Hunting Nessie – Real-Time Abnormality Detection from Webcams

Michael D. Breitenstein[1] Helmut Grabner[1] Luc Van Gool[1,2]
[1]Computer Vision Laboratory ETH Zurich
[2]ESAT-PSI / IBBT KU Leuven

Abstract

*We present a data-driven, unsupervised method for unusual scene detection from static webcams. Such time-lapse data is usually captured with very low or varying framerate. This precludes the use of tools typically used in surveillance (*e.g.*, object tracking). Hence, our algorithm is based on simple image features. We define usual scenes based on the concept of meaningful nearest neighbours instead of building explicit models. To effectively compare the observations, our algorithm adapts the data representation. Furthermore, we use incremental learning techniques to adapt to changes in the data-stream. Experiments on several months of webcam data show that our approach detects plausible unusual scenes, which have not been observed in the data-stream before.*

1. Introduction

Novelty detection is a classical task in computer vision. Unfortunately, most approaches are not suitable for permanent, time-lapsed data-streams like webcam footage. First, many methods are based on a constant, previously trained model of normality [15, 19, 31, 33]. However, the *normality concept* in permanent data-streams might change constantly (concept drift), thus the model needs to be learnt online and adapted permanently. Second, 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 [5, 18] or object tracking [13, 21, 28].

To handle these difficulties, we present a purely data-driven approach, which is not object-class specific. The method is based on simple, static features and aims at detecting atypical configurations in a scene. Although a single incident may be too small to be captured directly, it can affect a greater part of the scene, thus changing the context (*c.f.*, [30]). Similarly, unusual motion (*e.g.*, the speed of a car) can also cause abnormal constellations, which can be found. For example, our system can detect the illegally parked car in Fig. 1 (top left, in the center of the image), because other cars are forced to cross the lane markings.

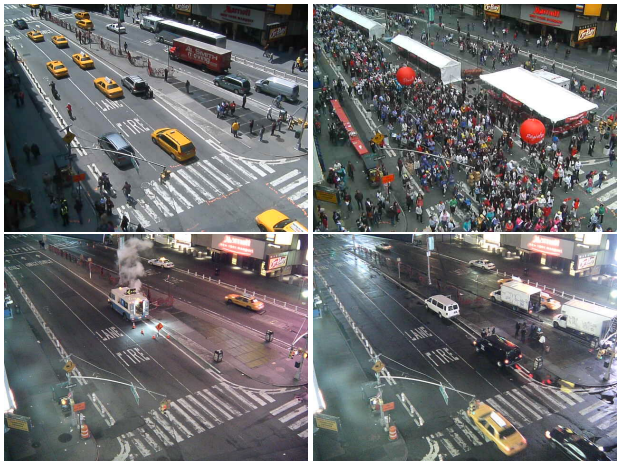


Figure 1. Can you spot the abnormalities in these scenes, which are captured from a public webcam – like our system?

Over the last years, data-driven methods that use large collections of publicly available images (*e.g.*, from Flickr) have been applied to many tasks in computer vision (*e.g.*, [12, 24, 30]). Recent years also witnessed the proliferation of publicly accessible webcams and surveillance cameras. They record time-lapse data over long periods of time. A growing amount of work is being published to explore such data as well. However, most work focuses on estimating imaging conditions such as illumination and color changes, which can be used, *e.g.*, for camera geo-location [16, 29], shadow removal [32], seasonal variation detection [14], or video synopsis [25]. In contrast to previous work, we are primarily interested in capturing and reasoning about the content of the scene itself. Our target is to automatically learn what usually happens in a scene, and to detect anomalies, such as those shown in Fig. 1.

In this paper, we first give our definition of unusual scene detection and then propose a new approach for permanent data streams, which employs the concept of meaningful nearest neighbours [6, 22]. The main idea is that everything is usual that has been observed in the past (or in a specified time-frame, respectively). Our algorithm checks if a new observation is a statistical outlier and thus should be classified as unusual, while constantly adapting its model

of normality. There exists a substantial body of work to effectively compare high-dimensional data by learning a data-dependent similarity measure, *e.g.*, [20, 9]. In contrast, we keep the similarity measure fixed and adapt the data representation instead. For this purpose, our algorithm selects the most representative subset of a hierarchical feature set, depending on the complexity of the data.

