Fast Algorithm for Nearest Neighbor Search Based on a Lower Bound Tree

Yong-Sheng Chen

Presenter: *Yen-shuo Huang*

Multimedia Communication and Computer Lab, NTNU CSIE

Outline

- Introduction
- Multilevel Structure and LB-Tree
- Agglomerative Clustering
- Data Transformation
- Winner-Update Search
- Conclusions

Reference

- Fast Algorithm for Nearest Neighbor Search Based on a Lower Bound Tree, Yong-Sheng Chen, Yi-Ping Hung, etc., *ICCV 2001*

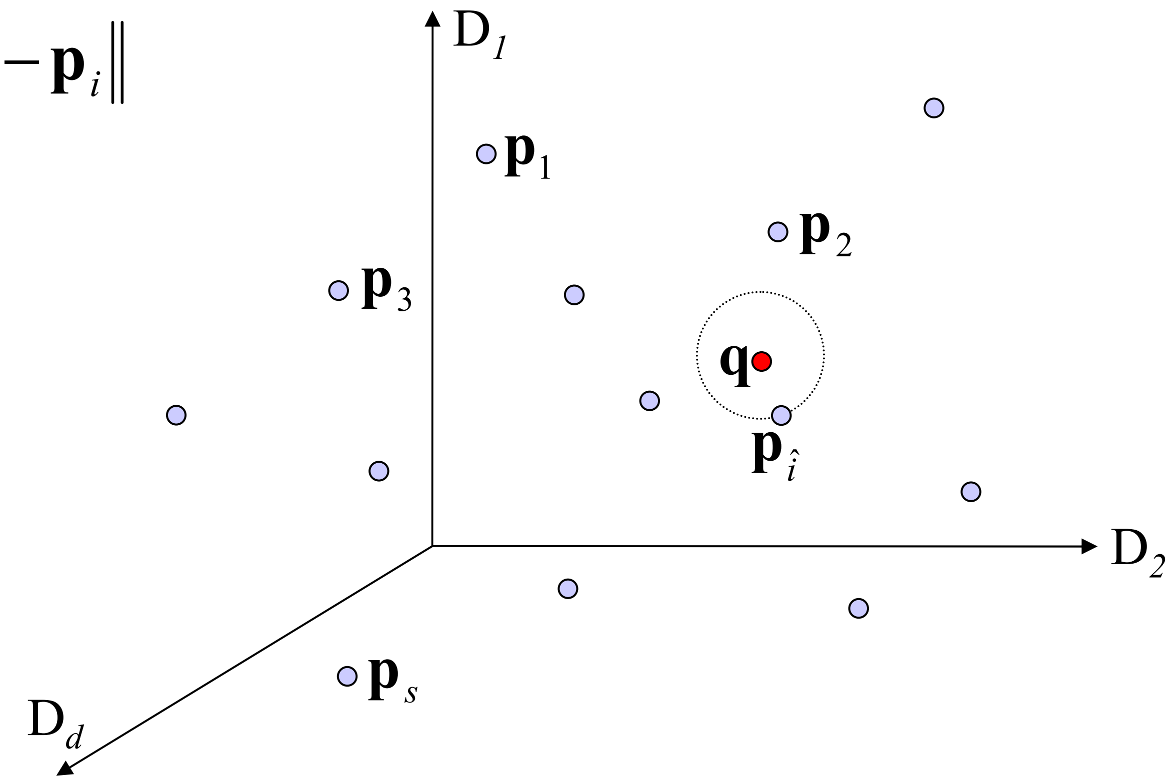
Applications of Nearest Neighbor Search

- Object (pattern) recognition
- Image matching
- Data compression
  - Block motion estimation
  - Vector quantization
- Information retrieval in database systems
  - Image and video databases
  - DNA sequence databases

Nearest Neighbor Search Problem

- Given a fixed set of s points in R^d, $P=\{\mathbf{p}_1,\mathbf{p}_2,\ldots,\mathbf{p}_s\}$
- For a query point $\mathbf{q}$, find in P the point, $\mathbf{p}_{\hat{i}}$, closest to $\mathbf{q}$

$$\hat{i} \equiv \arg\min_{i=1\cdots s} \|\mathbf{q}-\mathbf{p}_i\|$$



Literature Review

- Space partition methods
  - *k*-d tree, 1975
  - R-tree, 1984
  - SS-tree, 1996
  - SR-tree, 1997
  - Pyramid, 1998
- Elimination-based methods
  - Branch and bound, 1975
  - Projection elimination, 1975, 1987
  - Threshold rejection, 1997

