ImMens:

Scribe: Yiru Chen

Comments:
So far, we have explored:
1- polaris:
2- vega-lite:
3- draco:
4: voyager:
These systems focused on how to construct visualizations, either by providing grammar of graphics, constraint based input, building recommendation engine that combine both to find the best set of visualizations that would facilitate data exploration.
Immens focuses on the scalability problem and performance issue.

Q: How to choose number of bins?
A: They treat bin count as an adjustable parameter bounded by the screen pixels. They don't go into details on how the bins are chosen. Although Constrained by texture size: 256 bins per dimension for 3D data and 64 bins per dimension for a 4D tile.

Q: Does the number of bin matter?
A: Depending on how you set the bin, you can see artifacts. So, deciding on the bin is a technical problem.
A: The reduction does not depend on the axis you are selecting.

Q: When they say they reduce, how is the reduction strategy?
A: For here if you have all 5d cubes, that will give you things that within this interface you do not need. If you focus on the visualization, you do not need all the combination in these cubes. Example, part of the cube is enough to update a view.
A(supplement): In the 5 D cubes, you have interactions to filter by month, year…. You can have a lot of combinations. Here, it is the interface that does not allow you to do that. You can only filter it by one view and one view... Because they restrict you on what you can do. They do not need to compute the whole thing. For example, four 3-dimensional cubes can cover all the possible brushing and linking scenarios in paper. In that case, computing the low dimension cube is sufficient to support interactions. So now what you can do is you have low dimension cubes. You have quadratic pairs.

Q: I am interested in how they support panning and zooming. If zooming, the bin would be smaller. If the number is fixed by the programmer. How to figure changes when zooming?
A: Since it is precomputed, the zoom level is pre-defined. You zoom in and zoom out, you can load different tiles according to the zoom level. When zooming, it will load something that already computed. You can also compute on the fly. But they did not do it here.

Q: Kyrix claims “Multidimensional data tiles/cubes [LJH13, LKS13, PSSC17, Bre16] have been widely adopted to support interactive aggregation queries. However, due to huge amounts of memory used, the index structures proposed cannot support complex pan/zoom interactions where frequent querying of visual objects falling in a rectangular viewport is needed.”. Is it wrong?
A: The word choosing is wired. If they are talking about aggregation visualization like this, it is wrong. But they are not focusing on general aggregation visualizations, they are showing individual points. The paper shows general visualization. Under that context, they say this.

Q: Is the bin the same size across them? If you are geological thing, you may need finer grain bin size.
A: It is equi-width. The reason you do that is you want the index to do the optimization. The benefit is you can look up by one index look up which is very efficient. But the downside is that you cannot control the granularity.
A: Data cube has been around for decades. I am sure there are papers talking about different granularities.

The expressiveness:
Immens does not do everything. What class of sql queries that it can express?
A: Specific class.
SELECT a1, [a2], count(*)
WHERE a3=? GROUP BY a1, [a2]
Aggregation. If you want to complex query, nested query, join tables…, it is not so good. It is worth to think about what expressiveness it supports. Here it is very specific. it is a single table, and you can only filter one thing. You cannot filter by multiple things.

Q: Do you make this choice because something cannot be scaled?
A: This paper is about technique. You find a technique, and then you find a use case which can do this. If you have this technique, you can consider what you can do.
A: Scalability is not a task. Scalability is a technical problem that you want to solve this to achieve the task. The mechanism is I want a visualization of this type so that I can learn something. The task is what you try to learn.
A: For small data you do not need anything. How to define your interface, it depends on the domain.

Applicability
- When are the techniques (binning, cubing) not applicable?
  - Arbitrary brushes
  - Combination of filters
  - Arbitrary aggregates
  - Scatterplot with transparency.

A: Data may spread four pixels. It is not the same as you are directly separate the data. Equi-width bin's alternative would be ranges. It would make look up very expensive. But you can certainly do that.

Motivation: pan/zoom are powerful interaction, data nowadays are getting larger -> doesnt fit in-memory

State-of-the art systems has two main drawbacks:

- Lack scalability
- Lack of data driven primitives
    - Q: what are data driven primitives? A specifications for visualizations.

Contributions:

- General zoom/pan tool
    - Support for general data types and visualizations
    - Does not restrict to aggregation's functions
- Data driven primitive language (claim: easy to use)
    - Zoom is equivalent to changing layer
    - Canvas are composed of layers, layers can be:
        - Dynamic (have placement function)
        - Static (e.g. title)
- Support backend database optimization
    - How to only fetch visual objects that the user might interact with as the user is interacting with the data?
    - Considerations: memory size, cost of lookup (CPU, bandwidth)
    - Issues:
        - Memory size -> precomputed on disk R tree
        - CPU & BW:
            - caching in-frontend and
            - incremental view maintenance
                - As the view changes the frontend fetches new data and removes stale data

Evaluations/Experiments

- Ease of use of the declarative language proposed
- Performance study:
    - Scalability
    - Effect of caching and incremental view maintenance
- Performance on task: demonstrate real-time support

Discussions:

- Is it general?
    - This paper democratizes applications that are based on panning and zooming interactions. It might need extra effort to map and application to these two main interactions, and even if we manage to do that it wouldn't be necessarily performant.
- Do we need generalizable solutions for panning and zooming? Google map already does that pretty well, how useful it would be?

- Is the optimization targeted towards specific tasks?
- What do we think about how long it took to complete the task? It is reasonable. One might compare how long it would take them to build the system vs. how long it took from the users to do that and then decide based on that.
- Are the tasks given in the user study sufficient? What if the user wants to implement a new application from scratch?
- Which class of SQL queries does Kyrix fundamentally express?
    - Anything as they claim?
    - Can we implement imMens using Kyrix?
    - Wu: SELECT * WHERE point WITHIN viewport
- How general is this zoomable UI class of interfaces?
    - They optimized for specific system that uses mainly zoom/pan operations
- Core contribution:
    - Task oriented programming API for XUI
    - Minimum needed to “do the rest”
    - Layer needs only:
        - Select * condition
        - How to position an item?
        - How to render an item?
- Contrast with Immens:
    - Immens: specific optimization, still need to build the app
    - Kyrix: task -specific api that handles the rest