Experiment Project: Indexing Images for Content Based Retrieval

Instructor: Xiaoming Jin
xmjin@tsinghua.edu.cn
Teaching Assistant:
Zihan Zhang: zhangzihan.zac@gmail.com
Changsheng Xiang xcszbdnl@gmail.com
Due Date: May 23, 2016

Three most important things:

- Before you start this project, please read this document carefully, and make sure your submission meets all the requirements.
- Intra-group collaboration is encouraged, but inter-group "collaboration" in this project is not permitted.
- All the materials provided by this project are not permitted to be redistributed for any purpose.

The rest of this document is organized as follows: section 1 presents a general description about the task. Section 2 discusses some related knowledge and techniques of this project in detail. All required and optional tasks of this project are specified in Section 3. Submission policy is presented in section 4. Section 5 provides some useful hints and public resources, which you can skip if you have already had sufficient prior knowledge about content based image retrieval.

1 Introduction

This assignment gives you an opportunity to learn how to manage large database, particularly to index and search images by content. The assignment is divided into two parts:

- **Image Indexing**: you should try to extract several features from images and index the features using R-tree.
- **Image Searching**: you should provide an interface (not necessary to be a graphical interface) for querying by example. That is, given an image as query (which is selected from the provided dataset), your program should return a sorted list of images according to their relevance to the query.

2 Details

2.1 Images

The provided image dataset consists of 5613 images downloaded from the Internet. All the images are stored in the data\Image.zip file.

The category information of each image is indicated by the string before '_' of its filename. Take 'n03877845_251.JPEG' as an example, its category is n03877845. This information can tell whether a retrieved image is really relevant to the query image. But please note that you can **NOT** use this category information when processing the query, because in real systems we never know the category label of a query image.

2.2 Features

To build index and retrieval, you need to extract features from the images first. We recommend that you use color histogram features. The dimensionality of features should be adjustable so that your empirical study (see Section 3) can be accomplished. Some sample codes for basic image accessing are provided to help you get started. The codes are stored in src\LoadImage directory, including both Java and VC++ .NET versions. By running the code, an image can be loaded into main memory and each pixel of it can

be accessed in a simple way. This may help you design and implement an algorithm to build the image features.

You can also make a quick startup by using the features provided by the project. The feature we offer is: *color moment features*, which is of 9 dimensions. In order to try out the task, you can ignore the details of the image features if you are not interested in. The features are stored in the file data\color_feature.txt, with each line corresponding to one image file. Take the image file n03877845_101.JPEG as an example, its color moment feature is a 9-dimensional vector:

(1553 4367 6897 2239 3278 2712 3094 3553 3062)

The feature file also contains several commands used by the example program introduced in Section 2.3, e.g.

```
tech nmos
<< polysilicon >>
rect
<< end >>
```

You can just ignore these commands in your program. Please remember that the feature file provided here is just to help you make a quick start, you have to build your own features for the images.

2.3 R-tree

Second, you are asked to implement the algorithms for managing R-tree. For your convenience in completing this task, an implementation of R-tree developed by Guttman[1] is provided. You could first skim the implementation, and then develop your own version based on your understanding.

The source codes are saved in src\rtree directory. Since the original version can only be compiled under Unix system, if you want to use it you have to install a Unix/Linux system or a simulator, e.g. Cygwin. After the environment is properly setup, please run "make linear" in the src\rtree directory to build the project. The implementation provided here has been modified and built by the teaching assistant. The new version is currently set up to index 9-dimensional rectangles and can support query by examples. To run the program, you need to specify three parameters on the command line:

./linear.exe third/half File1 File2

The third/half parameter is used to set the minimum node fill number (i.e. m), which can be one half or one third of the maximum entries number per node (i.e. fan-out M). File1 contains the features for the images to be indexed. File2 contains the features for the images you need to search after the indexing step is completed. For example, you can type in "./linear.exe third color_feature.txt color_feature.query" to see the results. Other parameters, such as maximum entries number per node and disk block size, etc, are set in the source codes.

The original R-tree package supports only exact query rather than range query, which is insufficient for you to complete this project. If you want to use this package, you have to make proper modifications and add some necessary implementations such as those for supporting range query.

2.4 Alternative R-tree packages

We also provide two alternative implementations (C++[2] and JAVA[3]) of R-tree, which can be found in the folder src\rtree_alternative_package. In this project, you are allowed to use any other packages from Internet as long as they are used under corresponding license. Please be noted, most packages are not our task oriented, so we suggest you first have a good understanding about the implementation, and then utilize these packages to develop your own system to complete all experiments in Section 3.

2.5 Performance evaluation

The efficiency of R-tree based query processing can be evaluated approximately in terms of disk access number, instead of the CPU time consumed. Moreover, please note that you don't need to implement a disk storage procedure to count the number of disk access exactly. Instead, you could simply count the number of node access to estimate the time complexity. This is reasonable since each node of R-tree is fitted in one disk block in real applications.

One widely used method to judge whether two images are relevant is to

compare their category information. If and only if they come from the same category, they are regarded as relevant.

3 Problems

Please do the experiments to solve the following problems and accomplish an experiment report.

