# LIRE: Open Source Image Retrieval in Java

Mathias Lux
Klagenfurt University
Universitätsstr. 65-67
Klagenfurt, Austria
mlux@itec.uni-klu.ac.at

## ABSTRACT

Content based image retrieval has been around for some time. There are lots of different test data sets, lots of published methods and techniques, and manifold retrieval challenges, where content based image retrieval is of interest. LIRE is a Java library, that provides a simple way to index and retrieve millions of images based on the images' contents. LIRE is robust and well tested and is not only recommended by the websites of ImageCLEF and MediaEval, but is also employed in industry. This paper gives an overview on LIRE, its use, capabilities and reports on retrieval and runtime performance.

## Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval

## Keywords

Image retrieval, image features, image search, image indexing

## 1. INTRODUCTION

How to search in large image data sets? While Google image search[1] is ready at hand and yields impressive results, software for doing the same in arbitrary applications and scenarios is rare. Search techniques based on meta data have been explored a lot. Retrieval is then based on time and date or GPS coordinates, i.e. from EXIF data embedded by the camera, or relies on user assigned keywords. However, there are scenarios, where automatically generated meta data is not sufficient and manual annotation is sparse. Then, content based image retrieval (CBIR) is one avenue to pursue.

In this paper I introduce LIRE[2] [15], short for *Lucene Image Retrieval*, a library for content based image retrieval. LIRE is based on *Lucene*, a well-known, robust and fast text search engine. It's also already well known in research, teaching and industry and has been applied to many different scenarios over the last years. Recently even a book on content based image retrieval with LIRE has been published [15] aimed at developers and students. Furthermore, LIRE is free, open source and known to work in Java VMs on Linux, Mac OS and Windows. LIRE aims to make life of software developers easier on a high level. Developers do not have to know specifics of content based image retrieval, but use the library to (i) index the images and (ii) search in the index. Parameter settings for features and indexing are hidden for users on the highest API level and are pre-tuned to yield reasonable performance. However, control on a lower level is possible and easily done. This document present LIRE in terms of general methods, usage and performance charts.

[1]see `http://images.google.com/`
[2]see `https://code.google.com/p/lire/`

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profi or commercial advantage and that copies bear this notice and the full citation on the firs page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specifi permission and/or a fee. Request permissions from permissions@acm.org.
*MM'13,* October 21–25, 2013, Barcelona, Spain.
Copyright is held by the owner/author(s). Publication rights licensed to ACM.
ACM 978-1-4503-2404-5/13/10...$15.00
http://dx.doi.org/10.1145/2502081.2502226 .

## 2. LIRE FEATURES

There are many different ways to implement a content based image search engine or integrate content based image retrieval capabilities into applications. The intuitive, but also most time-consuming way to do so, is to implement extraction, storage and retrieval of features oneself. However, a lot of time and effort can be saved if a library is used. Currently there are several libraries available, that allow for content based image retrieval. There is for instance the *Gnu Image Finding Tool* (GIFT)[3] developed by the University of Geneva, or *ImageTerrier* and *OpenIMAJ* [9]. Other libraries were developed with intentions different from providing content based image retrieval, i.e. libraries for computer vision or image processing. Nevertheless, those libraries can provide means for content based image retrieval. Examples are *OpenCV*[4] [3] or *scikit-image*[5].

Still, while there are several possibilities available, both open source and commercial, LIRE has two main features, that make it stick out of the crowd. First LIRE is extremely simple to use. With just a few lines of code an indexing routine with a selection of features is written, that makess optimal use of multi-core systems. Even more so LIRE provides an extensive offline GUI demo, which allows for visually testing different features on arbitrary data sets without programming knowledge. Second, LIRE is based on the well-known, well-supported and extremely efficient *Lucene* text search engine [10]. There is no need for an extra database or

[3]see `http://www.gnu.org/software/gift/`
[4]see `http://opencv.willowgarage.com/wiki/`
[5]see `http://scikit-image.org/`

special file system and Lucene specific features, like memory mapped indexes, distributed indexes, multi-threaded access, etc. can be used with LIRE. LIRE can even be integrated in existing Lucene based search engines.

