Data Structures for Dynamic and Kinetic Multidimensional Point Sets

David Mount

University of Maryland, College Park

MURI Kickoff Meeting – 2008

Motivation

Latent-Space Embedding for Large Networks

Performed through iterative search. Large networks involve:

- solution spaces of very high dimension
- a large number of iterations to achieve convergence
- inner-loop computations of quadratic complexity

Recent results in spatial data structures

- Simple and practical dynamic structures
- Intrinsic data structures for kinetic updates
- Fast approximations through hierarchical sketching
- Space/query-time tradeoffs

Motivation

Latent-Space Embedding for Large Networks

Performed through iterative search. Large networks involve:

- solution spaces of very high dimension
- a large number of iterations to achieve convergence
- inner-loop computations of quadratic complexity

Recent results in spatial data structures

- Simple and practical dynamic structures
- Intrinsic data structures for kinetic updates
- Fast approximations through hierarchical sketching
- Space/query-time tradeoffs

Motivation

Latent-Space Embedding for Large Networks

Performed through iterative search. Large networks involve:

- solution spaces of very high dimension
- a large number of iterations to achieve convergence
- inner-loop computations of quadratic complexity

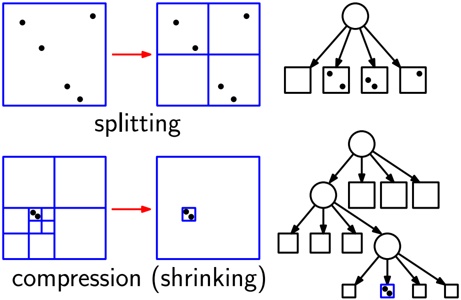
Recent results in spatial data structures

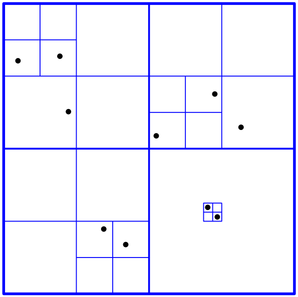
- Simple and practical dynamic structures ←
- Intrinsic data structures for kinetic updates ←
- Fast approximations through hierarchical sketching
- Space/query-time tradeoffs

Basic Structures: Compressed Quadtree

Quadtree: Hierarchical structure based on subdivision of squares.

Path Compression: Chains of trivial splits are compressed.

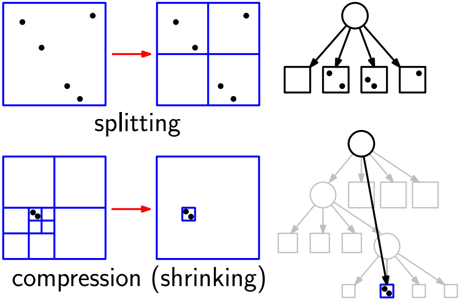


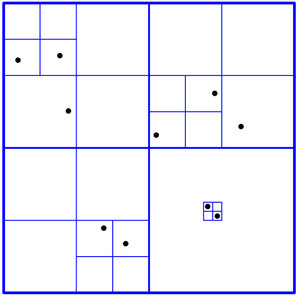


Basic Structures: Compressed Quadtree

Quadtree: Hierarchical structure based on subdivision of squares.

Path Compression: Chains of trivial splits are compressed.





Basic Structures: Balanced Partition Trees

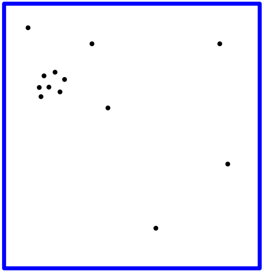
Problem: Compressed quadtrees may have depth $\Omega(n)$.

Balanced Structures

Achieving $O(\log n)$ depth:

BBD-tree: [AMN98] Combines splitting with centroid shrinking. Produces an inner cell and outer cell.

BAR-tree: [DGK01] Uses slanted splitting planes to split clusters. (Produces convex cells.)



