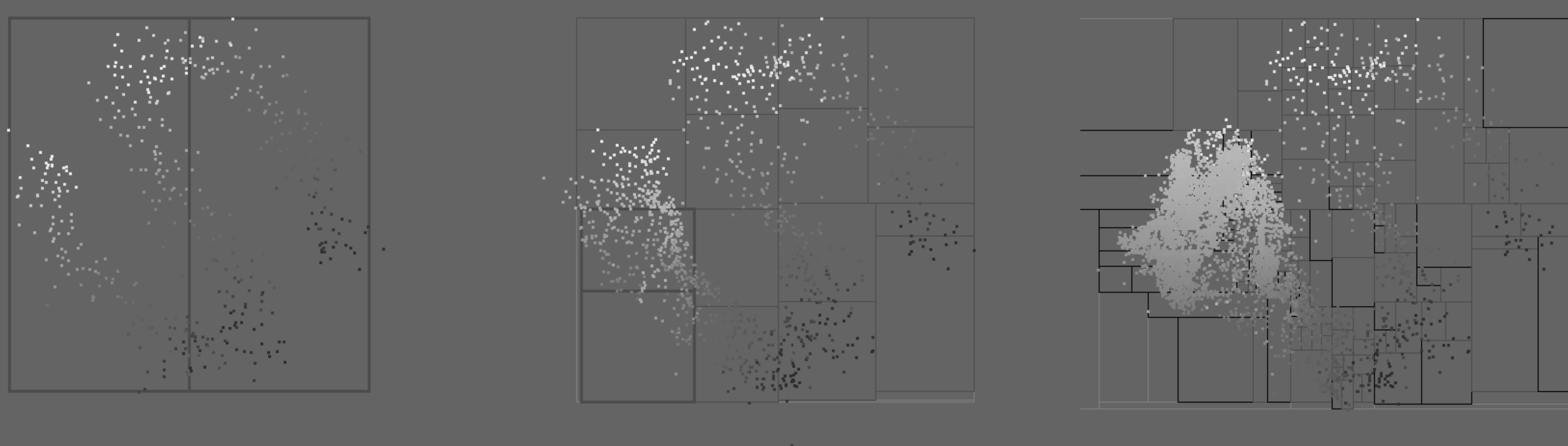
MDSteer: Steerable and Progressive Multidimensional Scaling

Matt Williams and Tamara Munzner

University of British Columbia
Imager Lab



Outline

- Dimensionality Reduction
- Previous Work
- MDSteer Algorithm
- Results and Future Work

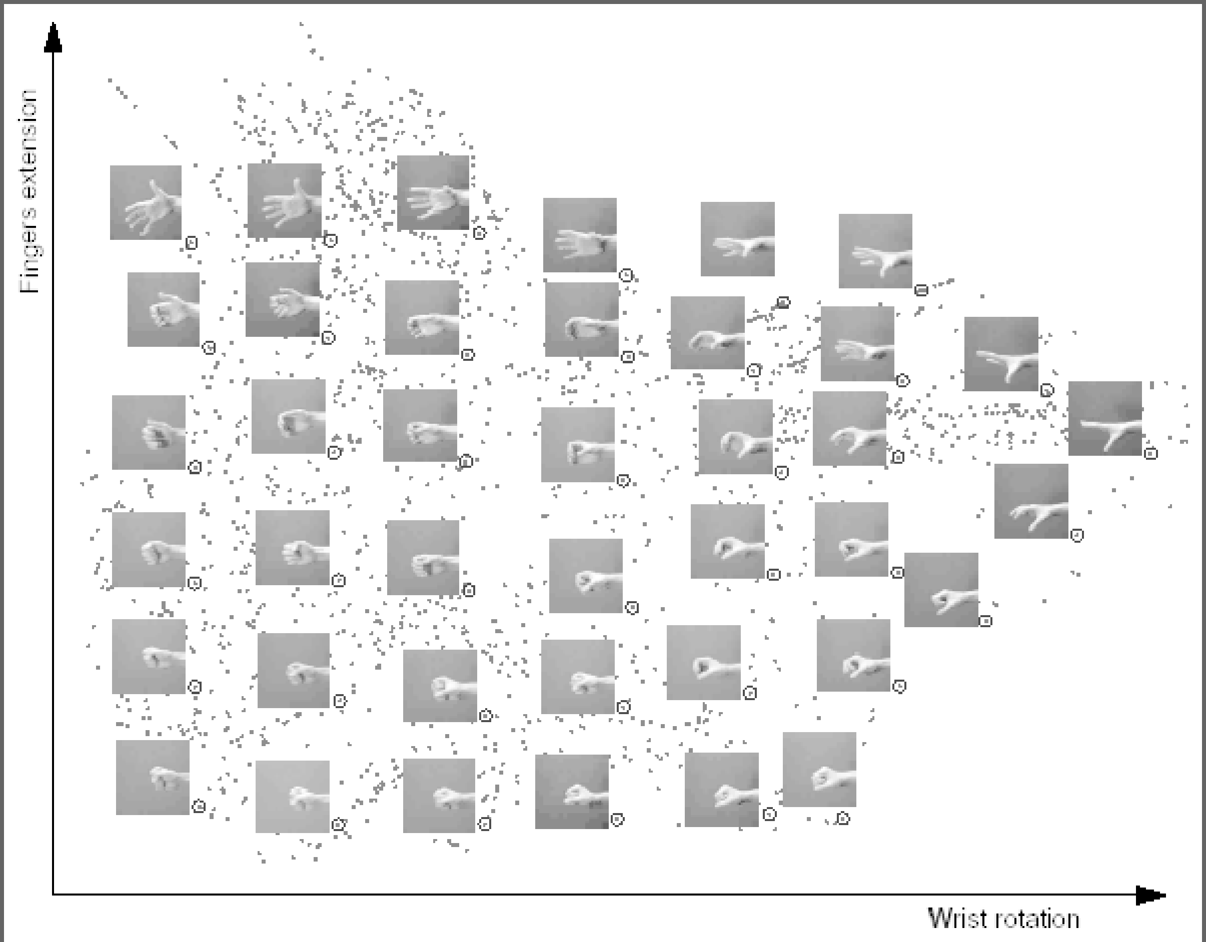
Dimensionality Reduction

- mapping multidimensional space into space of fewer dimensions
  - typically 2D for infovis
  - keep/explain as much variance as possible
  - show underlying dataset structure

- multidimensional scaling (MDS)
  - minimize differences between interpoint distances in high and low dimensions

Dimensionality Reduction Example

- Isomap: 4096 D to 2D [Tenenbaum 00]



[A Global Geometric Framework for Nonlinear Dimensionality Reduction. Tenenbaum, de Silva and Langford. *Science* 290 (5500): 2319-2323, 22 December 2000, isomap.stanford.edu]

Outline

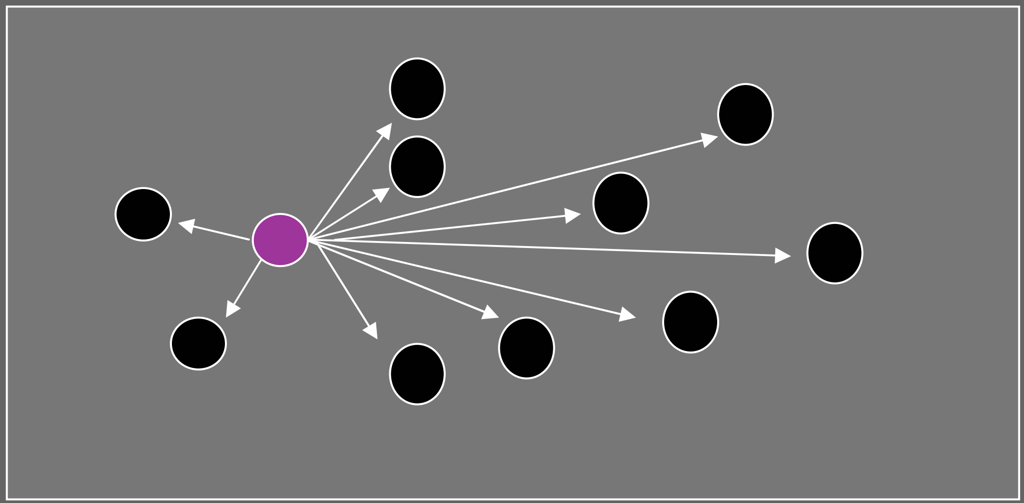
- Dimensionality Reduction
- Previous Work
- MDSteer Algorithm
- Results and Future Work

