# I know what you did last summer: object-level auto-annotation of holiday snaps

Stephan Gammeter[1] gammeter@vision.ee.ethz.ch
Lukas Bossard[1] lbossard@student.ethz.ch
Till Quack[1,2] quack@kooaba.com
Luc Van Gool[1,3] vangool@esat.kuleuven.be

[1]ETH Zurich, Zurich, Switzerland
[2]kooaba AG, Zurich, Switzerland
[3]KU Leuven, Leuven, Belgium

## Abstract

*The state-of-the art in visual object retrieval from large databases allows to search millions of images on the object level. Recently, complementary works have proposed systems to crawl large object databases from community photo collections on the Internet. We combine these two lines of work to a large-scale system for auto-annotation of holiday snaps. The resulting method allows for automatic labeling objects such as landmark buildings, scenes, pieces of art etc. at the object level in a fully automatic manner. The labeling is multi-modal and consists of textual tags, geographic location, and related content on the Internet. Furthermore, the efficiency of the retrieval process is optimized by creating more compact and precise indices for visual vocabularies using background information obtained in the crawling stage of the system. We demonstrate the scalability and precision of the proposed method by conducting experiments on millions of images downloaded from community photo collections on the Internet.*

## 1. Introduction

These days, an increasing amount of photos is being stored on desktops and the Web. For instance, the social networking site facebook [1] reports that 30 million photos are uploaded to the site by its users – daily. Most of these photo organization tools also allow for some form of tagging (labeling) with keywords to facilitate search in the photo collection. However, tagging is a tedious process, and the rapidly growing amount of digital photos calls for some form of automated annotation.

In this paper we present a system which tags photos automatically with keywords referring to places, landmark buildings or events present in a photo. The annotation reaches down to the object level, *i.e*. recognized items are outlined with a bounding box in the image. The whole system works without any manual creation of a reference database and can annotate query images with data from millions of reference images in seconds. Figure 1 shows a typical example outcome from our annotation system. To achieve this functionality we combine two lines of recent work. First, in order to automatically create a reference database we build on large-scale data crawling from community photo collections [18]. Second, for recognition from that database we integrate scalable visual vocabulary based recognition approaches [7, 6, 10, 15, 16, 17, 21, 8].

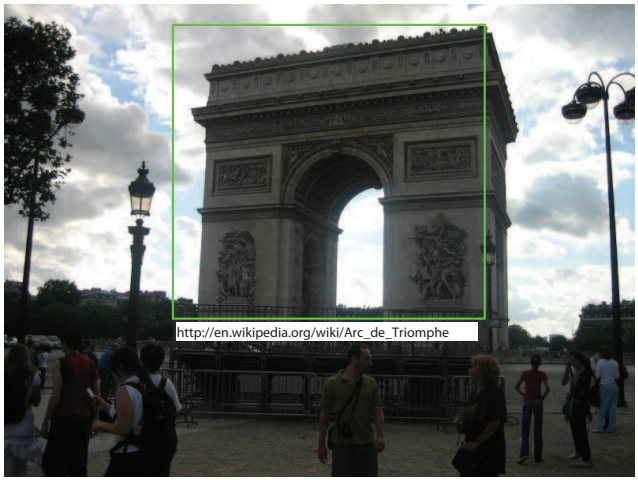


Figure 1. Example output from our system. Landmarks are automatically localized in the image and annotated with relevant content.

The first step, the crawling stage, enables us to create a large database of (exemplar) object models. Each object is represented as a cluster of images which show the same entity (object, event, scene *etc*.). In addition, the crawling stage also tells us what the cluster contains, by proposing labels, GPS location, and related content on the Internet without any manual intervention. This information is collected from the meta-data associated with the images from

[1] www.facebook.com



2009 IEEE 12th International Conference on Computer Vision (ICCV)
978-1-4244-4419-9/09/$25.00 ©2009 IEEE

the community photo collections.

The second stage, the retrieval stage, consists of a large scale retrieval system which is based on local image features. It indexes the entire database that was collected in the previous step. We further optimize the retrieval stage by integrating knowledge about the objects into the index creation process. This knowledge is also automatically acquired during the crawling stage. It results in more compact indices (only $33\%$ of the original size) without significant loss in precision. Any query image for annotation is first sent to the retrieval system, in order to identify matching objects. A verification step based on multiple view geometry refines the hypotheses. Finally, an annotation stage estimates the position of the object within the image, and annotates it with text, location, and related content from the database, resulting in the final annotation as the example shown in figure 1.

