# Large scale visual-based event matching

Riadh Trad, Alexis Joly, Nozha Boujemaa

# Large scale visual-based event matching

Mohamed Riadh Trad
INRIA Paris-Rocquencourt
Domaine de Voluceau, BP 105, 78153
Le Chesnay Cedex, France
Riadh.Trad@inria.fr

Alexis Joly INRIA
Paris-Rocquencourt
Domaine de Voluceau, BP 105, 78153
Le Chesnay Cedex, France
Alexis.Joly@inria.fr

Nozha Boujemaa INRIA
Paris-Rocquencourt
Domaine de Voluceau, BP 105, 78153
Le Chesnay Cedex, France
Nozha.Boujemaa@inria.fr

## ABSTRACT

Organizing media according to real-life events is attracting interest in the multimedia community. Event-centric indexing approaches are very promising for discovering more complex relationships between data. In this paper we introduce a new visual-based method for retrieving events in photo collections, typically in the context of User Generated Contents. Given a query event record, represented by a set of photos, our method aims to retrieve other records of the same event, typically generated by distinct users. Similarly to what is done in state-of-the-art object retrieval systems, we propose a two-stage strategy combining an efficient visual indexing model with a spatiotemporal verification re-ranking stage to improve query performance. For efficiency and scalability concerns, we implemented the proposed method according to the MapReduce programming model using Multi-Probe Locality Sensitive Hashing. Experiments were conducted on LastFM-Flickr dataset for distinct scenarios, including event retrieval, automatic annotation and tags suggestion. As one result, our method is able to suggest the correct event tag over 5 suggestions with a 72% success rate.

## 1. INTRODUCTION

An event can be described as an action that occurs at a specific time in a specific place. This notion is potentially useful for connecting individual facts and discovering complex relationships. It is worth noting that photos in User Generated Content (UGC) websites, as well as in personal collections, are often organized into events. Indeed, users are usually more likely to upload or gather pictures related to the same event, such as a given holiday trip, a music concert, a wedding, etc. This applies to professional contents such as journalism or historical data that are even more systematically organized according to hierarchies of events.
Defining new methods for organizing, searching and browsing media according to real-life events is therefore gaining interest in the multimedia community [13, 6]. In this paper, we address the problem of matching distinct records of the same event in picture datasets, typically in UGC's photo collections. Given a query event record represented by a set of photos, our method aims to retrieve other records of the same event, notably those generated by other actors or witnesses of the same real-world event. An illustration of two matching event records is presented in Figure 1. It shows how a small subset of visually similar and temporally coherent pictures might be used to match the two records, even if they include other distinct pictures covering different aspects of the event. Application scenarios related to such a retrieval paradigm are numerous. By simply uploading their own record of an event users might, for example, access to the community of other participants. They can then *revive* the event by browsing or collecting new data complementary to their own view of the event. If some previous event's records had already been uploaded and annotated, the system might also automatically annotate a new record or suggest some relevant tags. The proposed method might also have nice applications in the context of citizen journalism. Automatically detecting the fact that a large number of amateur users did indeed record data about the same event would be very helpful for professional journalists in order to cover breaking news. Finally, tracking events across different media has a big potential for historians, sociologists, politicians, etc.
Of course, in such scenarios, time and geographic information provided with the contents have a major role to play. Our claim is that using visual content as complementary information might overcome several limitations of approaches that rely only on metadata. First of all, distinct records of the same event are not necessarily located at the same place or can be recorded at different times. Some events might, for example, have wide spatial and temporal coverage such as a volcano eruption or an eclipse, so that geocoordinates and time stamps might be not sufficiently discriminant. This lack of discrimination can be problematic even for precisely located events, typically in crowded environments such as train stations, malls or tourist locations. In such environments, many records might be produced at the same time and place while being related to very distinct real-world events. Furthermore, in a wider meaning of the *event* concept, several instances of an event might be recorded at different times, e.g. periodical events or events such as "a trip to Egypt" illustrated in Figure 2. Finally, location and time information is not always available or might be noisy. The Flickr dataset used in the experiments reported in this paper notably does not contain any geographic information and contains noisy time information (as discussed in section