Previous Work

- MDS: iterative spring model (infovis)
  - [Chalmers 96, Morrison 02, Morrison 03]
  - [Amenta 02]
- eigensolving (machine learning)
  - Isomap [Tenenbaum 00], LLE [Roweis 00]
  - charting [Brand 02]
  - Laplacian Eigenmaps [Belkin 03]
- many other approaches
  - self-organizing maps [Kohonen 95]
  - PCA, factor analysis, projection pursuit

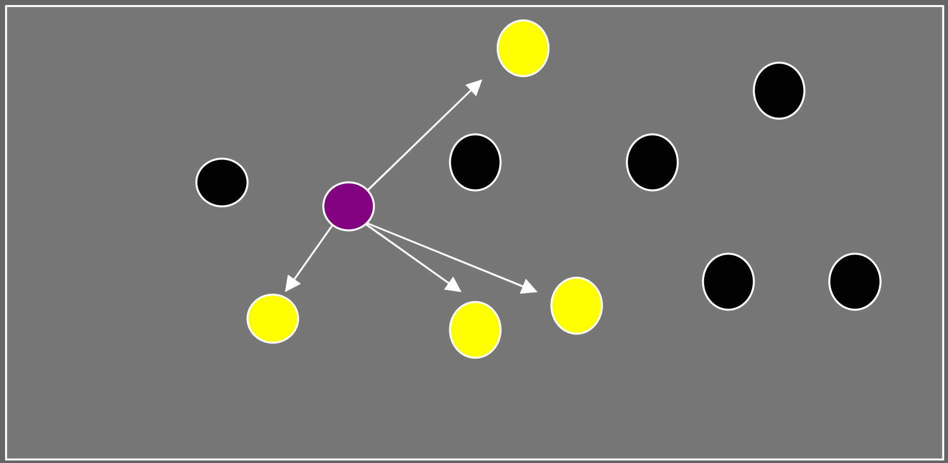
Naive Spring Model

- repeat for all points
  - compute spring force to all other points
    - difference between high dim, low dim distance
  - move to better location using computed forces
- compute distances between all points
  - $O(n^2)$ iteration, $O(n^3)$ algorithm



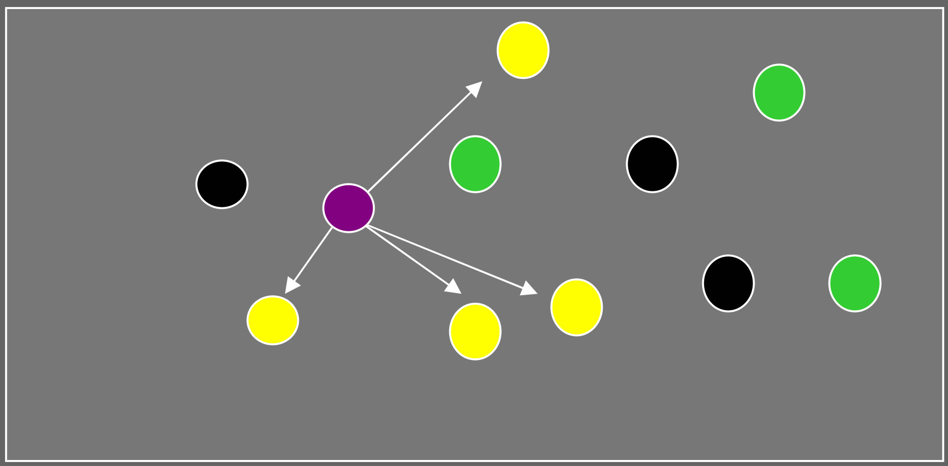
Faster Spring Model [Chalmers 96]

- compare distances only with a few points
  - maintain small local neighborhood set



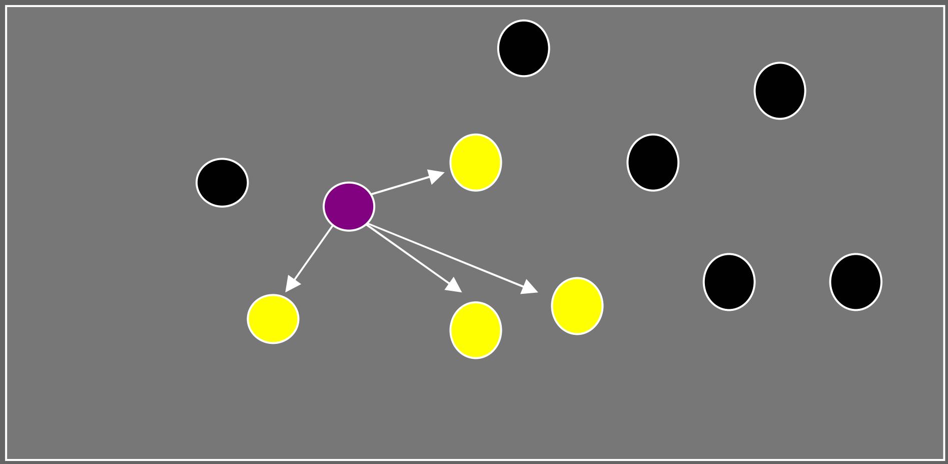
Faster Spring Model [Chalmers 96]

- compare distances only with a few points
  - maintain small local neighborhood set
  - each time pick some randoms, swap in if closer



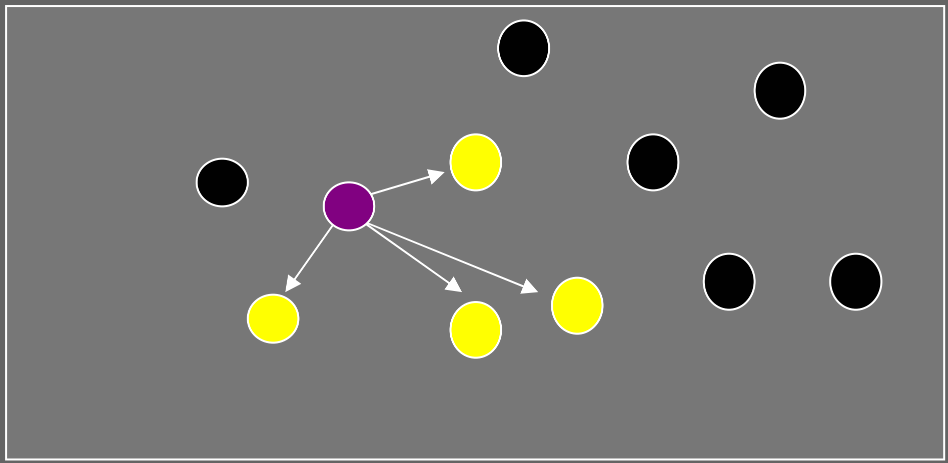
Faster Spring Model [Chalmers 96]

- compare distances only with a few points
  - maintain small local neighborhood set
  - each time pick some randoms, swap in if closer



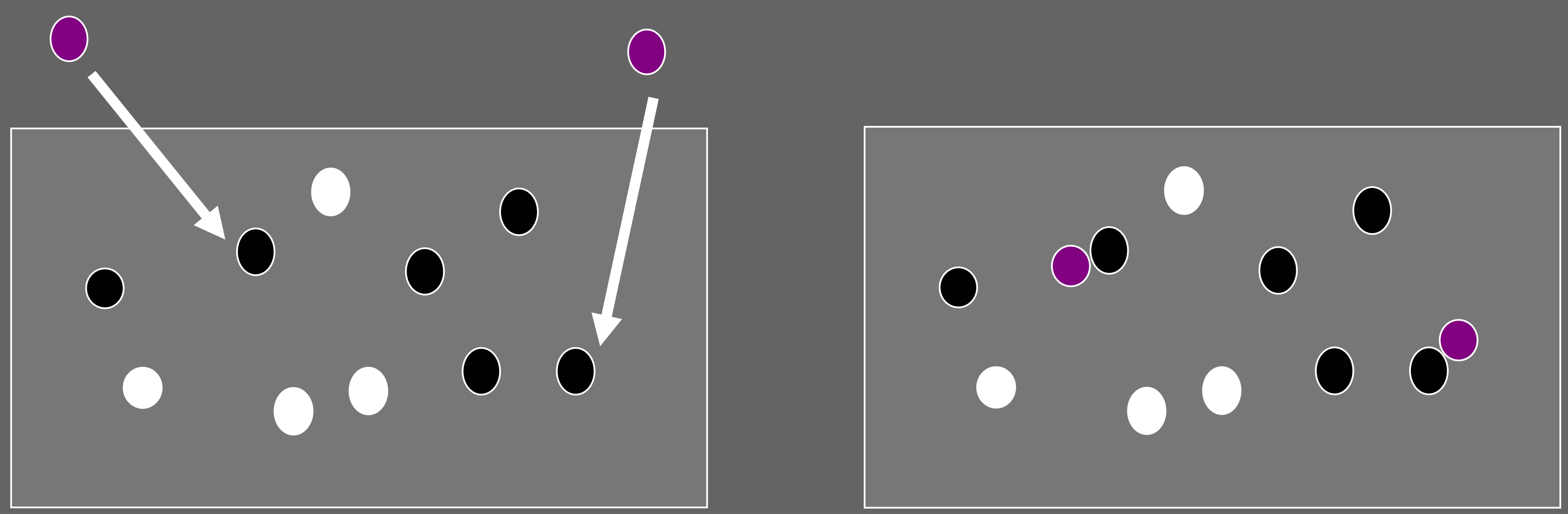
Faster Spring Model [Chalmers 96]

