Mining from large image sets (Keynote address)

Luc Van Gool
Computer Vision Laboratory
ETH Zurich – Switzerland

Michael D. Breitenstein
Computer Vision Laboratory
ETH Zurich – Switzerland

Stephan Gammeter
Computer Vision Laboratory
ETH Zurich – Switzerland

Helmut Grabner
Computer Vision Laboratory
ETH Zurich – Switzerland

Till Quack
Computer Vision Laboratory
ETH Zurich – Switzerland

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

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

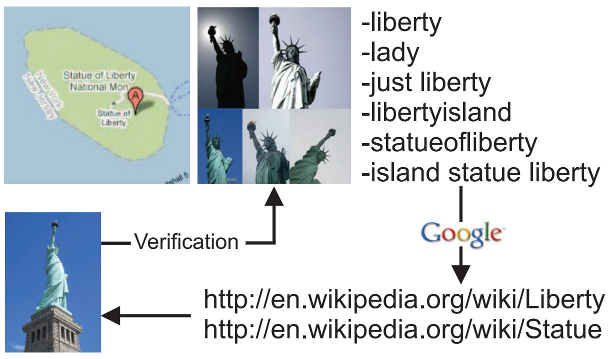


Figure 1: Example object cluster. *Top:* mined photo clusters with GPS location and tags. *Bottom right:* related content from Wikipedia. *Bottom left:* The images from the articles are used to match back to the clusters, as proposed in [21], which serves as a verification step for the articles' relevance.

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)

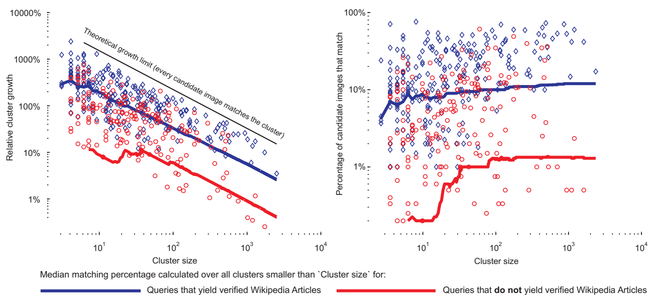


Figure 2: ***Left side:*** **The Y axis indicates by how much (in percent relative to the original cluster size) a cluster has grown by adding non geo-tagged photos. The black line is the theoretical limit of how much a cluster can grow, given that we consider at most 100 candidate images per itemset to enrich the cluster.** ***Right side:*** **Individual matching probabilities vs. cluster size, *i.e.*, what is the chance that an image returned as a search result matches the cluster.**

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 candidate 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.

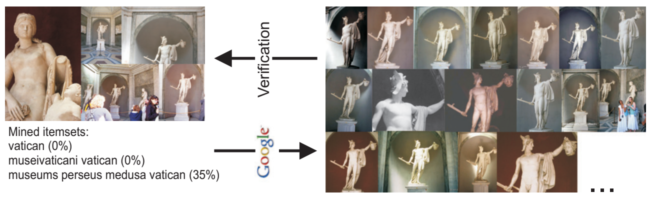

Verification
Mined itemsets:
vatican (0%)
museivaticani vatican (0%)
museums perseus medusa vatican (35%)
Google
...

Figure 3: A small cluster consisting only of 5 images showing the Perseus Statue in the Vatican Museum is enriched by 35 additional non geo-tagged images that were found via Google images. In this particular instance, this not only increase the cluster size but also the accuracy.

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.

