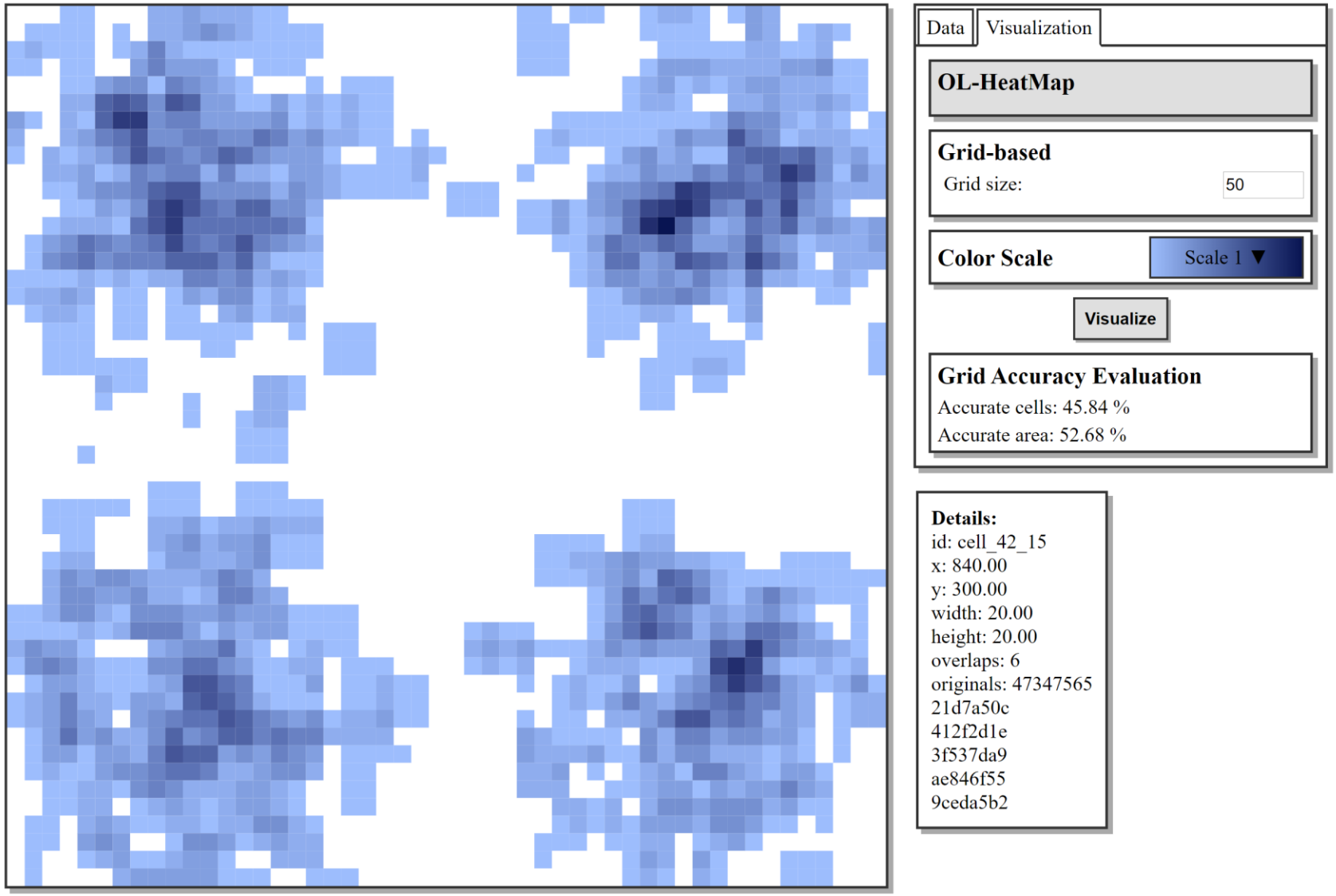
User interface





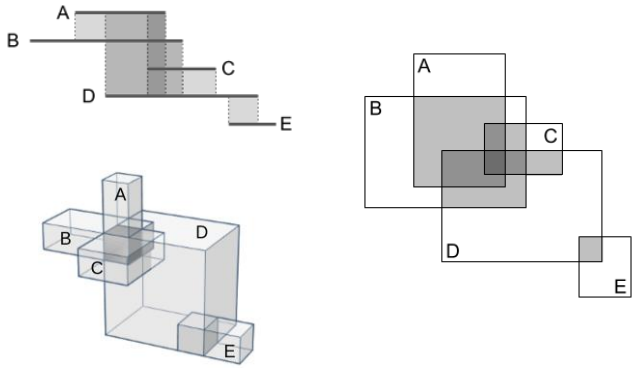
Input Data UI

User interface

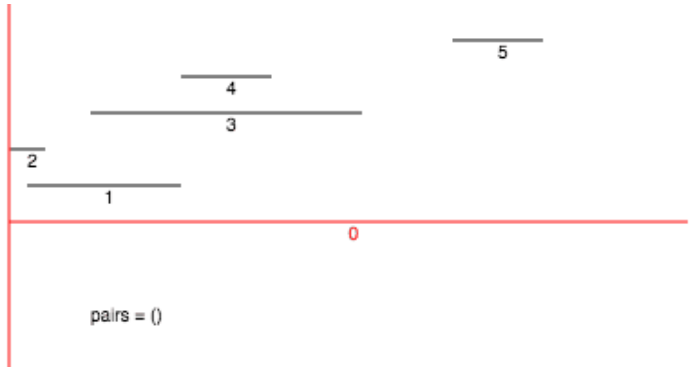


Visualization UI

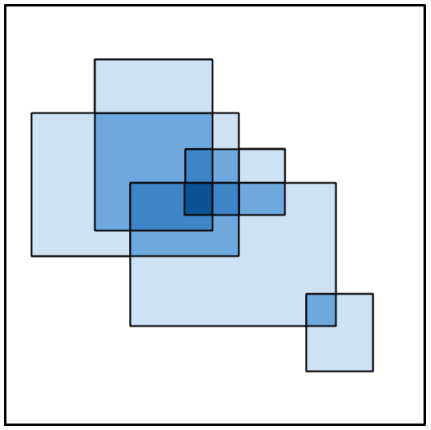
Take-away message



Finding multiple axis-aligned object intersections



OL-HeatMap – a powerful sweep-line based algorithm for finding density



Faster visualization rendering

OL-HeatMap properties:

- fast
- exact
- versatile

Thank you!

Questions?

YORK
UNIVERSITÉ
UNIVERSITY