Using meaningful nearest neighbours for classification has several advantages. Most importantly, it allows the model to adapt permanently, and to classify outliers without having to manually define a nearest neighbour distance threshold, which would have to change constantly as more data is observed. Instead, our approach classifies a data point relative to previous observations by considering the distribution of nearest neighbour distances in the memory.

We show that our approach is suitable to effectively detect plausible unusual scenes in webcam footage. Possible applications include a real-time alert system for surveillance cameras, where a human observer typically has to monitor a large number of cameras at the same time [8]. To support a human supervisor, our algorithm can rank different video streams. Thus, the supervisor can focus his attention to those streams where it is much likely needed currently. Furthermore, more cameras can be observed by one supervisor. A second application is a system to automatically generate a summary of the past day for a particular webcam, by selecting interesting, out-of-the-ordinary images (see [10]).

The paper is structured as follows. In Sec. 2, we present our approach, which raises the issue of data representation, addressed in Sec. 3. In Sec. 4, we describe the implementation of the system and show that it is suitable to detect plausible unusual scenes in Sec 5.

2. Abnormality Detection from Continuous Data-Streams

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. Many existing methods for abnormality detection work with offline data and previously learn a static *model* (or *normal concept*), which remains the same after a training phase is completed. However, data distributions in continuous data-streams are hardly ever constant. Therefore, it is likely that the normal concept changes over time, either temporally or permanently [27].

2.1. Abnormal, Unusual and Rare Scenes

To define an *abnormal* scene, we differentiate between the terms *unusual* and *rare*. A scene is usual if a very *similar* scene has been observed at least once in the past (*c.f.*, [33]), formulated as follows:

$$\exists \mathbf{x}_i, i \in \{1, \ldots, t - \Delta t\} : \mathbf{x}_i \sim \mathbf{x}_t \tag{1}$$

Because data points that appear temporally close may be correlated, we introduce a time delay Δt (*e.g.*, a day). However, a usual scene can appear frequently or rarely. Thus, the frequency of a scene $\mathbf{x}_t$ is proportional to the number of data points it is similar to

$$\mathrm{card}(\{\mathbf{x}_i | \mathbf{x}_i \sim \mathbf{x}_t, i \in \{1, \ldots, t - \Delta t\}\}). \tag{2}$$

The differentiation between unusual and rare is important especially in the context of concept drift. 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. In this work, we primarily concentrate on finding *unusual scenes* but still show how to integrate *scene frequency* in our method. Similar to our definitions for scenes, the integration of temporal information would lead to the definition of the terms *activities* and *events*, which however are not the scope of this paper.

2.2. Outlier Classification using Meaningful Nearest Neighbours

A simple approach to detect an unusual scene from a continuous data stream based on these definitions is to store all observed data in a memory

$$\mathcal{M} = \{\mathbf{x}_1, \ldots, \mathbf{x}_{t-\Delta t}\}, \tag{3}$$

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

$$s^\star(\mathbf{x}_t) = \max_{\mathbf{x}_i \in \mathcal{M}} \mathrm{sim}(\mathbf{x}_t, \mathbf{x}_i) \tag{4}$$

is compared to a threshold for classification, where $\mathrm{sim}(\cdot, \cdot)$ denotes the similarity of two data points. Different similarity measures could be used, see [26] for a review.

However, it is not clear if the nearest neighbour is *meaningful* for a high-dimensional space [6], 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. Furthermore, such a similarity threshold would have to change permanently, as more data is observed and included in the memory.