*ICMR'11* April 17-20, Trento, Italy

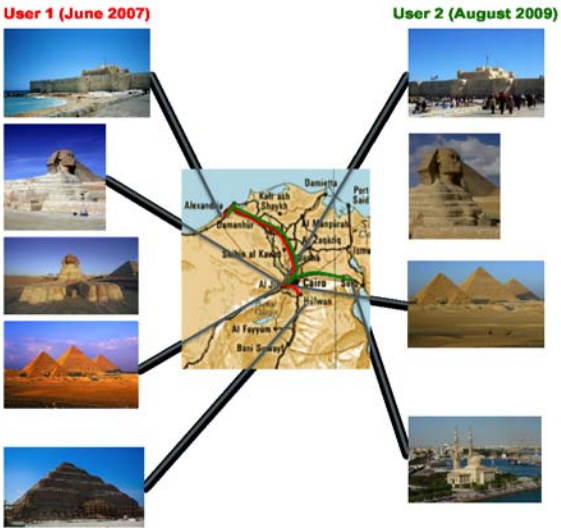


**Figure 2: Two records of the event "a trip in Egypt"**

5).
The retrieval method we propose makes use of both visual content and contextual meta-data. Visual content is used in the first stage to detect potential matches whereas geo-temporal metadata are used in the second stage to re-rank the results and therefore estimate the spatiotemporal offset between records. It is important to notice that our method allows spatiotemporally coherent records to be retrieved even if they were not produced at the same time and place (such as the examples discussed above).

## 2. RELATED WORKS

To the best of our knowledge, this work is the first to deal with visual based event matching in picture collections. Some recent works are more generally concerned with event models and ontology-based event retrieval, such as [13] or [6]. In the latter, a joint content-event model is proposed to facilitate the automatic enrichment of event elements with information extracted by automatic analysis of multimedia content segments. The automatic analysis of an event's visual contents is not however addressed in this work. In a survey paper on event mining in multimedia streams, Xie et al. [14] give more insights on how visual features might be used to contribute to multi-cue events detection. But here again, they just mention some high level principles and do not experiment any realization. The only work we found that really uses and experiments the use of visual content in the context of event based indexing is that by [11]. In this work the authors present a multimedia mining framework allowing the discovery of picture clusters from multiple cues including visual content, text content and metadata. The produced clusters can then be classified in either *object* or *event* type and further annotated by linking their content to wikipedia pages. Retrieving different instances of an event is thus not really addressed.

We also mention that a large number of studies focused on detecting or recognizing events in videos [12, 15, 8, 16] notably human actions [12], sports events [15] or video-surveillance events [8, 16]. It might be interesting to build upon some aspects of these methods in the context of image-based event records but they are clearly not directly applicable to our context. Most of them involve video specific algorithms such as tracking, space-time visual features, etc.

Finally, our work is, to some extent, related to object retrieval in picture collections. Our method is indeed very similar to state-of-the-art large-scale object retrieval methods combining efficient bag-of-words or indexing models with a spatial verification re-ranking stage to improve query performance [10, 7]. We might give the following analogy: images are replaced by event records (picture sets), local visual features are replaced by global visual features describing each picture of a record, spatial positions of the local features are replaced by the geo-coordinates and time stamps of the pictures. Matching spatially and temporally coherent event records is finally equivalent to retrieving geometrically consistent visual objects.

## 3. VISUAL BASED EVENT MATCHING

We first describe the proposed method in the general context of event records composed of a set of geo-tagged and time coded pictures. We further restrict ourselves to time coded only pictures since our experimental dataset did not include geo-tags.
We consider a set of $N$ event records $E_i$, each record being composed of $N_i$ pictures $I_j^i$ captured from the same real-world event. Each picture is associated with a geo-coordinate $\mathbf{x}_j^i$ and a time stamp $t_j^i$ resulting in a final geo-temporal coordinate vector $\mathbf{P}_j^i = (\mathbf{x}_j^i, t_j^i)$. The visual content of each image $I_j^i$ is described by a visual feature vector $\mathbf{F}_j^i \in \mathbb{R}^d$ associated with a metric $d : \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}$.
Now let $E_q$ be a query event record represented by $N_q$ pictures, with associated visual features $\mathbf{F}_j^q$ and geo-temporal metadata $\mathbf{P}_j^q$. Our retrieval method works as follows:

**STEP 1 - Visual Matching:** Each query image feature $\mathbf{F}_j^q$ is matched to the full features dataset thanks to an efficient similarity search technique (see section 4). It typically returns the approximate $K$-nearest neighbors according to the used metric $d$ (i.e the $K$ most similar pictures). When multiple matches occur for a given query image feature and a given retrieved record, we only keep the best match according to the feature distance. The visual matching step finally returns a set of candidate event records $E_i$, each being associated with $M_i^q$ picture matches of the form $(I_m^q, I_m^i)$.