In summary, the contributions of this work consist of:
1) The combination large-scale object mining from the Internet with scalable retrieval to obtain an auto-annotation system driven by the wisdom of crowds.
2) The exploitation of knowledge collected during the mining stage to improve object retrieval.
3) The automatic annotation of objects in holiday snaps at the object level - with bounding box, related text, related Internet content and geographic location.

The remainder of this paper is organized as follows. A discussion of related work follows in the next section. In section 2 we discuss how a reference database is mined which is used in the object annotation stage. In section 3 we discuss object retrieval and in section 4 we introduce several improvements to the retrieval based on the knowledge from the mining stage. In section 5 we combine the results from mining and retrieval stages to create a large-scale auto-annotation system.

### 1.1. Related Work

Our annotation method relates to other works in several aspects. In general, auto-annotation is one of the most relevant applications for object recognition methods, and accordingly a large amount of work has been devoted to that problem [2, 4, 5, 11, 12, 19, 23, 24, 25]. Types of annotation cover a wide range, from scene classification [5, 24, 2], over assigning names to faces [4], to learning general correlations between words and image content [2, 12]. While many of the classic auto-annotation works deal with collections such as the Corel database, some of these works also integrate information from Web-image collections [25].

Work with photos from community photo collections on the Internet for retrieval of landmarks, locations *etc*. is a topic which is gaining increasing attention recently. For instance, in [9] the geographic location an image was taken at is estimated by comparing it to an enormous database of images downloaded from Flickr. The overall objective is to find near duplicate images of the same scene very efficiently. The images are encoded using several global feature types. The location of the picture is estimated by finding the nearest neighbor(s) in the database. This results in recognition rates of up to $16$ $\%$ for locating an unseen test-image within $200$ $km$ of its correct location. However, these recognition rates are not sufficient for the kind of auto-annotation applications we have in mind. A more precise, but more costly processing is possible with local image features. For example, in [20, 22] a method for clustering images from community photo collections was proposed using multi-view geometry based matching of local features. The goal was to derive canonical views for certain landmarks and to use those as entry points for browsing. Initial image collections for clustering were retrieved by querying photo collections with known keywords such as "Rome", "Pantheon", *etc*. Quack *et al*. proposed an approach, which relies on geo-tagged images in [18], thereby avoiding the need for manual query generation. Both works focus solely on the database creation process.

In this paper, we combine a data collection process similar to [9, 18, 20, 22] with object retrieval methods [7, 6, 10, 15, 16, 17, 21, 8] to obtain a large-scale, retrieval-based auto annotation system. The resulting method differs from earlier annotation systems in several ways. First, the task we set out to solve is not general annotation of images with words (such as tiger or grass), but rather the labelling of specific objects such as landmark buildings (*e.g*. Eiffel Tower, Teatro di Marcello), as they are often present in typical holiday snaps. Second, the annotation happens at the object level, *i.e*. the object is outlined with a bounding-box in the image. Besides textual labels, the annotation also includes related web-sites describing the object and its GPS position. Third, the annotation of a query image happens within seconds using a database created from millions of images.

## 2. Automatic object mining

The first element of our annotation system is a large collection of photos for objects (*e.g*. landmark buildings, statues, mountains, scenery, ...), which will serve as a reference database for later annotation of query images. The key success factor is, to somehow eliminate all the irrelevant information, such as dog pictures, party pictures *etc*. To collect such a high-quality database automatically, we follow the approach proposed by Quack *et al*. in [18]. Their work describes a system, which crawls community photo collections on the Internet to identify clusters of images referring to a common object or event. The clusters are created based on the images' pair-wise visual similarities, and the metadata of the clustered photos is used to derive a labeling for the clusters and to crawl related content on the Internet. The

whole process is automated and does not require any manual intervention. The beauty of this method is that it combines the stored collective intelligence of Internet users to distill a reference database of objects. In summary, the system proposed performs the following steps:

1. A geospatial grid is overlaid over the earth. For each grid tile's center a query is sent to Flickr, to retrieve geo-tagged photos from that area

2. For each tile, the retrieved photos are matched pairwise using local visual features and a homography as geometric filter on the features' positions. The number of inliers after homography verification gives a similarity measure for each pair of photos. The resulting distance matrix is used to cluster photos into groups of images showing the same object or scene. Since this expensive pair-wise matching step is done only per geographic cell (which typically contains significantly fewer than a thousand images in average) it is scalable and can be executed in parallel for each geographic tile. Figure 2 shows typical image clusters as they are created in this stage. For sake of simplicity, we will refer to these clusters as "object clusters" in the following (in spite that they might also contain images of scenery, events *etc*.).

3. The meta-data of each object cluster's photos is used to label the object clusters automatically. Textual labels are estimated using frequent itemset mining [1] on the associated text (tags, titles *etc*.).