We propose the following approach based on the concept of *meaningful nearest neighbours* (Fig. 2). 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

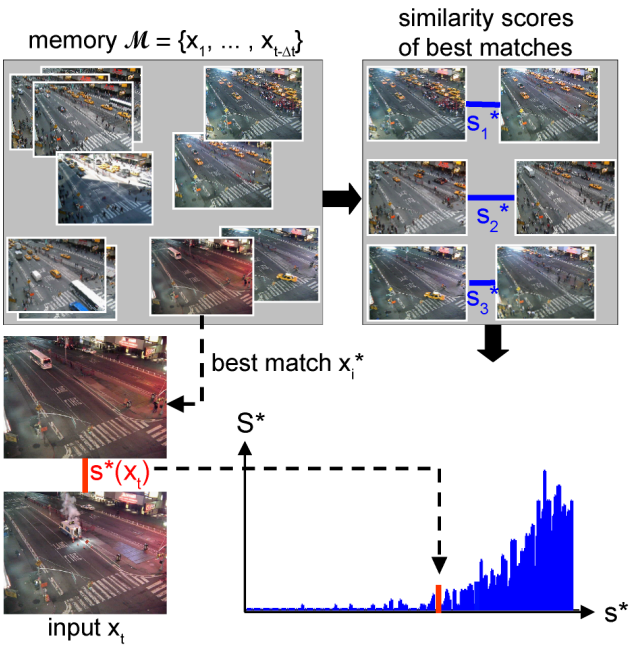


Figure 2. Our approach for unusual scene detection using meaningful nearest neighbours: The similarity score $s^\star(\mathbf{x}_t)$ of the input $\mathbf{x}_t$ and $\mathbf{x}_i^\star$, the best matching sample from the memory $\mathcal{M}$, is compared to the distribution $S^\star$ of similarity scores $s^\star$.

$\mathcal{M}$ obtained so far. In the beginning, $\mathcal{M}$ 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.

Then, a new data point $\mathbf{x}_t$ is 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 $S^\star$

$$P_{usual}(s^\star(\mathbf{x}_t)) = \int_{s=0}^{s^\star(\mathbf{x}_t)} S^\star(s)ds = F_{S^\star}(s^\star(\mathbf{x}_t)) \quad (5)$$

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

$$P_{usual}(s^\star(\mathbf{x}_t)) < p_{alarm}. \quad (6)$$

This can be rewritten using Eq. (5) and the inverse function $F_{S^\star}^{-1}(\cdot)$ of $F_{S^\star}(\cdot)$ to compute a threshold for the similarity score:

$$s^\star(\mathbf{x}_t) < F_{S^\star}^{-1}(p_{alarm}) = \theta_{alarm}. \quad (7)$$

This approach has several benefits. First, the normal model is purely data-driven. Thus, 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. Second, no prior knowledge is necessary, but could be incorporated by previously adding manually selected data to the memory. Third, no scene-specific, manually tuned similarity threshold for 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 in the memory. Thus, the model adapts automatically and permanently.

3. Adaptive Data Representation

In general, the observed data $\mathbf{x}_i \in \mathbb{R}^n$ spans a high dimensional space, which is populated only sparsely. Therefore, all data points are equivalently distant from each other, thus $S^\star$ might not be significant. Thus, we propose to adapt the *data representation*, depending on the model complexity and the observed data.

Model. A model $M := \langle \mathcal{C}, R(\cdot) \rangle$ consists of K cluster centers $\mathcal{C} = \{\mathbf{c}_1, \ldots, \mathbf{c}_K\} \subseteq \mathcal{X}$ and the representation function

$$R(\cdot) : \mathbb{R}^n \to \mathbb{R}^m, \quad (8)$$

which transforms the data points to a m-dimensional subspace (usually $m \ll n$). Clustering is applied to the transformed space, however, the cluster centers $\mathbf{c}$ are (selected) images themselves.

We evaluate how accurately the model M represents the data $\mathcal{X}$ by computing the *mean of the maximal similarity*

$$mms(\mathcal{X}, \mathcal{C}, R(\cdot)) = \frac{1}{|\mathcal{X}|} \sum_{\mathbf{x} \in \mathcal{X}} \max_{\mathbf{c} \in \mathcal{C}} \mathrm{sim}(R(\mathbf{x}), R(\mathbf{c})). \quad (9)$$

The mms-criterion measures the average similarity of observations $\mathbf{x} \in \mathcal{X}$ and their best matching cluster centers $\mathbf{c} \in \mathcal{C}$. If the mms-value is high, the data is represented exactly.