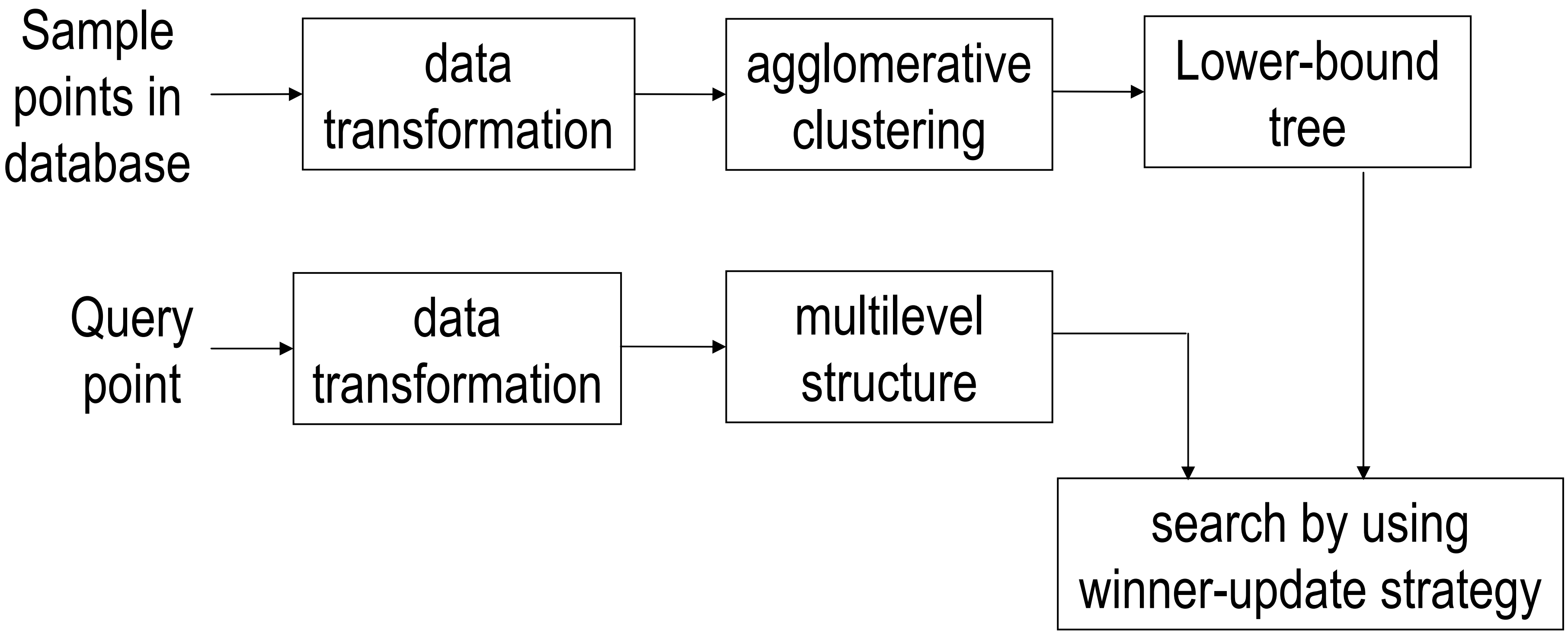
Central Idea of This Work

- We skip distance calculation for a point in database by calculating and comparing its distance lower bound.
  - Distance lower bound was derive from Minkowski's inequality.
  - Computational cost of the distance lower bound is less than that of the distance itself.
  - *Optimal search* is guaranteed.