4. Possibly related Wikipedia articles are crawled using the text labels from the previous steps as keyword queries in order to search for Wikipedia articles. A final verification is performed by extracting images from each article and trying to match them back to the associated photo cluster. If matching images can be found, the article is assumed to be relevant for the cluster. Note that this final step is essential, to obtain relevant related content for each object cluster.

We implemented the same pipeline as proposed in [18], but extended it with some modifications. The most important modification is that we do not rely on a regular geographic grid only to crawl photos from Flickr. Instead, we also instantiate crawling from the locations of geo-tagged Wikipedia articles. We start out a list of all articles on Wikipedia which are annotated with their geographic location. Each location is then taken as a "seed point" to query it's vicinity for photos on Flickr. The reasoning behind this extension over [18] is that many relevant articles are already tagged with their location, but we simply lack a sufficient amount of photos for the given topic. (A large amount of photos for each object cluster improves recognition of the

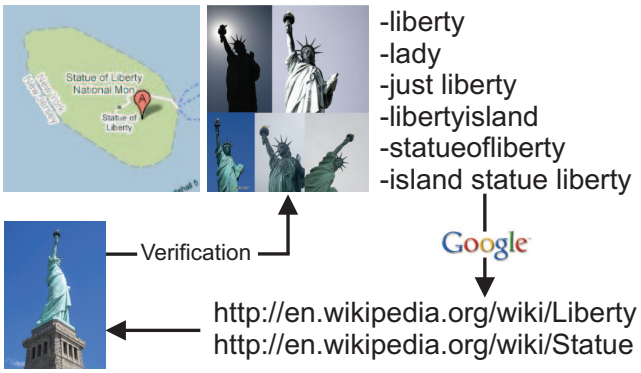


Figure 2. Example object cluster. Top: mined photo clusters with GPS location and tags. Bottom right: related content from Wikipedia. Bottom left: The images from the articles are used to match back to the clusters, as proposed in [18], which serves as a verification step for the articles' relevance.

object from arbitrary viewpoints.) In fact, this way we add an inverse processing step to the system of [18], by starting out with Wikipedia articles and then adding images from community photo collections.

We also extended step 4. of the above approach, by not only searching for related Wikipedia articles through Google, but also querying articles geographically by distance from the cluster. This is done using the mean location of all photos assigned to a given object cluster as a query point and retrieving all geo-tagged Wikipedia articles within a given radius .

In summary, the data collection steps outlined in this section leave us with a set of object clusters, each comprised of photos describing a given object or event. Each object cluster is associated with relevant tags and many of them also with related Wikipedia articles.

## 3. Scalable object cluster retrieval

In order to search the database of object clusters, we employ state-of-the art image retrieval methods based on visual vocabulary techniques. Visual vocabularies are usually created by clustering the descriptor vectors of local visual features such as SIFT [13] or SURF [3]. This approach has been applied very successfully in the domain of video retrieval [21] using k-means clustering to build vocabularies. Nister *et al*. demonstrated the benefits of larger vocabularies using a hierarchical k-means variant [15], which allowed retrieval for millions of images. The main advantage of the hierarchical approach is that it allows both for the efficient creation and search of the visual vocabulary. Recently, Philbin *et al*. showed [16] that a vocabulary obtained from a flat clustering still outperforms the hierarchical approach in precision. To handle large vocabulary sizes without an efficient hierarchical clustering method, they proposed approximate nearest neighbor search to speed-up cluster assign-

ment using a kd-forest as an index on the cluster centroids. (This is done while clustering as well as for searching the vocabulary with a query point).

Due to its superior performance, we build on the approximate k-means (AKM) idea [16] to index our database of images. We re-implemented the method and achieved similar performance on the Oxford test-set from [16]. (We obtained $0.53$ mAP without geometric constraints *etc*. compared $0.6$ mAP in [16] and $0.5$ mAP in [10], both also without geometric constraints).

Typically the retrieval results are ranked using a TF*IDF [14] scheme and a geometric consistency check for the arrangement of the local features in the image. In our system we use TF*IDF with

$$TF = 1 \quad (1)$$

$$IDF = -\sum_{v \in D} \log df(v) \quad (2)$$

where the candidate document $D$ contains the *set* of visual words $v_1 \dots v_n$ and $df(v)$ is the document frequency of visual word $v$.[2] We then apply a geometric consistency check by estimating a homography between candidate and query image using RANSAC. We retain only candidates when the number of inliers exceeds a given threshold.

The main difference between our work and others is, that we don't focus on a ranked list of images as a final result from retrieval. Instead, we want the system to tell us, which object is present in the query image. To that end, we return a ranked list of object clusters instead of images.