**STEP 2 - Stop List:** Only the retrieved records with at least two image matches are kept for the next step, i.e

$$\{E_i \mid M_i^q \geq 2\}_{1 \leq i \leq N}$$

**STEP 3 - Geo-temporal consistency:** For each remaining record, we compute a geo-temporal consistency score by estimating a translation model between the query record and the retrieved ones. The resulting scores $S_q(E_i)$ are used to produce the final records ranking returned for query $E_q$. The translation model estimation is based on a robust re-

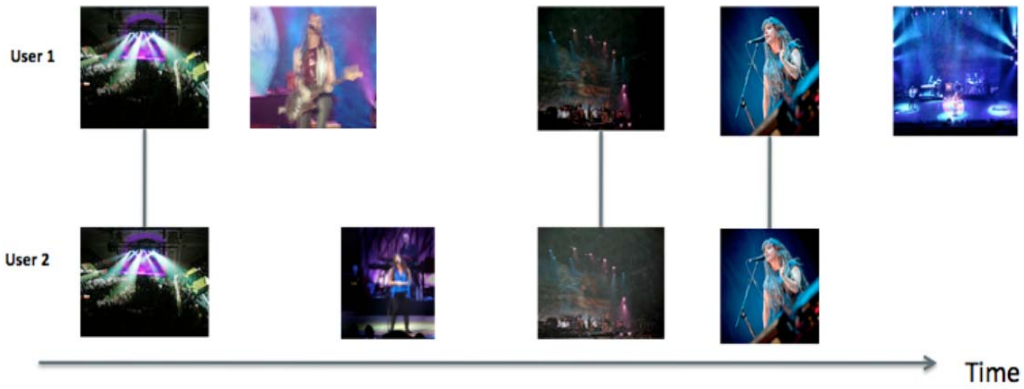


**Figure 1: Two events records of Alanis Morissette concert**

gression and can be expressed as:

$$\widehat{\Delta}(E_q, E_i) = \arg\min_{\Delta} \sum_{m=1}^{M_i^q} \rho_\theta \left( \mathbf{P}_m^q - (\mathbf{P}_m^i + \Delta) \right) \quad (1)$$

where $\mathbf{P}_m^q$ and $\mathbf{P}_m^i$ are the geo-temporal coordinates of the $m$-th match $(I_m^q, I_m^i)$. The cost function $\rho_\theta$ is typically a robust $M$-estimator allowing outliers to be rejected with a tolerance $\theta$ (in our experiments we used Tukey's robust estimator). The estimated translation parameter $\hat{\Delta}$ should be understood as the spatial and temporal offset required to register the query event record $E_q$ with the retrieved event record $E_i$. Once this parameter has been estimated, the final score of an event $E_i$ is finally computed by counting the number of inliers, i.e the number of visual matches that respect the estimated translation model:

$$S_q(E_i) = \sum_{m=1}^{M_i^q} \left( \left\| \mathbf{P}_m^q - (\mathbf{P}_m^i + \widehat{\Delta}) \right\| \leq \theta \right) \quad (2)$$

where $\theta$ is a tolerance error parameter, typically the same as the one used during the estimation phase. In practice, we use a smooth counting operator to get a better dynamic on resulting scores. When we restrict ourselves to temporal metadata (as was done in the experiments), equation 1 can be simplified to:

$$\hat{\delta}(E_q, E_i) = \arg\min_{\delta} \sum_{m=1}^{M_i^q} \rho_\theta \left( t_m^q - (t_m^i + \delta) \right) \quad (3)$$

where $\hat{\delta}$ represents the estimated temporal offset between $E_q$ and $E_i$ and $\theta$ is now a temporal tolerance error whose value is discussed in the experiments. Since $\delta$ is a single mono-dimensional parameter to be estimated, equation 3 can be resolved efficiently by a brut force approach testing all possible solutions $\delta$.
Final scores then become:

$$S_q(E_i) = \sum_{m=1}^{M_i^q} \left( \left| t_m^q - (t_m^i + \hat{\delta}) \right| \leq \theta \right) \quad (4)$$

**STEP 4 - Prior constraints:** Depending on the application context, strong effectiveness improvements might be obtained by adding prior constraints on the tolerated values for $\widehat{\Delta}$. Rejecting events with too large a spatial and/or temporal offset from the query record is indeed a good way to reduce the probability of false alarms. In our experiments we study the impact of such a constraint on the estimated temporal offsets. Concretely, we reject from the result list all retrieved event records having an estimated offset above a given threshold $\delta_{max}$ (regardless the matching score $S_q(E_i)$).

# 4. ENABLING SCALABILITY

To allow fast visual matching in large picture datasets, we implemented a distributed similarity search framework based on Multi-Probe Locality Sensitive Hashing [9, 7] and the MapReduce [4] programming model.

