ABSTRACT
So far, most image mining was based on interactive querying. Although such querying will remain important in the future, several applications need image mining at such wide scales that it has to run automatically. This adds an additional level to the problem, namely to apply appropriate further processing to different types of images, and to decide on such processing automatically as well. This paper touches on those issues in that we discuss the processing of landmark images and of images coming from webcams. The first part deals with the automated collection of images of landmarks, which are then also automatically annotated and enriched with Wikipedia information. The target application is that users photograph landmarks with their mobile phones or PDAs, and automatically get information about them. Similarly, users can get images in their photo albums annotated automatically. The object of interest can also be automatically delineated in the images. The pipeline we propose actually retrieves more images than manual keyword input would produce. The second part of the paper deals with an entirely different source of image data, but one that also produces massive amounts (although typically not archived): webcams. They produce images at a single location, but rather continuously and over extended periods of time. We propose an approach to summarize data coming from webcams. This data handling is quite different from that applied to the landmark images.

Categories and Subject Descriptors
H.3.1 [Information Systems]: Information Storage and Retrieval—Content Analysis and Indexing; I.4.8 [Computing Methodologies]: Image Processing and Computer Vision—Scene Analysis

General Terms
Image collection mining, Computer Vision, Large-scale processing

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 profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

Keywords
Image collection mining, Computer Vision, large-scale processing, webcam summarization, landmark recognition, vision for mobile phones, learning from continuous data streams

1. INTRODUCTION
Image mining has started off pretty much as an interactive process. This still is the case, also in contributions that can be regarded as state-of-the-art. Examples are collecting images from public repositories by typing in queries [24] and from movies by delineating objects by hand [23]. Several applications need images to be mined at too large a scale for such interaction to be affordable. Going back to the PhotoTourism example in [24], one would like to automatically find out about the existence of landmarks at a worldwide scale, and then mine images equally automatically for all of those. This means that there is a need for automation in the mining from the very first steps, i.e., some systems should autonomously know what to mine where and when.

Already in [21] we have proposed a mechanism to distinguish between images of “landmarks” vs. images of events vs. the rest. In this paper, we discuss a bit further how we plan to differentiate the treatment of automatically collected images, according to their nature. As in [21], a first part of this contribution focuses on landmark images. We demonstrate how the set of retrieved landmark images can be further enlarged and how to apply it to annotation. The second part then focuses on an important image type, which is automatically rejected by the landmark selection procedures from [21], namely those coming from webcams. These are classified as such automatically as well (many images taken over extended periods of time and at a more or less fixed temporal rate, taken by a single source - and at the same place if geo-tagged). Typically, images from webcams are not archived. So, we discuss a webcam summarization tool, which selects images that together give an overview of what the webcam typically sees, as well as of the unusual scenes it observes.

So far, we have discussed the handling of landmarks images from public repositories and images from webcam streams. Another example would be events. This category is again automatically detected by the method of [21]. Here, one would also need dedicated types of processing. For example, for images of a birthday party, it is probably relevant to figure out who appears in which image. Tags and keywords could be used to find out about their names [22]. In the case of a concert, the focus may be on where and when it was, and on which band is seen performing. One can only start to
imagine what would be required to deal with a sports match or an art vernissage. We give these examples merely to show that much work is still required to make fully automated mining a reality.

Next, we discuss the automatic landmark and webcam mining procedures in more detail (sections 2 and 3, respectively).

2. LARGE-SCALE LANDMARK MINING

In [21] we have proposed a way to automatically mine landmark images from public repositories. An important step in the process was the automated distinction between landmark (object) images, vs. event images vs. others. Here, the focus is again on the landmark images. We discuss two important extensions. On the one hand, we try to enlarge the set of images that are retrieved for each landmark. In [21], only geo-tagged images were considered for the visual clustering. Section 2.1 gives a short summary of that earlier work. Here, images without geo-tags are added, as described in section 2.2. As a second extension, an example application is built on top of the automated landmark detection and image mining. We discuss a web application for automated holiday snap annotation, c.f., section 2.3.

2.1 Automatic landmark mining: past

In Quack et al. [21] we proposed a system to automatically crawl community photo collections and mine images of landmarks, while discarding the many dog, cat, party, etc. pictures. Textual image annotations are used to find out more about the landmarks, in particular to retrieve relevant Wikipedia pages. The approach exploits the stored, collective intelligence of Internet users to distill a reference database of joint landmark images and descriptions.