# 4. Object knowledge from the wisdom of crowds

The database we use in this paper differs from databases used in the retrieval works in an important point: it is not organized by individual images but by object clusters. We can use this partly redundant information from within the clusters to obtain a better understanding of the objects appearance in an unsupervised manner. Most image retrieval systems in earlier works did neither keep any specific information on the relationships between the items in the database, nor did they exploit it. The most similar work in that respect is maybe [7], where initial query results are used to expand a query with additional visual words in order to increase recall. In contrast, in this section, we show how we can use redundancies in the database in order to segment objects from the scene and to create more compact inverted indices. Note that having as compact as possible indices is very desirable when scaling to Internet-scale retrieval and annotation systems with millions of images.

[2]This corresponds to the set of words document model

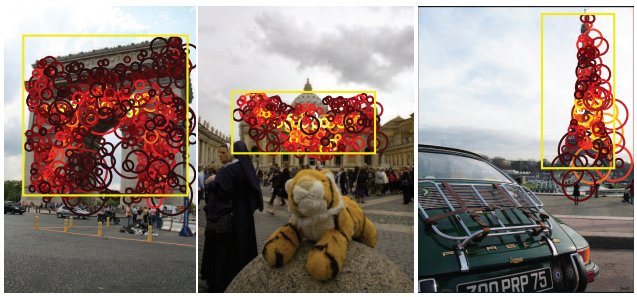

Figure 3. Visualization of object specific feature confidence scores for a few well known touristic sights. The brighter the color, the higher the score.

## 4.1. Object-specific feature confidence score

The key idea is to use the feature matches from the pairwise matching step in section 2. Using the information from these matches for each image we can derive a score for each feature, how likely it is to belong to the object that is represented by the images in the cluster. Only features which match to many of their counterparts in other images will receive a high score. In cases where many of the photos are taken from varying viewpoints around the objects, the background will receive less matches. As a result the object is "segmented" from the scene.

We define an object-specific confidence value for feature $f$ in image $i$ simply as the number of inlying feature matches stemming from all other images:

$$c^o_{if} = \| \{(u,v) \mid (u,v) \in I_{ij}, j = 1 \dots N^o \wedge u = f\} \| \quad (3)$$

where $N^o$ is the number of images in the current object cluster $o$. $I_{ij}$ is the set of inlying feature matches for image pair $ij$. We can now estimate a bounding box based on a threshold on that confidence value, where the threshold $t^o_i$ for object $o$ in image $i$ is defined as

$$t^o_i = max\left(t_{min}, \alpha * \frac{\sum_{f=1}^{M_i} c^o_{if}}{M_i}\right), M_i = \|\{f | c^o_{if} > 0\}\| \quad (4)$$

where $t_{min}$ and $\alpha$ are parameters. We obtained good results with $t_{min} = 1$ and $\alpha = \frac{1}{3}$. The bounding box is drawn around all features with confidence higher than $t^o_i$, in other words around all features that have a confidence higher than a fraction $\alpha$ of the mean confidence value.

Examples of the resulting confidence values and estimated bounding boxes are shown in Figure 3. Features, which lie within the estimated bounding box area are more likely to lie on the object and can also be used to create improved inverted indices for later retrieval of the object, as we will demonstrate in the following section.

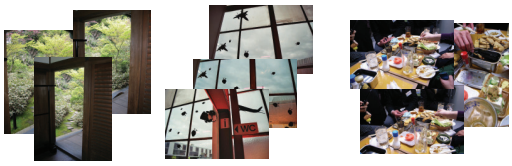

Figure 4. Bad "object" clusters

### 4.2. Better indices through object-specific feature sampling

We can use the gained knowledge on the object clusters to improve the efficiency and precision of our indexing stage.

The estimated bounding boxes can help to compact our inverted index of visual words. Since features which lie outside the bounding box are less likely to belong to the actual object, we simply remove their visual words from the inverted index. This results in a roughly $33\%$ smaller inverted index without any significant loss in precision. (Exact numbers are given in section 6).

An additional improvement in terms of index quality can be achieved by removing irrelevant data. Here, we resort again to the meta data collected alongside the images from flickr. Besides the associated tags, titles *etc*. it also tells us which user took the photo. This is valuable information, since it allows us to decide how relevant a given object cluster is for our auto-annotation system. We observed, that clusters which contain images taken only by a single user on a single day sometimes distract the retrieval. This is particularly bad when such a cluster contains hundreds of near duplicate images, as illustrated by the example in Figure 4. We thus simply remove these object clusters from ranked lists returned by the index lookup, or rank them lower.