## 4.1 Multi-Probe LSH

To process Nearest Neighbors search efficiently, we use an approximate similarity search structure, namely Multi-Probe Locality Sensitive Hashing (MP-LSH) [9, 7]. MP-LSH methods are built on the well-known LSH technique [3], but they intelligently probe multiple buckets that are likely to contain results. Such techniques have been proved to overcome the over-linear space cost drawback of common LSH while preserving a similar sub-linear time cost (with complexity $O(N^\lambda)$).
Now, let $\mathcal{F}$ be the dataset of all visual features $\mathbf{F} \in \mathbb{R}^d$ (i.e. the one extracted from the pictures of the N event records $E_i$). Each feature $\mathbf{F}$ is hashed with a hash function $g : \mathbb{R}^d \rightarrow \mathbb{Z}^k$ such that:

$$g(\mathbf{F}) = (h_1(\mathbf{F}), ..., h_k(\mathbf{F})) \quad (5)$$

where individual hash functions $h_j$ are drawn from a given locality sensitive hashing function family. In this work we used the following binary hash function family which is known to be sensitive to the inner product:

$$h(\mathbf{F}) = sgn(\mathbf{W}.\mathbf{F}) \quad (6)$$

where $\mathbf{W}$ is a random variable distributed according to $\mathcal{N}(0, \mathbf{I})$. The produced hash codes $\mathbf{g}_i = g(\mathbf{F}_i)$ are thus binary hash codes of size $k$.

At indexing time, each feature $\mathbf{F}_i$ is mapped into a single hash table $\mathbf{T}$ according to its hash code value $\mathbf{g}_i$. As a result, we obtain a hash table of $\mathbf{N}_b$ buckets where $\mathbf{N}_b \leq 2^k$.

At query time, the query vector $\mathbf{F}_q$ is also mapped onto the hash table $\mathbf{T}$ according to its hash code value $\mathbf{g}_q$. The

multi-probe algorithm then selects a set of $\mathbf{N}_p$ buckets $\{(\mathbf{b}_j)\}_{j=1..N_p}$ as candidates that may contain objects similar to the query according to :

$$d_h(\mathbf{g}_q, \mathbf{b}_j) < \delta_{MP} \quad (7)$$

where $\mathbf{d}_h$ is the hamming distance between two binary hash codes and $\delta_{MP}$ is the multi-probe parameter (i.e. a radius of hamming space).

A final step is then performed to filter the features contained in the selected buckets by computing their distance to the query and keeping the $K$ Nearest Neighbors.

## 4.2 The MapReduce framework

MapReduce is a programming model introduced by Google to support distributed batch processing on large data sets. A MapReduce job splits the input dataset into independent chunks which are processed by the *map* tasks in a parallel manner. The framework sorts the outputs of the maps, which are then input to the *reduce* tasks. Chunks are processed based on key/value pairs. The *map* function computes a set of intermediate key/value pairs and, for each intermediate key, the *reduce* function iterates through the values that are associated with that key and outputs 0 or more values. The *map* and *Reduce* tasks scheduling is performed by the framework. In a distributed configuration, the framework assigns jobs to the nodes as slots become available. The number of *map* and *reduce* slots as well as chunk size can be specified for each job, depending on the cluster size. With such a granularity, large data sets processing can be distributed efficiently on commodity clusters.

## 4.3 Multi-Probe LSH in the MapReduce framework

The hash table T in the MapReduce framework is stored in a text file where each line corresponds to one single bucket. Each bucket is represented by a $< key, value >$ pair:

$$< \mathbf{b}, ((id(\mathbf{F}_1), \mathbf{F}_1), (id(\mathbf{F}_2), \mathbf{F}_2), \dots ) > \quad (8)$$

where b is the hash code of the bucket.

In order to be processed by the MapReduce framework, the table $T$ has to be divided into a set of splits. The number of splits is deduced by the MapReduce framework according to a set of input parameters as the number of available slots and the minimal input split size which is related to the file system block size. However, in order to be entirely processed by a mapper, a bucket cannot spill over different splits.

Since MapReduce is mainly dedicated to batch processing, setting up tasks could be expensive due to process creation and data transfer. Therefore, our implementation processes multiple queries at a time, typically sets of pictures belonging to the same records.

The hash codes of all query features are computed and passed to the *map* instances to be executed on the different slots. The number of *map* instances is computed by the MapReduce framework according to the number of input splits.