## 2.1 Image Features

A main contribution of LIRE is the extensive set of global image features supported out of the box. Most prominent global features include:

1. Color Histograms supporting different color spaces, like HSV and the Opponent color space [21], and quantization routines as well as different metrics for distance computation.

2. MPEG-7 descriptors Scalable Color, Color Layout and Edge Histogram, see [4].

3. The Tamura texture features coarseness, contrast and directionality, see [20]

4. Color and edge directivity descriptor, CEDD, see [5] and Fuzzy color and texture histogram, FCTH, see [6], which are joint histograms combining fuzzy color and edge histograms, as well as the JCD feature, which combines both descriptor to a joint histogram [7].

5. Auto color correlation feature defined by Huang et al. [11]

6. Pyramid Histogram of Oriented Gradients (PHOG) as defined by Bosch et al. [2].

For all global features in LIRE parameters have been set to default values, that led to good results on the SIMPLIcity data set [22] and the UCID data set [18].

For local features LIRE integrates the jopensurf[6] library implementing the Speeded Up Robust Features [1] and code from the JavaSIFT project[7] implementing the scale-invariant feature transform approach [14]. An included multi-threaded k-means implementation and indexing abstraction routines allow for using the *bag of visual words* search approach [19].

## 2.2 Indexing and Search

For each of the implemented image features LIRE defines an optimized, byte-array based descriptor. The CEDD feature for instance is defined as a 144 dimensional double vector. Still due to quantization it's possible to put two values into one byte. Moreover, trailing zero are omitted resulting in a descriptor size significantly smaller than $s144/2 = 77$ bytes. Tests on the data set associated with LIRE have shown that that omission of trailing zeros leads to a further reduction of 7.2% in size on average.

Locality Sensitve Hashing [8] has been implemented for LIRE in two different ways. (i) Random projections allows nearest neighbor search in euclidean spaces, while (ii) bit sampling [17] allows for approximation of search with the cosine coefficient.

[6]see `https://code.google.com/p/jopensurf/`

[7]see `http://fly.mpi-cbg.de/~saalfeld/Projects/javasift.html`

**Table 1: Tests on the Wang SIMPLIcity data set [13]. Best value per column in bold face.**

| Feature | map | p@10 | er |
|---|---|---|---|
| CEDD | 0.513 | 0.706 | 0.193 |
| Color Correlogram | 0.498 | **0.740** | **0.159** |
| Color Layout | 0.439 | 0.612 | 0.303 |
| Edge Histogram | 0.333 | 0.500 | 0.401 |
| FCTH | 0.499 | 0.703 | 0.207 |
| JCD | **0.520** | 0.730 | 0.183 |
| Joint Histogram | 0.449 | 0.689 | 0.197 |
| OpponentHistogram | 0.450 | 0.635 | 0.270 |
| PHOG | 0.352 | 0.535 | 0.368 |
| RGB Color Histogram | 0.450 | 0.704 | 0.191 |
| Scalable Color | 0.305 | 0.470 | 0.464 |

# 3. USAGE

As already indicated indexing can be done utilizing available system resources in a few lines of code. Investigating runtime performance for the MIRFlickr [12] data set for extracting the global features Color Layout, Edge Histogram, JCD, Joint Histogram, Opponent Histogram, PHOG, and RGB Color Histogram and hashing the descriptors with LSH bit sampling shows that extraction takes about 44 ms per image on a current system[8]. Interpolating runtime the whole MIRFlickr data set can be indexed in under 13 hours. Approximate search using the hashes and re-ranking of the results returned by Lucene takes well below 100 ms running in a local Java VM. However, in the LIRE web demo[9] a search on 300,000 images from the MIRFlickr data set takes roughly 150 ms due to current configuration of the Java servlet container (the search is simply called within a Java servlet).

# 4. PERFORMANCE

The full source code of the evaluated features is available in the LIRE SVN, so results are reproducible. Test results for the Wang SIMPLIcity data [13] and the UCID data set [18] are given in Table 1 and Table 2. The benchmarking code is also part of LIRE and can be found in the classes *TestUCID* and *TestWANG*[10]. The tables give the performance in terms of mean average precision (map), precision at ten (p@10), and error rate (er). Note that the precision at ten is considerably lower with the UCID data set as the ground truth only features queries with very few – often only one – result. As can be seen easily for both test data sets CEDD, FCTH, JCD, and especially the Color Correlogram perform well. Still, all other features have their uses. Figure 1 for instance shows how well the PHOG feature can identify flower shapes. Figure 2 provides anecdotal evidence that the MPEG-7 Edge Histogram can be employed for finding logos having a similar overall structure.