The last step of the retrieval stage consists of selecting the best object cluster as a final result. The naïve approach of simple voting with retrieved images for their parent clusters has the obvious problem of normalization. Normalizing by cluster size is not feasible either, since some large clusters cover a wide range of viewpoints for a given item. Since only photos taken from a similar viewpoint as the query would match, the normalization by the full cluster would punish these large clusters (which are often the ones of highest quality). A simple but effective solution we found works as follows: only the votes of the $T$ images per cluster with the highest retrieval scores (*i.e*. TF*IDF) are counted. We obtained good results with $T = 5$.

## 5. Object-level auto-annotation

The final annotation stage consists of two steps: bounding box estimation and labelling. The bounding box for the object's location in the query image is estimated in the same way as the bounding boxes for the database images. To that end, the query image is matched to a number of images in the cluster returned at the top of the retrieval results to refine the initial voting from the retrieval stage. The mean number of votes for all features in the bounding box serves as a score for the bounding box hypothesis.

Since the crawling stage already added location, related text and related content to each object cluster, we can simply copy this information to serve as labels for the query image.

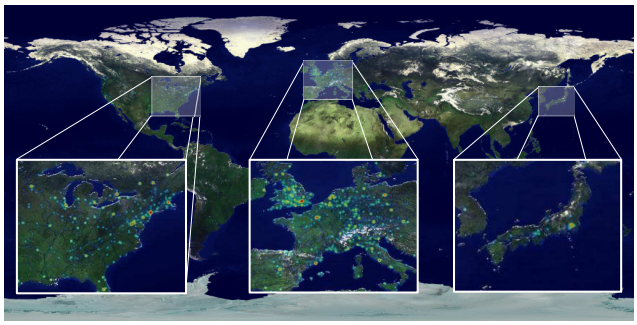

Figure 5. Crawling hotspots.

| | |
|---|---|
| # Images crawled | 4'482'582 |
| # Images processed to date | 996'341 |
| # Object Clusters | 63'232 |

Table 1. Dataset statistics.

## 6. Experiments and Results

We report experiments for all steps of the annotation process. First we evaluate how well our object retrieval stage retrieves the correct object clusters from the database. For the retrieved object clusters we then measure the quality of the object-level annotation.

The experiments were conducted on a large dataset collected from Flickr ( `www.flickr.com` ) as described in section 2. We crawled and downloaded over $4$ million images at $500$px resolution to date. Figure 5 shows the hotspots (*i.e*. regions with large numbers of photos) encountered by our crawling method. The exact statistics of the set are shown in Table 1.

We report results on a subset of of roughly $1$ million images, since the processing for the mining stage had not been finished processing at the time of writing this paper. These images are contained in $63'232$ object clusters identified by the mining stage and the estimated bounding boxes cover on average $52\%$ of each image. For this database, we collected a challenging test-set of $674$ images from the Internet. We selected a number of object clusters and for each we manually searched Picasa Web-Albums ( `www.picasaweb.com` ) for images of the same object, ensuring that the images were not contained in our database. By using images from Picasa, we obtain typical holiday snaps shared by users on that platform. The testset is in-

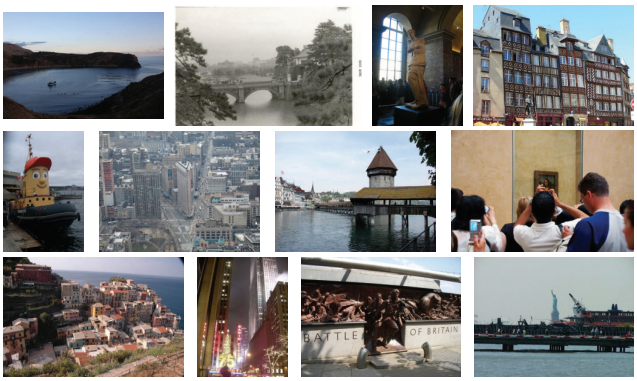

Figure 6. Examples from our test dataset.

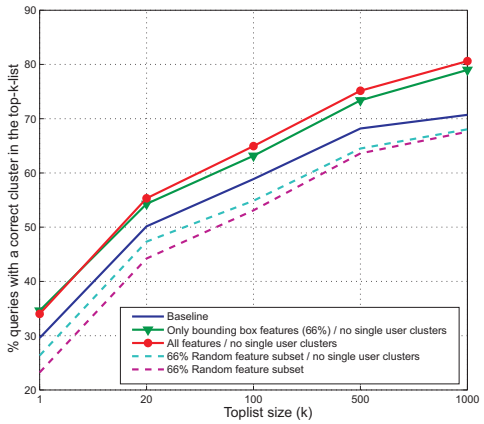


Figure 7. Recall in top $k$ results after inverted index, no geometry.

tended to simulate typical photos tourists would take during a holiday trip, including examples taken from varying viewpoints, with partial occlusions of the objects, bad lighting conditions *etc*. A few examples are shown in figure 6.