Basic Structures: Balanced Partition Trees

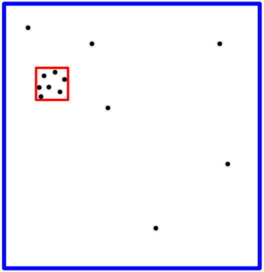
Problem: Compressed quadtrees may have depth $\Omega(n)$.

Balanced Structures

Achieving $O(\log n)$ depth:

BBD-tree: [AMN98] Combines splitting with centroid shrinking. Produces an inner cell and outer cell.

BAR-tree: [DGK01] Uses slanted splitting planes to split clusters. (Produces convex cells.)



Basic Structures: Balanced Partition Trees

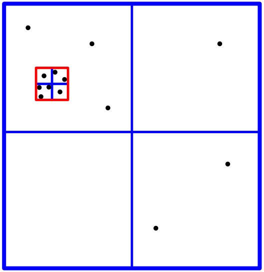
Problem: Compressed quadtrees may have depth $\Omega(n)$.

Balanced Structures

Achieving $O(\log n)$ depth:

BBD-tree: [AMN98] Combines splitting with centroid shrinking.
Produces an inner cell and outer cell.

BAR-tree: [DGK01] Uses slanted splitting planes to split clusters. (Produces convex cells.)



Basic Structures: Balanced Partition Trees

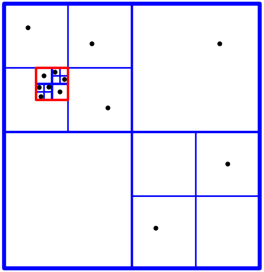
Problem: Compressed quadtrees may have depth $\Omega(n)$.

Balanced Structures

Achieving $O(\log n)$ depth:

BBD-tree: [AMN98] Combines splitting with centroid shrinking. Produces an inner cell and outer cell.

BAR-tree: [DGK01] Uses slanted splitting planes to split clusters. (Produces convex cells.)



Basic Structures: Balanced Partition Trees

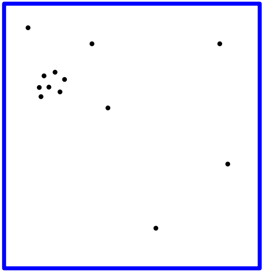
Problem: Compressed quadtrees may have depth $\Omega(n)$.

Balanced Structures

Achieving $O(\log n)$ depth:

BBD-tree: [AMN98] Combines splitting with centroid shrinking.
Produces an inner cell and outer cell.

BAR-tree: [DGK01]
Uses slanted splitting planes to split clusters. (Produces convex cells.)



Basic Structures: Balanced Partition Trees

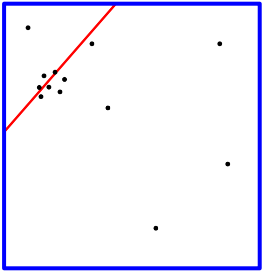
Problem: Compressed quadtrees may have depth $\Omega(n)$.

Balanced Structures

Achieving $O(\log n)$ depth:

BBD-tree: [AMN98] Combines splitting with centroid shrinking. Produces an inner cell and outer cell.

BAR-tree: [DGK01] Uses slanted splitting planes to split clusters. (Produces convex cells.)



Approximate Spherical Range Searching

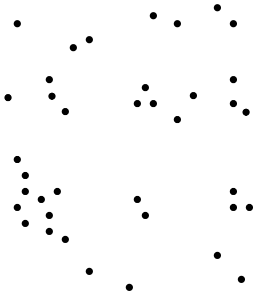
Example: Count (or report) the points lying within a given spherical range. Allow an error of ε.

Preprocessing: Build BBD tree.

Query Processing:

- Find maximal cells lying within the outer range and covering the inner range
- Access counts for each cell
- Return the total

Query time: $O(\log n + (1/\varepsilon)^{d-1})$.