# 5. DEMOS

Currently there are two different demos available for LIRE. One is the LireDemo GUI, which allows for indexing and

[8]an Intel Core i7, 16GB RAM, SSD, Windows 7, 64bit JVM running 8 threads

[9]see `http://demo-itec.uni-klu.ac.at/liredemo/`

[10]both to be found in *net.semanticmetadata.lire.benchmarking*, SVN rev. 88

Table 2: Tests on the UCID data set [18]. Best value per column in bold face.

| Feature | map | p@10 | er |
|---|---|---|---|
| CEDD | 0.431 | 0.420 | 0.553 |
| Color Correlogram | **0.586** | **0.480** | **0.370** |
| Color Layout | 0.277 | 0.285 | 0.679 |
| Edge Histogram | 0.180 | 0.202 | 0.813 |
| FCTH | 0.447 | 0.415 | 0.531 |
| JCD | 0.470 | 0.435 | 0.508 |
| Joint Histogram | 0.348 | 0.313 | 0.603 |
| Opponent Histogram | 0.319 | 0.309 | 0.649 |
| PHOG | 0.238 | 0.238 | 0.714 |
| RGB Color Histogram | 0.403 | 0.358 | 0.550 |
| Scalable Color | 0.172 | 0.183 | 0.840 |

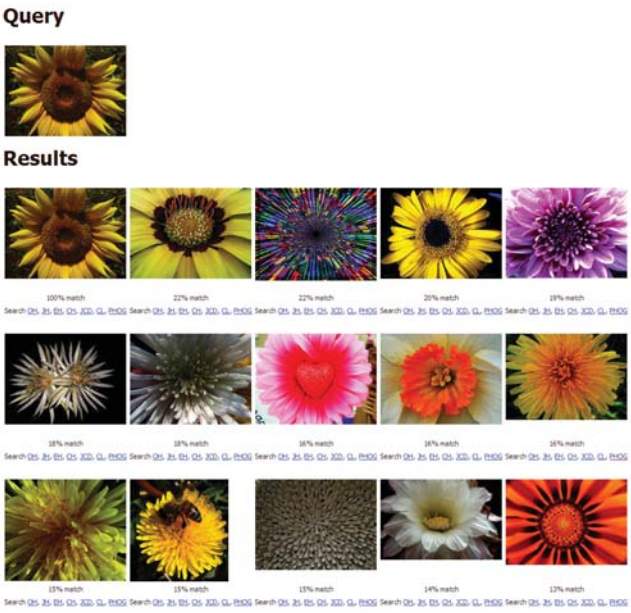



**Figure 1: Results for a query-by-example search in the MIRFlickr data set [12] using the PHOG feature [2] taken from the LIRE web demo.**

searching for arbitrary images, as seen in Figure 3. The second one, the LIRE web demo as shown in Figure 4, has been implemented recently and allows for searching in the first 300,000 images of the MIRFlickr data set based on selected features.

## 6. CONCLUSIONS

LIRE has been around for some time now. It started as a small side project and gradually became bigger and more important. It has been – and still is – employed in academia (i.e. as recommended library for imageCLEF [16]), in teaching, and industry (i.e. in the products of mediamid[11] or in the upcoming WIPO trademark search[12]).

LIRE provides a well balanced trade-off between (i) simplicity for the developer using the library and (ii) control over parameters at indexing and search time. With approximate nearest neighbor search based on locality sensitive hashing, retrieval times for data sets with more than 1.5 million images smaller than 70 ms have been observed.

LIRE is actively developed and available at `https://code.google.com/p/lire/`.