1. Try to compare the performance of R-tree based query processing in terms of disk access number, with respect to the number of features used (e.g. 4, 8, 12, 16, 20), and the number of data objects inserted (e.g. 1000, 2000, 3000, 4000, 5000). The query strategy is range-query. Each time, you should provide the average results over a group of queries (e.g. 5000). (60%)

2. Implement the extracting methods for one more type of image features that you believe most useful for image retrieval. You could utilize either existing feature schemas or new ones designed by yourself. The only requirement is that the extracted features should be in vector form. Once the features are extracted, try to analyze its effectiveness and compare it with the features provided by this project (*color moment features*). (20%)

3. Try to study the influence of features on the relevance ranking. Specifically, you should use some different sets of features in indexing and querying (e.g., by varying the number of features, or by varying the type of features), and calculate the relevance between the query image and the search results. For this task, you could use either the features provided by this project or the features extracted by yourself. (20%)

4. (Optional) Improve the performance of R-tree. Your method should be of novelty. Moreover, both empirical and theoretical studies of the proposed method are required to justify the superiority of your method

over the original R-tree. (10% each, with a 20% ceiling in total)

(a) Identify factor(s) that affects the number of node splits during data insertion and illustrate how; propose your own idea, if any, to reduce the number of node splits with empirical evaluation.

(b) Identify factor(s) that affects the level of MBR overlapping and illustrate how; propose your own idea, if any, to reduce the level of MBR overlapping with empirical evaluation.

(c) Implement a query processing algorithm for one more similarity (relevance) function that you think useful for this task. Investigate how the performance of similarity query varies with different similarity functions.

(d) Modify the R-tree codes to implement an index structure utilizing bounding shape other than MBR, e.g. bounding spheres. Investigate how the performance of similarity query varies with different bounding shapes.

(e) Any other possible improvement on the indexing scheme of R-tree that can better support any type of query. Evaluate your idea with empirical results.

4 Submission

- Please submit your experiment report together with source code before 23:59:59, May 23, 2016 to course website. Each submission can have at most THREE names on it. If you like, you could also specify the weights of the group members according to their contributions.
- The experiment report should include:

  - Outline of your experimental system. It includes but is not limited to the system structure and the definitions of the features you have considered for problem 2.
  - Your observations corresponding to the three required problems in section 3 respectively.
  - Your analysis and concluding remarks.
  - (Optional) Your novel idea on indexing. Please describe your method clearly, and provide the analyzing remarks that support your idea.
  - Necessary references.

- You can write the report in either English or Chinese. But it should be clear and be limited within **3** pages using **the given template** [4], which can be found in the template folder. **If not using template**, 10% ceiling for overall score will be penalized. Both LaTeX and Microsoft Word templates are provided. Please convert your report to PDF format for submission.

- For this project, either implementation by yourself or use of existing packages is allowed. However, note that, in the latter case, explicit declaration of used packages is required. The evaluation will mainly be based on YOUR work rather than the downloaded packages. That is, the grading will depend mainly on how meaningful your work is, instead of how "nice" your results seem to be. This policy is also effective for the optional question (question 3). Failing in citing referenced work will be regarded as plagiarizing and thus result in immediate ZERO credit.

- There is a 120% ceiling for overall score of this assignment. That is, if your work is so excellent that your raw score exceeds 120%, we will record 120%. And of course, your overall score for this course will not exceed 100.

- Important dates:
  - Report & project code submission: 23:59:59, May 23, 2016

Your submission is worth full credit before the due date, 75% credit before 3 days later (23:59:59, May 26, 2016), half credit in a week (23:59:59, May 30, 2016) and **ZERO** credit after that.

5 Hints and Resources

- Survey [5] may help you get an overview of content-based multimedia information retrieval, including its history, concepts and techniques. The details of these techniques can be found in the references listed in the paper. Perhaps you can dig into some of them and develop your implementations to extract features and improve your system.
- Survey [6] [7] examines various spatial indexes proposed in literature and presents taxonomy of these structures. It reviews each structure by brief summary, comparison with similar structures, characteristics and algorithms for various operations. It also presents a comparative analysis of all of these structures. It is very helpful for you to get a full understanding of spatial indexing methods.
- Survey [8] introduces various kinds of content-base image retrieval systems. You may have a look at these systems if you are interested.
- Packages [9] can help you study and develop spatial access methods (SAM). Many types of R-Tree are implemented in these packages. However, you are encouraged to learn from these implementations and then develop your own versions based on your understanding.

References

[1] Antonin Guttman. R-trees: A dynamic index structure for spatial searching. In *ACM SIGMOD International Conference on Management of Data*, pages 47–57, 1984.

[2] http://superliminal.com/sources/sources.htm#C & C++ Code.

[3] http://jsi.sourceforge.net/.

[4] http://www.acm.org/chapters/policy/toolkit/template.html.

[5] Michael Lew and et al. Content-based multimedia information retrieval: State of the art and challenges. *ACM Transactions on Multimedia Computing, Communications, and Applications*, pages 1–19, 2006.

[6] Volker Gaede and Oliver Gnther. Multidimensional access methods. *ACM Computing Surveys (CSUR)*, 30(2), 1998.

[7] Jayendra Venkateswaran. A survey of recent multidimensional access methods. Technical report, University of Missouri-Rolla.

[8] Remco C. Veltkamp and Mirela Tanase. A survey of content-based image retrieval systems. Technical report, Department of Computer Science, Utrecht University, 2000.

[9] http://www.rtreeportal.org/code.html.