Approximate Spherical Range Searching

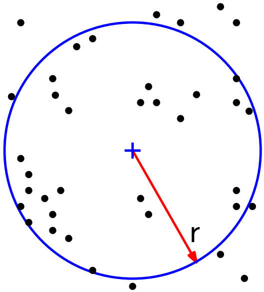
Example: Count (or report) the points lying within a given spherical range. Allow an error of ε.

Preprocessing: Build BBD tree.

Query Processing:

- Find maximal cells lying within the outer range and covering the inner range
- Access counts for each cell
- Return the total

Query time: $O(\log n + (1/\varepsilon)^{d-1})$.



Approximate Spherical Range Searching

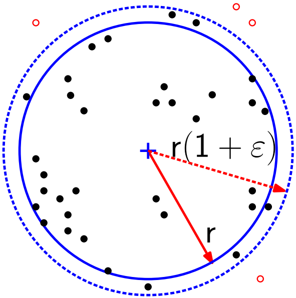
Example: Count (or report) the points lying within a given spherical range. Allow an error of ε.

Preprocessing: Build BBD tree.

Query Processing:

- Find maximal cells lying within the outer range and covering the inner range
- Access counts for each cell
- Return the total

Query time: $O(\log n + (1/\varepsilon)^{d-1})$.



Approximate Spherical Range Searching

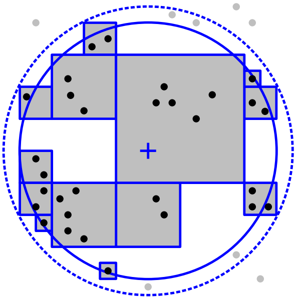
Example: Count (or report) the points lying within a given spherical range. Allow an error of ε.

Preprocessing: Build BBD tree.

Query Processing:

- Find maximal cells lying within the outer range and covering the inner range
- Access counts for each cell
- Return the total

Query time: $O(\log n + (1/\varepsilon)^{d-1})$.



Approximate Spherical Range Searching

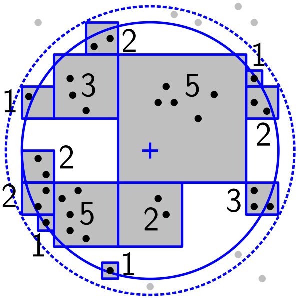
Example: Count (or report) the points lying within a given spherical range. Allow an error of ε.

Preprocessing: Build BBD tree.

Query Processing:

- Find maximal cells lying within the outer range and covering the inner range
- Access counts for each cell
- Return the total

Query time: $O(\log n + (1/\varepsilon)^{d-1})$.



Approximate Spherical Range Searching

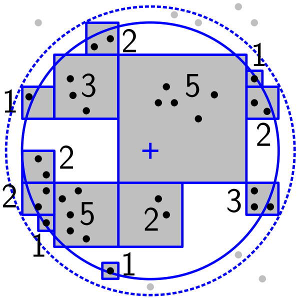
Example: Count (or report) the points lying within a given spherical range. Allow an error of ε.

Preprocessing: Build BBD tree.

Query Processing:

- Find maximal cells lying within the outer range and covering the inner range
- Access counts for each cell
- Return the total

Query time: $O(\log n + (1/\varepsilon)^{d-1})$.



Approximate Spherical Range Searching

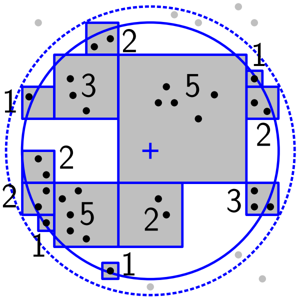
Example: Count (or report) the points lying within a given spherical range. Allow an error of ε.

Preprocessing: Build BBD tree.

Query Processing:

- Find maximal cells lying within the outer range and covering the inner range
- Access counts for each cell
- Return the total

Query time: $O(\log n + (1/\varepsilon)^{d-1})$.



Dynamic (Balanced) Structures

How to support insertion and deletion?

Analogy — Skiplist [Pug90]

A randomized structure for 1-dimensional data. Build linked lists for successive random samples.

