Interactive Editing of Large Point Clouds

Michael Wand

Max Planck-Center VCC
Stanford University / MPII Saarbrücken

Alexander Berner, Martin Bokeloh, Arno Fleck, Mark Hoffmann, Philipp Jenke, Benjamin Maier, Dirk Staneker, Andreas Schilling

WSI/GRIS, University of Tübingen

SYMPOSIUM ON POINT-BASED GRAPHICS

Overview

Talk Overview

- Introduction
- Data Structure & Algorithms
- Software Architecture
- Results
- Conclusions & Future Work

Introduction

Problem Statement

Goal of this work:

- Interactive editor for large point clouds

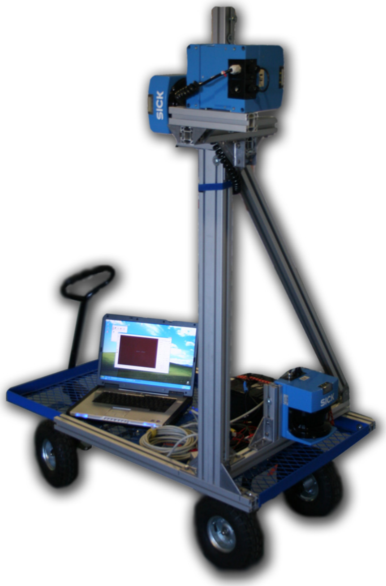
Two Contributions:

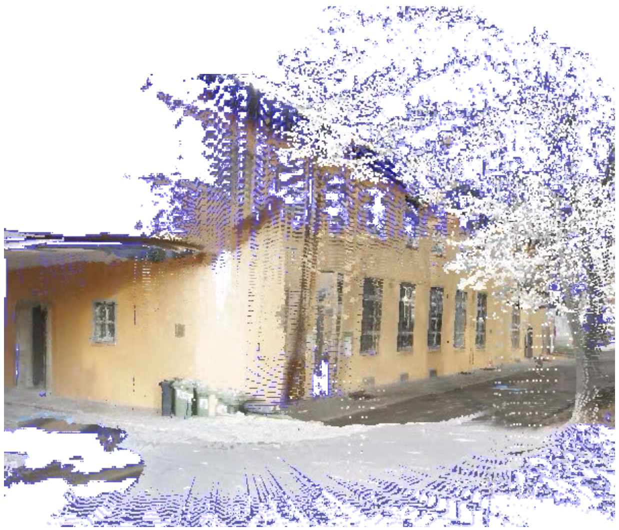
- Fully dynamic out-of-core multi-resolution data structure for point clouds
- Large scene editor architecture

Application Scenario

Example: Urban Scanning

- "Drive-by" 3D scanning devices
- Can acquire a whole city
- Easy to get 10 - 100 GB of 3D data
- Terabytes for a complete model
- "Google Earth"

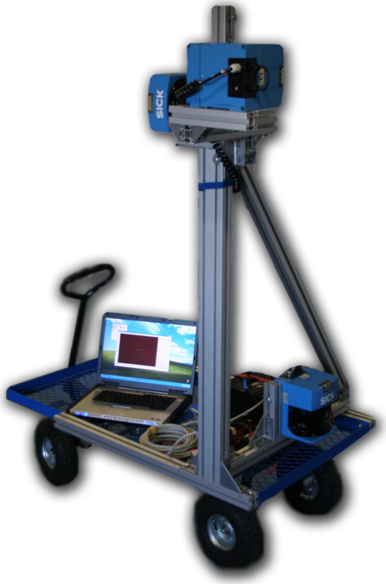


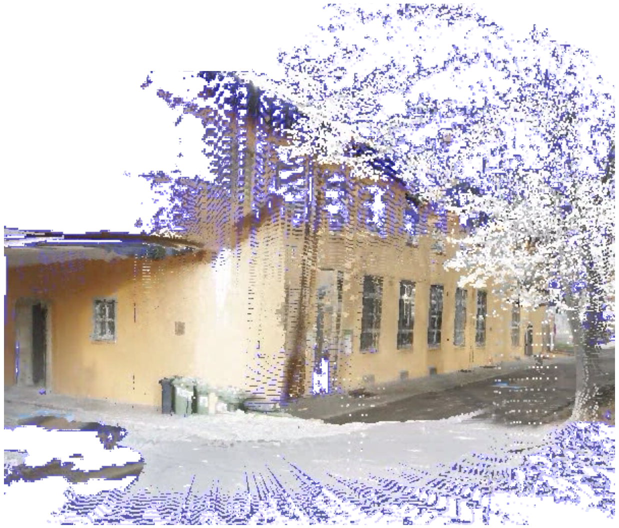


Application Scenario

Typical Data Processing

- Automatic processing
  - Filtering, normal estimation, hole filling, etc.
- Manual processing
  - Remove people driving/walking by
  - Paint over license plates numbers
- Global and local operations





Our Approach

New Data Structure:

- Real-time rendering of large models
- *Local* modifications in *real-time*
- *Global* operations in *batch mode*

Software System:

- Automatic *scripting* for *interactive global* modifications

Related Work

Out-of-Core Multi-Resolution Data Structures

- [Hoppe 98, Lindstrom 00, Shaffer et al. 01, Lindstrom 03, Yoon et al. 04, Cignoni et al 04, Guthe et al. 04., Shaffer et al. 05]

Point-Based Approaches

- Multi-Resolution [Pfister et al. 00, Rusinkiewicz et al. 00, Wand et al. 01]
- Out-of-core [Rusinkiewicz et al. 01, Gobbetti et al. 04, Wimmer et al. 06]

Related Work

Point-Based Editing:

- Pointshop [Zwicker et al. 02, Pauly et al. 03, Weyrich et al. 04]
- Attribute painting [Boubekeur et al. 06]

Multi-Resolution Editing:

- Wavelet-based Image/Terrain editing [Berman et al. 94, Atlan et al. 06]
- Multi-Resolution Surfaces [Zorin et al. 97, Pauly et al. 06]

Related Work

Processing Huge Models:

- Octree partitioning [Cignoni et al. 03]
- Stream processing of meshes [Isenburg et al. 03, Isenburg et al. 05]
- Streaming processing of point clouds [Pajarola 05]

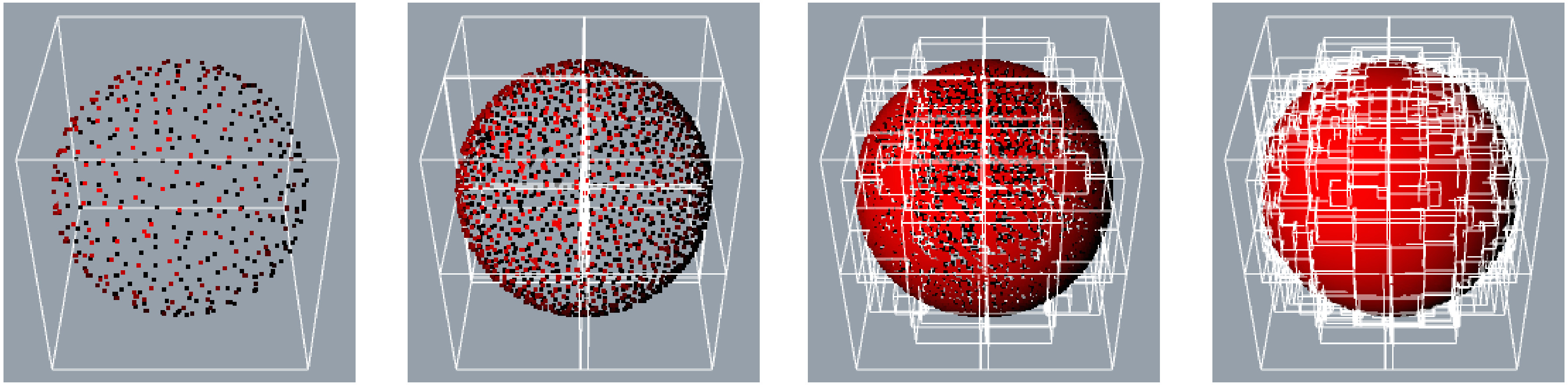
Data Structure & Algorithms

New Data Structure

Overview

- “Surfels” hierarchy [Pfister et al. 00]
- Create multi-resolution representation (MRR) by quantization [Rossignac et al. 93]
- Dynamic octree [Samet 90]
- Update MRR through voxel counting

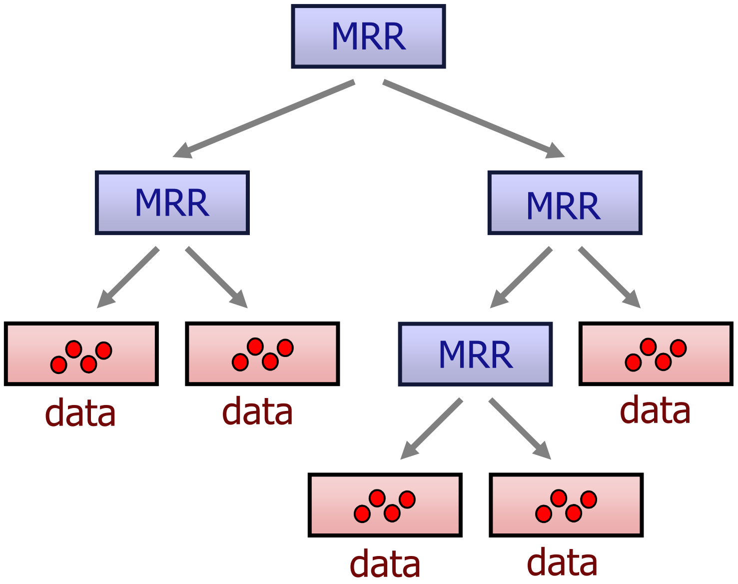
Surfels Hierarchy



Surfel Hierarchy: [Pfister et al. 00]

- Octree hierarchy
- Sample spacing ~ node side length
- Resolution increases with depth
- Rendering: Decent to pixel resolution (on-screen)