Central Idea of This Work

- Reduce the number of distance lower bounds actually calculated by using
  - an *LB-tree* constructed in the preprocessing stage
  - the ***winner-update search strategy*** for traversing the constructed LB-tree
- Tighten distance lower bounds by
  - constructing the LB-tree with an *agglomerative clustering* method
  - transforming each data point with
    - Wavelet transform
    - Principal Component Analysis

Proposed Algorithm for NN Search



Multilevel Structure of a Point

- For a point $\boldsymbol{p}=[p_1, p_2, \ldots, p_d]$ in R^d, $d=2^L$, we denote its multilevel structure:

  $\{\boldsymbol{p}^0, \boldsymbol{p}^1, \ldots, \boldsymbol{p}^L\}$

EX

$$\mathbf{p}^0 = [p_1] \qquad \|\mathbf{p}^0-\mathbf{q}^0\|_2 \qquad \mathbf{q}^0 = [q_1]$$

$$\leq$$

$$\mathbf{p}^1 = [p_1, p_2] \qquad \|\mathbf{p}^1-\mathbf{q}^1\|_2 \qquad \mathbf{q}^1 = [q_1, q_2]$$

$$\leq$$

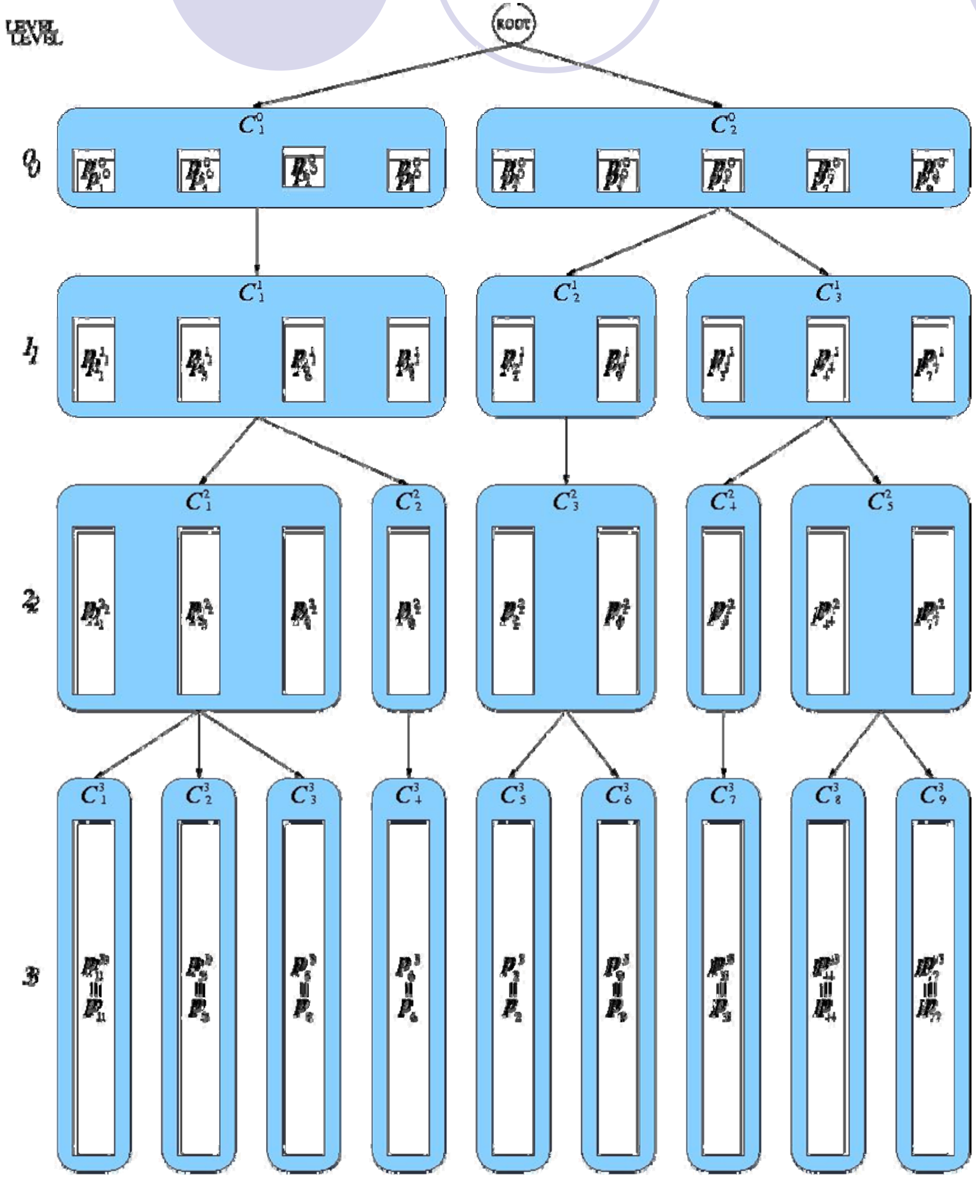
$$\mathbf{p}^2 = [p_1, p_2, p_3, p_4] \qquad \|\mathbf{p}^2-\mathbf{q}^2\|_2 \qquad \mathbf{q}^2 = [q_1, q_2, q_3, q_4]$$

