Blazingly Fast Image Copyright Enforcement

Herwig Lejsek, Friðrik Ásmundsson, Björn Thór Jónsson, Laurent Amsaleg

Blazingly Fast Image Copyright Enforcement

Herwig Lejsek, Fridrik H. Ásmundsson, Björn Thór Jónsson
Reykjavík University
Ofanleiti 2
IS-103 Reykjavík
Iceland
{herwig, fridrik01, bjorn}@ru.is

Laurent Amsaleg
IRISA–CNRS
Campus de Beaulieu
35042 Rennes
France
laurent.amsaleg@irisa.fr

ABSTRACT

Many photo agencies use the web to sell access to their image collections. Despite significant security measures, images may be stolen and distributed, making it necessary to detect copyright violations. This demonstration paper describes a content-based system for large-scale automatic copyright enforcement. The paper briefly describes the image description, indexing and retrieval algorithm that lie at the heart of the system. It also describes our proposed demonstration, which is a realistic scenario of copyright violations of a large image collection.

Categories and Subject Descriptors: H.2.4 [Database Systems]: Systems—*Multimedia Databases*

General Terms: Algorithms, Performance

Keywords: NV-tree, copyright protection

1. INTRODUCTION

Today, many photo agencies expose their collections on the web for the purpose of selling access to the images. Enforcing intellectual property rights and fighting against copyright violations is particularly important for these agencies as the images are a key source of revenue. With the proliferation of high-speed internet access, however, piracy of multimedia data has developed into a major problem.

Although significant security measures are typically employed to prevent violations, skilled hackers may find their way into the web sites, making it necessary to detect copyright violations. Today, photo agencies employ people to examine important newspapers, magazines, and web sites to look for pictures that might originate from their collections. Because of its manual nature, and because an ever-increasing number of images must be controlled, the current copyright enforcement process is very tedious.

In contrast, an automatic system enforcing property rights reduces the cost of the enforcement and improves the violation detection rate. This demonstration paper describes a content-based back-end system for copyright enforcement. The paper briefly describes the image description, indexing and retrieval algorithms that lie at the heart of the system. It also describes a realistic scenario of copyright violations that will be illustrated by the demonstration.

MM'06, October 23–27, 2006, Santa Barbara, California, USA.
ACM 1-59593-447-2/06/0010.

2. COPYRIGHT ENFORCEMENT REQUIREMENTS

A copyright enforcement system processes a stream of incoming images, that are retrieved from the web or scanned and entered by users (the process of obtaining images is out of the scope of our work). The system searches its collection for matches and raises alarms when it believes it has found images for which the publisher is not registered as having purchased that image. These alarms must then be processed by human experts to check whether there is a real violation of copyrights. To streamline this process, the system must meet the following requirements.

- First, it must be *robust* to image modifications, as images are typically modified to make the piracy less obvious, e.g., using line or column removal, or image rotations.
- Second, the system must also be *effective*. For detection of large scale piracy, it is not necessary to have 100% detection rate, but the system should return as few false alarms as possible without missing many violations.
- Third, the system must be very *efficient*, as the time needed to check a particular image must be small. The system must also be *scalable* as photo agencies have very large image collections to protect, which very many images must be checked against.
- Finally, the system must be *dynamic*. The most recent images typically have high commercial value and are likely to attract copyright violators. The system must therefore handle frequent and large updates to the image collection.

3. SYSTEM OVERVIEW

In this section, we describe a system which satisfies the requirements above. First, we briefly present the *robust* and *effective* descriptors used to describe images. Then we describe the new NV-tree (Nearest Vector tree) index, the *efficient* and *dynamic* data structure central to our system.[1] Finally, we describe our *scalable* distributed system.

[1]The NV-tree is a descendant of the PvS-index used in [1], which yields better query performance and result quality. A patent application is pending.

3.1 The Eff^2 Descriptors

The Eff^2 descriptors we use in this demonstration are based on 72-dimensional histograms of gradients computed around points of interests determined using the Difference of Gaussian scales [1]. To retrieve the images that are similar to a query image, a k nearest neighbor search is run for each local descriptor computed on the query image. Each nearest neighbor "votes" for the image it is associated with and the most similar images are found according to their number of votes. Overall, the Eff^2 descriptors allow finding similar images despite modifications that include rotations, JPEG-compression, various filtering attacks, severe croppings, change of illumination, rescaling, occlusions and affine transforms. While they are quite similar in spirit to the SIFT scheme devised by Lowe [2], the Eff^2 descriptors have shown better recognition power for large scale retrieval [1].