- compare distances only with a few points
  - maintain small local neighborhood set
  - each time pick some randoms, swap in if closer
- small constant: 6 locals, 3 randoms typical
  - O(n) iteration, $O(n^2)$ algorithm



Parent Finding [Morrison 2002, 2003]

- lay out a root(n) subset with [Chalmers 96]
- for all remaining points
  - find “parent”: laid-out point closest in high D
  - place point close to this parent
- O($n^{5/4}$) algorithm



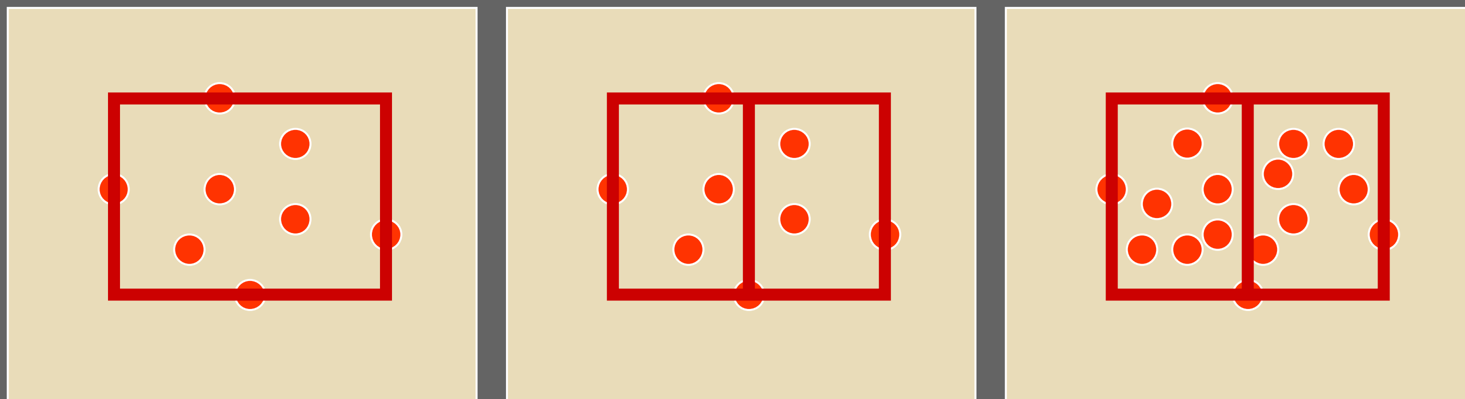
Scalability Limitations

- high cardinality and high dimensionality: still slow
  - motivating dataset: 120K points, 300 dimensions
  - most existing software could not handle at all
  - 2 hours to compute with $O(n^{5/4})$ HIVE [Ross 03]
- real-world need: exploring huge datasets
  - last year's questioner wanted tools for millions of points
- strategy
  - start interactive exploration immediately
    - progressive layout
  - concentrate computational resources in interesting areas
    - steerability
  - often partial layout is adequate for task

Outline

- Dimensionality Reduction
- Previous Work
- MDSteer Algorithm
- Results and Future Work