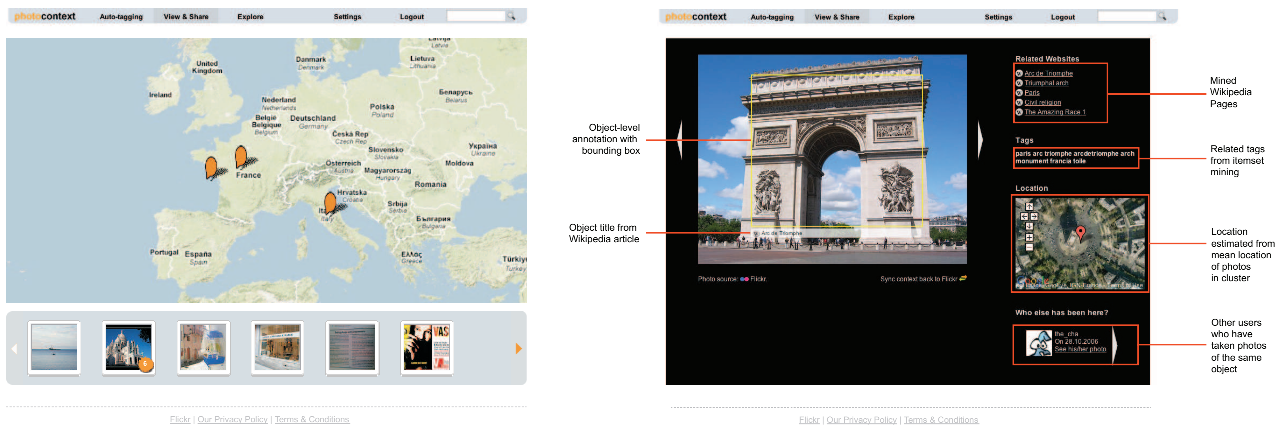

photocontext
Auto-tagging
View & Share
Explore
Settings
Logout
Flickr | Our Privacy Policy | Terms & Conditions
Object-level annotation with bounding box
Object title from Wikipedia article
Arc de Triomphe
Photo source: Flickr.
Sync context back to Flickr
Related Websites
Arc de Triomphe
Triumphal arch
Paris
Civil religion
The Amazing Race 1
Tags
paris arc triomphe arcdetriomphe arch monument francia toile
Location
Who else has been here?
the_cha
On 28.10.2006
See his/her photo
Mined Wikipedia Pages
Related tags from itemset mining
Location estimated from mean location of photos in cluster
Other users who have taken photos of the same object

Figure 4: Annotation tool user interface. *Left:* when annotating sets of images, users are asked to give limited supervision by dragging the set to the country the photos were taken in. While this is not absolutely necessary for good annotation results, it decreases recognition time per image. *Right:* typical annotation result. The multi-modal context of the image that was obtained from the data mining stage can be used to describe the full context of the photo, including related Wikipedia articles, relevant tags, location, and social features.

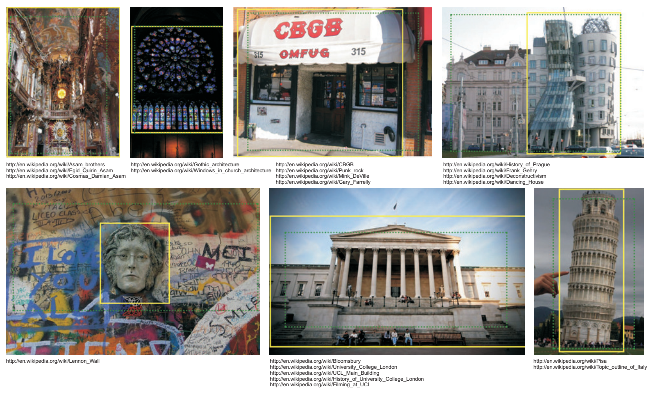

http://en.wikipedia.org/wiki/Asam_brothers
http://en.wikipedia.org/wiki/Egid_Quirin_Asam
http://en.wikipedia.org/wiki/Cosmas_Damian_Asam
http://en.wikipedia.org/wiki/Gothic_architecture
http://en.wikipedia.org/wiki/Windows_in_church_architecture
http://en.wikipedia.org/wiki/CBGB
http://en.wikipedia.org/wiki/Punk_rock
http://en.wikipedia.org/wiki/Mink_DeVille
http://en.wikipedia.org/wiki/Gary_Farrelly
http://en.wikipedia.org/wiki/History_of_Prague
http://en.wikipedia.org/wiki/Frank_Gehry
http://en.wikipedia.org/wiki/Deconstructivism
http://en.wikipedia.org/wiki/Dancing_House
http://en.wikipedia.org/wiki/Lennon_Wall
http://en.wikipedia.org/wiki/Bloomsbury
http://en.wikipedia.org/wiki/University_College_London
http://en.wikipedia.org/wiki/UCL_Main_Building
http://en.wikipedia.org/wiki/History_of_University_College_London
http://en.wikipedia.org/wiki/Filming_at_UCL
http://en.wikipedia.org/wiki/Pisa
http://en.wikipedia.org/wiki/Topic_outline_of_Italy

Figure 5: Results of automatic object-level annotation with bounding boxes. Groundtruth annotation is shown as solid yellow lines, the auto-annotation results are shown as green dashed lines.