$$\leq$$

$$\mathbf{p}^3 = [p_1, p_2, p_3, p_4, p_5, p_6, p_7, p_8] \qquad \|\mathbf{p}^3-\mathbf{q}^3\|_2 \qquad \mathbf{q}^3 = [q_1, q_2, q_3, q_4, q_5, q_6, q_7, q_8]$$

$$\|\mathbf{p}^l-\mathbf{q}^l\|_2 \leq \|\mathbf{p}-\mathbf{q}\|_2,\ l=0,\ldots,L$$

Lower Bound Tree



- Multilevel structures of every points in the database, $\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_s$
- Idea of node reduction
  - Select representatives
- Hierarchical, agglomerative clustering

Hierarchical, Agglomerative Clustering

Ex:

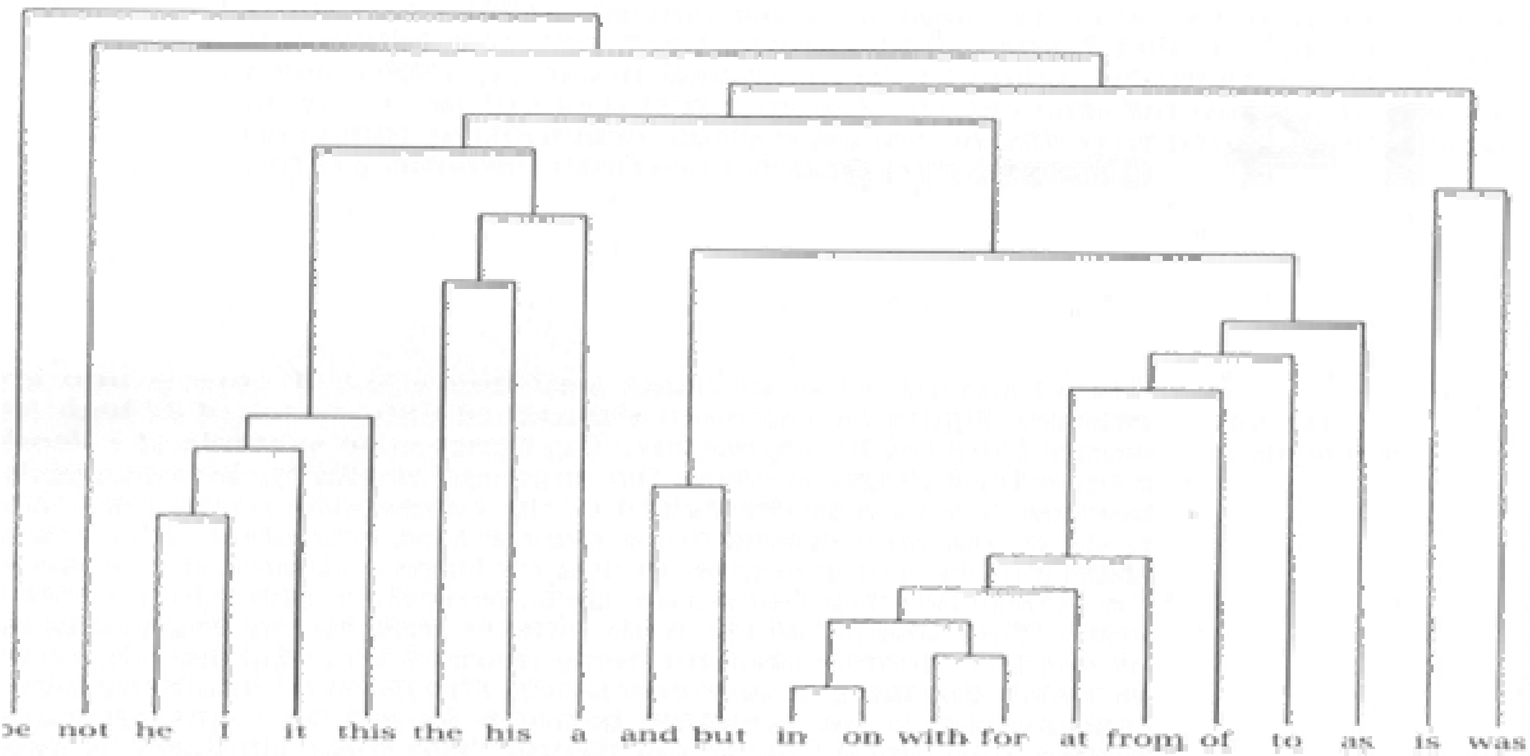
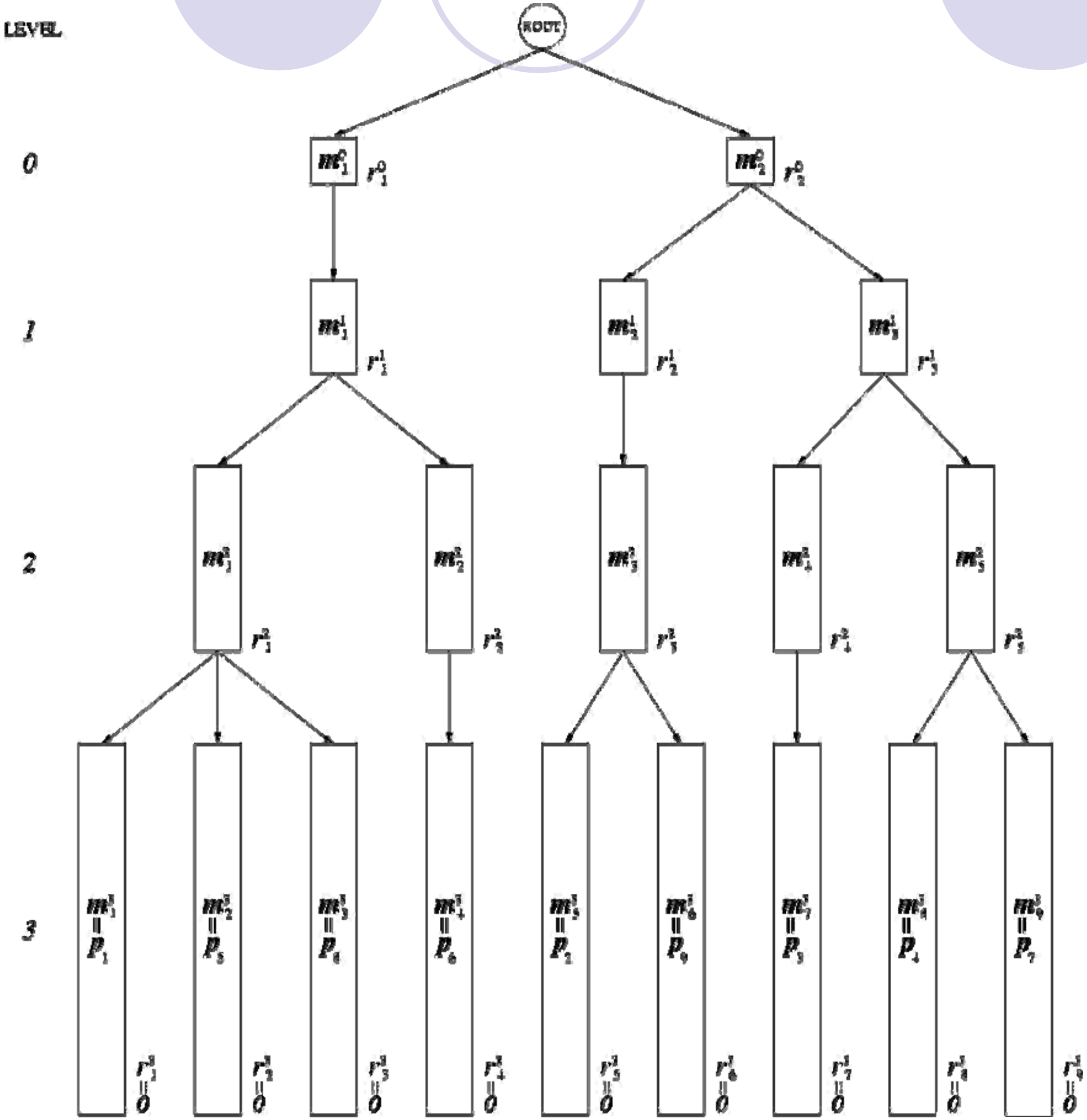


Figure 14.1 A single-link clustering of 22 frequent English words represented as a dendrogram.