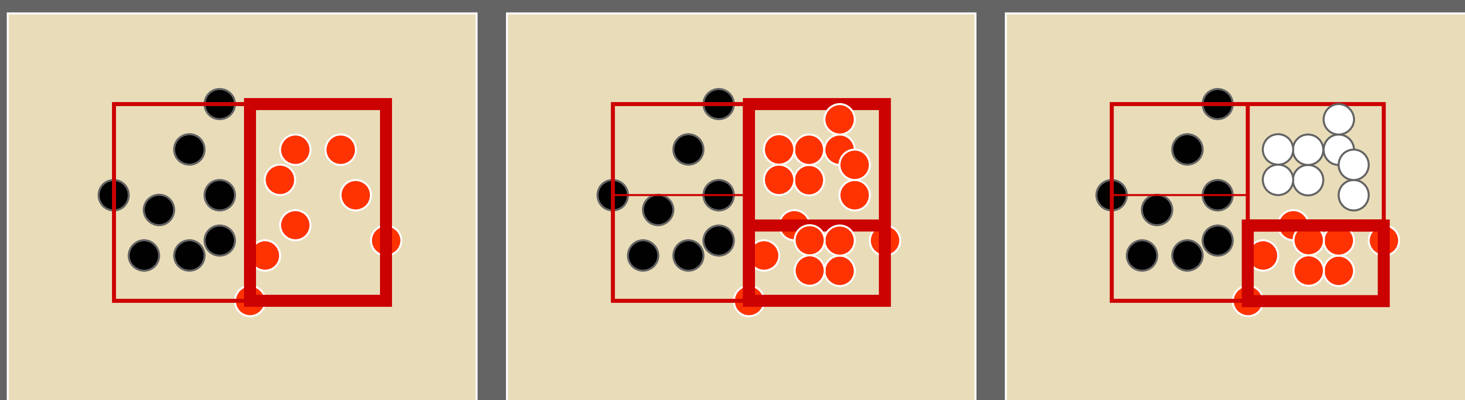
MDSteer Overview



lay out
random subset

subdivide bins

lay out another
random subset

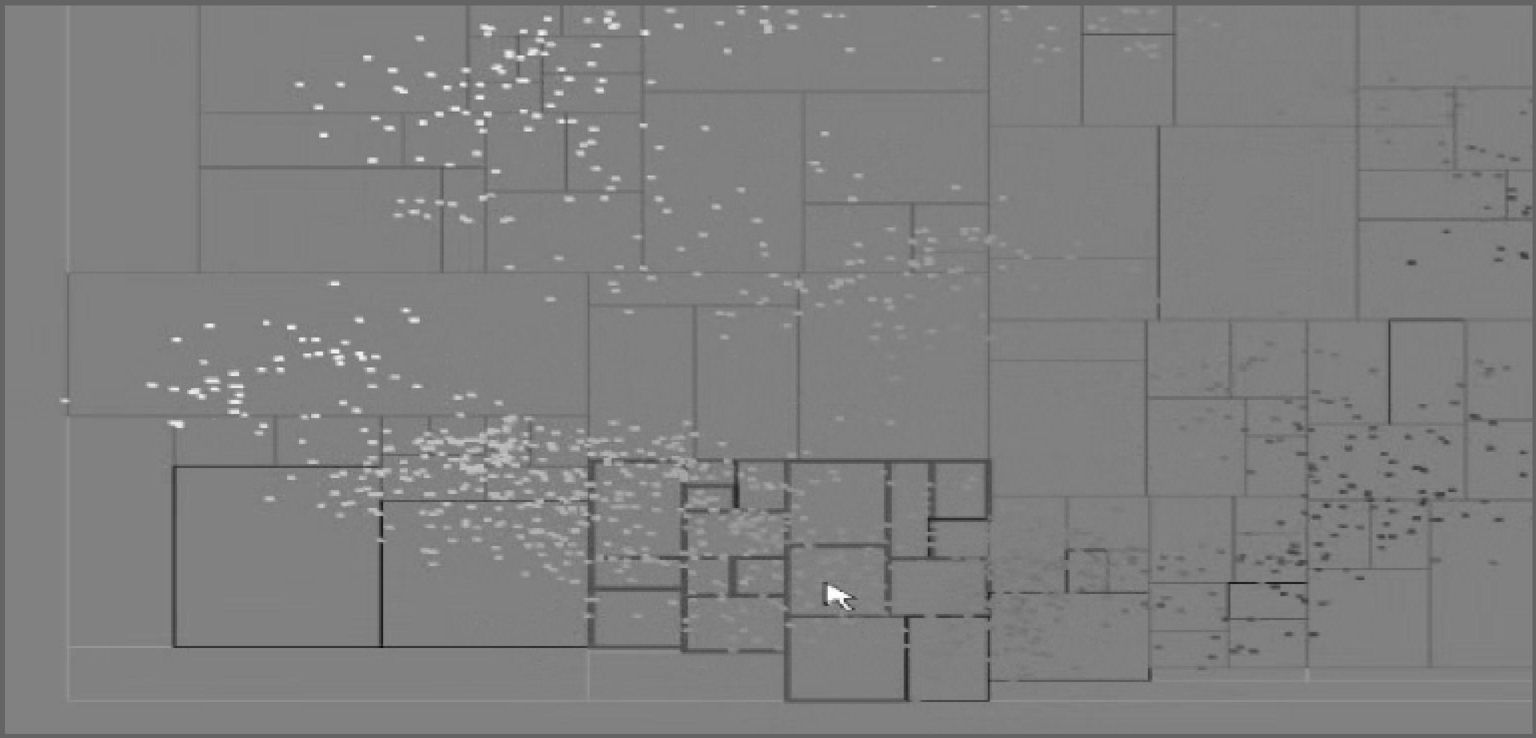


user selects active
region of interest

more subdivisions
and layouts

user refines
active region

Video 1



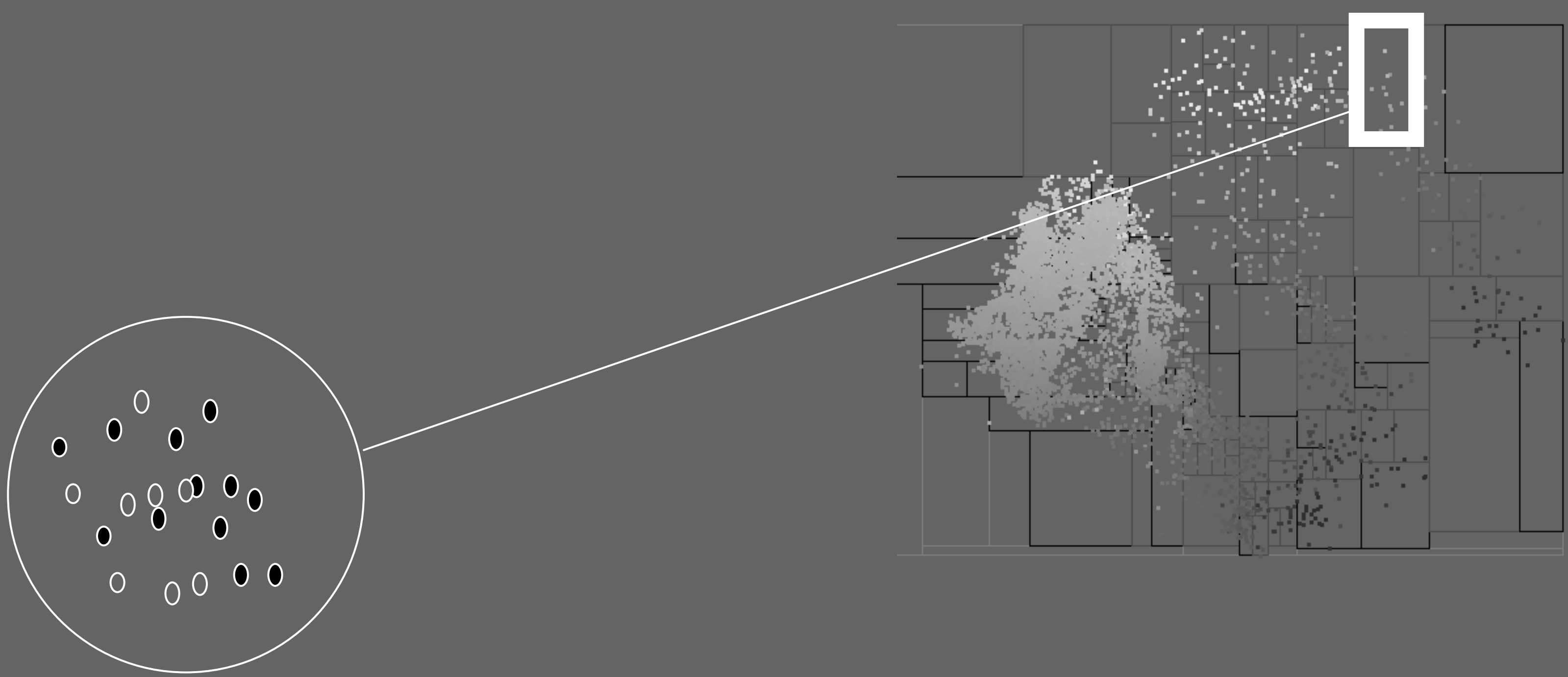
Algorithm Outline

```
lay out initial subset of points
loop {
        lay out some points in active bins
             - precise placement of some

        subdivide bins, rebin all points
            - coarse placement of all
            - gradually refined to smaller regions
}
```

Bins

- screen-space regions
  - placed points: precise lowD placement with MDS
  - unplaced points: rough partition using highD distances

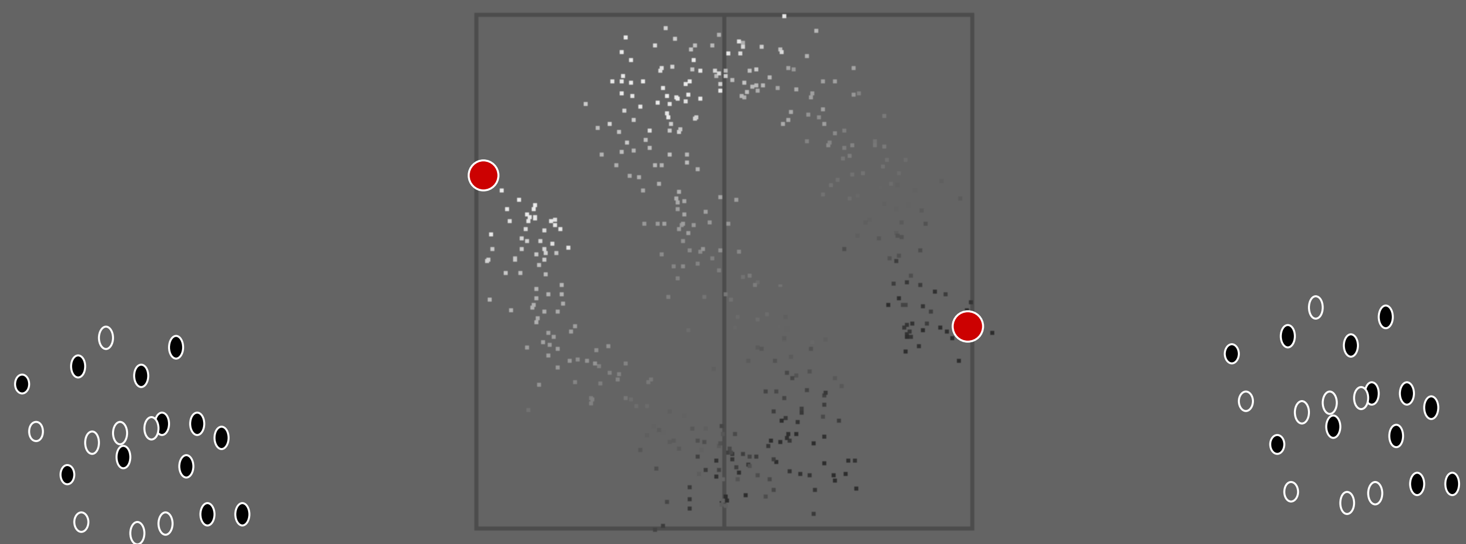


Bins

- incremental computation
  – unplaced points partitioned
  – cheap estimate of final position, refine over time
- interaction
  – user activates screen-space regions of interest
- steerability
  – only run MDS on placed points in active bins
  – only seed new points from active bins
- partition work into equal units
  – roughly constant number of points per bin
  – as more points added, bins subdivided