Our approach looks for images that are sufficiently different from any images seen coming from the webcam before. These novel images automatically produce the mix we are after. At the beginning, the images may simply be as typical and usual as any other image. Yet, the system has not observed such images and will at that stage consider them sufficiently novel. Suppose the webcam observes a crossroad. Four months later, when it starts to snow, this will be unusual to the system as well. Hence, after a while the system will have collected a set of normal, but representative images of the scene. However, if something truly unusual happens, we also want to archive those images, again by detecting the novelty in comparison to all prior, collected images.

Although novelty detection is a classical task in computer vision, most previous approaches are not suitable for permanent, time-lapsed data streams. Many methods work with offline data and first learn a static *normal model* (or *normal concept*), which remains the same after a training phase is completed [12, 15, 28, 30]. The normal model represents typical situations observed in the data stream. However, for permanent data streams it might need to change constantly (*concept drift*), thus the model needs to be learnt online and adapted permanently. Secondly, most algorithms are based on information that is very difficult to extract robustly from webcams. Typically, the framerate is very low or even not constant, which prevents the use of optical flow [2, 14] or object tracking [10, 16, 25]. To handle these difficulties, our approach is purely data driven and based on simple, static features.

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 $S = \{\mathbf{x}_1, \ldots, \mathbf{x}_{t-1}\}$ (*e.g.*, of images that are represented by feature vectors $\mathbf{x}_i \in \mathbb{R}^n$ in a high dimensional space), our target is to classify the next example $\mathbf{x}_t$ 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 Δt (*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-

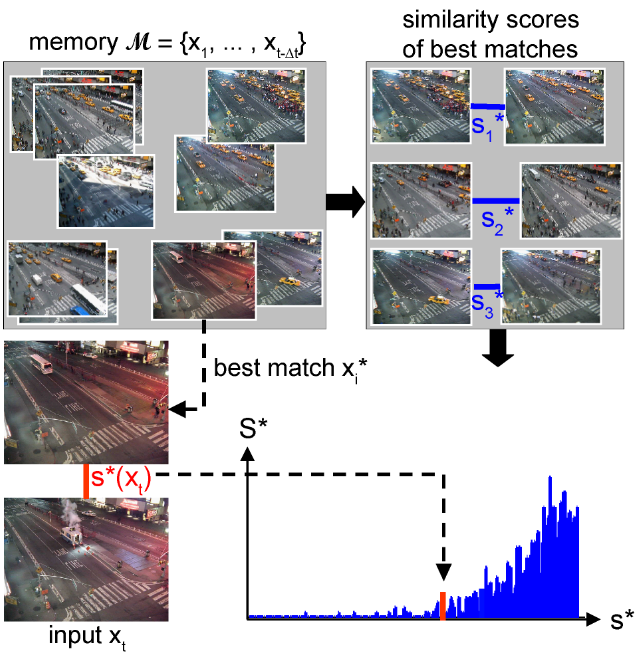

memory $\mathcal{M} = \{x_1, \dots, x_{t-\Delta t}\}$
similarity scores of best matches
s_1^*
s_2^*
s_3^*
best match x_i^*
S^*
$s^*(x_t)$
s^*
input x_t

Figure 6: Our approach for unusual scene detection using meaningful nearest neighbours: The similarity score $s^\star(\mathbf{x}_t)$ of the input $\mathbf{x}_t$ and $\mathbf{x}_i^\star$, the best matching sample from the memory $\mathcal{M}$, is compared to the distribution $S^\star$ of similarity scores $s^\star$.

ning, more or less all scenes are equivalently frequent; over time the algorithm finds a more and more representative subset.

3.2 Meaningful Nearest Neighbours

Data Representation Because images from webcams are typically captured with low resolution and low, even non-constant framerate, our method is based on static, low level features. The algorithm computes *Histogram of Oriented Gradient* 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 $\mathbf{f}_1, \mathbf{f}_2$ of two data points are compared using normalized cross correlation.