Lower Bound Tree



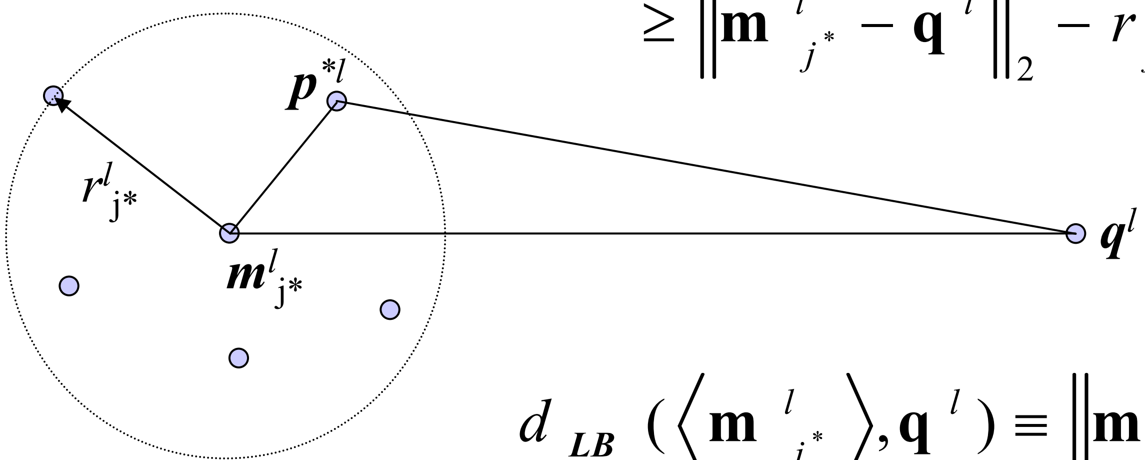
- Representative for each cluster
  - mean point $\boldsymbol{m}^{l}_{j*}$
  - distance r^{l}_{j}* of the farthest point in the cluster from $\boldsymbol{m}^{l}_{j*}$
- Each point $\boldsymbol{p}^{*l}$ satisfies

$$\left\| \mathbf{p}^{*l} - \mathbf{m}^{l}_{j^*} \right\|_2 \le r^{l}_{j^*}$$

Distance Lower Bound Using LB-Tree

- For each point $\mathbf{p}^*$, we can derive the lower bound of its distance to the query point $\mathbf{q}$:

$$\begin{aligned}\left\|\mathbf{p}^* - \mathbf{q}\right\|_2 &\geq \left\|\mathbf{p}^{*l} - \mathbf{q}^l\right\|_2 \\ &\geq \left\|\mathbf{m}^l_{j^*} - \mathbf{q}^l\right\|_2 - \left\|\mathbf{p}^{*l} - \mathbf{m}^l_{j^*}\right\|_2 \\ &\geq \left\|\mathbf{m}^l_{j^*} - \mathbf{q}^l\right\|_2 - r^l_{j^*}\end{aligned}$$