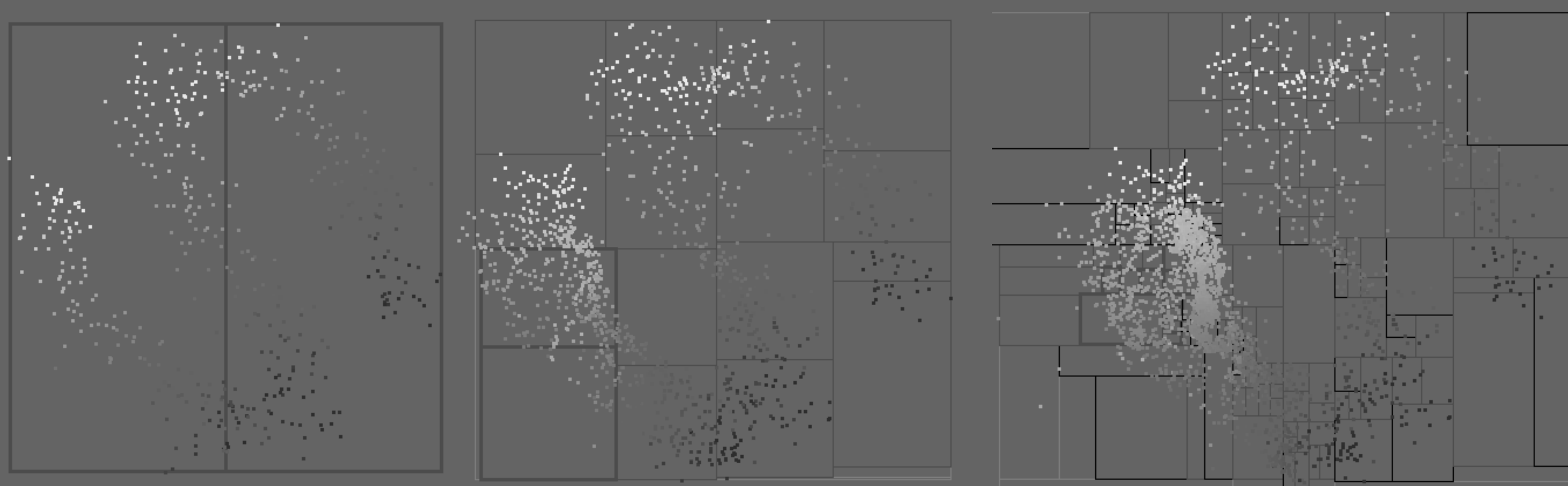
Rebinning

- find min and max representative points
  - alternate between horizontal and vertical
- split bin halfway between them
- rebin placed points: lowD distance from reps
- rebin unplaced points: highD distance from reps



Recursive Subdivision

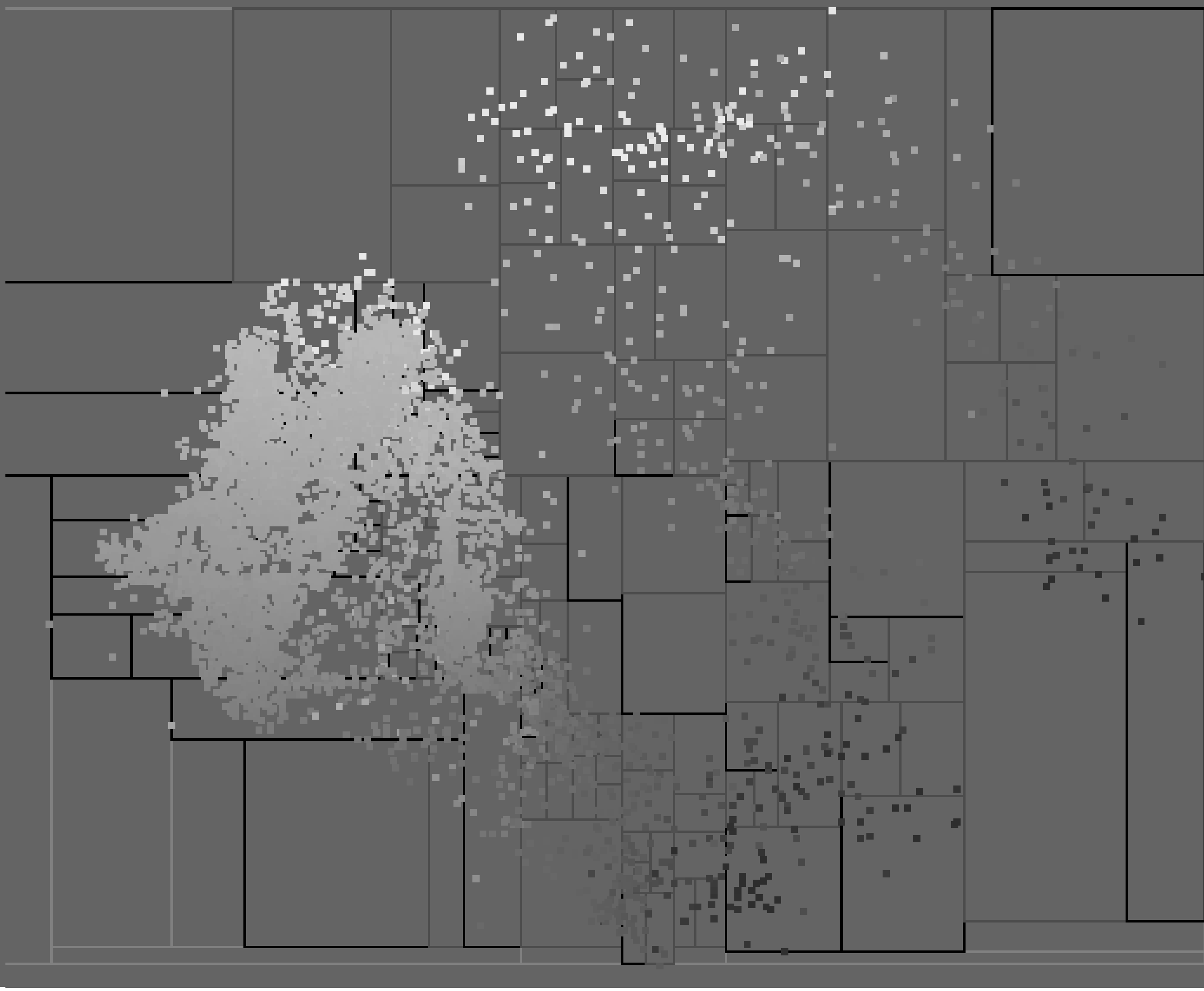
- start with single top bin
  - contains initial root(n) set of placed points
- subdivide when each new subset placed



Irregular Structure

– split based on screen-space point locations

– only split if point count above threshold



Steerability

- user selects screen-space bins of interest
- screen space defines “interesting”
  - explore patterns as they form in lowD space
  - points can move between bins in MDS placement
    - MDS iterations stop when points move to inactive bins

Steerability

- approximate partitioning
  - point destined for bin A may be in bin B's unplaced set
  - will not be placed unless B is activated
- allocation of computation time
  - user-directed: MDS placement in activated areas
  - general: rebinning of all points to refine partitions
  - rebinning cost grows with
    - dimensionality
    - cardinality
- traditional behavior possible, just select all bins