Classification A simple approach to detect an unusual scene based on the previous definitions is to store all observed features in a memory

$$\mathcal{M} = \{\mathbf{x}_1, \dots, \mathbf{x}_{t-\Delta t}\}, \tag{1}$$

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

$$s^\star(\mathbf{x}_t) = \max_{\mathbf{x}_i \in \mathcal{M}} \operatorname{sim}(\mathbf{x}_t, \mathbf{x}_i) \tag{2}$$

is compared to a threshold for classification, where $\operatorname{sim}(\cdot,\cdot)$ 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 $s^\star(\mathbf{x}_t)$ of the current input and the best match in the memory with the distribution $S^\star$ 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 $S^\star$ is not meaningful yet. However, as time goes by, 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 $\mathbf{x}_t$ is then classified as an inlier (and thus as *usual*), if its similarity score $s^\star(\mathbf{x}_t)$ is likely to be generated from the distribution $S^\star$. Consequently, our algorithm computes the inlier probability P_{usual} of a new data point $\mathbf{x}_t$ by comparing its similarity score $s^\star(\mathbf{x}_t)$ with the cumulative probability distribution $F_{S^\star}(\cdot)$ of $S^\star$

$$P_{usual}(s^\star(\mathbf{x}_t)) = \int_{s=0}^{s^\star(\mathbf{x}_t)} S^\star(s)ds = F_{S^\star}(s^\star(\mathbf{x}_t)) \tag{3}$$

Finally, $\mathbf{x}_t$ is classified as *unusual* using the probability threshold p_{alarm}

$$P_{usual}(s^\star(\mathbf{x}_t)) < p_{alarm}. \tag{4}$$

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

$$s^\star(\mathbf{x}_t) < F_{S^\star}^{-1}(p_{alarm}) = \theta_{alarm}. \tag{5}$$

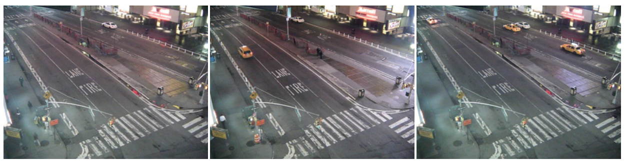
Advantages Using meaningful nearest neighbours for novelty detection has several benefits. Most importantly, no scene-specific, manually tuned similarity threshold for the classification based on the distance to the nearest neighbour is used. Such a threshold would have to be changed permanently, as 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.

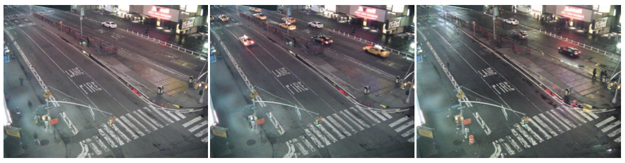
3.3 Clustering

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 Δt (*e.g.*, a day), which is divided by its age (*i.e.*, a multiple of Δt). 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 centers 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.


(a) After the first day.


(b) After one week.

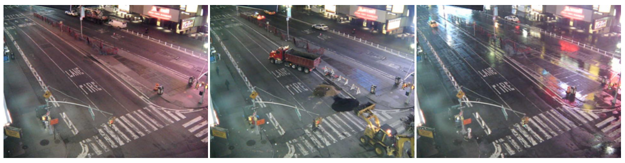

(c) After two months.

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.

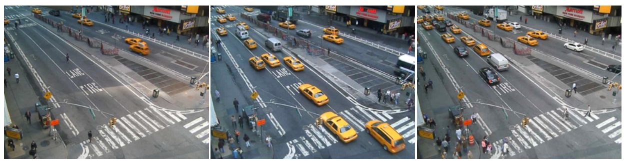
3.4 Implementation

The system consists of two parts. A *maintenance phase* is conducted regularly after Δt, *e.g.*, daily, during which the clustering algorithm integrates newly observed data. Then, $S^\star$ and θ_{alarm} are computed from the updated model. Note that θ_{alarm} can be calculated for the first time after $2\Delta t$ because it relies on $S^\star$. During the online *detection phase*, the algorithm classifies outliers in real-time based on Eq. (5).