### 6.1. Efficiency and Precision of Recognition

To achieve good annotation results, the goal of the retrieval stage must be to return the correct object cluster with the highest possible precision, *i.e*. we rather retrieve no object at all than a wrong result. The object clusters that were collected during the mining process allow us to reach these high levels of precision, since they contain a large number of images covering the objects from many viewpoints, lighting conditions *etc*. However, since ultimately we are interested in world-scale auto annotation of holiday snaps, the system needs to be not only precise, but also efficient. Thus, our evaluation emphasizes both criteria.

As introduced in section 3 the retrieval stage is composed of object cluster candidate retrieval followed by geometric verification. In order to evaluate the object cluster retrieval precision, we look at the position of the correct cluster in the retrieved ranked list of candidates. This is illustrated in figure 7. We plot the percentage of query results that contain the correct groundtruth cluster on the y-axis. This value is set in relation to the rank $k$ in the retrieved list. As a baseline we perform image retrieval on the entire dataset using $TF * IDF$-ranking on a $500K$ visual vocabulary as it is used in other works. The vocabulary is trained on the features of a random subsample of the 1 million images. We compare this to our optimized indices based on the two improvements described in section 4. The first optimization – discarding clusters with photos from only one user from retrieval – increases precision by about $5\%$.

The second optimization reduces the index by about $33\%$ by building the inverted index only from features within the mined object bounding boxes. This significant memory efficiency improvement comes at nearly no loss in precision. We demonstrate the superiority of our approach over random subsampling by also plotting the results achieved with a $66\%$ random subsampling of features. The precision of this method is about $10\%$ lower.

| | Uncompressed | Compressed |
|---|---|---|
| Full Index | 3.7 GB | 1.8 GB |
| BB Features | 2.3 GB | 1.2GB |

Table 2. Inverted index sizes in GB. (BB = Bounding Box)

Note that the index can be even further compressed by using standard index compression techniques from text-retrieval, *e.g*. lossless compression techniques such as Simple-9 [14]. Table 2 compares the resulting index sizes. It demonstrates how the combination of our bounding-box feature sampling method together with Simple-9 results in a total index size reduction of $67\%$ compared to the naïve implementation where each index entry occupies $4$ bytes.

After this step, the ranked list contains between $55\%$ and $80\%$ of the correct clusters within the top $20$ and top $1000$ positions, respectively. This index lookup typically requires well below $1$ second in our current system. The following geometric verification step should move most of these candidates to the top spots.

For this geometric verification step we consider two scenarios. In the first we only use the matched visual words for the homography estimation. Since no more correspondences have to be computed this can be done very efficiently for large sets of images. Our current implementation performs this step for $1000$ images in about $2$ seconds. In the second scenario we trade speed for accuracy by matching the actual local features of the query to the candidate image features in order to increase the number of correspondences for the homography estimation. In principle this can be done in near real time (sub $4$ seconds) for $1000$ images using dedicated hardware like a GPU. On our testset we achieved $0\%$ false positives using a threshold of $8$ inliers for the first scenario and $20$ inliers for the second scenario. The final recognition rates (*i.e*. correct clusters at position $1$)

| | Feat. Based | Vis. Word Based |
|---|---|---|
| Full Index | 452 (67.1%) | 324 (48.1%) |
| BB Features | 447 (66.3%) | 325 (48.2%) |

Table 3. Final recognition rates after geometric verification. (BB = Bounding Box)

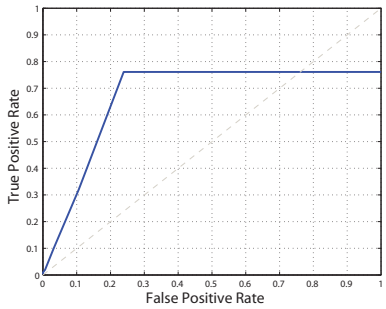


Figure 8. ROC plot for object-level auto-annotation.

after the retrieval stage are summarized in table 3. Considering the extremely challenging nature of our testset, these results are of good quality.