- $S_0 \leftarrow S$ (the original point set).
- $S_1 \leftarrow$ sample S_0 with probability $\frac{1}{2}$.
- $S_2 \leftarrow$ sample S_1 with probability $\frac{1}{2}$.
- The process ends after $O(\log n)$ stages in expectation.

Dynamic (Balanced) Structures

How to support insertion and deletion?

Analogy — Skiplist [Pug90]

A randomized structure for 1-dimensional data. Build linked lists for successive random samples.

- $S_0 \leftarrow S$ (the original point set).
- $S_1 \leftarrow$ sample S_0 with probability $\frac{1}{2}$.
- $S_2 \leftarrow$ sample S_1 with probability $\frac{1}{2}$.
- The process ends after $O(\log n)$ stages in expectation.

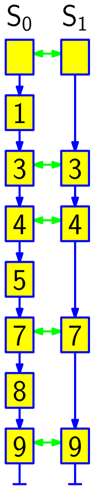
Dynamic (Balanced) Structures

How to support insertion and deletion?

Analogy — Skiplist [Pug90]

A randomized structure for 1-dimensional data. Build linked lists for successive random samples.

- $S_0 \leftarrow S$ (the original point set).
- $S_1 \leftarrow$ sample S_0 with probability $\frac{1}{2}$.
- $S_2 \leftarrow$ sample S_1 with probability $\frac{1}{2}$.
- The process ends after $O(\log n)$ stages in expectation.



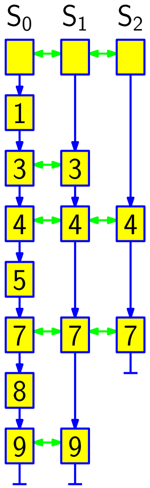
Dynamic (Balanced) Structures

How to support insertion and deletion?

Analogy — Skiplist [Pug90]

A randomized structure for 1-dimensional data. Build linked lists for successive random samples.

- $S_0 \leftarrow S$ (the original point set).
- $S_1 \leftarrow$ sample S_0 with probability $\frac{1}{2}$.
- $S_2 \leftarrow$ sample S_1 with probability $\frac{1}{2}$.
- The process ends after $O(\log n)$ stages in expectation.



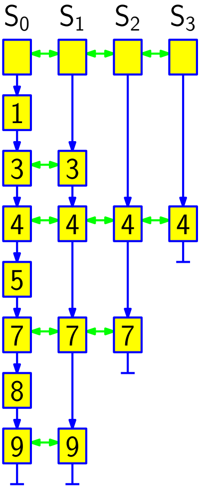
Dynamic (Balanced) Structures

How to support insertion and deletion?

Analogy — Skiplist [Pug90]

A randomized structure for 1-dimensional data. Build linked lists for successive random samples.

- $S_0 \leftarrow S$ (the original point set).
- $S_1 \leftarrow$ sample S_0 with probability $\frac{1}{2}$.
- $S_2 \leftarrow$ sample S_1 with probability $\frac{1}{2}$.
- The process ends after $O(\log n)$ stages in expectation.



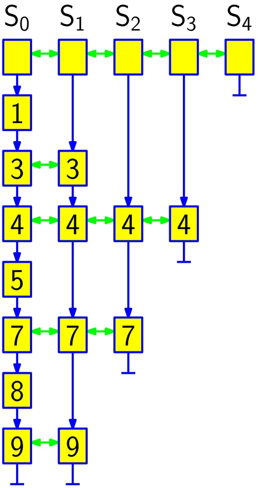
Dynamic (Balanced) Structures

How to support insertion and deletion?

Analogy — Skiplist [Pug90]

A randomized structure for 1-dimensional data. Build linked lists for successive random samples.

- $S_0 \leftarrow S$ (the original point set).
- $S_1 \leftarrow$ sample S_0 with probability $\frac{1}{2}$.
- $S_2 \leftarrow$ sample S_1 with probability $\frac{1}{2}$.
- The process ends after $O(\log n)$ stages in expectation.