[11]see `http://www.mediamid.com/`
[12]see `http://www.wipo.int/`

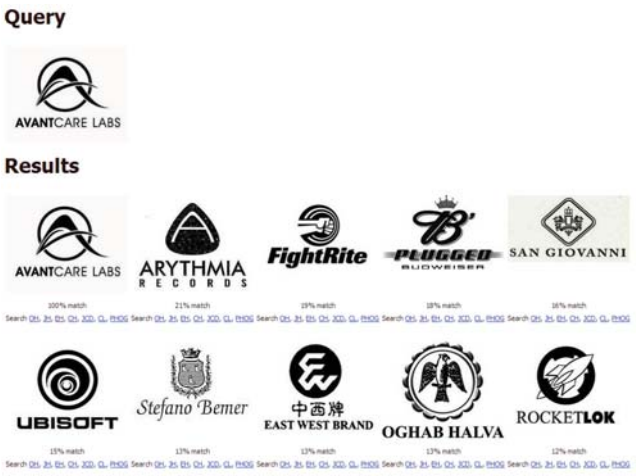



**Figure 2: Results for a query-by-example search in a trademark logo image data set using the MPEG-7 Edge Histogram feature.**

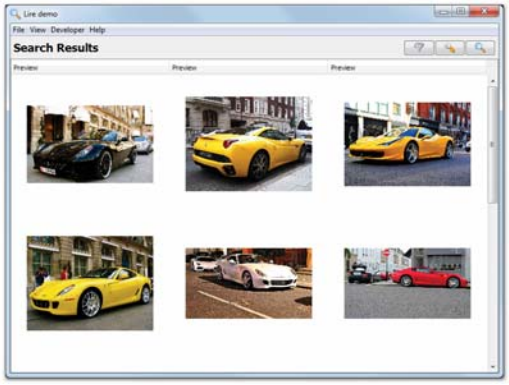



**Figure 3: LireDemo user interface screen shot showing the first two rows of results.**

## Acknowledgments

I'd like to thank at least some of the numerous people having contributed code and ideas to Lire or having published their work as open source allowing integration into LIRE: Anna-Maria Pasterk, Arthur Li, Arthur Pitman, Bastian Hösch, Benjamin Sznajder, Christian Penz, Christine Keim, Christoph Kofler, Dan Hanley, Daniel Pötzinger, Fabrizio Falchi, Franz Graf, Giuseppe Amato, James Charters, Janine Lachner, Katharina Tomanec, Lukas Esterle, Manuel Oraze, Marian Kogler, Marko Keuschnig, Michael Riegler, Rodrigo Carvalho Rezende, Roman Divotkey, Roman Kern, Savvas Chatzichristofis and Sandeep Gupta.

This work was supported by Lakeside Labs GmbH, Klagenfurt, Austria and funding from the European Regional Development Fund and the Carinthian Economic Promotion Fund (KWF) under grant KWF-20214 22573 33955.

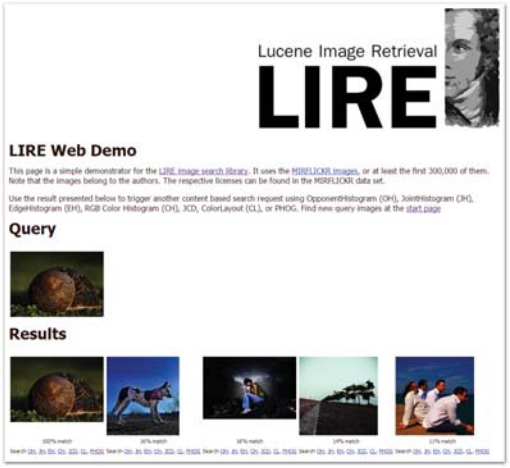



**Figure 4: LIRE web demo user interface screen shot showing a query and the first row of results.**

## 7. REFERENCES


[1] H. Bay, T. Tuytelaars, and L. Van Gool. Surf: Speeded up robust features. In *Computer Vision - ECCV 2006*, volume 3951 of *Lecture Notes in Computer Science*, pages 404–417. Springer, 2006.

[2] A. Bosch, A. Zisserman, and X. Munoz. Representing shape with a spatial pyramid kernel. In *Proceedings of the 6th ACM international conference on Image and video retrieval*, CIVR '07, pages 401–408, New York, NY, USA, 2007. ACM.

[3] G. Bradski and A. Kaehler. *Learning OpenCV: Computer vision with the OpenCV library*. O'Reilly Media, Incorporated, 2008.