Each *map* process iterates over its assigned input split and for each query selects the candidate buckets that are likely to contain similar features according to Equ.7. It then computes the distance to each feature within the selected buckets. For each visited feature $\mathbf{F}_i$, the *map*function outputs a $< key, value >$ pair of the form:

$$< id(\mathbf{F}_q), (dist(\mathbf{F}_q, \mathbf{F}_i), id(\mathbf{F}_i)) > \quad (9)$$

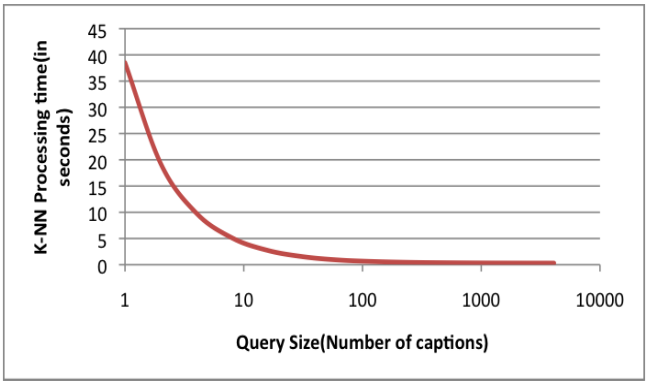


**Figure 3: Processing time per image according to query size**

where $id(\mathbf{F})$ denotes the picture identifier associated to feature F and $dist(\mathbf{F}_q, \mathbf{F}_i)$ the distance between $\mathbf{F}_q$ and $\mathbf{F}_i$. For each query identifier $id(\mathbf{F}_q)$ the *reduce* instance sorts the set of emitted values for all *map* instances and filters the K-nearest neighbors.

Figure 3 gives the average response time per K-NN search according to the total number of queries batched within the same MapReduce job. It shows that the MapReduce framework becomes profitable from about 50 grouped queries. The average response time becomes almost constant for more than 400 grouped queries. In our experiments, the number of images per event record range from about 5 to 200. That means that using the MapReduce framework is still reasonable for the online processing of a single event record.
Finally, many MapReduce implementations materialize the entire output of each *map* before it can be consumed by the *reducer* in order to ensure that all *maps* successfully completed their tasks. In [1], Condel et al. propose a modified MapReduce architecture that allows data to be pipelined between operators. This extends the MapReduce programming model beyond batch processing, and can reduce completion times while improving system utilization for batch jobs as well.

# 5. EXPERIMENTS

We evaluated our method on a *Flickr* image dataset using *last.fm* tags as real-world events ground truth. It was constructed from the corpus introduced by Troncy et al. [13] for the general evaluation of event-centric indexing approaches. This corpus mainly contains events and media descriptions and was originally created from three large public event directories (*last.fm*, *eventful* and *upcoming*). In our case, we only used it to define a set of Flickr images labeled with *last.fm* tags, i.e. unique identifiers of music events such as concerts, festivals, etc. The images themselves were not provided in the data and had to be crawled resulting in some missing images. Unfortunately, in this corpus, only a small fraction had geo-tags so that we evaluated our method using only temporal metadata. We used the EXIF *creation date* field of the pictures to generate the time metadata used in our method. Only about 50% of the crawled images had such a valid EXIF (others had empty or null date fields). In Table 1, we report the statistics on the original, crawled

and filtered dataset. To gather the pictures in relevant event records, we used both the *last.fm* identifier and the *Flickr* author field provided with each picture. An event record is then defined as the set of pictures by a given author having the same LastFM label. Our final dataset contains 41,294 event records related to 34,034 distinct LastFM events.

**Table 1: Test dataset Vs Original dataset**

| | Total | Crawled | Filtered |
|---|---|---|---|
| photos | 1 667 317 | 1637585 | 828902 |
| users | 23 060 | 22676 | 10257 |

## 5.1 Experimental settings

We used 6 global visual features to describe a picture's visual content (including HSV Histogram[5], Hough histogram[5], Fourier histogram[5], edge orientation histogram[5]). Each feature was $L2$-normalized and hashed into a 1024 bits hash code using the same hash function as the one used to construct the hash table (see Equ.6). The 6 hash codes were then concatenated into a single hash code of 6144 bits. We used the Hamming distance on these hash code as visual similarity.
From the full set of 41,294 event records in the dataset, the only queries we kept, were the records being tagged with *last.fm* events and having at least 7 records in the dataset. We finally get 172 query records $E_q$. This procedure was motivated by the fact that a very large fraction of events were represented by only one record and therefore not usable for experiments.

In all experiments, we used a leave-one-out evaluation procedure and measured performances with 2 evaluation metrics: Mean Average Precision (MAP) and Classification Rate (CR). MAP is used in most information retrieval evaluations and measures the ability of our method to retrieve all the records related to the same event as the query one. Classification rate is obtained by using our method as a nearest neighbors classifiers. The number of occurrences of retrieved events is computed from the top 10 returned records and we keep the event with the maximum score as the best prediction. It measures the ability of our method to automatically label some unknown query event record. We extend this measure to the case of multiple labels suggestion. In addition to the best retrieved event we also return the following events by decreasing scores (i.e decreasing number of occurrences found within the top-10 returned records). In this case, the success rate is measured by the percentage of query records where the correct event was retrieved among all suggested event tags. It measures the performance of our method in the context of tags suggestion rather than automatic annotation.
Finally, we used the Hadoop[1] MapReduce implementation on a 5-node cluster. Nodes are equipped with Intel Xeon X5560 CPUs as well as 48Gb of RAM.

