Tools for Image Retrieval in Large Multimedia Databases

by Carles Ventura Royo

Directors:
Verónica Vilaplana
Xavier Giró

Tutor:
Ferran Marqués

Barcelona, September 2011
Index

- Identifying the problem
- State of art: indexing techniques
- State of art: Hierarchical Cellular Tree (HCT)
- Modifications to the original HCT
- Experimental results
- Implemented tools
- Conclusions and future work lines
Identifying the problem (I)
Identifying the problem (II)

QUERY

Visual + textual descriptors

K

TARGETS
Identifying the problem (III)

- K nearest neighbor problem
- Solution: Sequential scan
  - **Drawback**: Computational time for large databases (10 s for a 200,000 elements)
- Approximate K nearest neighbor problem
  - Indexing techniques
Requirements of the solution

- Dynamic approach
  - Multimedia databases are not static
  - Insertions and deletions

- High dimensional feature spaces
  - “curse of dimensionality” problem
  - MPEG-7 visual descriptors are high-dimensional feature vectors
Index

- Identifying the problem
- **State of art: indexing techniques**
- State of art: Hierarchical Cellular Tree (HCT)
- Modifications to the original HCT
- Experimental results
- Implemented tools
- Conclusions and future work lines
Indexing techniques (I)

- Hierarchical data structures
  - Spatial Access Methods (SAMs)
    - K-d tree, R-tree, R*-tree, TV-tree, etc.
  - **Drawbacks**:
    - Items have to be represented in an N-dimensional feature space
    - Dissimilarity measure based on a $L_p$ metric
    - SAMs do not scale up well to high dimensional spaces
Indexing techniques (II)

- Hierarchical data structures
  - Metric Access Methods (MAMs)
    - VP-tree, MVP-tree, GNAT, M-tree, etc.
    - More general approach than SAMs
      - Assuming only a similarity distance function
    - MAMs scale up well to high dimensional spaces
  - **Drawbacks:**
    - Static MAMs do not support dynamic changes
    - Dependence on pre-fixed parameters
Indexing techniques (III)

- **Locality Sensitive Hashing**
  - It uses hash functions
  - Nearby data points are hashed into the same bucket with a high probability
  - Points faraway are hashed into the same bucket with a low probability
  - **Drawback**: It does not solves the K nearest neighbor problem, but the $\epsilon$-near neighbor problem.
Index

- Identifying the problem
- State of art: indexing techniques
- **State of art: Hierarchical Cellular Tree (HCT)**
- Modifications to the original HCT
- Experimental results
- Implemented tools
- Conclusions and future work lines
Solution adopted

- Hierarchical Cellular Tree (HCT)
  - MAM-based indexing scheme
  - Hierarchical structure
  - Self-organized tree
  - Incremental construction in a bottom-up fashion
  - Unbalanced tree
  - Not dependence on a maximum capacity
  - Preemptive cell search algorithm for insertion
  - Dynamic approach

HCT: Cell Structure (I)

- Basic container structure
- Undirected graph
- Minimum Spanning Tree (MST)
- Cell nucleus
- Covering radius
HCT: Cell Structure (II)

- Cell compactness
  \[ CF_C = f(\mu_C, \sigma_C, r_C, \max(w_C), N_C) \geq 0 \]
- Maturity size
- Mitosis operation
HCT: Level Structure

- Representatives for each cell from the lower level
- Responsible for maximizing the compactness of its cells
- Compactness threshold

\[ CThr_L = \frac{1}{k_0} \text{Median}(CFC | \forall C \in S_M) \]
HCT Operations (I)

- Item insertion
  - Find the most suitable cell

- Most Similar Nucleus vs Preemptive Cell Search

![Diagram with invalid labels]
HCT Operations (II)

- **Item insertion**
  - Find the most suitable cell

- **Most Similar Nucleus**
  - vs **Preemptive Cell Search**

\[ C_2: d(O, O^2_N) - r(O^2_N) < d_{\text{min}} \]
\[ C_3: d(O, O^3_N) - r(O^3_N) > d_{\text{min}} \]
HCT Operations (III)

- Item insertion
  - Find the most suitable cell
  - Append the element
  - Generic post-processing check
    - Mitosis operation
    - Nucleus change
HCT Operations (IV)

- Item removal
  - Cell search algorithm not required
  - Remove the element
  - Generic post-processing check
    - Mitosis operation
    - Nucleus change
HCT: Retrieval scheme

- **Progressive Query**
  - Periodical subqueries over database subsets
  - Query Path formation
    - Based on Most Similar Nucleus
- **Ranking aggregation**
Index

- Identifying the problem
- State of art: indexing techniques
- State of art: Hierarchical Cellular Tree (HCT)
- **Modifications to the original HCT**
- Experimental results
- Implemented tools
- Conclusions and future work lines
Modifications to the HCT (I)

- Covering radius
  - Original definition gives an approximation by defect
  - Consider all the elements belonging to the subtree
    - High computational cost
  - Approximation by excess

\[
r_C = \max(r_C(S_N), d(O_1, O_N) + r_C(S_1), \ldots, d(O_M, O_N) + r_C(S_M))
\]
Modifications to the HCT (II)

- Covering radius
Modifications to the HCT (III)

- Covering radius

\[ r_C(C_9) = \max \left( d(E,B) + r_C(C_1), r_C(C_2), d(E,H) + r_C(C_3) \right) \]
Modifications to the HCT (III)

* HCT construction
  * Preemptive Cell Search over all the levels
  * A method for updating the covering radius
    * To reduce the searching time
    * It can be performed after the HCT construction or periodically
Modifications to the HCT (IV)