This evaluation is very efficient, because θ_{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 correspond to the observation (*i.e.*, having the same time modulo the pre-specified Δt). This not only results in speed-up, but also adds *time dependency* 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 640×480 pixels, resulting in more than 0.5 TB of data. The datasets covers scenes that capture a large variety of potentially interesting events under a large 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.


(a) 8 am – 9 am

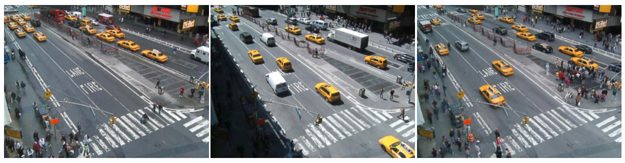
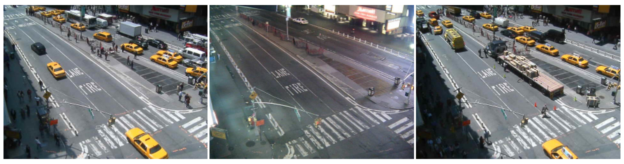

(b) 12 pm – 1 pm

Figure 8: The most different of the 20 most frequent clusters (representative images) found by our algorithm for different times of the day after observing the scene for two months.


(a) Input image.

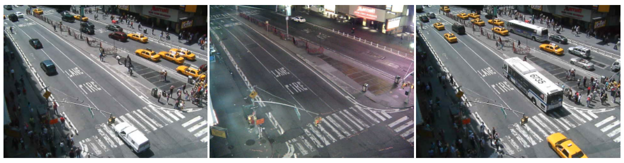

(b) Best matching cluster center.

Figure 9: ***(a)*** **Input images from the** ***Time Square*** **dataset that are classified as usual, together with their best matches** ***(b)*****. The right image should rather be classified as unusual but is too similar to the best match.**

Parameter Settings We choose the parameters $\theta_{mms} = 0.95$ and $p_{alarm} = 1\%$, 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_{usual} 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(q), 10(r)). 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

[1] Time Square Webcam. http://images.earthcam.com/ec_metros/ourcams/mpstudiosouth.jpg, 2008/04/01 – 2008/05/20.

[2] A. Adam, E. Rivlin, I. Shimshoni, and D. Reinitz. Robust real-time unusual event detection using multiple fixed-location monitors. *Trans. PAMI*, 30(3):555–560, 2008.

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

[4] K. Beyer, J. Goldstein, R. Ramakrishnan, and U. Shaft. When is "nearest neighbor" meaningful. In *Int. Conf. on Database Theory*, 1999.

[5] M. Breitenstein, H. Grabner, and L. V. Gool. Hunting nessie – real-time abnormality detection from webcams. Technical report, Computer Vision Laboratory, ETH Zürich, 2009.

[6] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *CVPR*, 2005.

[7] S. Gammeter, L. Bossard, T. Quack, and L. Van Gool. I know what you did last summer: object level auto-annotation of holiday snaps. In *ICCV*, 2009.

[8] M. Goesele, N. Snavely, B. Curless, H. Hoppe, and S. M. Seitz. Multi-view stereo for community photo collections. In *ICCV*, pages 265–270, Rio de Janeiro, Brazil, 2007. IEEE.

[9] I. D. Guedalia, M. London, and M. Werman. An on-line agglomerative clustering method for nonstationary data. *Neural Computation*, 11(2):521–540, 1999.

[10] W. Hu, X. Xiao, Z. Fu, D. Xie, F.-T. Tan, and S. Maybank. A system for learning statistical motion patterns. *Trans. PAMI*, 28(9):1450–1464, 2006.

[11] N. Jacobs, N. Roman, and R. Pless. Consistent temporal variations in many outdoor scenes. In *CVPR*, 2007.

[12] N. Johnson and D. Hogg. Learning the distribution of object trajectories for event recognition. In *BMVC*, 1996.

[13] J. Lalonde, S. Narasimhan, and A. Efros. What does the sky tell us about the camera? In *ECCV*, 2008.

[14] J. Li, S. Gong, and T. Xiang. Scene segmentation for behaviour correlation. In *ECCV*, 2008.