In summary, the system in [21] performs the following steps:

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

2. For each tile, the retrieved photos are matched pair-wise using local visual features and a homography as geometric filter on the features’ positions. The number of inlier features 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 on average) it is scalable enough. Moreover, it can be executed in parallel for each geographic tile. Figure 1 shows typical image clusters as they are created in this stage. The clusters for landmarks are automatically split from event images and others.

3. The metadata of each landmark cluster’s photos is used to automatically label it. Textual labels are derived using frequent itemset mining [3] on the associated text (tags, titles etc.).

4. These frequent itemsets are then used as queries to find related 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.

2.2 ...and present: extended mining

Geo-tagging as required by [21] is quickly becoming general practice, but it isn’t yet. On the other hand, applications built on top of the image clusters often benefit from a dense covering of the landmark by the images. Thus, for the landmark images, it is important to go beyond those that have been collected through their geo-tags. Here, we propose a strategy that does. As a matter of fact, it also goes beyond the traditional, typed query results. Each cluster of [21] will be expanded on the basis of several, automatically generated queries. For instance, in our case, it is perfectly possible to generate queries in different languages. Take the Mona Lisa as an example. With the ‘Mona Lisa’ query only a subset of images will be retrieved. Additional queries like ‘Monna Lisa’ and ‘Joconde’ will produce more.

Note that a limitation of our approach still is that a sufficiently large cluster of visually matched, geo-tagged images must be available to kickstart the process. Missing out on smaller clusters, which would then also have few geo-tagged images, is a problem suffered by other state-of-the-art approaches as well [18].

In order to enlarge the landmark image clusters without increasing the computational time, we improve the pipeline proposed in [21] (section 2.1) through several modifications:

Step 1. A first modification is made to the first step. Instead of using a fine regular grid with cell radii of only a few hundred meters we start off with a very coarse grid with cell radii of hundreds of kilometers. Before we download images from a cell we first query the cell only for the number of images it contains. If the cell contains more than a certain threshold \( T \) it is split into four equally large cells. This continues recursively until a cell has less than \( T \) images, in that case we start retrieving the images of the cell. The advantage of this method is that vast empty regions can be covered very quickly. This helps freeing the necessary time to add the following modifications.

Step 2. For the second step, we no longer 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 by using their respective location as a “seed point” to query their vicinity for photos (Wikipedia pages not yet linked with image clusters). The reasoning behind this extension is that many relevant articles are already tagged with their location, but we simply lack a sufficient amount of photos for the given topic and hence no appropriate queries are generated. This approach retrieves additional but small clusters of images, still geo-tagged, and connected to a Wikipedia site (as normally achieved through Step 4).

Step 3. Our extension to step 3 uses the textual frequent itemsets generated during the mining stage to find more (non geo-tagged)
images to enrich the cluster, instead of only using them to find Wikipedia pages. We use the itemsets as queries for an image search service (e.g., Google images, Flickr etc.) and match the top-N results against the images of the existing cluster. This extension is also linked with Step 4 in that queries that produce relevant Wikipedia pages are also the most successful ones in retrieving additional images.

**Step 4.** We also extended the fourth step of the original 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.

Fortunately, the Step 3 extension works especially well for small clusters, where the addition of images has the most impact. This appears to be because the probability of a candidate image matching against any image already inside a cluster does not depend strongly on the cluster size. Figure 2 demonstrates this. We applied the aforementioned procedure to a random subset of 332 clusters and evaluated how much each cluster has grown. We also made a distinction between queries that yield validated Wikipedia articles and queries that do not. In this setting we use the 5 itemsets with largest support as search queries for Google images and limit the maximum number of candidate images to 100 per query. Adding all 100 images for all itemsets yields the theoretically maximum performance in the left part of the figure. We see that small clusters grow comparably as close to this maximum as larger clusters do. Looking at how the median matching rate changes with cluster size, we find that even very small clusters yield stable matching rates for queries that are known to lead up to validated Wikipedia articles. As a consequence, especially small clusters can grow a lot relative to their original size. Large clusters usually don’t need to. These data have been corrected for the bias created by the fact that relevant Wikipedia pages are guaranteed to yield at least one matching image. Thus, a local copy of Wikipedia can be used to a priori assess the potential value of a query before any external service like Google or Flickr has to be queried.