Representation. Every point belonging to the underlying data generation process should be represented *as exactly as possible* by our model. More formally, given a fixed K, the representation function $R(\cdot)$ should be chosen such that the mms-value is above a threshold θ_{mms}, *i.e.*,

$$mms(\mathcal{X}, \mathcal{C}, R(\cdot)) \geq \theta_{mms}. \quad (10)$$

On the other hand, the cluster centers should be *diverse* in order to capture the whole variety of the data. For example, although the trivial representation $R(\mathbf{x}) = 1$ fulfills Eq. (10), every data point would collapse to the same point. This is obviously not a good representation, although resulting in $mms(\mathcal{X}, \mathcal{C}, R(\cdot)) = 1$ (the maximum value). Thus, we are interested in a representation that describes the data as exact as possible but is still diverse enough to differentiate between cluster centers, *i.e.*, the individual cluster centers should be distant to each other.

We propose to adjust the system parameter θ_{mms} to adapt the diversity of the resulting model. To find a good tradeoff between diversity and accuracy, we select an appropriate data representation according to

$$R^{\star}(\cdot) = \arg\min_{R(\cdot)} |mms(\mathcal{X}, \mathcal{C}, R(\cdot)) - \theta_{mms}|. \quad (11)$$

Selection. We assume that the operator $R(\cdot)$ turns the raw image data into a hierarchy of extracted feature values (*e.g.*, a tree). When going to the next level, the features there convey more detailed information about the image, *e.g.*, by extracting information from image subparts, which can then be subdivided further at the next level. Thus, the resulting representation can be refined step-wise to capture more and more information from the underlying levels. Given this kind of image representation, we optimize Eq. (11) to let $R(\cdot)$ select the most relevant parts of that information. In order to make this selection, we use a greedy, breadth-first search, as follows. Having selected a set of subparts at the level where the optimization process has arrived, we then check for all their subparts at the next level. Those parts for which subparts are selected, are from then on only represented through these subparts, *i.e.*, the original part is deactivated and only the subparts at the next level are now selected. This process continues until we have selected subparts at the lowest (most refined) level or until adding a level no longer decreases $|mms(\mathcal{X}, \mathcal{C}, R(\cdot)) - \theta_{mms}|$.

Discussion. The number of cluster centers K is usually fixed in practice (restricted by the memory, as each center is a representative image). Therefore, our approach has only one free parameter θ_{mms}, which specifies the tradeoff between generalization and discrimination of the model. The optimal representation $R(\cdot)$, which projects the data to the m-dimensional space, depends on the number of cluster centers K and the threshold θ_{mms}:

$$m \sim K, \quad m \sim \frac{1}{\theta_{mms}}. \quad (12)$$

Our method is related to subspace clustering [23]. However, it constrains the search space by making use of the hierarchical structure of possible data representations.

3.1. Representative Concatenated Histograms

As our specific choice of feature hierarchy, we build pyramids of feature histograms organized using a quad-tree data structure (akin to [17]). Hence, a refinement step here means splitting a part into its 4 quadrants and replacing the part's histogram by a selection of quadrant histograms. Those quadrants that are not selected are said to be masked. The histograms of all selected image parts (throughout the hierarchy) are simply concatenated. We coin this representation *representative concatenated histograms* or *RCH*. The procedure is visualized in Fig. 3.

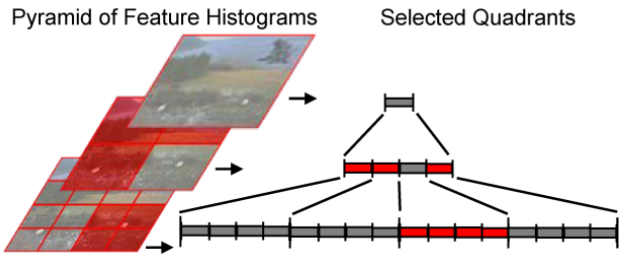


Figure 3. Representative concatenated histogram selection: Based on a generated pyramid of feature histograms, quadrants (red) are selected by our algorithm, which are composed to the final feature vector.

Applied to our task of unusual scene detection, the *RCH* method has several advantages. An image region deviating strongly from the corresponding regions in other images can be represented more closely by refining respective image quadrants. Furthermore, the calculated image representation is still interpretable, which allows to localize image regions that do not match well.