Multi-Resolution

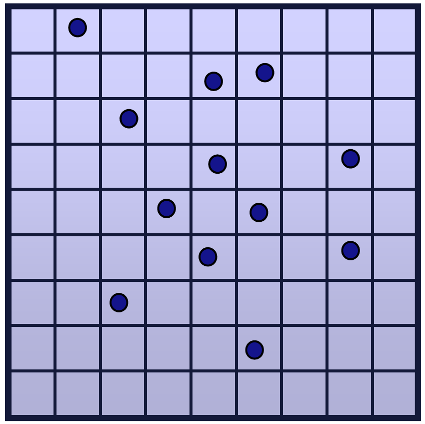


Multi-Resolution Representation (MRR):

- Stored in inner nodes
- Fixed spatial resolution

Quantization:

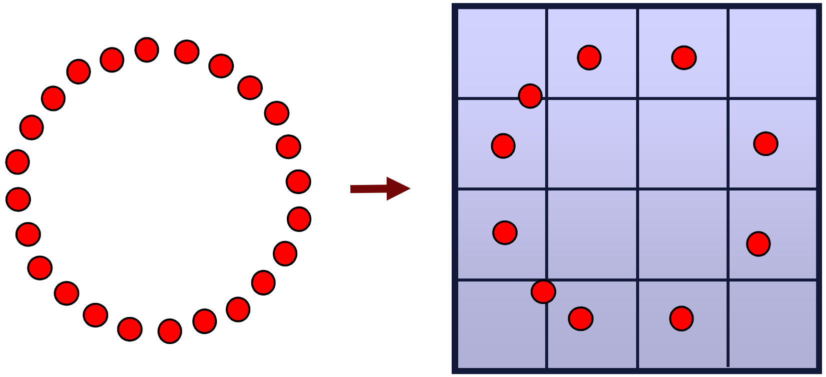
- 3D Grid
- Sparse storage: hash table
- Fully dynamic

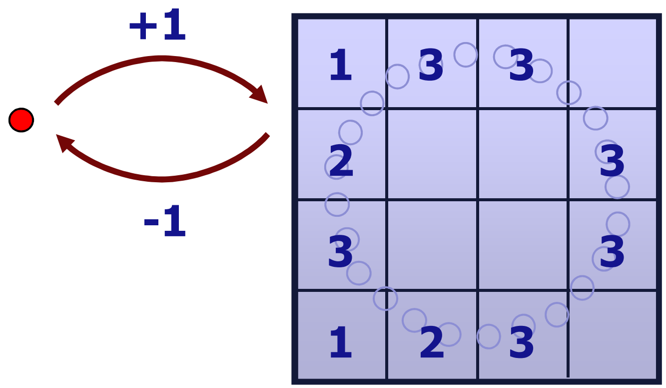


Quantization

Quantization Grid:

- Quantize coordinates on grid
- Store representative point (random, average, centered)
- Dynamic: count points
  - Point counter per cell
  - Add / subtract one
  - Remove representative at zero

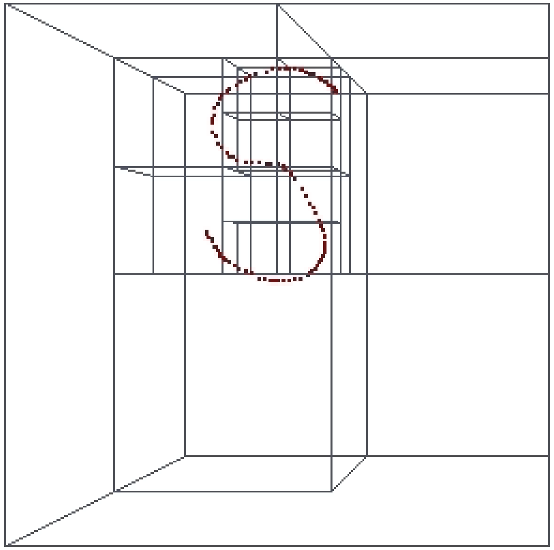




Dynamic Octree

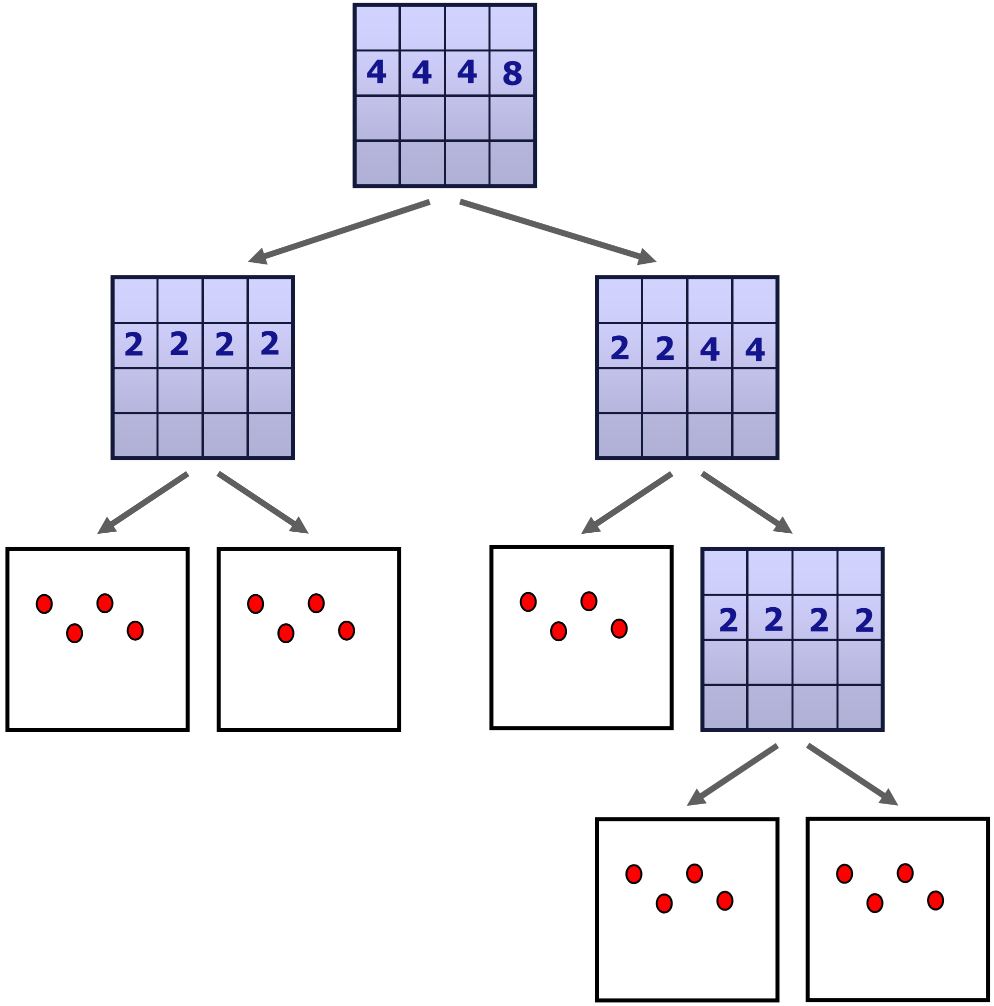
Dynamic Octree: [Samet 90]

- Spatial octree, points stored in leafs
- Maximum n_{max} points per node
- Dynamic operations: *insert* / *delete* points



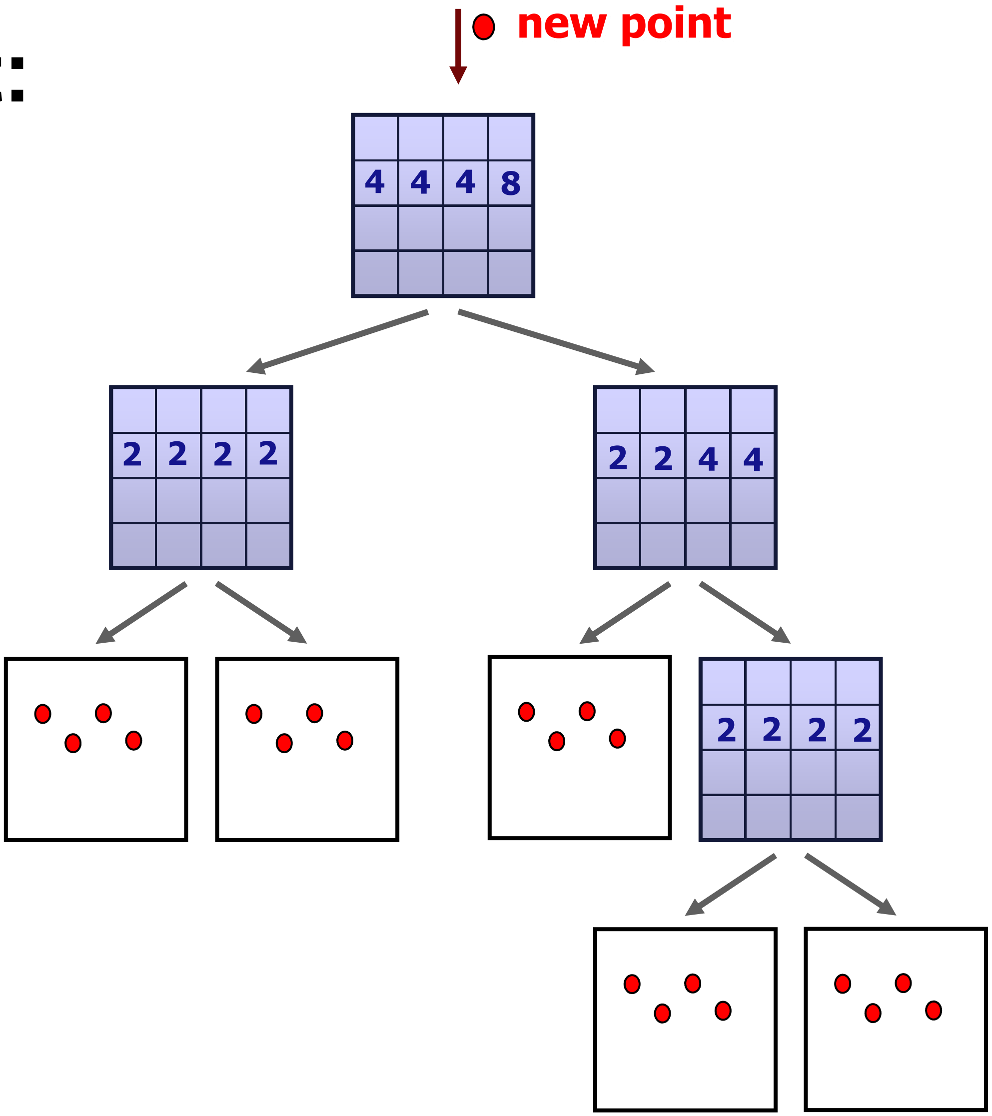
Dynamic MR-Octree

Dynamic MRR:



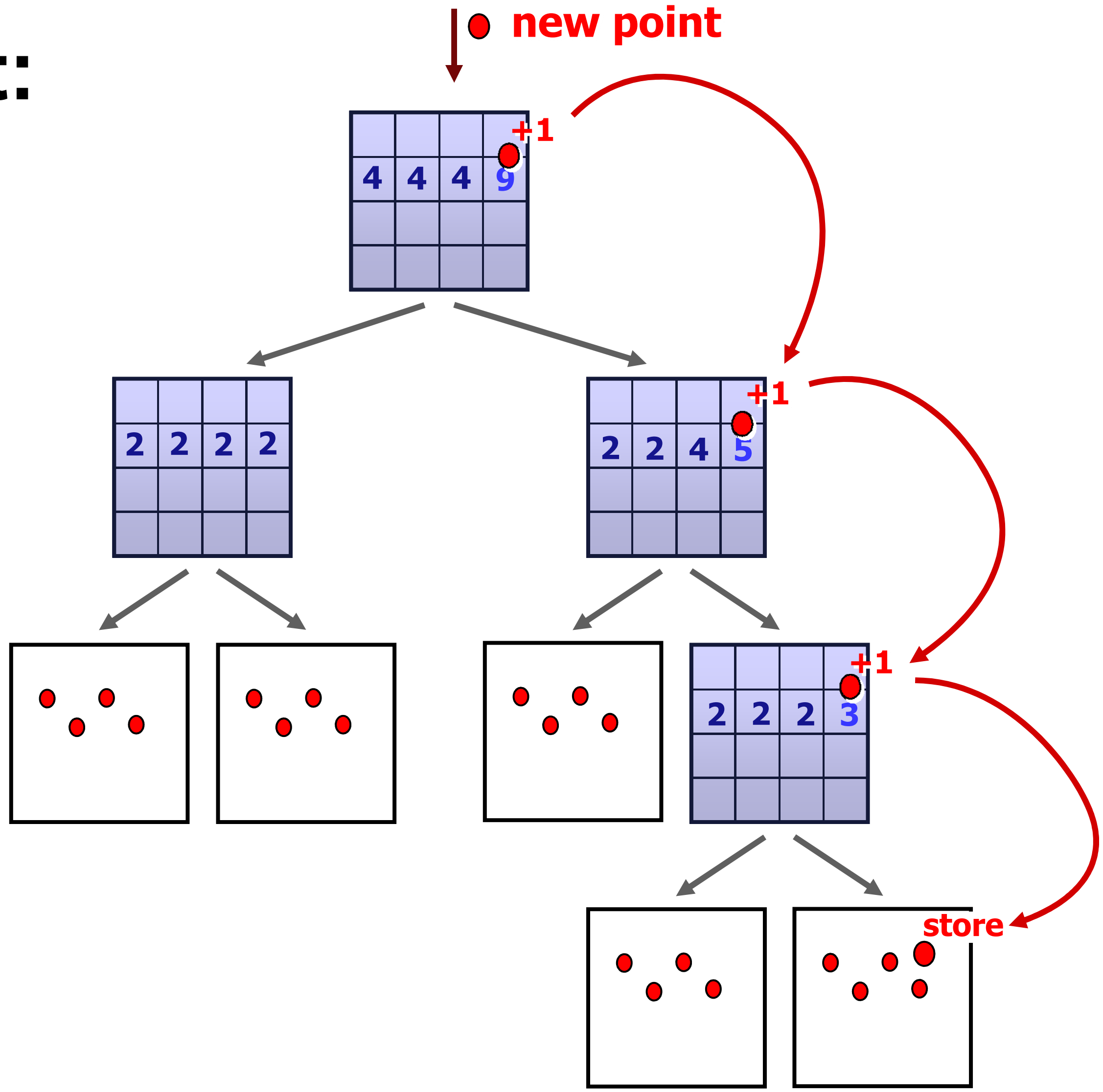
Dynamic MR-Octree

Add Point:



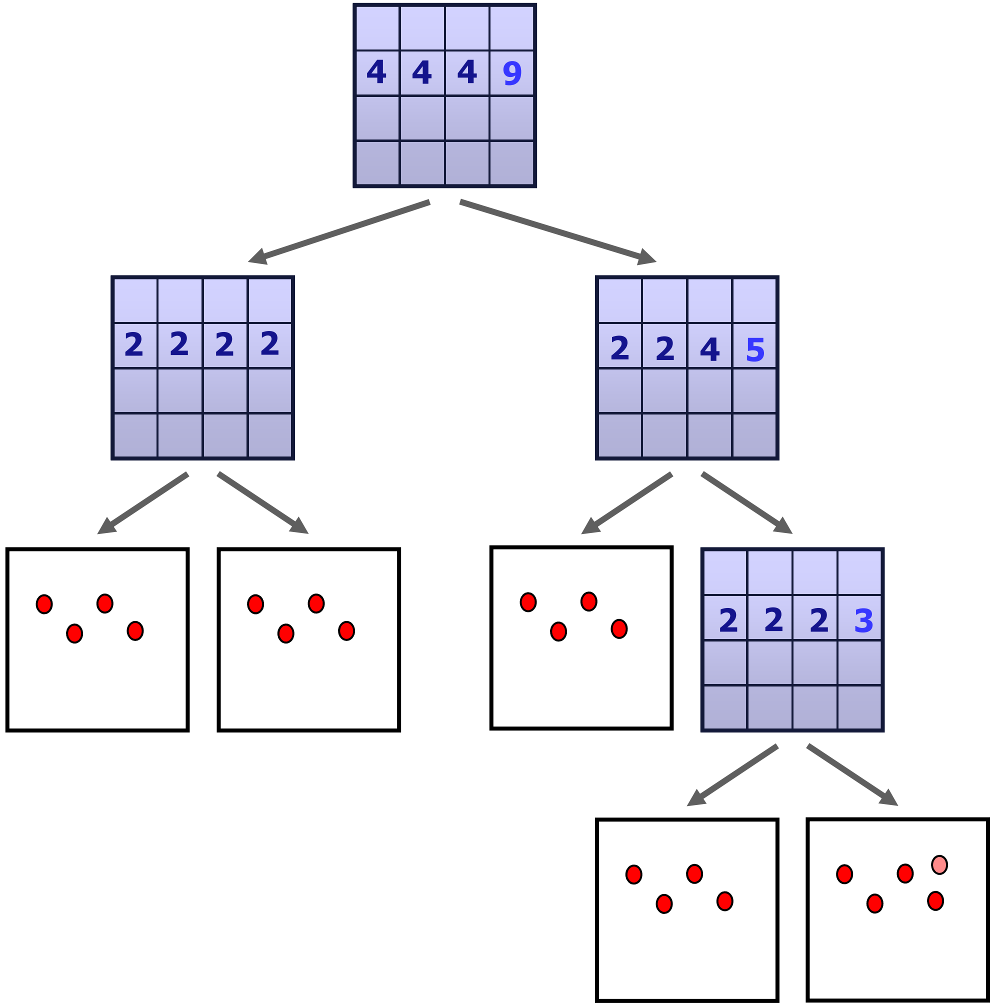
Dynamic MR-Octree

Add Point:



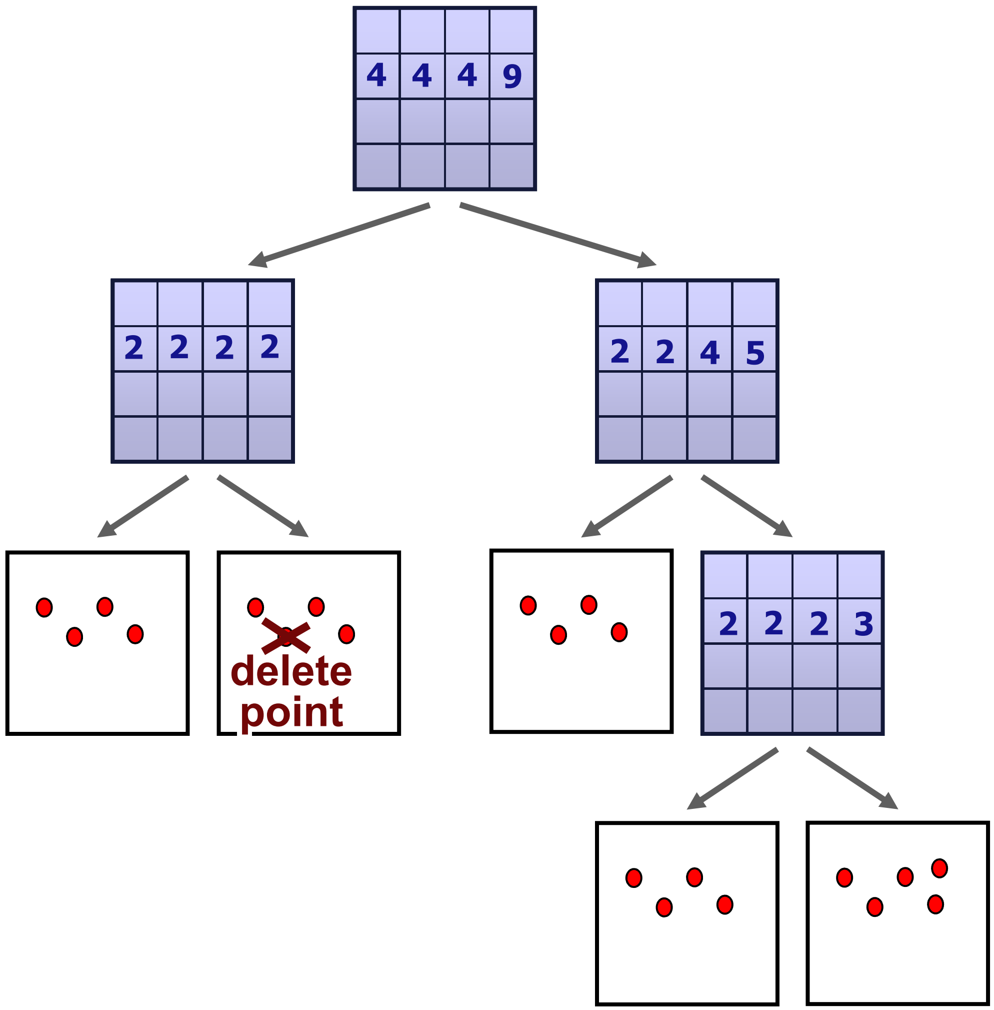
Dynamic MR-Octree

Add Point:



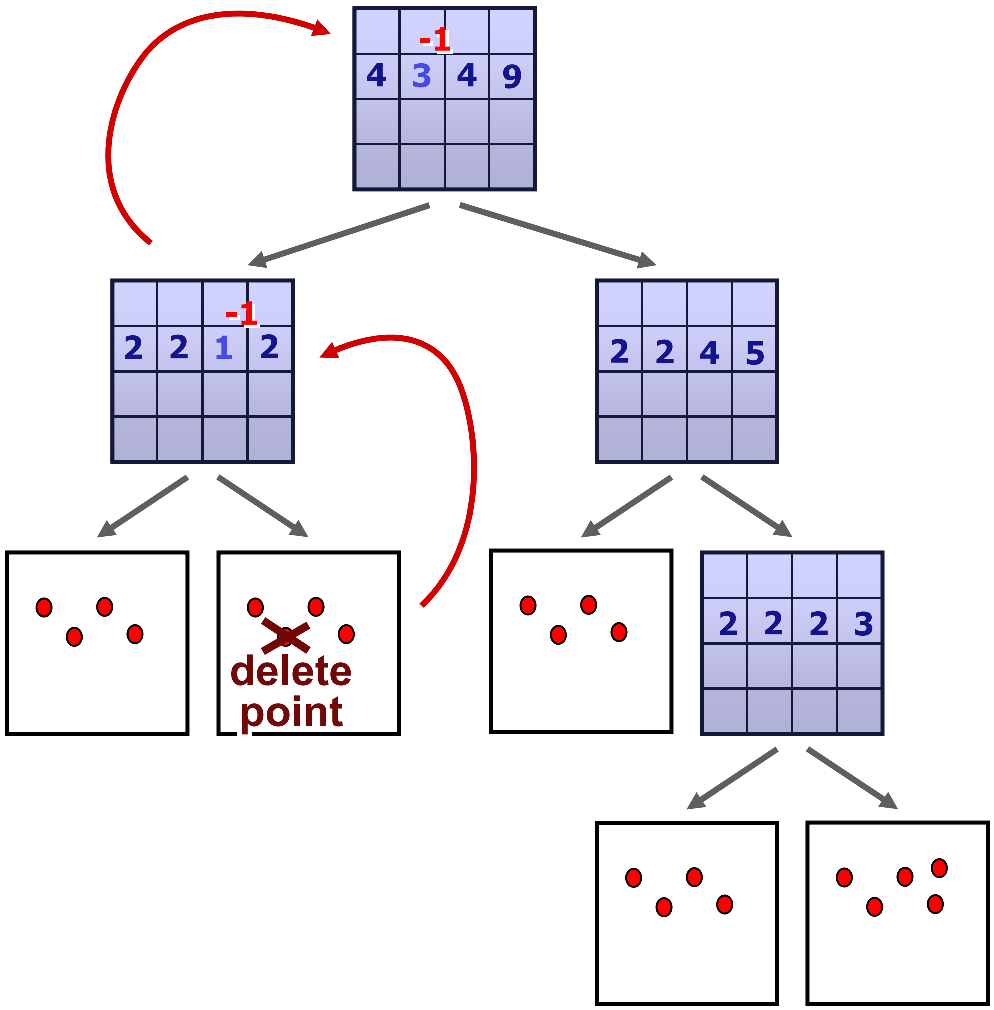
Dynamic MR-Octree

Delete Point:



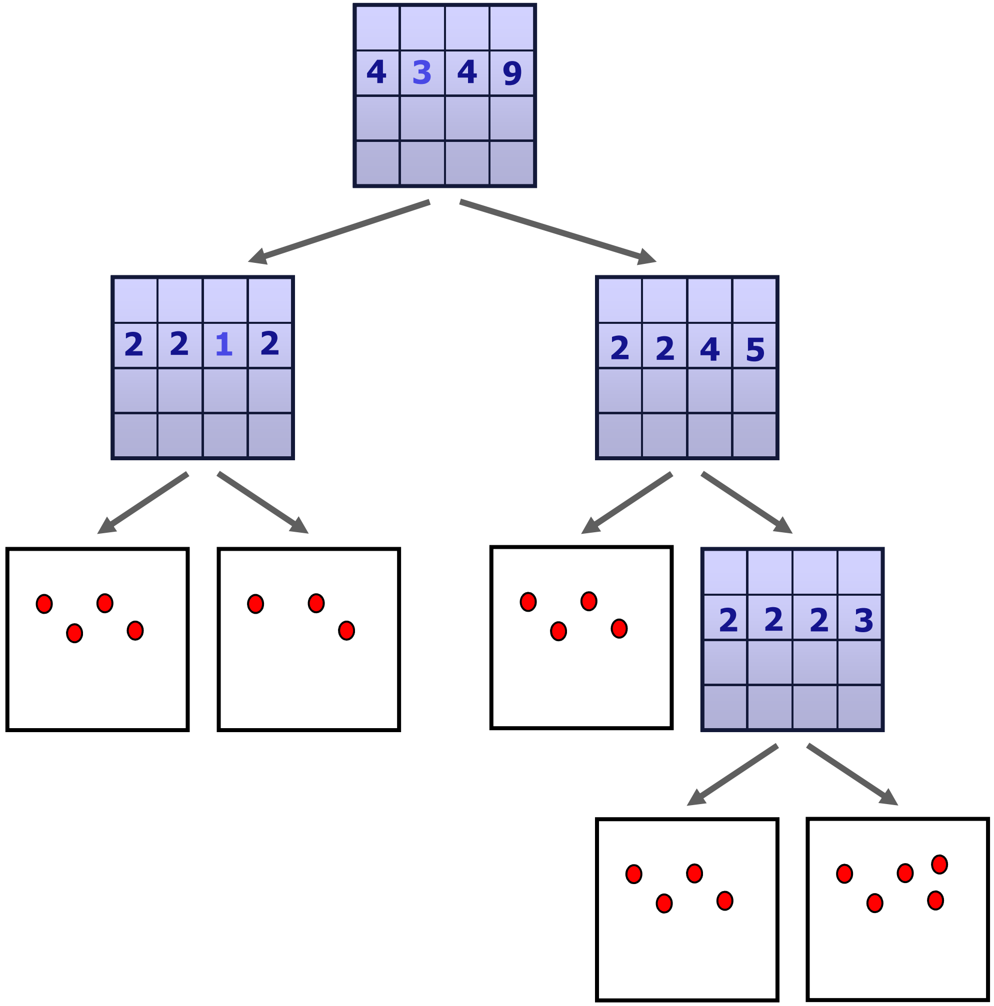
Dynamic MR-Octree

Delete Point:



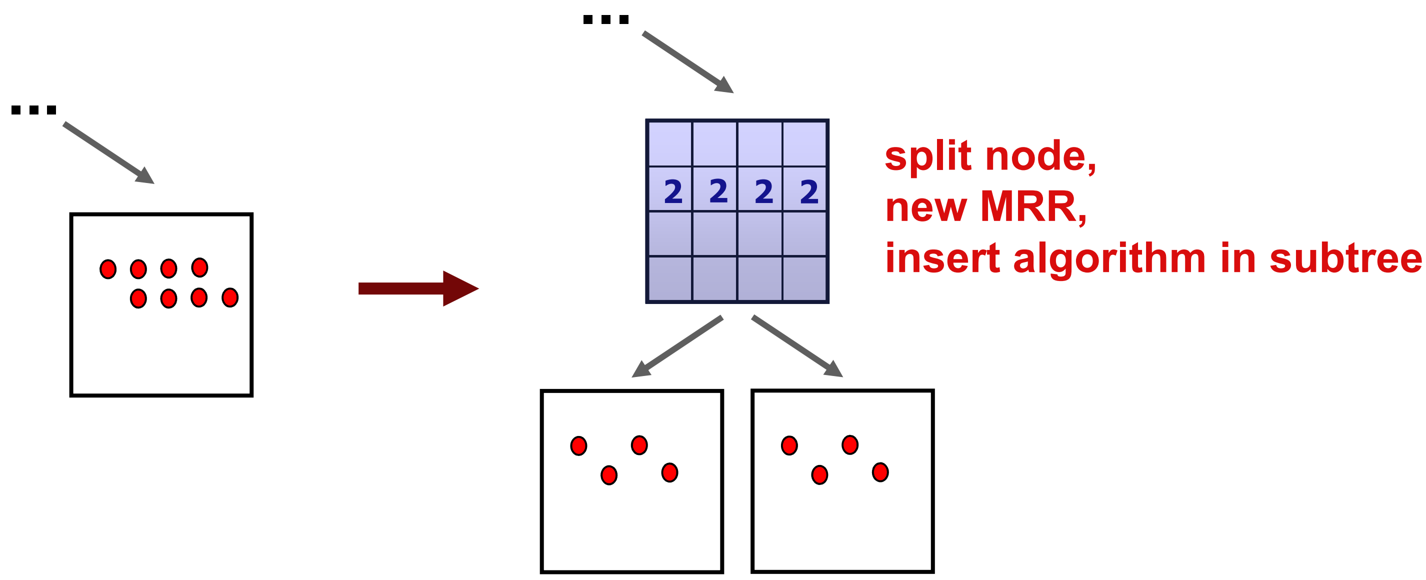
Dynamic MR-Octree

Delete Point:



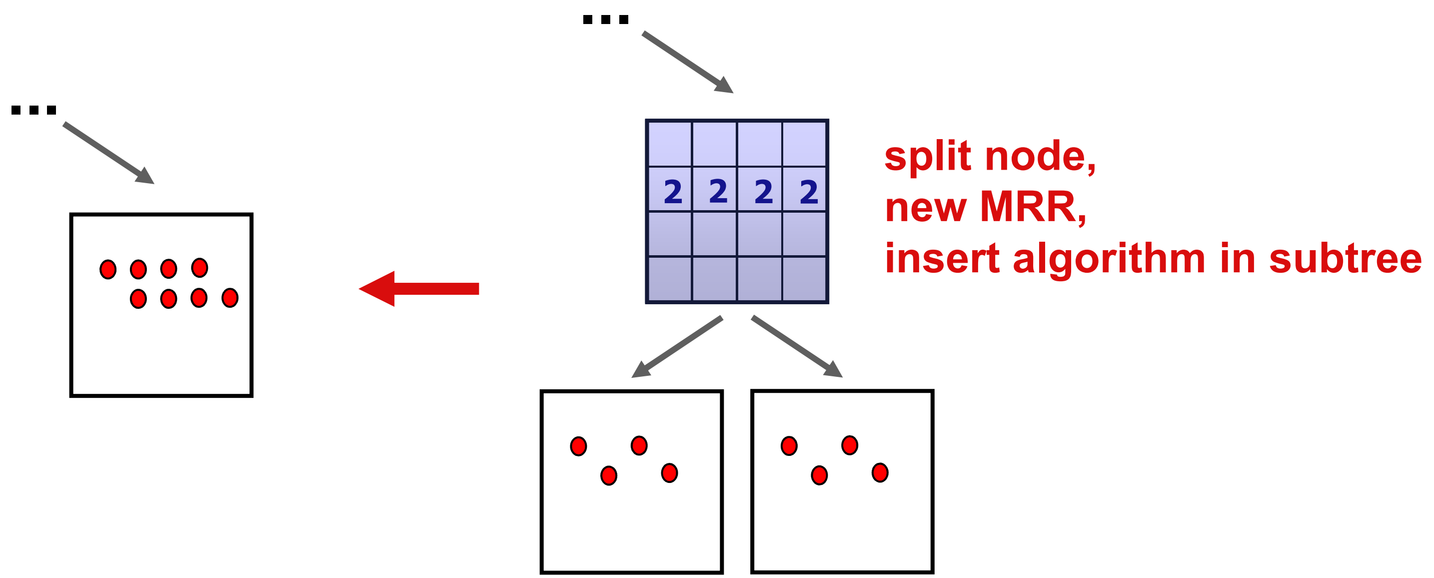
Special Cases

Special cases: Split overfull node



Special Cases

Special cases: Split overfull node

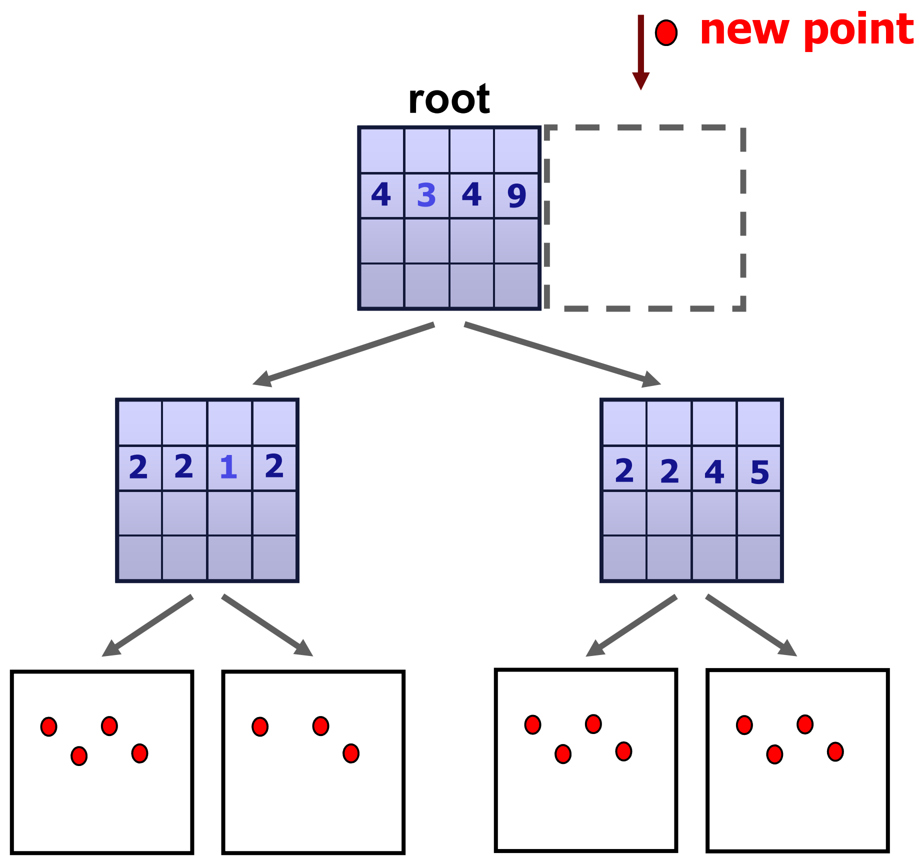


Other case: Combine empty nodes

$\Rightarrow$ no MRR update necessary

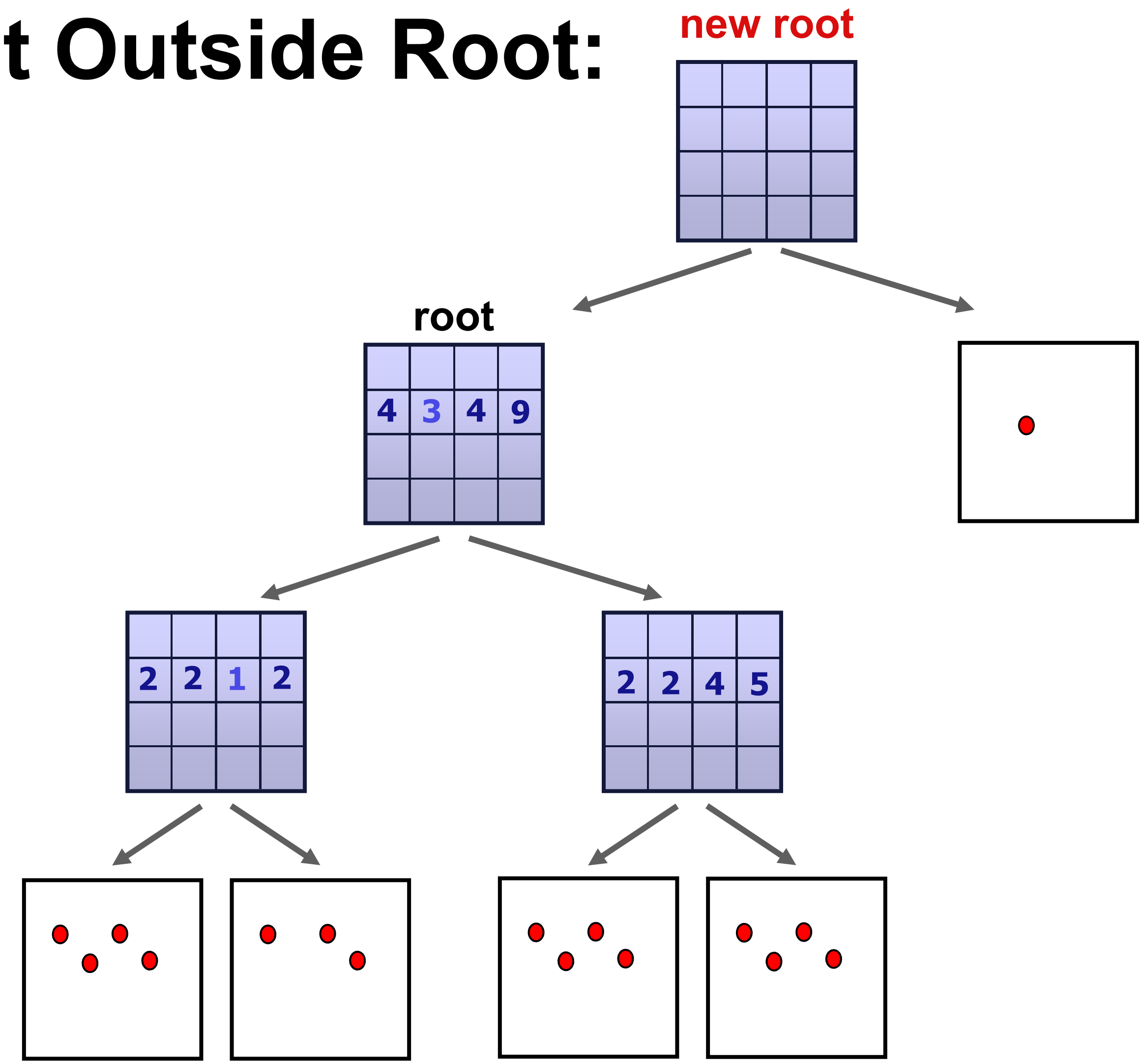
Special Cases

New Point Outside Root:



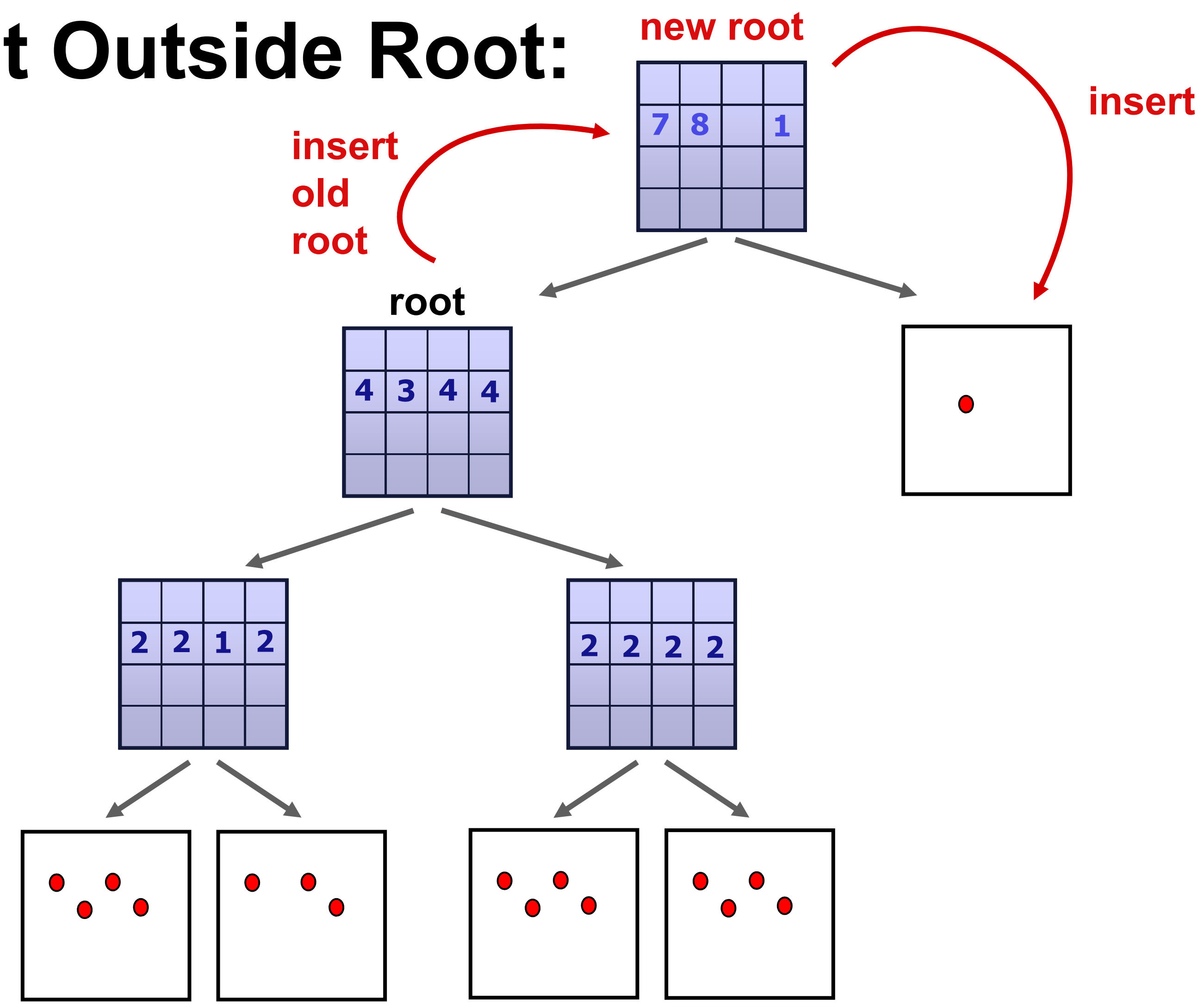
Special Cases

New Point Outside Root:



Special Cases

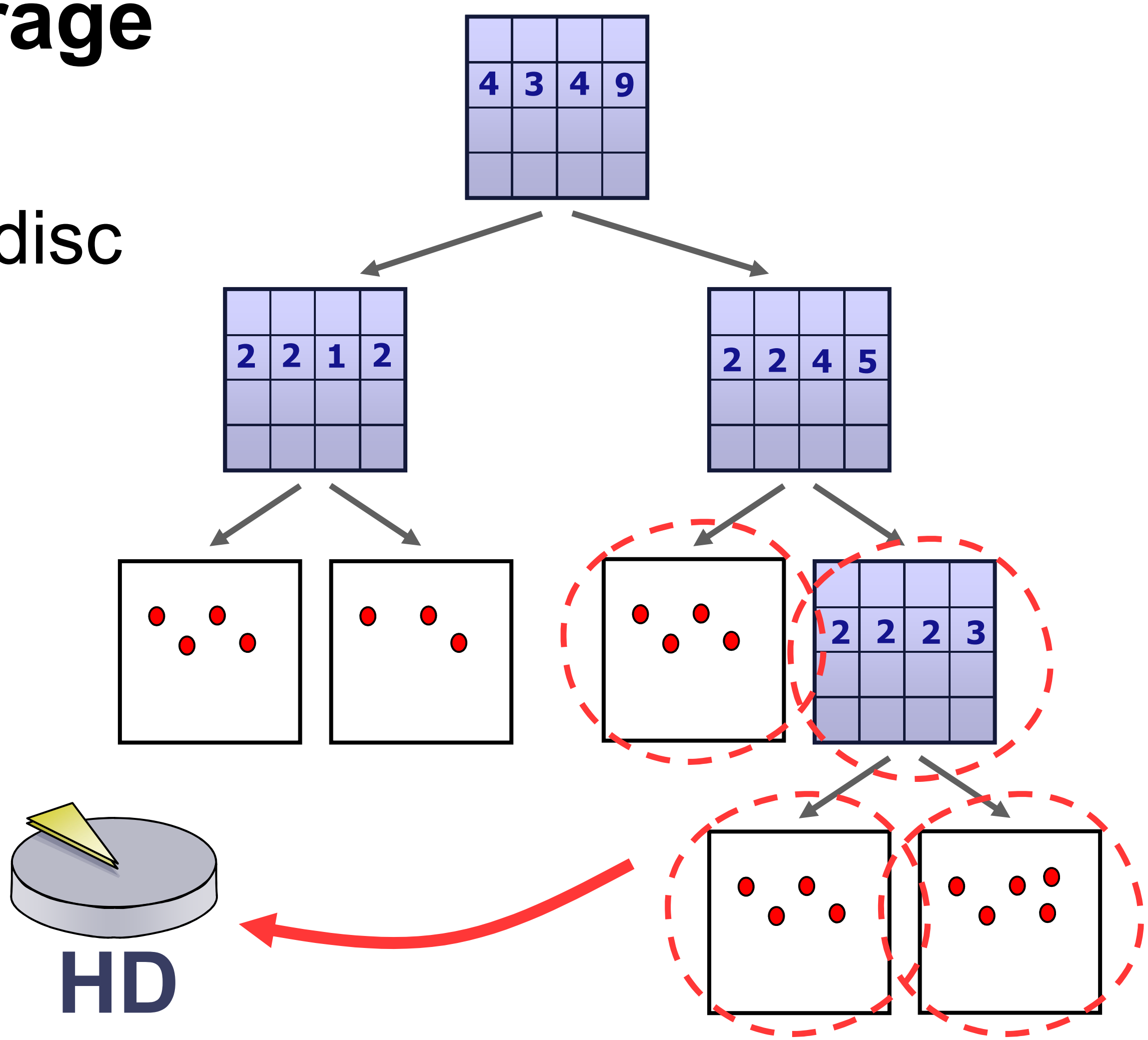
New Point Outside Root:



Out-of-Core

Out-of-Core Storage

- Simple idea:
  swap nodes to disc
- LRU-scheme
  for scheduling



Out-of-Core

Question: How to choose parameters?

- Parameters: n_{max}, grid size
- Trade-Off: *latency* vs. *throughput*
- Standard HD:
  - ~ 50 MB/s throughput
  - ~ 10 ms latency
- ⇒ 50% efficiency at 500 KB block size
- Typically: Choose 1-5 MB blocks
- Same ratio for GPUs (~ 50 GB/s vs. ~ 10 µs)

Nested Hierarchies

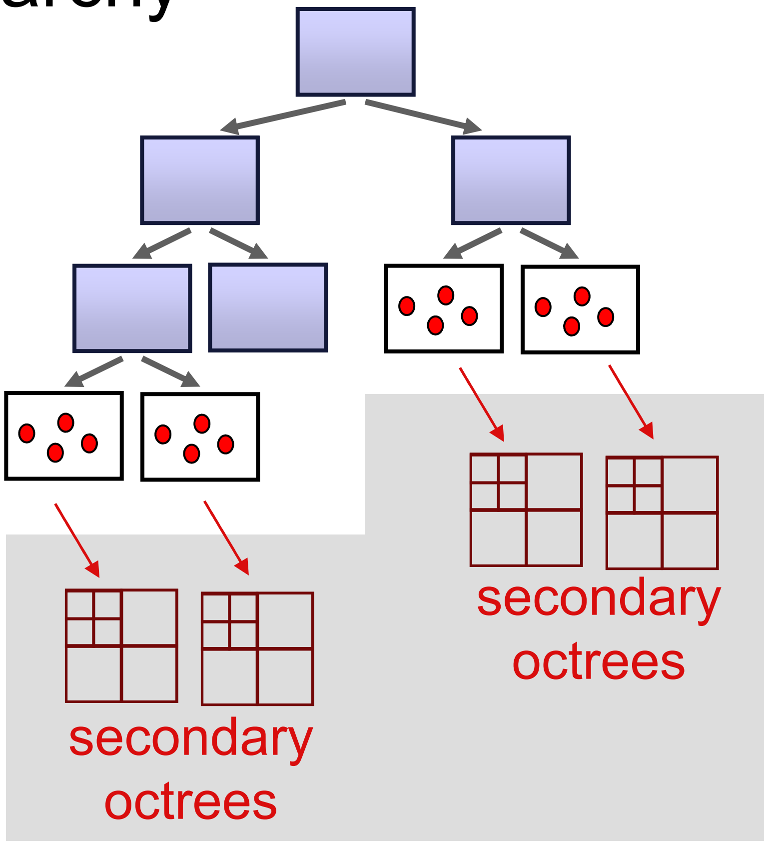
What if the granularity does not match?