A specific example of how much this processing can help is given in figure 3. The query used to retrieve these images was “museums perseus medusa vatican” with a matching rate of 35% while the other two mined itemsets did not yield any results. There are two things that should be noted here. First, none of the individual words in this query would produce nearly as many matching candidates images compared to the actual, composition query. Second, the initial cluster contains an incorrect image and thus is only 80% accurate, but by adding more not geo-tagged images, the cluster’s accuracy increased to 95%. This demonstrates the effectiveness of using the mined itemsets as queries even for small clusters.

### 2.3 Auto-annotation application

With the database of landmark image clusters in place, we can use it for applications such as auto-annotation [7] or 3D reconstruction [27, 8]. In [7] we propose such a retrieval-based auto-annotation system, which annotates landmarks in query images. The annotation system combines mining and state-of-the-art large-scale object retrieval, i.e., visual vocabulary techniques [23]. Due to its superior performance, we build on the approximate k-means (AKM) idea [19] to index our database of images.

The final annotation stage consists of two steps: bounding box estimation and labeling. To estimate the bounding box for the landmark in the query image, this image is matched to a number of images in the cluster returned at the top of the retrieval results. The mean number of votes for all features in the bounding box serves as a score for the bounding box hypothesis. Typical “raw” results of this annotation process are shown in figure 5. Note the wide variety of object and scene types that can be annotated by our system.

We implemented the system of [7] with the aforementioned extensions as a web service. Our auto-annotation application connects to the Flickr API and allows users to annotate their photos, e.g., their holiday snaps. 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 image to be annotated. The user interface for annotation and the viewer interface are shown in figure 4. The cross media information collected during the mining stage presents relevant contextual information for each annotated photo. Obviously, once the photos are annotated, a user’s photo-collection can be searched by keyword. Note that the contextual information is richer than what people would typically add by hand.

### 3. WEBCAM SUMMARIZATION

In the previous section, we discussed the large scale, automatic mining from public repositories of landmark images. As said, other image types would require a very different treatment. Images that are identified as coming from a webcam are a good case in point. They do not have to be scrambled together from many sources, but would require online monitoring and archival from a single source. Again, in order to deal with such data at a large scale, these image streams could each be replaced by a relatively small, representative set of images. Hence, the question is which images ought to be selected. We present a method that picks a representative mix of typical and unusual images.
Recent years witnessed the proliferation of publicly accessible webcams. However, most work to explore such data focuses on estimating imaging conditions such as illumination and color changes, which can be used, e.g., for camera geo-location [13, 26], shadow removal [29], seasonal variation detection [11], or video synopsis [20]. Our previous method [5] automatically detects anomalies in continuous data streams. We adapt this algorithm to summarize the data stream. The method described in the following sections builds a representative mix of usual and unusual imagery. As will be explained, such mix results from a single approach.

3.1 What is usual?

Our approach is that everything is usual that has been observed in the past (or within a specified time frame, respectively). Hence, our algorithm checks if a new observation is a statistical outlier and thus should be classified as a novel, unusual image, while constantly adapting the model of normality. We employ the concept of meaningful nearest neighbours [4, 17] for classification and maintain a normal model by applying online agglomerative clustering. The cluster centers will represent the typical images of a scene that could be used to summarize a data stream.

More formally, given a continuous (i.e., ongoing, permanent) data stream \( S = \{x_1, \ldots, x_{t-1}\} \) (e.g., of images that are represented by feature vectors \( x_i \in \mathbb{R}^n \) in a high dimensional space), our target is to classify the next example \( x_t \) as normal if it can be explained by previous data (i.e., it is similar) and as abnormal otherwise. A scene is usual if a very similar scene has been observed at least once in the past (c.f., [30]). Because data points that appear temporally close may be correlated, we introduce a time delay \( \Delta t \) (e.g., a day). However, a usual scene still can appear frequently or rarely. The first time a sunny day is observed from a London webcam, an image of the scene will look very differently than everything before. Thus, the scene is unusual. However, if more sunny days are observed, they are becoming usual but may still be rare. If they appear more often then they become also frequent. In this work, we primarily concentrate on finding usual, frequent scenes to summarize the online data stream from a webcam. In the begin-
3.2 Meaningful Nearest Neighbours

Data Representation Because images from webcams are typically captured with low resolution and low, even non-constant frame rate, our method is based on static, low level features. The algorithm computes Histogram of Oriented Gradient (HOG) features [6] by applying Sobel filters on 8 × 8 image cells and equally dividing the orientation range to 8 bins. The HOG features from all cells are then concatenated. For simplicity and efficiency, the concatenated features \( f_1, f_2 \) of two data points are compared using normalized cross correlation.