## 5.2 Results

[1]http://hadoop.apache.org/mapreduce/

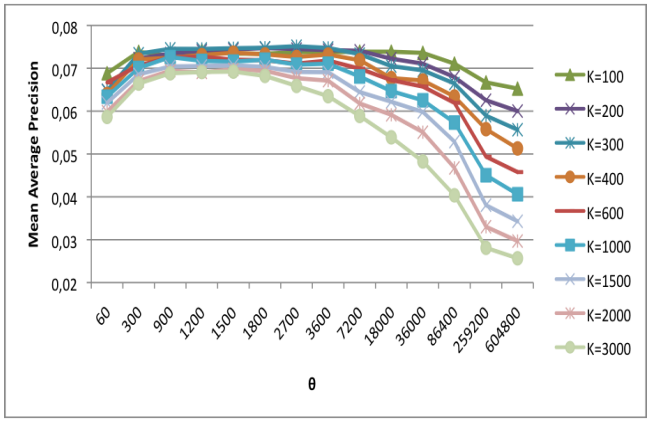


**Figure 4: Influence of temporal error to tolerance $\theta$**

### 5.2.1 *Parameters discussion*

In figure 4, we report the mean average precision for varying values of the $\theta$ parameter (Eq. 3) and different numbers of $K$-nearest neighbors used during the visual matching step. The results show that MAP values are at their optimal for $\theta \in [300, 1800]$ seconds. This optimal error tolerance value is coherent with the nature of the events in the *last.fm* corpus. Concert's picture records indeed usually range from one to several hours. On the other hand, above 5 minutes, real-world concert scenes are too much ambiguous to be discriminated by their visual content (or at least with the global visual features used in this study). In what follows, we fix $\theta$ to 1800 as an optimal value for visual matching.

We now study the impact of adding a prior constraint $\delta_{max}$ on the estimated temporal offsets $\hat{\delta}$. Most events in *last.fm* dataset being music concerts, it is unlikely that the temporal offset between two records reach high values. We therefore study the impact of rejecting all retrieved records having a temporal offset higher than $\delta_{max}$. Figure 5 displays the new MAP curves for varying values of $\delta_{max}$. It shows that the mean average precision can be consistently improved from about 0.08 without any constraint to 0.18. The optimal value for $\delta_{max}$ is about 86,400 seconds which is exactly 1 day. That means that the records of a single real-word event might have a temporal offset of up to 1 day. Our interpretation is that the EXIF *creation date* field might be noisy due to the different reference times of the used devices (users from different countries, etc.). It is worth noting that our method is by its very nature robust to such temporal offsets since we mainly consider temporal coherence rather than absolute time matching. On the other hand, rejecting records with temporal offsets higher than 1 day allows many visual false positives to be rejected.

Figure 6 displays the results of the same experiment but for the classification rate (using a 10-NN classifier on retrieved records) rather than the mean average precision. This evaluates the ability of our method to automatically annotate a query event record rather than its ability to retrieve all records in the dataset. Here again the optimal classification rates are obtained when $\delta_{max}$=1 day. Furthermore, we see that the classification rate always increases with the number $K$ of closest visual matches (returned for