### 6.2. Annotation precision

In this section we evaluate how well our system localizes bounding boxes for the retrieved objects. We evaluate the localization precision by measuring the intersection-over-union (IOU) measure for the ground-truth and hypothesis overlap, as it is commonly done also in object class detection. An IOU value greater than $0.5$ counts as true positive, any other hypothesis counts as false positive. The evaluation is carried out on all images, that have been correctly recognized from the previous stages of the annotation system, *i.e*. in this step we measure only how well we can localize the object, given that we already determined its presence in the image. Figure 8 shows a ROC plot for this experiment. The curve is generated by varying the confidence (*c.f*. equation 4) threshold for the estimated bounding boxes. We reach up to $76.1\%$ localization rate at rather low false positive rate. Note that the false positives are exclusively generated from detections which do not have sufficient overlap with the groundtruth. The geometric verification step after the retrieval stage circumvents the generation of false positives. Figure 9 shows final localization and annotation examples. We show the ground truth bounding box in yellow, and our correct detections in green. We also show the Wikipedia article that was considered relevant for the detected object by our system. All objects are also labeled with relevant tag and with a GPS location which are omitted in this figure. Note the wide variety of objects, and the sometimes surprising matches with Wikipedia articles resulting in astonishing annotation results. (Remember, that from each Wikipedia article an image matched to our object cluster, which we used to verify the content assignment as described in section 2.)

## 7. Conclusions

In this paper we presented a full auto annotation pipeline for holiday snaps. The complete system functions fully automatically starting with data crawling up to object-level annotation with bounding box, relevant tags and Wikipedia articles, as well as GPS location. The system design allows scaling to millions of images in the reference database. To achieve this functionality, we combined a multi-modal data mining method for community photo collections with state of the art object retrieval based on visual vocabularies. The mining method groups photos of objects into object clusters which serve as exemplar models for items in the database. These object clusters allow us not only to recognize query images for annotation from a wide variety of viewpoints, but also to automatically localize the object in database images. We showed how to exploit this information to reduce the index size without loss in retrieval precision, enabling further scaling. The system was evaluated on challenging test-data and a large database in terms of correct recognition and annotation of objects as well as their localization within the query image.

**Acknowledgments:** The authors gratefully acknowledge the support by the Swiss SNF NCCR IM2 and the Swiss KTI.

## References

[1] R. Agrawal, T. Imielinski, and A. N. Swami. Mining association rules between sets of items in large databases. In *SIGMOD*, 1993.

[2] K. Barnard, P. D. Pinar, D. Forsyth, N. de Freitas, D. Blei, and M. Jordan. Matching words and pictures. *J. Mach. Learn. Res.*, 3:1107–1135, 2003.

[3] H. Bay, T. Tuytelaars, and L. Van Gool. Surf: Speeded up robust features. 2006.

[4] T. Berg, A. Berg, J. Edwards, M. Maire, R. White, T. Yee-Whye, E. Learned-Miller, and D. Forsyth. Names and faces in the news. In *CVPR*, 2004.

[5] A. Bosch, A. Zisserman, and X. Munoz. Scene classification via pLSA. 2006.

[6] O. Chum and J.Matas. Web scale image clustering. Technical report, Czech Technical University Prague, 2008.

[7] O. Chum, J. Philbin, J. Sivic, M. Isard, and A. Zisserman. Total recall: Automatic query expansion with a generative feature model for object retrieval. In *CVPR*, 2007.

[8] M. B. G. Schindler and R. Szeliski. City-scale location recognition. In *CVPR*, 2007.

[9] J. Hays and A. A. Efros. Im2gps: estimating geographic information from a single image. In *CVPR*, 2008.

[10] H. Jegou, M. Douze, and C. Schmid. Hamming embedding and weak geometric consistency for large scale image search. In *ECCV*, 2008.