4. Implementation

In this Section, we describe how the approach in Sec. 2 is implemented. Because not all data from a continuous datastream can be stored in practice, the memory $\mathcal{M}$ is replaced by the model from Sec. 3.

Features. We choose static, low level features because of the properties of our data source (webcams with low, varying framerate). Our algorithm computes *Histogram of Oriented Gradient* features [7] by applying Sobel filters on 8×8 image cells and equally dividing the orientation range to 8 bins. In practice, only the histograms of the lowest level are stored because higher-level histograms can be generated based on sub-histograms. For simplicity and efficiency, the features $\mathbf{f}_1, \mathbf{f}_2$ of two data points are compared using normalized cross correlation.[1]

Clustering. We apply an online, agglomerative clustering algorithm similar to [11]. 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

[1] We experienced that the performance did not improve by individually normalizing sub-histograms instead of the complete, assembled feature.

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 (see Sec. 2.1) that everything is usual that has been observed in the past. Alternatively, those clusters could be removed instead which were not best matches for a long time. In addition, similar cluster center statistics could be used to also classify *rare* scenes, according to Sec. 2.1.

4.1. System

The system consists of two parts. During an online *detection phase*, the algorithm classifies outliers in real-time, while the model is updated regularly (after Δt, *e.g.*, daily) during a *maintenance phase*.

Maintenance Phase. First, the clustering algorithm integrates newly observed data as described before. Second, the algorithm checks if a better data representation can be achieved according to Eq. (11). Instead of only refining a certain histogram by *splitting* parts (see Sec. 3.1), the algorithm also checks if a representation is more suitable according to Eq. (11) that is created by *merging* corresponding quadrant histograms. Finally, $S^\star$ 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 $S^\star$. As described in Sec. 2.2 and will be demonstrated in the experiments, the system will get increasingly stable as more and more scenes are observed.

Detection Phase. The first detection phase starts after having processed data from $2\Delta t$ (see above). During the online detection phase, an observation is classified based on Eq. (7). 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 in Sec. 2.1, 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 frequent cluster centers. Third, the search space can be reduced by only considering cluster centers with timestamps that correspond to the observation (*i.e.*, 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).

5. Experiments

To demonstrate how the most important parts of the system work, we perform two synthetic experiments using toy examples. Such controlled experiments allow to better understand certain properties than using real data, where several effects come together. Second, we evaluate the algorithm on two interesting and challenging datasets captured from public webcams.

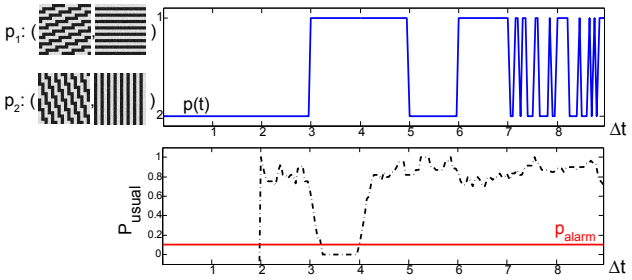


Figure 4. Samples are drawn time-dependently from two distributions p_1, p_2 (top). Based on P_{usual} and p_{alarm}, they are classified (below). After $2\Delta t$, samples from p_2 are in the memory and thus classified as inliers. In contrast, because p_1 has not been observed until $3\Delta t$, they are classified as outliers. After $4\Delta t$, samples from p_1 and p_2 are in the memory and thus classified as inliers.

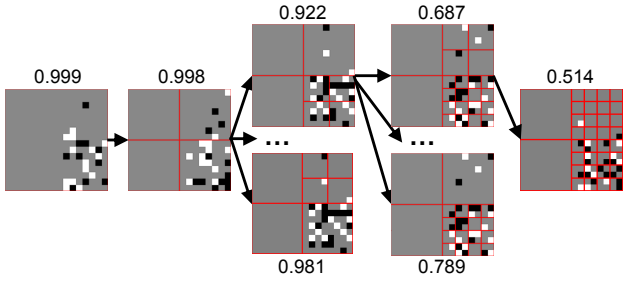


Figure 5. For a dataset with a varying amount of noise in the different image quarters, the generation of features from different resolution layers is illustrated. An image region deviating strongly from the corresponding regions in other images can be represented more closely by refining respective image quadrants. Our algorithm selects a representation based on the shown mms-values.