[4] S.-F. Chang, T. Sikora, and A. Puri. Overview of the MPEG-7 standard. *IEEE Transactions on Circuits and Systems for Video Technology*, 11(6):688–695, June 2001.

[5] S. A. Chatzichristofis and Y. S. Boutalis. CEDD: Color and edge directivity descriptor. a compact descriptor for image indexing and retrieval. In A. Gasteratos, M. Vincze, and J. Tsotsos, editors, *Proceedings of the 6th International Conference on Computer Vision Systems, ICVS 2008*, volume 5008 of *LNCS*, pages 312–322, Santorini, Greece, May 2008. Springer.

[6] S. A. Chatzichristofis and Y. S. Boutalis. FCTH: Fuzzy color and texture histogram a low level feature for accurate image retrieval. In *Proceedings of the 9th International Workshop on Image Analysis for Multimedia Interactive Services, WIAMIS 2008*, pages 191–196, Klagenfurt, Austria, May 2008. IEEE.

[7] S. A. Chatzichristofis, Y. S. Boutalis, and M. Lux. Selection of the proper compact composite descriptor for improving content based image retrieval. In *The Sixth IASTED International Conference on Signal Processing, Pattern Recognition and Applications SPPRA 2009*, 2009.

[8] M. Datar, N. Immorlica, P. Indyk, and V. S. Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. In *Proceedings of the twentieth annual symposium on Computational geometry*, pages 253–262. ACM, 2004.

[9] J. S. Hare, S. Samangooei, and D. P. Dupplaw. OpenIMAJ and ImageTerrier: Java libraries and tools for scalable multimedia analysis and indexing of images. In *Proceedings of the 19th ACM international conference on Multimedia*, MM '11, pages 691–694, New York, NY, USA, 2011. ACM.

[10] E. Hatcher, O. Gospodnetic, and M. McCandless. Lucene in action, 2004.

[11] J. Huang, S. R. Kumar, M. Mitra, W.-J. Zhu, and R. Zabih. Image indexing using color correlograms. In *Proceedings of the 1997 Conference on Computer Vision and Pattern Recognition, CVPR '97*, volume 00, pages 762–768, San Juan, Puerto Rico, June 1997. IEEE.

[12] M. J. Huiskes and M. S. Lew. The mir flickr retrieval evaluation. In *Proceedings of the 1st ACM international conference on Multimedia information retrieval*, pages 39–43. ACM, 2008.

[13] J. Li and J. Z. Wang. Automatic linguistic indexing of pictures by a statistical modeling approach. *IEEE Trans. Pattern Anal. Mach. Intell.*, 25:1075–1088, September 2003.

[14] D. Lowe. Object recognition from local scale-invariant features. In *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, volume 2, pages 1150 –1157 vol.2, 1999.

[15] M. Lux and O. Marques. Visual information retrieval using java and lire. *Synthesis Lectures on Information Concepts, Retrieval, and Services*, 5(1):1–112, 2013.

[16] H. Müller, P. Clough, T. Deselaers, and B. Caputo, editors. *ImageCLEF*. Springer, 2010.

[17] S. Rüger. Multimedia information retrieval. *Synthesis Lectures on Information Concepts, Retrieval, and Services*, 1(1):1–171, 2009.

[18] G. Schaefer and M. Stich. Ucid: an uncompressed color image database. In *Electronic Imaging 2004*, pages 472–480. International Society for Optics and Photonics, 2003.

[19] J. Sivic and A. Zisserman. Video google: A text retrieval approach to object matching in videos. *Computer Vision, IEEE International Conference on*, 2:1470, 2003.

[20] H. Tamura, S. Mori, and T. Yamawaki. Textural features corresponding to visual perception. *IEEE Transactions on Systems, Man, and Cybernetics*, 8(6):460–472, June 1978.

[21] K. E. van de Sande, T. Gevers, and C. G. Snoek. Evaluating color descriptors for object and scene recognition. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 32(9):1582–1596, 2010.

[22] J. Z. Wang, J. Li, and G. Wiederhold. Simplicity: Semantics-sensitive integrated matching for picture libraries. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 23(9):947–963, 2001.