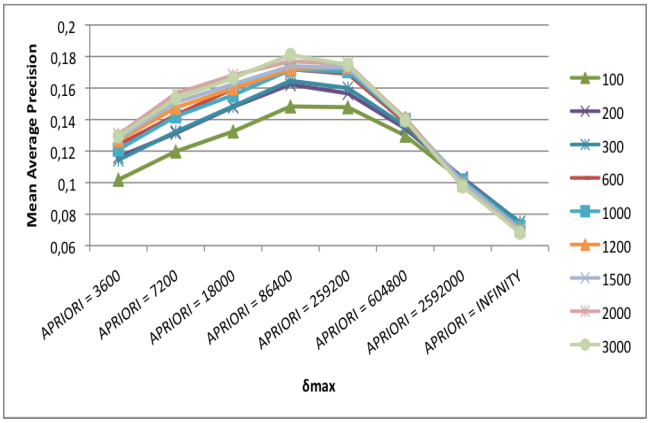


**Figure 5: Influence of temporal offset thresholding ($\delta_{max}$) on MAP**

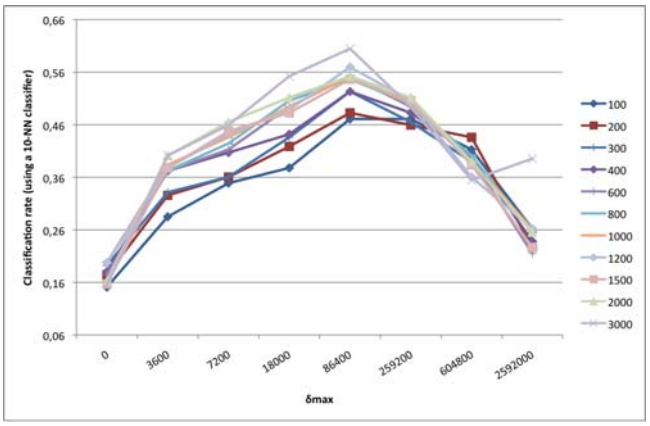


**Figure 6: Influence of temporal offset thresholding ($\delta_{max}$) classification rates**

each query image). The interpretation is that increasing $K$ improves recall without degrading precision too much thanks to the selectivity of our temporal consistency re-ranking step. We verified this from the results presented in this paper by studying the recall and the precision independently.

### 5.2.2 *Event suggestion in the MapReduce framework*

All the previous experiments were made using an exhaustive search for the k-NN search. In this section, we evaluate the performance of our full framework using MapReduce and the Multi-Probe LSH. As parameters, we used the optimal values discussed in the previous section (i.e. $\delta_{max}$=86.400, $K$=3000 and $\theta$=1800).

Table 2 displays the class rates using an exhaustive search as seen in the previous section as well as class rates using a Multi-Probe LSH-based similarity search for different values of $\delta_{MP}$.

As one might expect, all class rates values increase accordingly with the number of probes (i.e increasing $\delta_{MP}$ values) to surprisingly perform better than the exhaustive search for $\delta_{MP}$=8.

Overall, in the best case, our method is able to suggest

**Table 2: Suggestion rates**

| # of suggested events tags | 1 | 2 | 3 | 4 | 5 | 10 |
|---|---|---|---|---|---|---|
| Exhaustive | 0.60 | 0.66 | 0.69 | 0.71 | 0.72 | 0.73 |
| MP-Delta 0 | 0.39 | 0.48 | 0.50 | 0.51 | 0.52 | 0.54 |
| MP-Delta 1 | 0.45 | 0.55 | 0.57 | 0.58 | 0.59 | 0.63 |
| MP-Delta 2 | 0.48 | 0.59 | 0.61 | 0.65 | 0.66 | 0.69 |
| MP-Delta 4 | 0.51 | 0.61 | 0.63 | 0.67 | 0.67 | 0.70 |
| MP-Delta 8 | 0.61 | 0.67 | 0.70 | 0.72 | 0.72 | 0.74 |
| MP-Delta 16 | 0.59 | 0.66 | 0.69 | 0.71 | 0.72 | 0.73 |

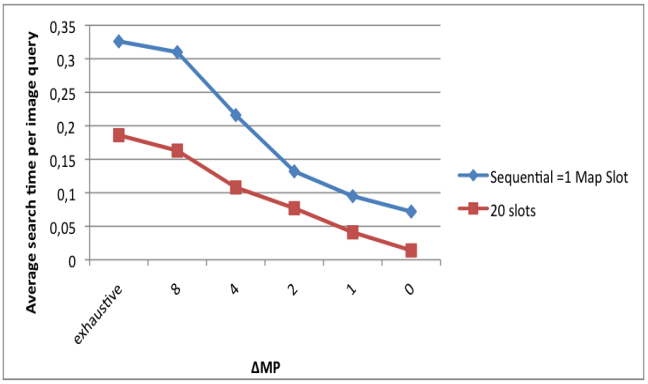


**Figure 7: K-NN search time per image ($k = 4000$)**

the correct event tag over 5 suggestions with a 72% success rate. Such performances are clearly acceptable from an application point of view.

Figure 7 displays the average search time per query for both distributed and centralized search. We compare the K-NN processing time per image for a centralized setting (number of *map* slots = 1) to the processing time in a distributed scheme (20 *map* slots available on the network) for both exact and approximate similarity search. Although the multi-probe might reduce the effectiveness down to 66%, it might also reduce the search time by a factor of 13.

## 6. CONCLUSION AND FUTURE WORK

The achieved performance gain is nonetheless still less than the performance gain obtained in usual centralized settings. First, in MapReduce approaches, probing multiple buckets generates more network overhead in addition to data transfers across the network. The second reason is due to bucket occupation. In fact, imbalanced buckets generate imbalanced map chunks leading to disproportionate map execution times. Such a problem is addressed in [2] and we plan to study such balancing methods in future work.