Skip Quadtree

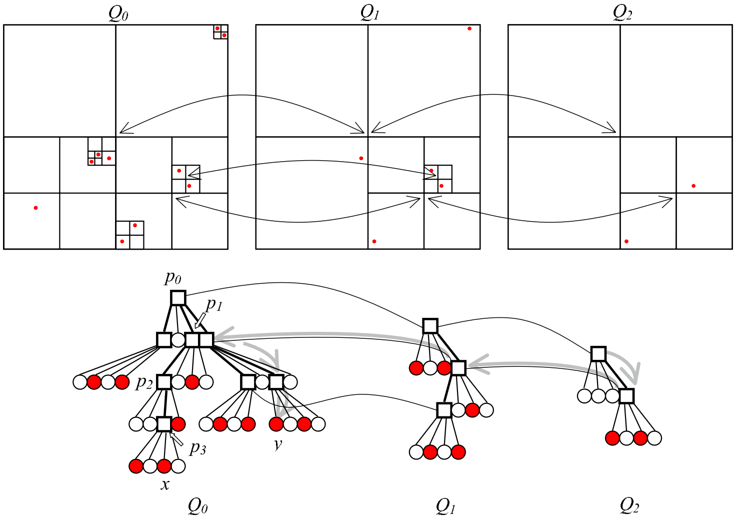
Skip Quadtree [EGS05]

Same idea, but applied to a sequence of compressed quadtrees.

- $Q_0 \leftarrow$ quadtree for S_0.
- $Q_1 \leftarrow$ quadtree for S_1.
- $Q_2 \leftarrow$ quadtree for S_2.
- ... Each node of Q_i is linked to its counterpart in Q_{i-1}.

Although each quadtree may be unbalanced (like a linked list) it is possible to access each node in $O(\log n)$ time through the links.

Skip Quadtree



Kinetic Structures

Latent embedding algorithms involve adjusting the location many points with each iteration. The motion of each point may be small.

Kinetic Updates

Update a data structure after a small motion involving many points of the set.

- Data structures that are defined in terms of a fixed coordinate frame are sensitive to changes in absolute position of points, even if the relative position remains unchanged.
- Can we define spatial data structures that are independent of any coordinate frame?

r-Net

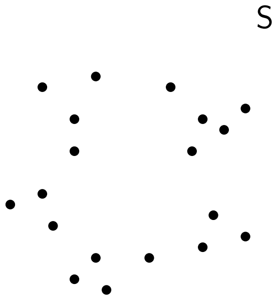
r-Net

. . . is a subset $X \subseteq S$ such that

(i) every point of S is within distance r of some point of X

(ii) the pairwise distance between any two points of X is $\geq r$

The balls of radius r form a cover, which is similar to the partition provided by the square cells of one level of a quadtree.



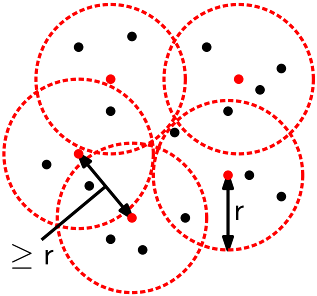
r-Net

r-Net

...is a subset $X \subseteq S$ such that

(i) every point of S is within distance r of some point of X

(ii) the pairwise distance between any two points of X is $\geq r$



The balls of radius r form a cover, which is similar to the partition provided by the square cells of one level of a quadtree.

r-Net

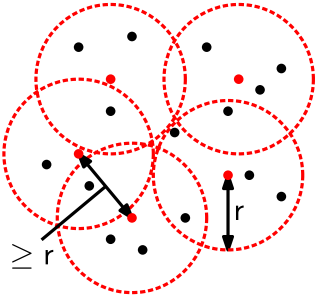
r-Net

...is a subset $X \subseteq S$ such that

(i) every point of S is within distance r of some point of X

(ii) the pairwise distance between any two points of X is $\geq r$

The balls of radius r form a cover, which is similar to the partition provided by the square cells of one level of a quadtree.