- OOC/Rendering:
  Large boxes (typ. 100K points / node)
- Nearest neighbor queries:
  Optimum at 20 points / node
- Other trade-offs: hardware changes
  (disk/graphics, raytracing, etc...)

Nested Hierarchies

Secondary Data Structures

- Attached to nodes of the hierarchy
- Updated, if content changes
- Dynamically or statically rebuild
- Currently: Secondary octrees (n_{max} = 20), rebuild on changes
- Virtually deeper hierarchy



Summary

What Does The New Data Structure Offer?

- Real-time rendering
  (mostly independent of scene size)
- Local editing in O(kh) time, real-time for small k
  [k points changed, h = height of octree]
- Externally efficient global editing if access pattern is spatially coherent (octree blocking)
- Efficient fine-granular range queries (kNN, rays, balls...)

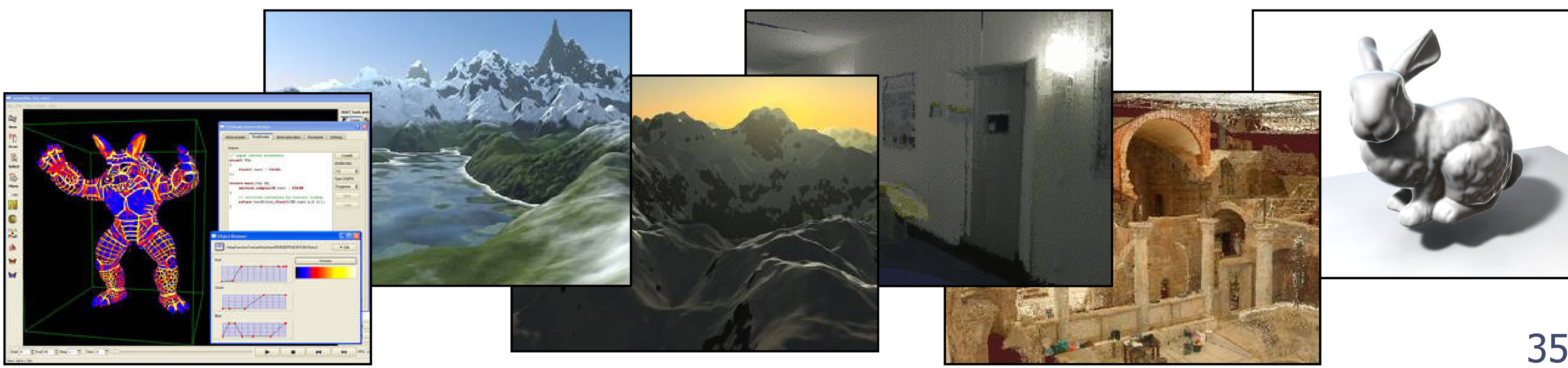
Software Architecture

Software System

Implementation:

- Prototype editor for large point clouds
- Part of a the "XGRT" software system
- Available online as open source (GPL) at:

http://www.gris.uni-tuebingen.de/xgrt



Point Cloud Editor

Editing Point Clouds:

- All changes are mapped to *insert* and *delete* operations
- User input / selection via hierarchical *range queries* (rays, cones, boxes, frustra etc...)
- Selection counters: Each octree node stores the count of selected points (for *selection queries*)

Handling Large Changes

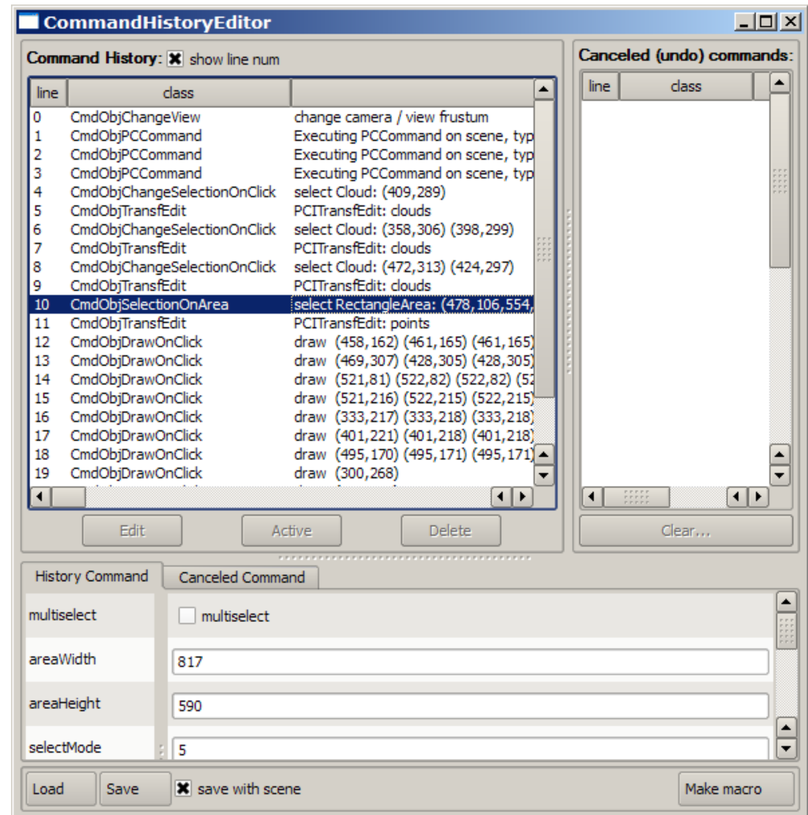
Open Problem: How to handle *large changes*

- Dynamic operations are O(kh)
  [k points changed, h = height of octree]
- Small, local changes in real-time
- Large changes: no interactive response
- $\Rightarrow$ *Command object architecture*

Command Scripting

Command Object Architecture

- Subset of [Meyers 98]
- All editing commands are recorded as *command objects*
- Can be *replayed* with *modified parameters*
- *Reflective software* architecture to reduce implementation complexity



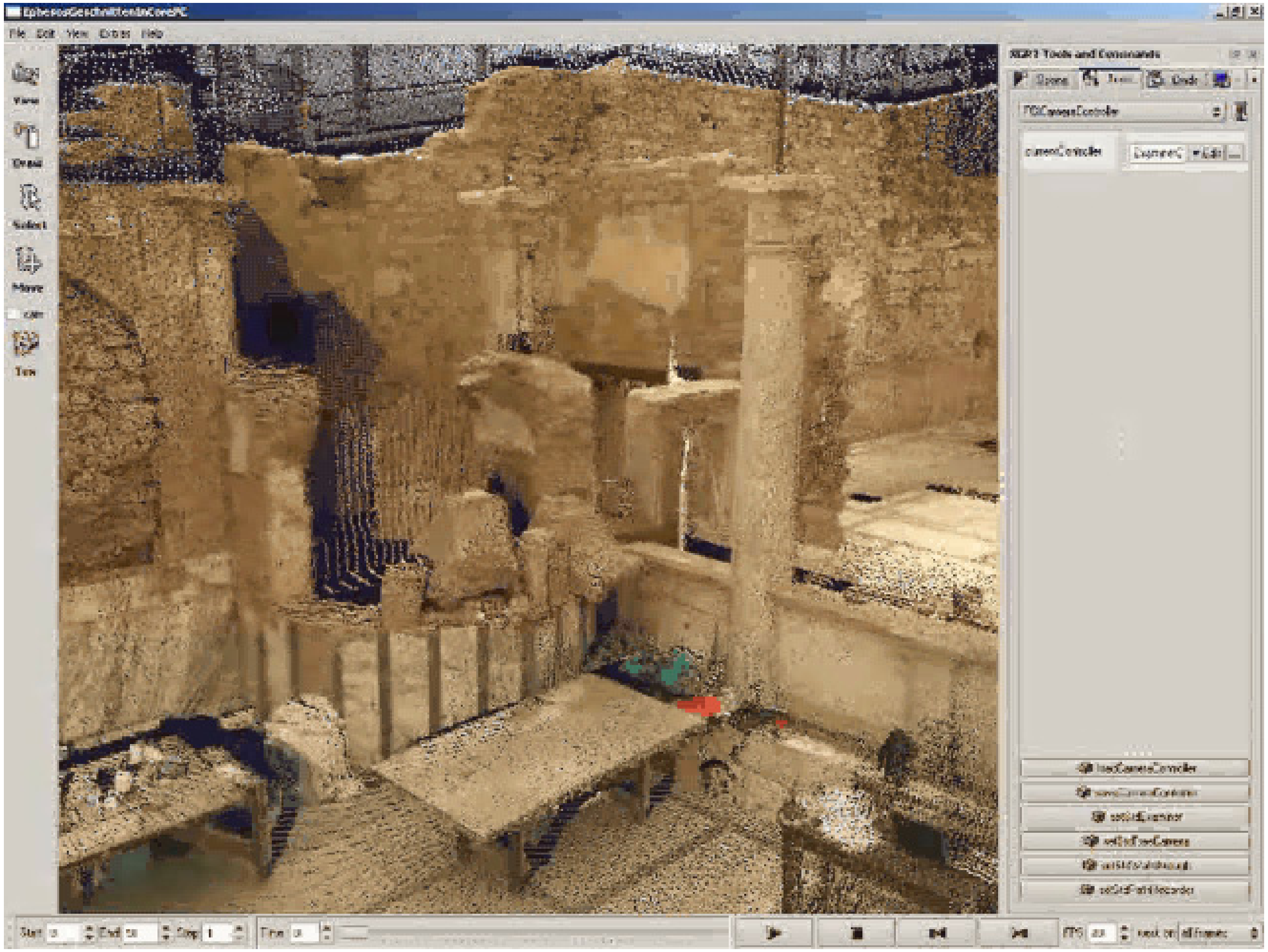
Command Scripting

Handling Large Changes:

- *Resample* input model to a fraction of its original size (say 1:100)
- Simple random streaming simplification
- Perform editing operations
- *Reexecute* command script with resampling command disabled
- All editing commands are *externally efficient*
- Full scripting language for more flexibility

Results

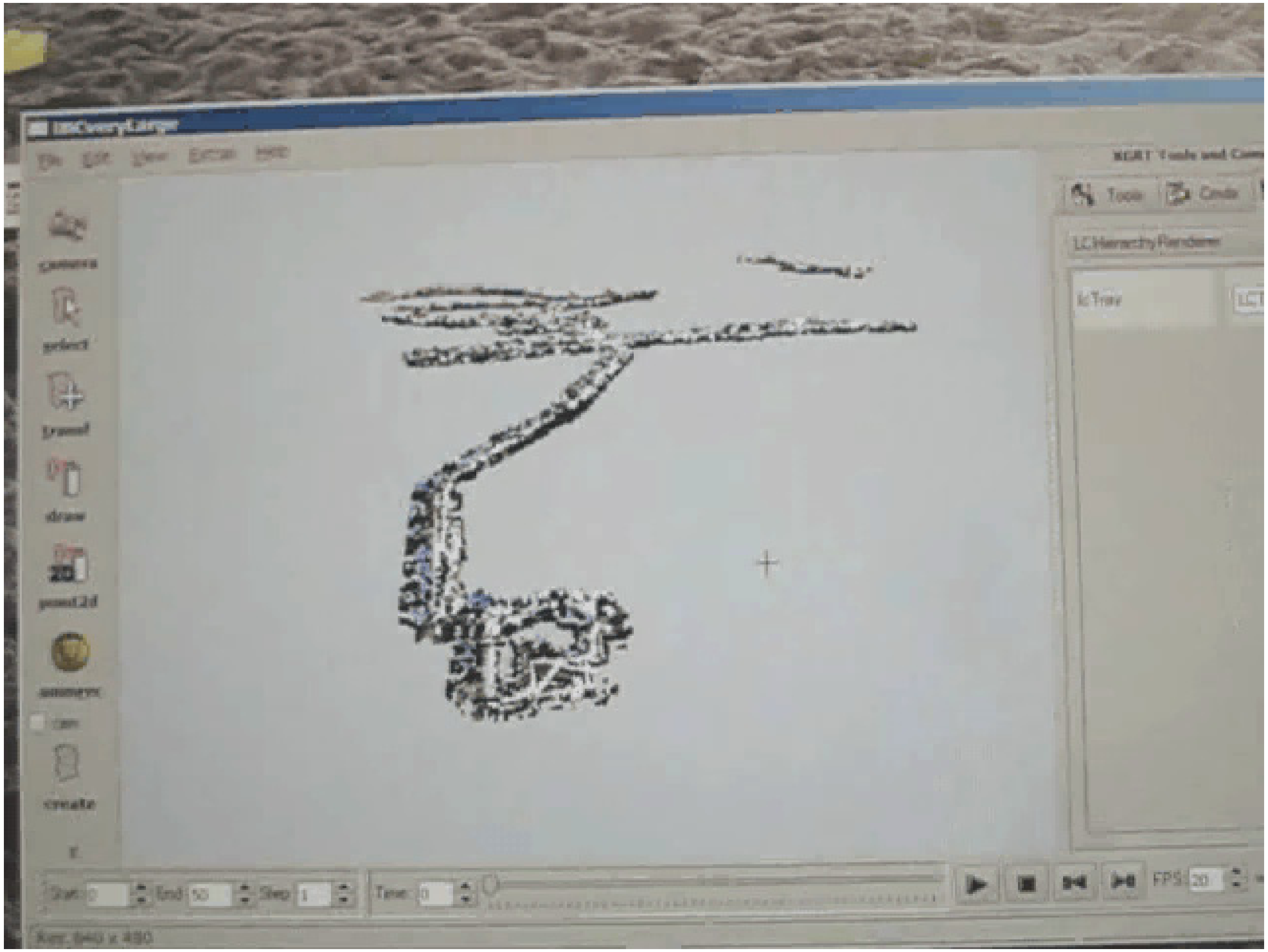
In-Core Example



Data set: Ephesos (14M pts / ~1GB), Courtesy of M. Wimmer, TU Vienna

[Core2 2.13Ghz, ATI X1300]

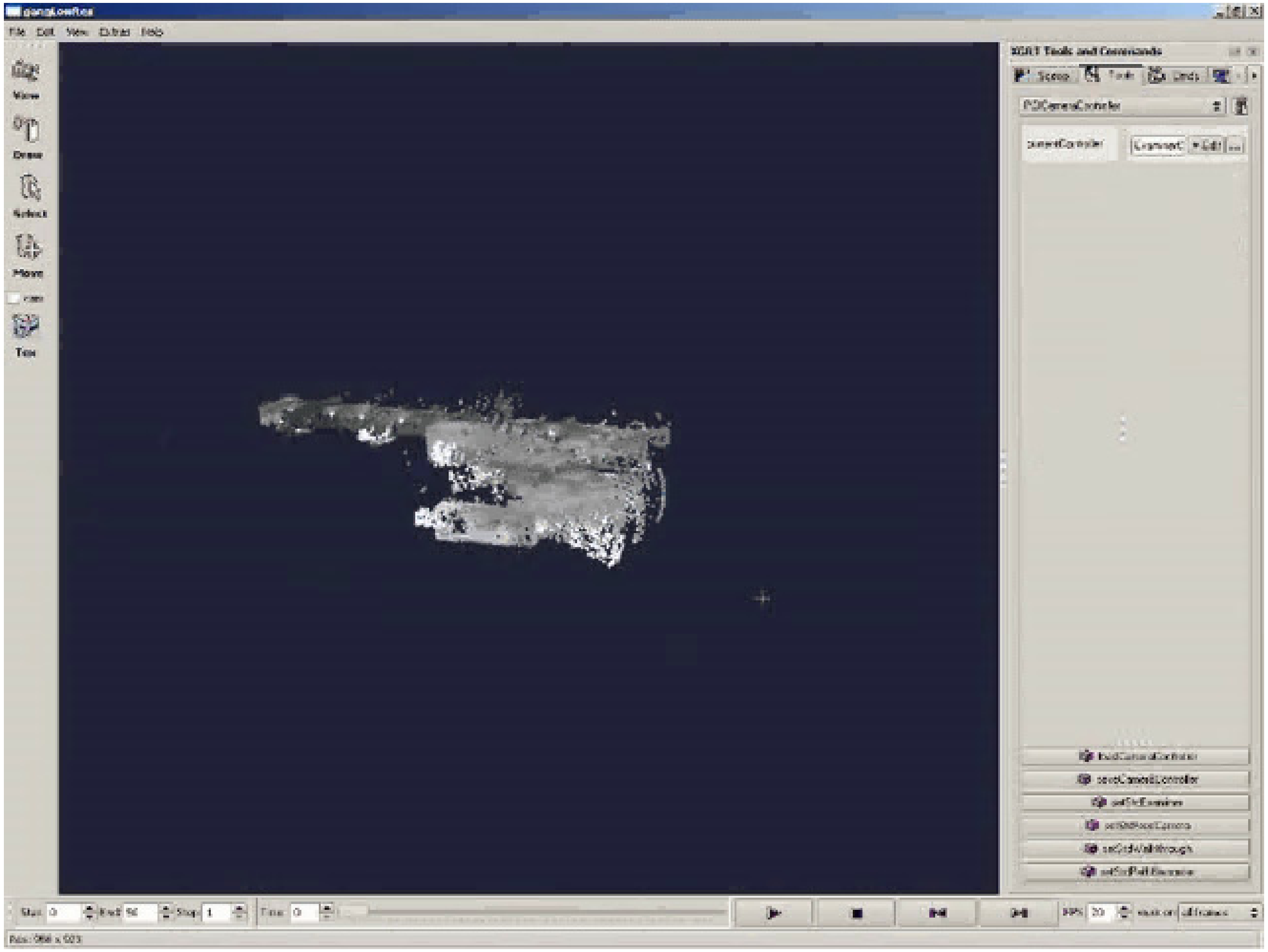
Out-of-Core Example



Data set: Outdoor Scan ($2.2 \cdot 10^9$ pts / 63.5 GB), J.-M. Frahm, UNC

[Pentium-4 3.4Ghz, QuadroFX 3450, 500GB/7200rpm SATA HD]

Command Scripting



Data set: Building Scan (76M pts/6.5 GB), P. Biber / S. Fleck, Univ. of Tübingen

[Core2 2.13Ghz, ATI X1300, 250GB/7200rpm SATA HD]

Conclusions & Future Work

Conclusions

Interactive Editor for Large Point Clouds

- Dynamic out-of-core multi-resolution data structure
- Efficient visualization & local editing
- Command scripting for global changes

Future Work:

- Better filtering (fractional weights)
- Handling triangle meshes
- Editing of complex models mostly unexplored

Acknowledgements

Thanks to: Peter Biber, Matthias Fisher, Jan-Michael Frahm, Sven Fleck, David Gallup, Marc Pollefeys, Wolfgang Straßer, Michael Wimmer

Funding: DFG grant “Perceptual Graphics”, BW-FIT grant “Gigapixel Displays”, the State of Baden Württemberg, Max Planck Center for Visual Computing and Communication.

Download:
http://www.gris.uni-tuebingen.de/xgrt