In this paper we presented a new visual-based method for retrieving events in photo collections, that might also be used for event tag suggestion or annotation. Our method proved to be robust to temporal offsets since we mainly rely on temporal coherence rather than absolute time matching. As one result, we are able to suggest the correct event tag with a success rate of at least 60% and even 72% if we allow multiple suggestions.

The proposed method is scalable, since it relies on efficient approximate similarity search techniques based on the MapReduce framework. We also investigated multi-probe techniques trading accuracy for efficiency, which might lead to a loss of 8.3% compared to a gain of 58.6%

Future work can focus on improving the suggestion rate as well as the efficiency of the approximate similarity search framework. The first issue can be addressed by including additional metadata during the re-ranking stage (notably geo-tags) and the use of more effective visual features. The second can be achieved through a better design of the hash functions to ensure a fair bucket occupation and therefore, balanced inputs for the *map* tasks.

We believe that the proposed method could have many other applications including other media event tracking or event mining in UGC's streams. Up to now, however we have found it difficult to collect relevant data for evaluation, and we work on that as well.

# 7. REFERENCES


[1] T. Condie, N. Conway, P. Alvaro, J. M. Hellerstein, K. Elmeleegy, and R. Sears. MapReduce Online. Technical Report UCB/EECS-2009-136, EECS Department, University of California, Berkeley, Oct 2009.

[2] M. Covell and S. Baluja. Lsh banding for large-scale retrieval with memory and recall constraints. In *Proceedings of the 2009 IEEE International Conference on Acoustics, Speech and Signal Processing*, ICASSP '09, pages 1865–1868, Washington, DC, USA, 2009. IEEE Computer Society.

[3] M. Datar and P. Indyk. Locality-sensitive hashing scheme based on p-stable distributions. In *In SCG'04: Proceedings of the twentieth annual symposium on Computational geometry*, pages 253–262. ACM Press, 2004.

[4] J. Dean and S. Ghemawat. Mapreduce: simplified data processing on large clusters. *Commun. ACM*, 51:107–113, January 2008.

[5] M. Ferecatu. *Image retrieval with active relevance feedback using both visual and keyword-based descriptors*. PhD thesis, Université de Versailles Saint-Quentin-en-Yvelines, jul 2005.

[6] N. Gkalelis, V. Mezaris, and I. Kompatsiaris. A joint content-event model for event-centric multimedia indexing. In *Fourth IEEE International Conference on Semantic Computing (ICSC 2010)*, Pittsburgh, PA, USA, 09/2010 2010.

[7] A. Joly and O. Buisson. A Posteriori Multi-Probe Locality Sensitive Hashing. In *ACM International Conference on Multimedia (MM'08)*, pages 209–218, Vancouver, British Columbia, Canada, oct 2008.

[8] J. Liu, J. Luo, and M. Shah. Recognizing realistic actions from videos "in the wild".

[9] Q. Lv, W. Josephson, Z. Wang, M. Charikar, and K. Li. Multi-probe lsh: Efficient indexing for high-dimensional similarity search. In *VLDB*, pages 950–961, 2007.

[10] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman. Object retrieval with large vocabularies and fast spatial matching. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2007.

[11] T. Quack, B. Leibe, and L. Van Gool. World-scale mining of objects and events from community photo collections. In *Proceedings of the 2008 international conference on Content-based image and video retrieval*, CIVR '08, pages 47–56, New York, NY, USA, 2008. ACM.

[12] C. Schuldt, I. Laptev, and B. Caputo. Recognizing human actions: A local svm approach. In *In Proc. ICPR*, pages 32–36, 2004.

[13] R. Troncy, B. Malocha, and A. T. S. Fialho. Linking events with media. In *Proceedings of the 6th International Conference on Semantic Systems*, I-SEMANTICS '10, pages 42:1–42:4, New York, NY, USA, 2010. ACM.

[14] L. Xie, H. Sundaram, and M. Campbell. Event Mining in Multimedia Streams. *Proceedings of the IEEE*, 96(4):623–647, 2008.

[15] C. Xu, J. Wang, K. Wan, Y. Li, and L. Duan. Live sports event detection based on broadcast video and web-casting text. In *Proceedings of the 14th annual ACM international conference on Multimedia*, MULTIMEDIA '06, pages 221–230, New York, NY, USA, 2006. ACM.

[16] K. W. L. Young-Kee Jung and Y.-S. Ho. Content based event retrieval using semantic scene interpretation for automated traffic surveillances.


# 8. ACKNOWLEDGMENTS


This work was funded by the EU through the Integrated Project GLOCAL[2] (contract number FP7 - 248984).


[2]http://www.glocal-project.eu/