Classification A simple approach to detect an unusual scene is to store all observed features in a memory

\[
\mathcal{M} = \{x_1, \ldots, x_{t-\Delta t}\},
\]

and to find the best matching sample \( x_t^* \) for a new data point \( x_t \) (i.e., the nearest neighbour in \( \mathcal{M} \)). Then, the similarity score of this best match

\[
s^* (x_t) = \max_{x_i \in \mathcal{M}} \text{sim}(x_t, x_i)
\]

is compared to a threshold for classification, where \( \text{sim}(\cdot, \cdot) \) denotes the similarity of two data points. However, it is not clear if the nearest neighbour is meaningful for a high-dimensional space [4], because most points are equally distant to a query point. Thus, it may be difficult to find an appropriate threshold to classify a query point as outlier based on its distance to the nearest neighbour.

Therefore, we propose the following approach based on the concept of meaningful nearest neighbours, as illustrated in figure 6.

We compare the similarity score \( s^* (x_t) \) of the current input and the best match in the memory with the distribution \( S^* \) of the similarity scores of all nearest neighbours in the memory \( \mathcal{M} \) obtained so far. In the beginning, the memory \( \mathcal{M} \) will not contain a representative set of images, hence \( S^* \) is not meaningful yet. However, as time goes on, more and more images are observed and integrated, such that the memory will contain a representative set of typical images of the scene.

A new data point \( x_t \) is then classified as an inlier (and thus as usual), if its similarity score \( s^* (x_t) \) is likely to be generated from the distribution \( S^* \). Consequently, our algorithm computes the inlier probability \( P_{\text{usual}} \) of a new data point \( x_t \) by comparing its similarity score \( s^* (x_t) \) with the cumulative probability distribution \( F_{S^*} (\cdot) \) of \( S^* \):

\[
P_{\text{usual}}(s^* (x_t)) = \int_{s=0}^{s^* (x_t)} S^* (s) \, ds = F_{S^*} (s^* (x_t))
\]

Finally, \( x_t \) is classified as unusual using the probability threshold \( p_{\text{alarm}} \)

\[
P_{\text{usual}}(s^* (x_t)) < p_{\text{alarm}}.
\]

This can be rewritten using Eq. (3) and the inverse function \( F_{S^*}^{-1} (\cdot) \) to compute a threshold for the similarity score:

\[
s^* (x_t) < F_{S^*}^{-1} (p_{\text{alarm}}) = \theta_{\text{alarm}}.
\]

Advantages Using meaningful nearest neighbours for novelty detection has several benefits. Most importantly, no scene-specific, manually tuned similarity threshold for the classification based on the distance to the nearest neighbour is used. Such a threshold would have to be changed permanently, as more data is observed and included in the memory. Instead, the meaningful nearest neighbour approach classifies a data point relative to previous observations, by comparing it to the distribution of nearest neighbour distances of previous observations. Secondly, the normal model is purely data driven. It is represented by the data points themselves, which allows the model to change in order to become more sensitive if more data is observed. Thus, the model adapts automatically and permanently.

3.3 Clustering

To summarize a continuous data stream and because not all data could be stored in practice, the memory \( \mathcal{M} \) is modified by applying an online, agglomerative clustering algorithm similar to [9]. As long as the maximum number of clusters \( K \) is not reached, every observation is saved. Afterwards, the most similar two cluster centers are merged and the new observation is added. However, because a cluster center represents an image in our case, the centers are not altered (i.e., merged), but the “weaker” center is removed. Our algorithm replaces a cluster center based on the following measure. We compute the number of times a cluster center was a best match, limited to once per time window \( \Delta t \) (e.g., a day), which is divided by its age (i.e., a multiple of \( \Delta t \)). Then, the cluster center with the lower value is deleted. Hence, a cluster center is only removed (i) if another nearby center exists and (ii) if it represents a scene that has not been observed for a long time. A cluster center that is very distant to every other center remains in the model, implementing our definition that everything is usual that has been observed in the past. Alternatively, those clusters which were not best matches for a long time could be removed instead.

The cluster centers with the most support are the typical, representative images.
ducted regularly after 3.4 Implementation

months. Three clusters (images) found by our algorithm are shown after observing the scene for one day, one week and two months.

Figure 7: Summary of a data stream from a public webcam at Time Square (3 am – 4 am). Three clusters (images) found by our algorithm are shown after observing the scene for one day, one week and two months.