- Searching techniques
  - PQ fails in solving the KNN problem efficiently
  - New searching techniques
    - Most Similar Nucleus
    - Preemptive Cell Search
    - Hybrid
  - Number of cells to be considered
    - Minimum number of cells
    - Cells hosting $2 \cdot K$ elements
    - Cellular structure is not kept
Index

- Identifying the problem
- State of art: indexing techniques
- State of art: Hierarchical Cellular Tree (HCT)
- Modifications to the original HCT
- Experimental results
- Implemented tools
- Conclusions and future work lines
Experimental results

- CCMA image database of 216,317 elements
- HCT building evaluation
  - Construction time
- Retrieval system evaluation
  - Retrieval time
  - Elements retrieved
HCT building evaluation

![Graph showing HCT building time vs. number of elements inserted. The graph includes lines for original covering radius, proposed covering radius, and proposed covering radius with update method every 5000, 10000, and 25000 insertions. The y-axis represents HCT building time in seconds, and the x-axis represents the number of elements inserted. The graph includes data points for various x and y values marked at specific coordinates.]
Retrieval system evaluation (I)
Retrieval system evaluation (V)
Retrieval system evaluation (VI)
Retrieval system evaluation (II)

- Evaluation with respect to exhaustive search
  - Mean Competitive Recall
    - Elements in common
  - Mean Normalized Aggregate Goodness
    \[ NAG(k, q, A) = \frac{W(k, q, E) - \sum_{p \in A(k, q, E)} d(p, q)}{W(k, q, E) - \sum_{p \in GT(k, q, E)} d(p, q)} \]
  - Kendall distance
    - Number of exchanges needed in a bubble sort
  - Query set of 1,082 images
Retrieval system evaluation (III)

- **Preemptive Cell Search**

<table>
<thead>
<tr>
<th></th>
<th>proposed covering radius</th>
<th>original covering radius</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>non updated</td>
<td>updated</td>
</tr>
<tr>
<td>Mean retrieval time (s)</td>
<td>1.2386</td>
<td>0.8319</td>
</tr>
<tr>
<td>Variance retrieval time (s)</td>
<td>0.1886</td>
<td>0.1466</td>
</tr>
<tr>
<td>Retrieved queries (%)</td>
<td>99.26</td>
<td>99.26</td>
</tr>
<tr>
<td>$CR^\overline{}$</td>
<td>28.09</td>
<td>27.51</td>
</tr>
<tr>
<td>NAG</td>
<td>0.9970</td>
<td>0.9967</td>
</tr>
<tr>
<td>Kendall</td>
<td>295.24</td>
<td>313.87</td>
</tr>
</tbody>
</table>
Retrieval system evaluation (IV)

- Searching techniques comparative

<table>
<thead>
<tr>
<th></th>
<th>MS-Nucleus</th>
<th>Hybrid (7 levels)</th>
<th>Hybrid (8 levels)</th>
<th>Hybrid (9 levels)</th>
<th>Preemptive</th>
<th>MS-Nucleus (20,000 el)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean retrieval time (s)</td>
<td>0.0072</td>
<td>0.2172</td>
<td>0.4144</td>
<td>0.6835</td>
<td>0.8319</td>
<td>1.3695</td>
</tr>
<tr>
<td>Variance retrieval time (s)</td>
<td>2.1e-05</td>
<td>0.0009</td>
<td>0.0093</td>
<td>0.0569</td>
<td>0.1466</td>
<td>0.0054</td>
</tr>
<tr>
<td>Retrieved queries (%)</td>
<td>5.00</td>
<td>37.99</td>
<td>57.95</td>
<td>84.20</td>
<td>99.26</td>
<td>31.61</td>
</tr>
<tr>
<td>$\bar{CR}$</td>
<td>1.35</td>
<td>9.83</td>
<td>14.22</td>
<td>20.81</td>
<td>27.51</td>
<td>12.33</td>
</tr>
<tr>
<td>NAG</td>
<td>0.9087</td>
<td>0.9727</td>
<td>0.9824</td>
<td>0.9917</td>
<td>0.9967</td>
<td>0.9776</td>
</tr>
<tr>
<td>Kendall</td>
<td>1530.93</td>
<td>1106.08</td>
<td>883.51</td>
<td>580.43</td>
<td>313.87</td>
<td>1025.71</td>
</tr>
</tbody>
</table>
Index

- Identifying the problem
- State of art: indexing techniques
- State of art: Hierarchical Cellular Tree (HCT)
- Modifications to the original HCT
- Experimental results
- Implemented tools
- Conclusions and future work lines
Implemented tools (I)

- **database_indexing tool**
  - Tool for indexing an image database
  - HCT is stored at disk

- **hct_query tool**
  - Tool for carrying out a search over an indexed database
  - HCT is read from disk and load at main memory
Implemented tools (II)

- A server/client architecture
  - Based on a messaging system: KSC
Index

- Identifying the problem
- State of art: indexing techniques
- State of art: Hierarchical Cellular Tree (HCT)
- Modifications to the original HCT
- Experimental results
- Implemented tools
- Conclusions and future work lines
Conclusions

- Hierarchical Cellular Tree implementation
  - To improve the retrieval times
  - Generic implementation for any kind of data
  - Modifications proposed

- HCT evaluation
  - Measures extracted from literature

- Preemptive Cell Search technique gives the best performance
  - It is essential not to use an underestimated value for the covering radius
Future work lines

- Very large databases
  - Not using only main memory
- Region-based CBIR system
  - Each image can be represented by a set of regions
- Browser application based on HCT
  - Take advantage of the hierarchical structure
  - Alternative way to retrieve elements
Thanks for your attention