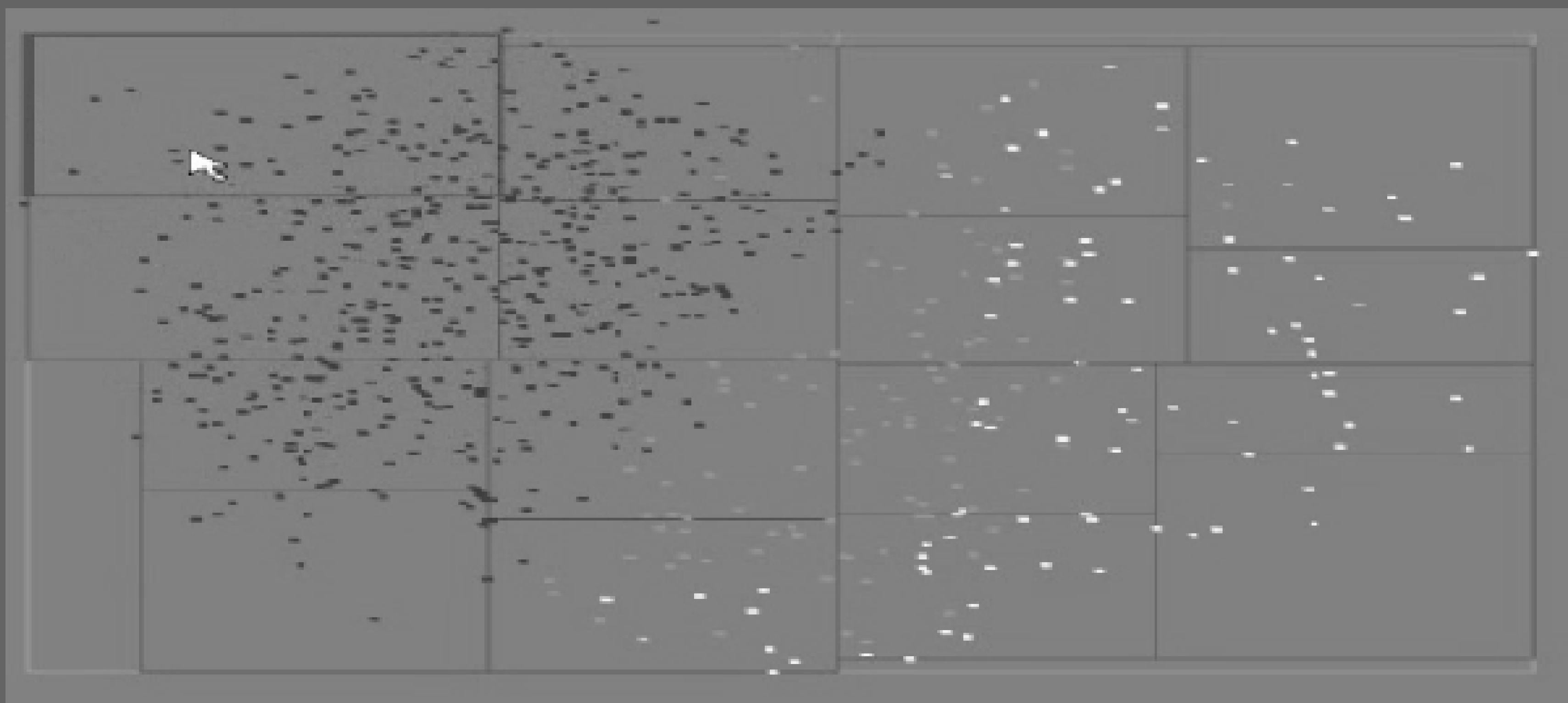
Algorithm Loop Details

```
until all points in selected bins are placed {
    add sampleSize points from selected bins
    until stress stops shrinking {
            for all points in selected bins {
                    run [Chalmers96] iteration
                    calculate stress } }
    divide all bins in half
    rebin all points }
```

Outline

- Dimensionality Reduction
- Previous Work
- MDSteer Algorithm
- Results and Future Work

Video 2



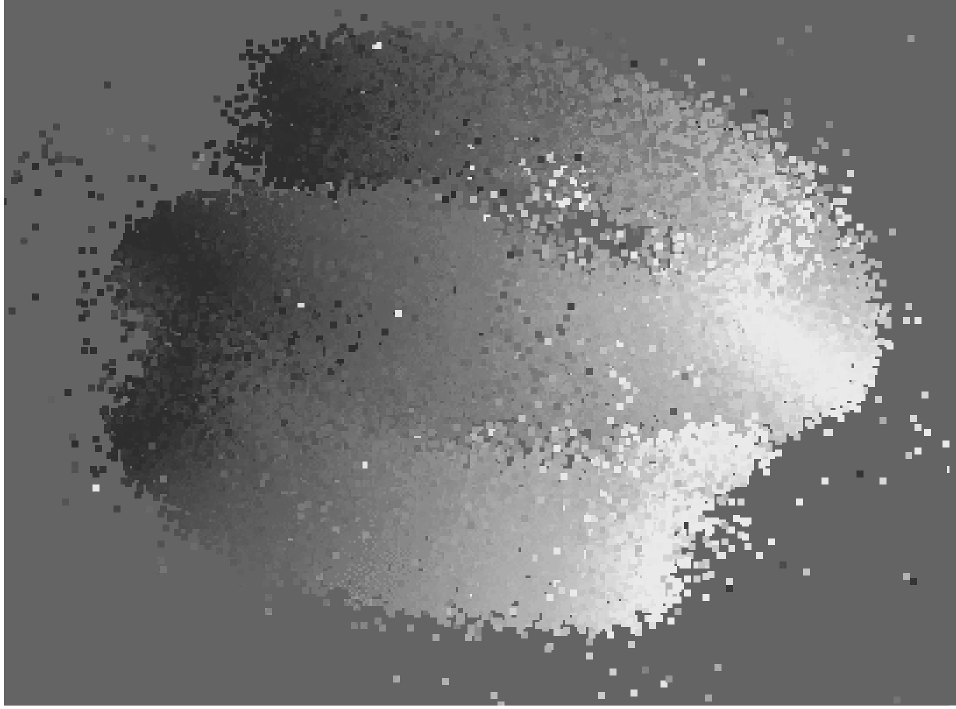
Comparison

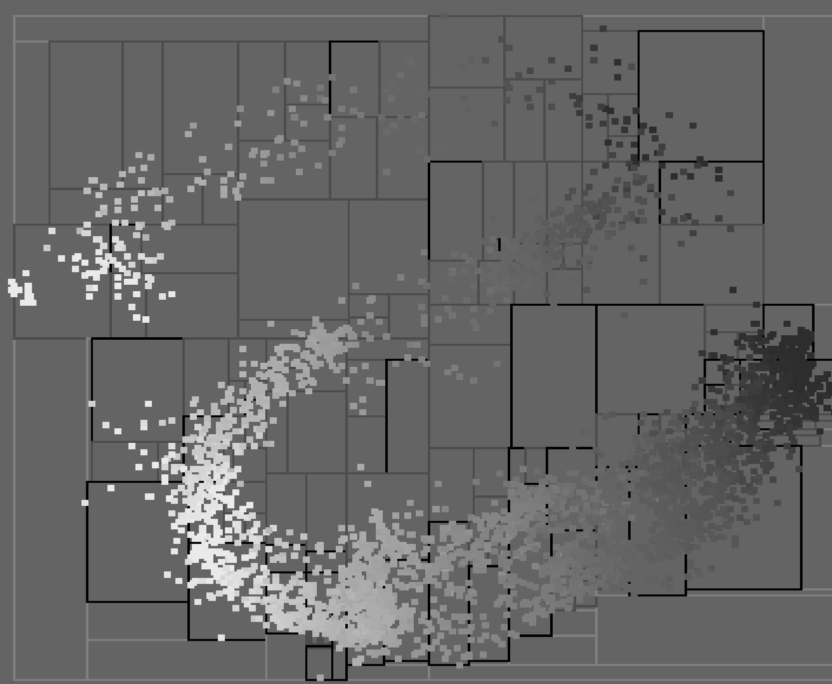
Standard MDS

- all points placed
- hours to compute for big data (100K points, 300 dim)

MDSteer

- user-chosen subset of points placed
- progressive, steerable
- immediate visual feedback