3.4 Implementation

The system consists of two parts. A maintenance phase is conducted regularly after $\Delta t$, e.g., daily, during which the clustering algorithm integrates newly observed data. Then, $S'$ and $\theta_{\text{alarm}}$ are computed from the updated model. Note that $\theta_{\text{alarm}}$ can be calculated for the first time after $2\Delta t$ because it relies on $S'$. During the online detection phase, the algorithm classifies outliers in real-time based on Eq. (5).

This evaluation is very efficient, because $\theta_{\text{alarm}}$ is computed in the maintenance phase. Second, the algorithm can usually classify an observation as inlier without finding its best match. According to our definition, a scene is usual if at least one similar data point has been observed in the past. For this purpose, the cluster statistics described above is used to first compare the input to the most frequently matching cluster centers. Third, the search space can be reduced by only considering cluster centers with timestamps that match. According

to our approach. Thus, a scene is usual only around a certain time of day (e.g., an empty street is normal at night but not during the day).

3.5 Experiments

We recorded a dataset from a public webcam in New York (US) [1] around the clock during two months with a framerate of 16 images per minute. An image has a resolution of $640 \times 480$ pixels, resulting in more than 0.5 TB of data. The dataset covers scenes that capture a large variety of potentially interesting events under a wide range of conditions (i.e., depending on the weather, illumination, time of day, etc.). Individual objects such as people, cars or animals have a size of only a few pixels, such that object detection and tracking is difficult. We demonstrate that our algorithm (i) finds representative images that are typical for a data stream and (ii) detects novel, unusual scene images.

Parameter Settings We choose the parameters $\theta_{\text{match}} = 0.95$ and $\theta_{\text{alarm}} = 1\%$, which remain identical for the sequences. The number of cluster centers is set to $K = 10,000$, limited by the physical memory. On a standard workstation (Intel P4 3.2 GHz, 2 GB), the runtime of the algorithm during the test phase is about 200 ms and the maintenance phase is in the order of minutes for our MATLAB implementation. Because the framerate of many publicly accessible webcams is below 5 fps, our algorithm is able to process their data streams in real-time. We disrupt the online evaluation phase once a day for the maintenance phase during a certain hour at night.

Evaluation In figure 7, we show some of the top-ranked cluster centers according to their frequency. As can be seen, a wide variety of situations as well as illumination and weather conditions are represented when observing the scene for a longer period of time. The longer the algorithm runs, the more variety is represented ($e.g.$, rainy situations). Also less frequent situations ($e.g.$, construction work) are represented because the respective cluster center is very different to the others and hence will not merged. These images can be used to summarize a data stream. In figure 8, the most distinctive of the 20 most frequent cluster centers are shown for two other times of the day.
Figure 9(b) shows typical matches found for the input images in figure 9(a). The first two scenes are correctly classified as usual. In the right image, a low-loading truck has stopped at the roadside, which intuitively should be an unusual scene. Because an exceptionally similar image is found (below), this scene is also classified as usual. However, by simply looking at this single image, humans probably would not detect this event neither.

In figure 10, we plot $P_{outlier}$ on a logarithmic scale for three representative hours of the day, and we show selected scenes classified as unusual. In the beginning of the timeline, some scenes are classified as unusual because nothing similar has been observed before, but they appear normal to a human. As soon as enough data is observed, the system is stable. Although it is hard to intuitively define when a scene should be unusual, the classified outliers are very plausible. Our algorithm detects unusual scenes with (i) global incidents, covering a large image region (10(f), 10(i), 10(j), 10(o)), or (ii) very atypical, local events (10(b), 10(d), 10(k), 10(n), 10(g), 10(c)). Local events that are too small to be detected directly as well as events related to motion can also be detected, if they affect a larger area of the scene. For example, our system can detect the illegally parked car in figure 10(p), because other cars are forced to cross the lane markings.

4. CONCLUSION

We have argued that several image mining applications call for a completely automatic pipeline. As examples, we have discussed ways to deal with images from landmarks and from webcams. As already stated in the intro, there are other source of images as well, which would require yet different ways of dealing with them.

Acknowledgments: The authors gratefully acknowledge the support by the Swiss SNF NCCR IM2 and the Swiss KTI for the extended landmark mining work, and the EC project HERMES (IST-027110) and Google for their support for the webcam summarization work.

5. REFERENCES

Figure 10: Detected unusual scenes from the Time Square dataset (see text). During the update, the images are subsequently added as new cluster centers to the memory, becoming usual. However, most probably they will not appear often and will thus be detected as rare.