3.2 The NV-tree Index

The NV-tree indices we use have been designed to efficiently return approximate k-nearest neighbors of high-dimensional descriptors. When the construction of an NV-tree index starts, all descriptors are considered to be part of a single temporary partition. The NV-tree indexing scheme is then based on a repeated combination of projecting and partitioning descriptors.

Descriptors belonging to one partition are first projected onto a single line through the high-dimensional space. The line is selected by determining the line that best distributes a sample of descriptors (chosen from a pre-existing pool of random lines for various performance reasons).

The projected values are then partitioned into disjunct chunks. The chunks vary in size, depending on the distribution of data, with large chunks covering dense areas of the space and small chunks covering sparse areas. Then, the high-dimensional descriptors are classified into a new set of temporary sub-partitions according to which chunk they have been projected to. Finally, overlapping sub-partitions are created for redundant coverage of partition borders.

This process of projecting and partitioning is repeated for all the new partitions. It stops when the number of descriptors in a sub-partition reaches a specified lower limit designed to be disk I/O friendly. At that time, the descriptors are written to a data file on disk on a sub-partition basis. Note that the NV-tree is typically an unbalanced data structure, depending on the data distribution.

The data structure keeps track of, at each level, which line was used for the projection and the cut points for all the partitions at that level. Within each leaf partition, descriptors are sorted according to the rank of their projected value. Overall, an NV-tree is a hierarchy of partition cut points and only the leaves contain references to descriptors.

3.3 NV-tree Operations

During query processing, the query descriptor traverses the NV-tree. At each level of the tree, the query descriptor is projected to the line associated with the current node. Using the cut points for that level, the search is then directed to the appropriate sub-level. This process of projection and choosing the right sub-level is repeated until the search reaches a leaf partition.

The leaf partition is fetched into memory and the search returns the k descriptors that have their rank closest to the final projection of the query point. More accurate results may be obtained if more than one NV-tree is used. In this case, results returned by individual NV-trees must be merged, e.g., using median rank aggregation [3].

When images are inserted, the new descriptors are stored in a temporary holding area which is organized by partitions and included in the search process. When the holding area fills, it is written to the index. On overflow, partitions are split into new sub-partitions and the tree structure is updated accordingly. Note that partitions fetched into memory have a fixed and strictly limited size, guaranteeing a fixed query processing time regardless of the size of the image collection. Larger collections need deeper NV-trees but the tree traversal cost is negligible.

To be usable in practice in high-throughput situations, NV-tree indices must be used in a distributed system [4]. The architecture of this system is composed of a set of workers, a coordinator and a server. Each worker stores a replica of all existing NV-trees. The server receives the query image, computes its description and sends it to the coordinator. The coordinator then assigns specific query descriptors to specific workers. The assignment policy takes into account which partitions each worker is likely to have currently in memory and also balances the load over all workers. Load balancing copes on-the-fly with worker's load variations and with the addition and removal of workers.

4. DEMONSTRATION OVERVIEW

The goal of the demonstration is to show that the system described above satisfies all the requirements for an image copyright protection system.

The image collection used in the demonstration consists of 287,268 high-quality news images, resulting in 169,159,548 descriptors of 72-dimensions. During the conference, we will take many "news" photos such that we are constantly updating our image collection. We will demonstrate, by "stealing" and modifying new and old images, that the system practically always finds a match to copyright violations.

Overall, this demonstration will show that our system offers robust and effective descriptions, dynamic storage and blazingly fast retrieval. More importantly, it will show that these desirable properties hold even at a very large scale.

5. ACKNOWLEDGEMENTS

We thank Morgunbladid for the use of their images, and Stefan Hackl for his help with the interface. This work was partly supported by Ranns Grant 60036021, INRIA Eff^2 Associate Teams grant, and GIDE Jules Verne travel grant.

6. REFERENCES

[1] H. Lejsek, F. H. Ásmundsson, B. Th. Jónsson, and L. Amsaleg. Scalability of local image descriptors: A comparative study. In *Proceedings of ACM Multimedia*, Santa Barbara, CA, USA, 2006.

[2] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

[3] R. Fagin, R. Kumar, and D. Sivakumar. Efficient similarity search and classification via rank aggregation. In *Proceedings of ACM SIGMOD*, San Diego, CA, USA, 2003.

[4] F. H. smundsson. NV-Network: A distributed architecture for high throughput image retrieval. Master's thesis, Reykjavk University, August 2006.