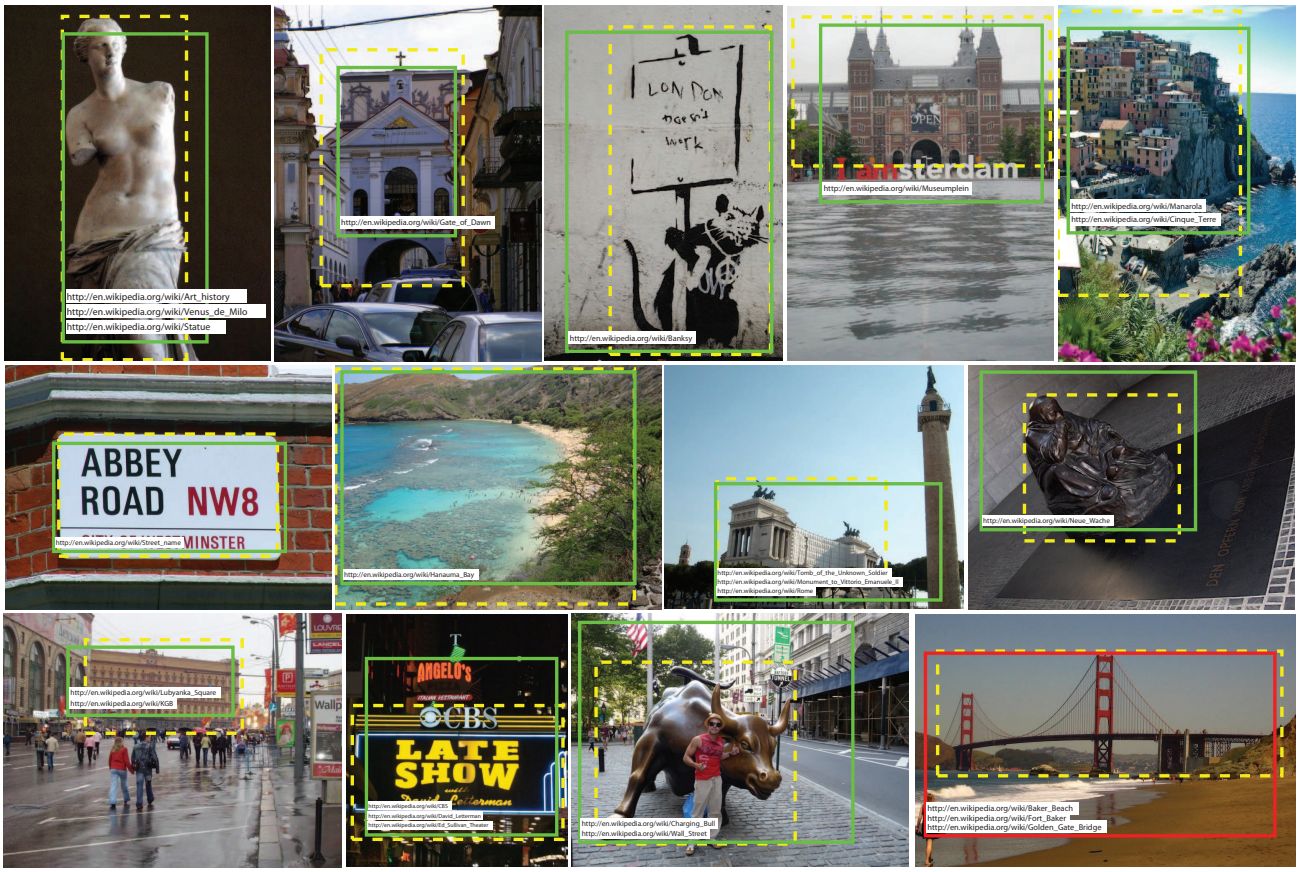


Figure 9. Results of automatic object-level annotation with bounding boxes. Groundtruth annotation is shown with dashed lines, correct detection with solid green lines, false detections with solid red lines. Auto-annotation with related Wikipedia articles is also shown. All results are also labeled with their GPS position and estimated tags (not shown here).


[11] Y. Jin, L. Khan, L. Wang, and M. Awad. Image annotations by combining multiple evidence and wordnet. In *ACM Multimedia*, 2005.

[12] V. Lavrenko, R. Manmatha, and J. Jeon. A model for learning the semantics of pictures. In *NIPS*, 2003.

[13] D. G. Lowe. Object recognition from local scale-invariant features. In *ICCV*, 1999.

[14] C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to Information Retrieval*. Cambridge University Press, 2008.

[15] D. Nistér and H. Stewénius. Scalable recognition with a vocabulary tree. In *CVPR*, 2006.

[16] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman. Object retrieval with large vocabularies and fast spatial matching. In *CVPR*, 2007.

[17] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman. Lost in quantization: Improving particular object retrieval in large scale image databases. In *CVPR*, 2008.

[18] T. Quack, B. Leibe, and L. Van Gool. World-scale mining of objects and events from community photo collections. In *CIVR*, 2008.

[19] F. Schaffalitzky and A. Zisserman. Multi-view matching for unordered image sets, or "how do i organize my holiday snaps?". In *ECCV*, 2002.

[20] I. Simon, N. Snavely, and S. M. Seitz. Scene summarization for online image collections. In *ICCV*, 2007.

[21] J. Sivic and A. Zisserman. Video google: a text retrieval approach to object matching in videos. In *ICCV*, 2003.

[22] N. Snavely, S. Seitz, and R. Szeliski. Photo tourism: Exploring photo collections in 3d. *ACM Trans. on Graphics*, 25(3), 2006.

[23] A. Torralba, R. Fergus, and W. Freeman. 80 million tiny images: a large dataset for non-parametric object and scene recognition. *PAMI*, 30(11):1958–1970, 08.

[24] J. Vogel and B. Schiele. Semantic modeling of natural scenes for content-based image retrieval. *IJCV*, 72(2):133–157, 2007.

[25] C. Wang, F. Jing, L. Zhang, and H. Zhang. Image annotation refinement using random walk with restarts. In *ACM Multimedia*, 2006.