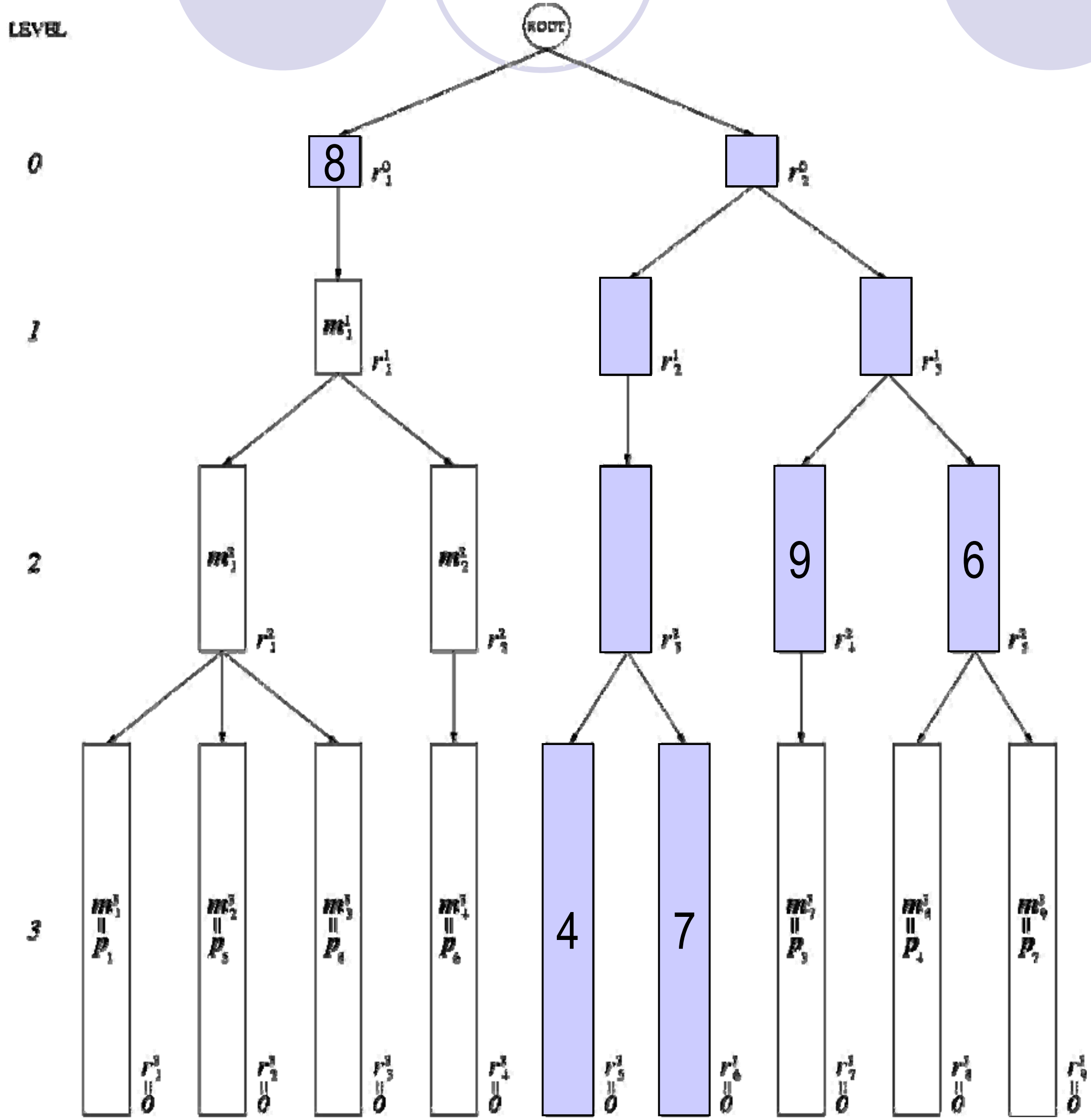


$$d_{LB}\left(\left\langle \mathbf{m}^l_{j^*} \right\rangle, \mathbf{q}^l\right) \equiv \left\|\mathbf{m}^l_{j^*} - \mathbf{q}^l\right\|_2 - r^l_{j^*}$$

Data Transformation

- Why Data Transformation?
  - Make anterior dims more discriminative than posterior dims.
  - Lower bound can be tightened.
- By data content, two transformations used in this work:
  - Haar wavelet transform(autocorrelated data)
  - Principal Component Analysis (object recognition)

Winner-Update Search for LB-Tree Traversal



Given a query point $\mathbf{q}$

Calculate d_{LB} for all the nodes in level 0

Choose the node having the minimum d_{LB} as the temporary winner

While the winner is not at the bottom level

- Replace the winner node with its children
- Calculate d_{LB} for each new child node
- Choose the node having the minimum d_{LB} as the temporary winner

Output the final winner

Other Query Types

- Winner-update algorithm can be easily extended to support other useful query types:
  - Progressive search for k-nearest neighbors
  - Search for k-nearest neighbors within a distance threshold

Conclusions

- Fast nearest neighbor search techniques, including
  - Lower bound tree
  - Winner-update search strategy
  - Data transformation
    - Wavelet transform
    - Principal component analysis
  - Various useful query types
- According to our experiments on Nayar's object recognition database, our algorithm can be more than one thousand faster than the full search algorithm.

Thank you.