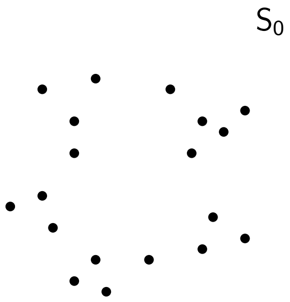


Net Tree

Net Tree [CG06, GGN04, HM06]

- The leaves are $S_0 = S$.
- Let r be the minimum distance between any two points of S.
- Let S_1 be an r-net of S_0.
- Let S_2 be a $(2r)$-net of S_1.
- Let S_j be a $(2^j r)$-net of S_{j-1}.
- ...
- Until only one remains — the root.

Similar to the quadtree in spirit, but intrinsic to the point set.

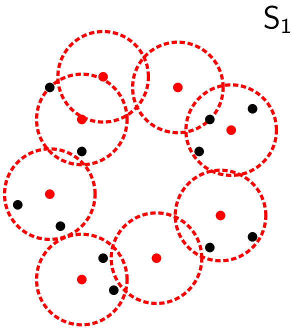


Net Tree

Net Tree [CG06, GGN04, HM06]

- The leaves are $S_0 = S$.
- Let r be the minimum distance between any two points of S.
- Let S_1 be an r-net of S_0.
- Let S_2 be a $(2r)$-net of S_1.
- Let S_j be a $(2^j r)$-net of S_{j-1}.
- ...
- Until only one remains — the root.

Similar to the quadtree in spirit, but intrinsic to the point set.

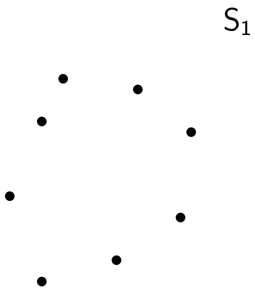


Net Tree

Net Tree [CG06, GGN04, HM06]

- The leaves are $S_0 = S$.
- Let r be the minimum distance between any two points of S.
- Let S_1 be an r-net of S_0.
- Let S_2 be a $(2r)$-net of S_1.
- Let S_j be a $(2^j r)$-net of S_{j-1}.
- ...
- Until only one remains — the root.

Similar to the quadtree in spirit, but intrinsic to the point set.

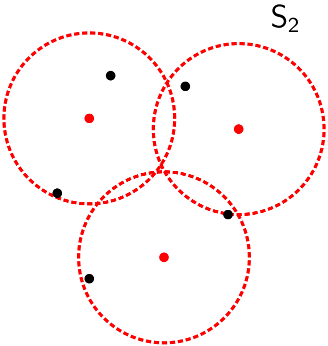


Net Tree

Net Tree [CG06, GGN04, HM06]

- The leaves are $S_0 = S$.
- Let r be the minimum distance between any two points of S.
- Let S_1 be an r-net of S_0.
- Let S_2 be a $(2r)$-net of S_1.
- Let S_j be a $(2^j r)$-net of S_{j-1}.
- ...
- Until only one remains — the root.

Similar to the quadtree in spirit, but intrinsic to the point set.

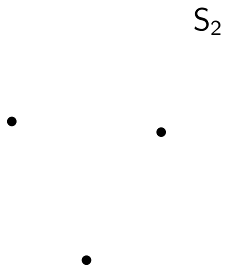


Net Tree

Net Tree [CG06, GGN04, HM06]

- The leaves are $S_0 = S$.
- Let r be the minimum distance between any two points of S.
- Let S_1 be an r-net of S_0.
- Let S_2 be a $(2r)$-net of S_1.
- Let S_j be a $(2^j r)$-net of S_{j-1}.
- ...
- Until only one remains — the root.

Similar to the quadtree in spirit, but intrinsic to the point set.