[15] D. Makris and T. Ellis. Learning semantic scene models from observing activity in visual surveillance. *Trans. on Systems, Man, and Cybernetics*, 35(3):397–408, 2005.

[16] N. Oliver, B. Rosario, and A. Pentland. A bayesian computer vision system for modeling human interactions. *Trans. PAMI*, 22(8):831–843, 2000.

[17] D. Omercevic, O. Drbohlav, and A. Leonardis. High-dimensional feature matching: Employing the concept of meaningful nearest neighbors. In *ICCV*, 2007.

[18] M. Perdoch, O. Chum, and J. Matas. Efficient representation of local geometry for large scale object retrieval. In *CVPR*, 2009.

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

[20] Y. Pritch, A. Rav-acha, A. Gutman, and S. Peleg. Webcam synopsis: Peeking around the world. In *ICCV*, 2007.

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

[22] J. Sivic, M. Everingham, and A. Zisserman. “Who are you?” – Learning person specific classifiers from video. In *CVPR*, 2009.

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

[24] N. Snavely, S. M. Seitz, and R. Szeliski. Photo tourism: Exploring photo collections in 3d. In *SIGGRAPH*, 2006.

[25] C. Stauffer and W. E. L. Grimson. Learning patterns of activity using real-time tracking. *Trans. PAMI*, 22(8):747–757, 2000.

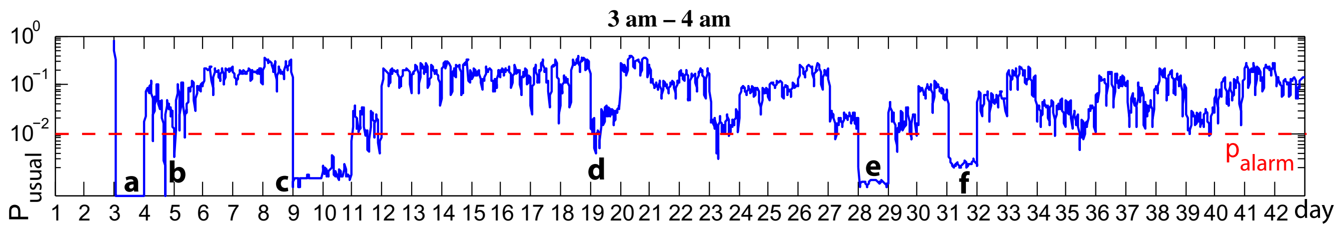
[26] K. Sunkavalli, F. Romeiro, W. Matusik, T. Zickler, and H. Pfister. What do color changes reveal about an outdoor scene? In *CVPR*, 2008.

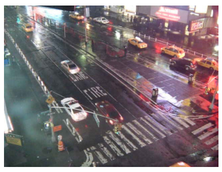
[27] M. Vergauwen and L. Van Gool. Web-based 3d reconstruction service. *MVA*, 17(6):411–426, 2006.

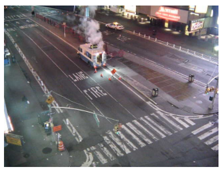
[28] X. Wang, K. Ma, G. Ng, and W. Grimson. Trajectory analysis and semantic region modeling using a nonparametric bayesian model. In *CVPR*, 2008.

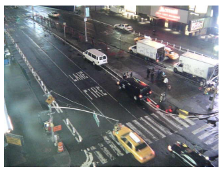
[29] Y. Weiss. Deriving intrinsic images from image sequences. In *ICCV*, 2001.

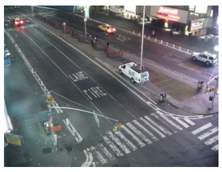
[30] H. Zhong, J. Shi, and M. Visontai. Detecting unusual activity in video. In *CVPR*, 2004.

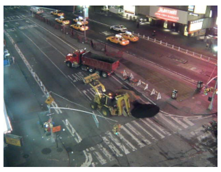


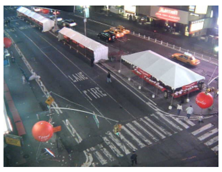

(a) Rain.