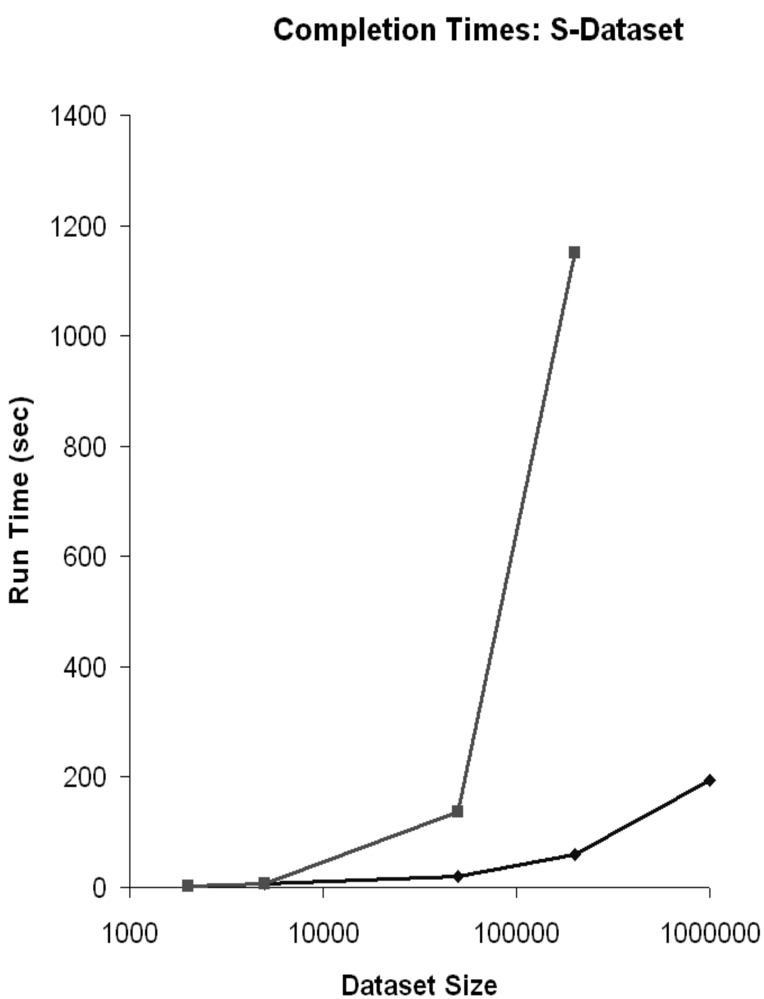




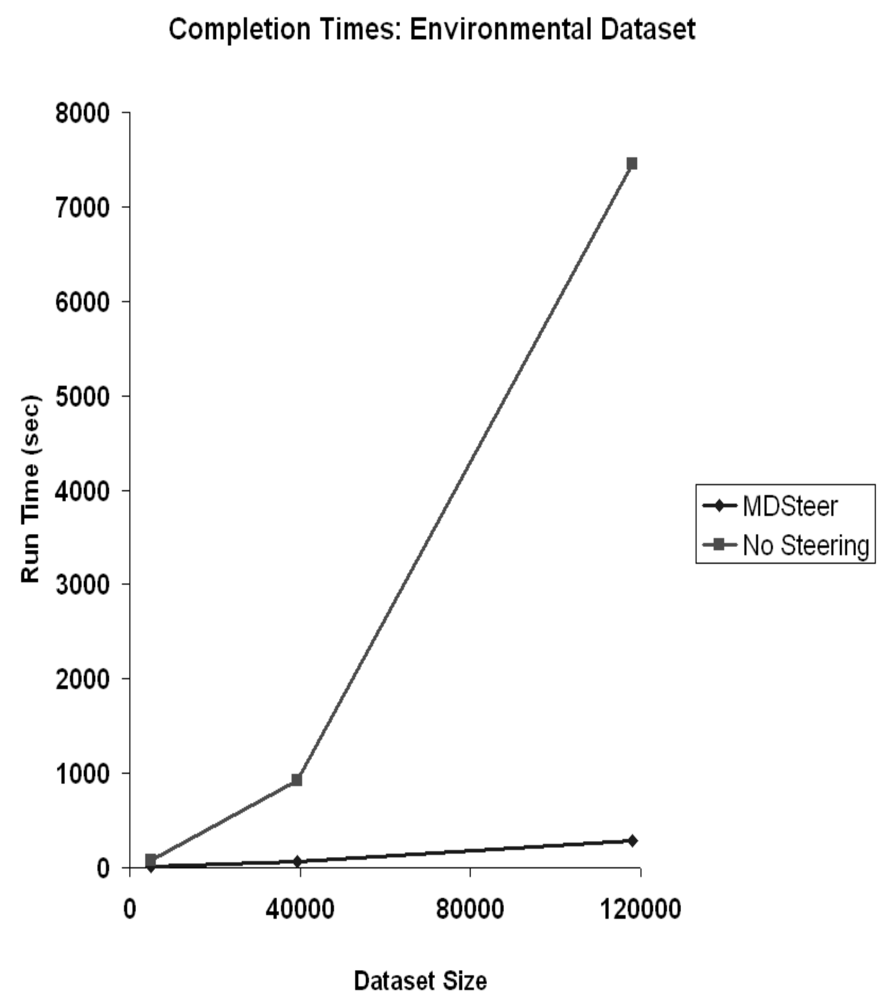
Results: Speed

- unsurprisingly, faster since fewer points placed

3 dimensional data



300 dimensional data

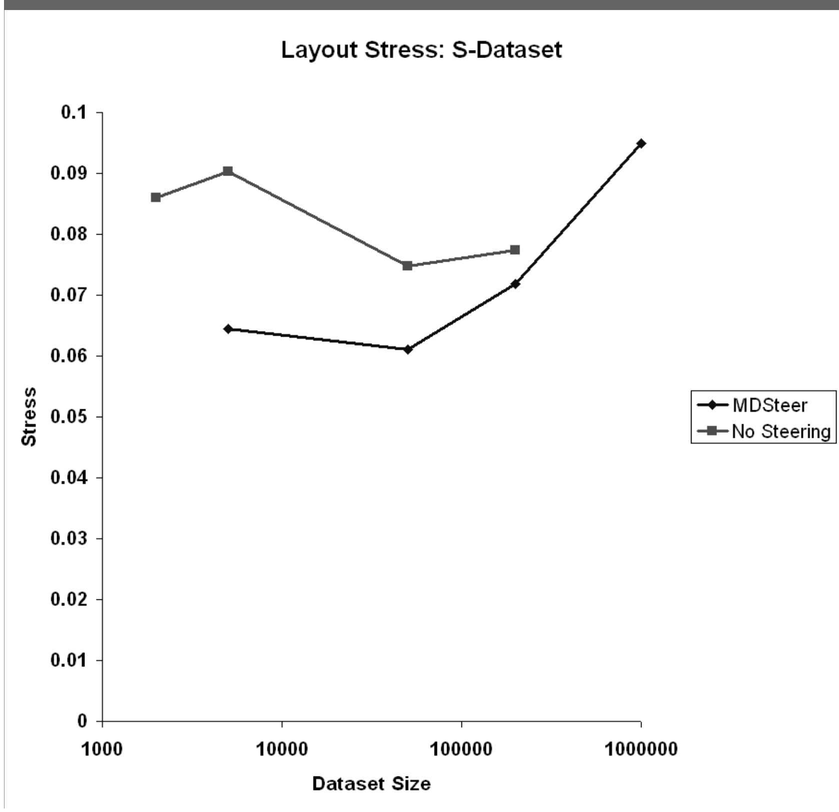


Results: Stress

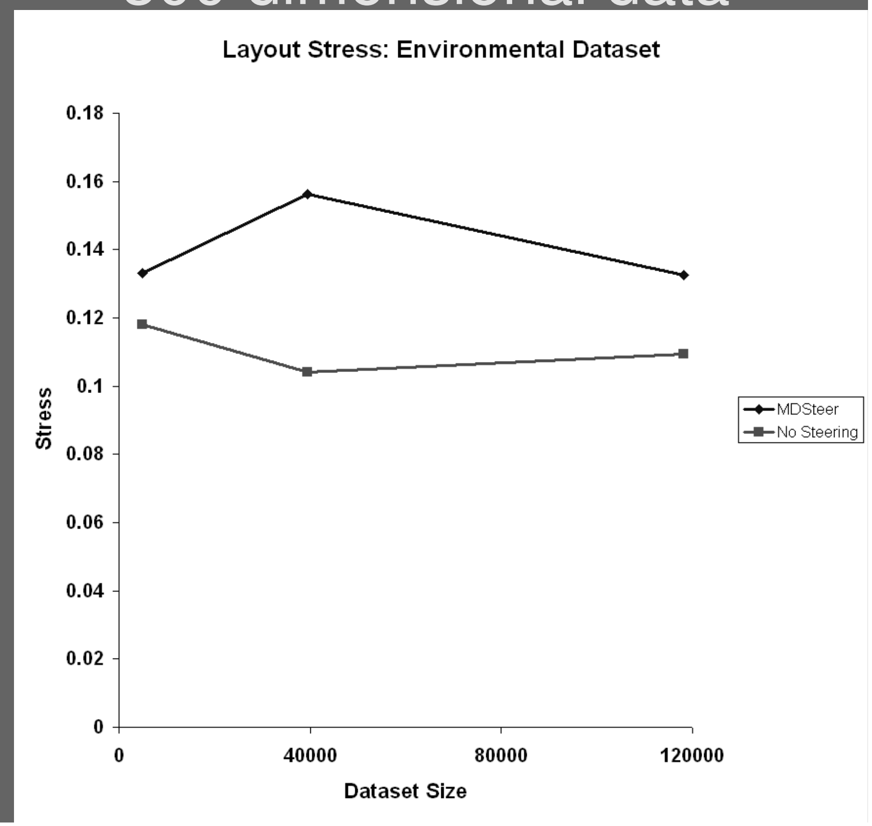
$$Stress = \frac{\sum_{i<j}(d_{ij} - p_{ij})^2}{\sum_{i<j} p_{ij}^2}$$