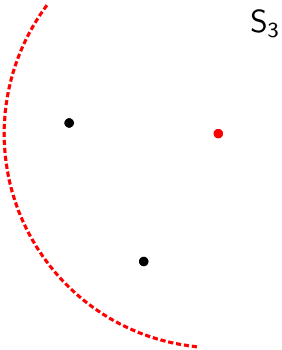


Net Tree

Net Tree [CG06, GGN04, HM06]

- The leaves are $S_0 = S$.
- Let r be the minimum distance between any two points of S.
- Let S_1 be an r-net of S_0.
- Let S_2 be a $(2r)$-net of S_1.
- Let S_j be a $(2^j r)$-net of S_{j-1}.
- ...
- Until only one remains — the root.

Similar to the quadtree in spirit, but intrinsic to the point set.



Net Tree

Net Tree [CG06, GGN04, HM06]

- The leaves are $S_0 = S$.
- Let r be the minimum distance between any two points of S.
- Let S_1 be an r-net of S_0.
- Let S_2 be a $(2r)$-net of S_1.
- Let S_j be a $(2^j r)$-net of S_{j-1}.
- ...
- Until only one remains — the root.

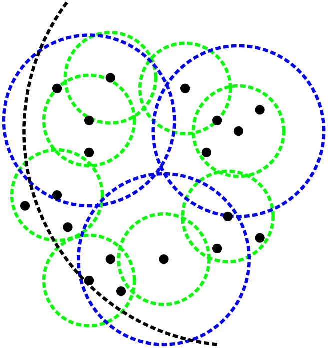
Similar to the quadtree in spirit, but intrinsic to the point set.

Net Tree

Net Tree [CG06, GGN04, HM06]

- The leaves are $S_0 = S$.
- Let r be the minimum distance between any two points of S.
- Let S_1 be an r-net of S_0.
- Let S_2 be a $(2r)$-net of S_1.
- Let S_j be a $(2^j r)$-net of S_{j-1}.
- ...
- Until only one remains — the root.

Similar to the quadtree in spirit, but intrinsic to the point set.



Challenges

Depth vs. Breadth: Most innovations have focused on maintaining structures of low depth. However, search times are dominated by the search's breadth, that is, the number of cells visited.

Tight vs. Slack: The best static structures achieve efficiency by enforcing tight constraints on subdivision properties (e.g. tree depth, cell size, aspect ratio). However, tight constraints result in frequent certificate failures, as used by kinetic structures.

The Right Mixture: What is the proper mixture of methods to achieve best overall performance, in terms of accuracy and execution times?

Bibliography

S. Arya, D. M. Mount, N. S. Netanyahu, R. Silverman, and A. Wu.
An optimal algorithm for approximate nearest neighbor searching.
J. Assoc. Comput. Mach., 45:891–923, 1998.

S. Arya and D. M. Mount.
Approximate range searching.
Comput. Geom. Theory Appl., 17:135–163, 2001.

Bibliography

R. Cole and L. Gottlieb.
Searching dynamic point sets in spaces with bounded doubling dimension.
In *Proc. 38th Annu. ACM Sympos. Theory Comput.*, pages 574–583, 2006.

C. A. Duncan, M. T. Goodrich and S. G. Kobourov,
Balanced Aspect Ratio Trees: Combining the Benefits of k-D Trees and Octrees.
J. Algorithms, 38:303–333, 2001.

D. Eppstein, M. T. Goodrich and J. Z. Sun
The Skip Quadtree: A Simple Dynamic Data Structure for Multidimensional Data.
In *Proc. 21st Symp. Comp. Geom.*, pages 296–305, 2005.

Bibliography

J. Gao, L. J. Guibas and A. Nguyen.
Deformable Spanners and Applications.
In *Proc. 20th Symp. on Comp. Geom.*, pages 179–199, 2004.

S. Har-Peled and M. Mendel.
Fast construction of nets in low dimensional metrics, and their applications.
SIAM J. Comput., 35(5):1148–1184, 2006.

W. Pugh.
Skip lists: A probabilistic alternative to balanced trees.
Commun. ACM, 33(6):668–676, 1990.