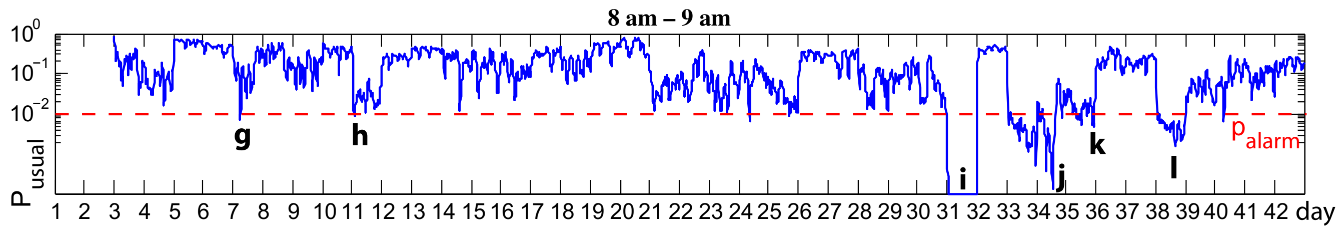

(b) Smoke.

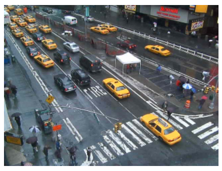

(c) Parking trucks.

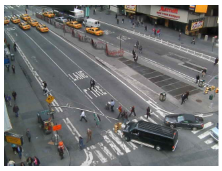

(d) Antenna car.

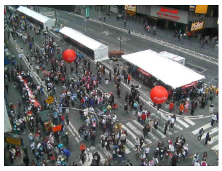

(e) Construction trucks.

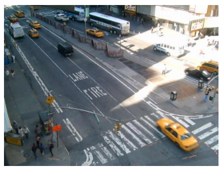

(f) Big tents.

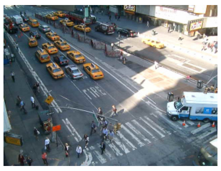


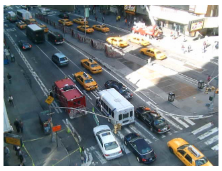

(g) Tent.

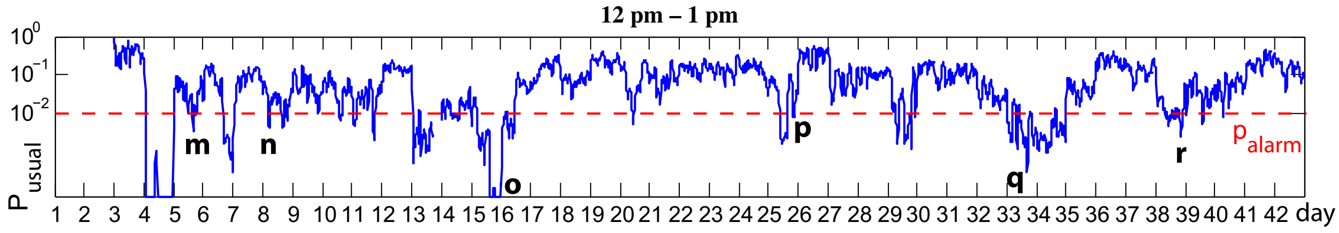

(h) Turning cars.

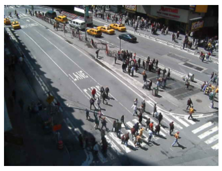

(i) Big tents.

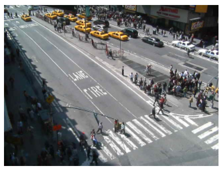

(j) Shadow shape.

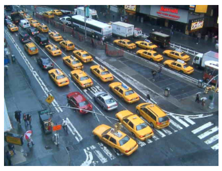

(k) Parked car.

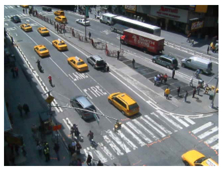

(l) Roadwork.

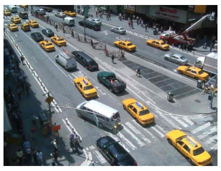



(m) People on street.


(n) Parking police cars.


(o) Jam.


(p) Stopping car.


(q) Crane.

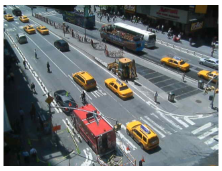

(r) Roadwork.

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.