- difference between high dimensional distance and layout distances
  - one measure of layout quality
- d_{ij} – high dim distance between i and j
- p_{ij} – layout distance between i and j

3 dimensional data



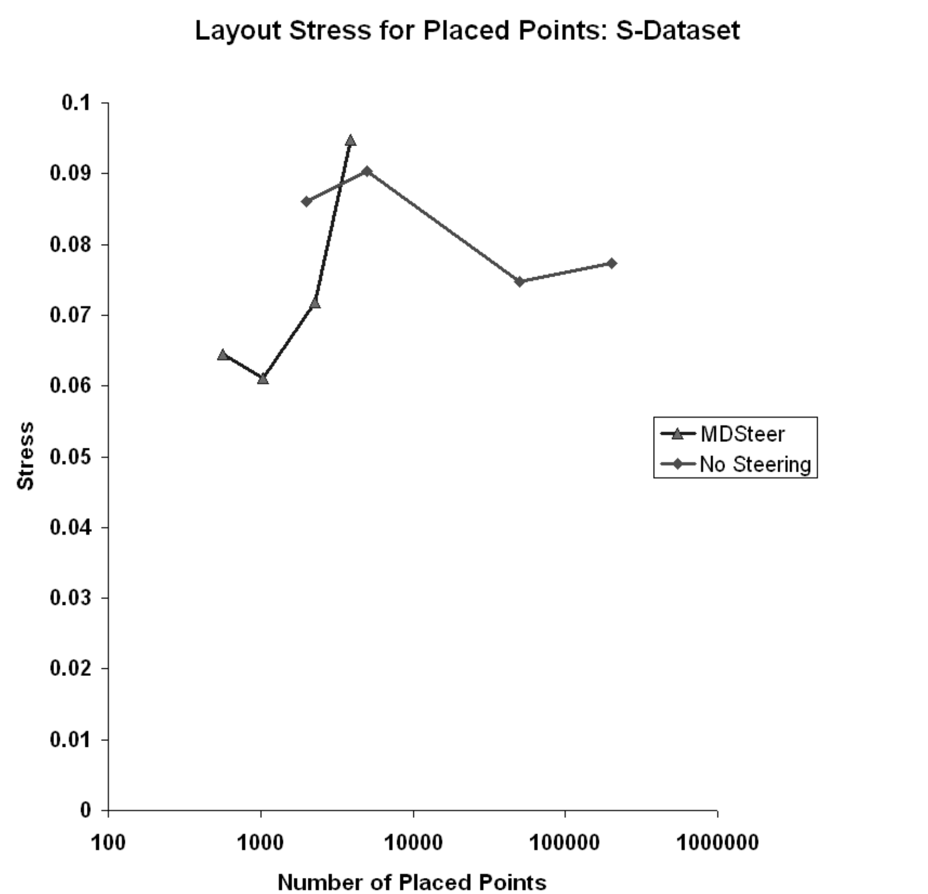
300 dimensional data



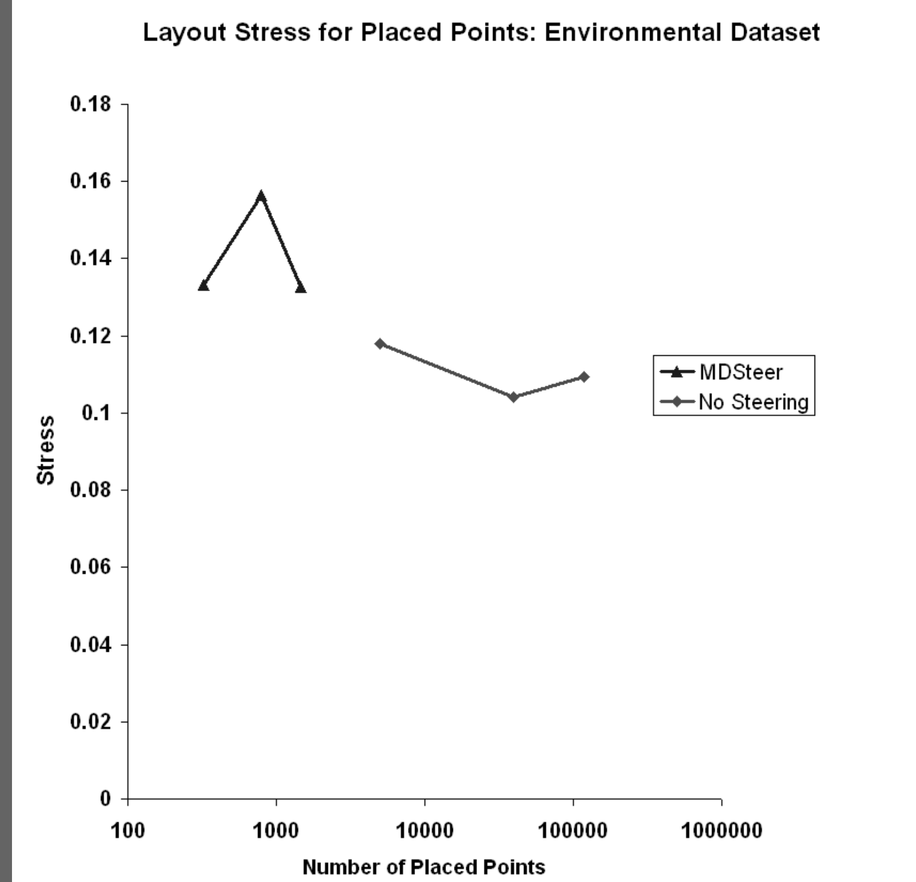
Results: Stress For Placed Points

- placed << total during interactive session
- passes sanity check: acceptable quality

3 dimensional data

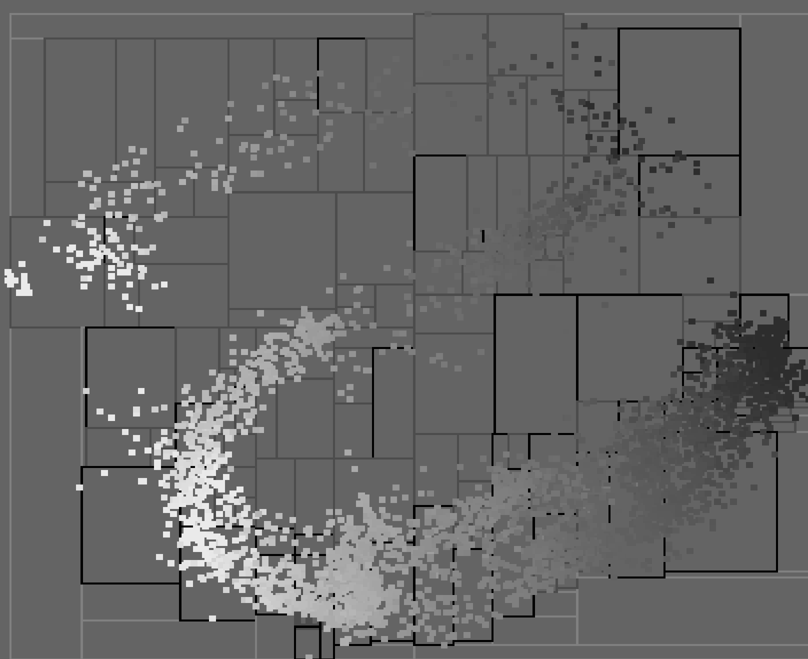


300 dimensional data



Contributions

- first steerable MDS algorithm
  - progressive layout allows immediate exploration
  - allocate computational resources in lowD space



Future Work

- fully progressive
  - gradual binning
  - automatic expansion of active area
- dynamic/streaming data
- steerability
  - find best way to steer
  - steerable eigensolvers?
- manifold finding

Acknowledgements

- datasets
  - Envision, SDRI
- discussions
  - Katherine St. John, Nina Amenta, Nando de Freitas
- technical writing
  - Ciaran Llachlan Leavitt
- funding
  - GEOIDE NCE (GEOmatics for Informed DEcisions)