5.1. Toy Examples

Outlier Detection (Sec. 2). We create two distributions p_1, p_2 and randomly draw 20 samples during a time span Δt from either p_1, p_2, or from both. Each distribution consists of a pair of patterns (see images shown in Fig. 4), from which one is drawn and altered by random noise. After each time period Δt, the memory (consisting of $K = 10$ samples) is updated, without adapting the feature resolution (Sec. 3) for simplicity. After updating the memory after $2\Delta t$, all samples from p_2 are classified as inliers based on P_{ususal} (see Fig. 4, p_{alarm} is 10%). In contrast, after $3\Delta t$, samples from p_1 are observed for the first time, which are detected as outliers. After the following maintenance phase, the memory contains samples from both distributions, such that both are detected as inliers henceforth. For now on, although only samples from one distribution would be drawn for a long time, samples from the other distribution will remain in the memory (because of our clustering method, see 4) and will be classified as inlier.

Adaptive Data Representation (Sec. 3). We perform an experiment using a data-stream that consists of 20 uniform images per time step. The images are altered by a

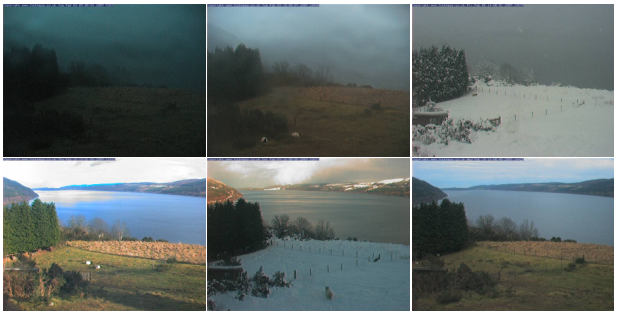


Figure 6. Typical scenes (cluster centers) for *Loch Ness*.

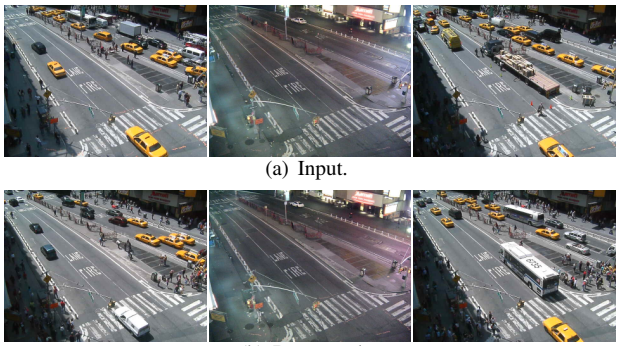
varying amount of random salt and pepper noise, depending on the different image quarters. While the left side of the image remains empty, noise is added that affects in average 10% of the pixels in the top right quarter and about 50% of the pixels in the bottom right quarter. The generated representations are illustrated in Fig. 5. The feature resolution is increased (from left to right) by replacing a single histogram with four sub-histograms to represent a certain image region more accurately, and vice versa. The *mms*-values characterize the resulting models, based on which our algorithm selects the data representation.

5.2. Real Data

Datasets. We recorded two datasets (resolution 640×480 pixels, more than 0.5 TB of data) from public webcams in New York (US) [4] and at lake Loch Ness (UK) [2]. The dataset *Time Square* is captured around the clock during two months with a framerate of 16 images per minute, and the dataset *Loch Ness* during one month with a framerate of 3 images per minute. As can be seen from sample images (Figs. 6, 7), the datasets cover 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.*). However, individual objects such as people, cars or animals have a size of only a few pixels, such that object detection and tracking is difficult. As mentioned before, the framerate is very low.

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. Dominant cluster centers for *Loch Ness* are visualized in Fig. 6. For *Time Square*, Fig. 7(b) shows



(a) Input.

(b) Best match.

Figure 7. These input scenes (a) from the *Time Square* dataset are classified as usual based on their best matches (b). However, the right image should intuitively be classified as unusual.

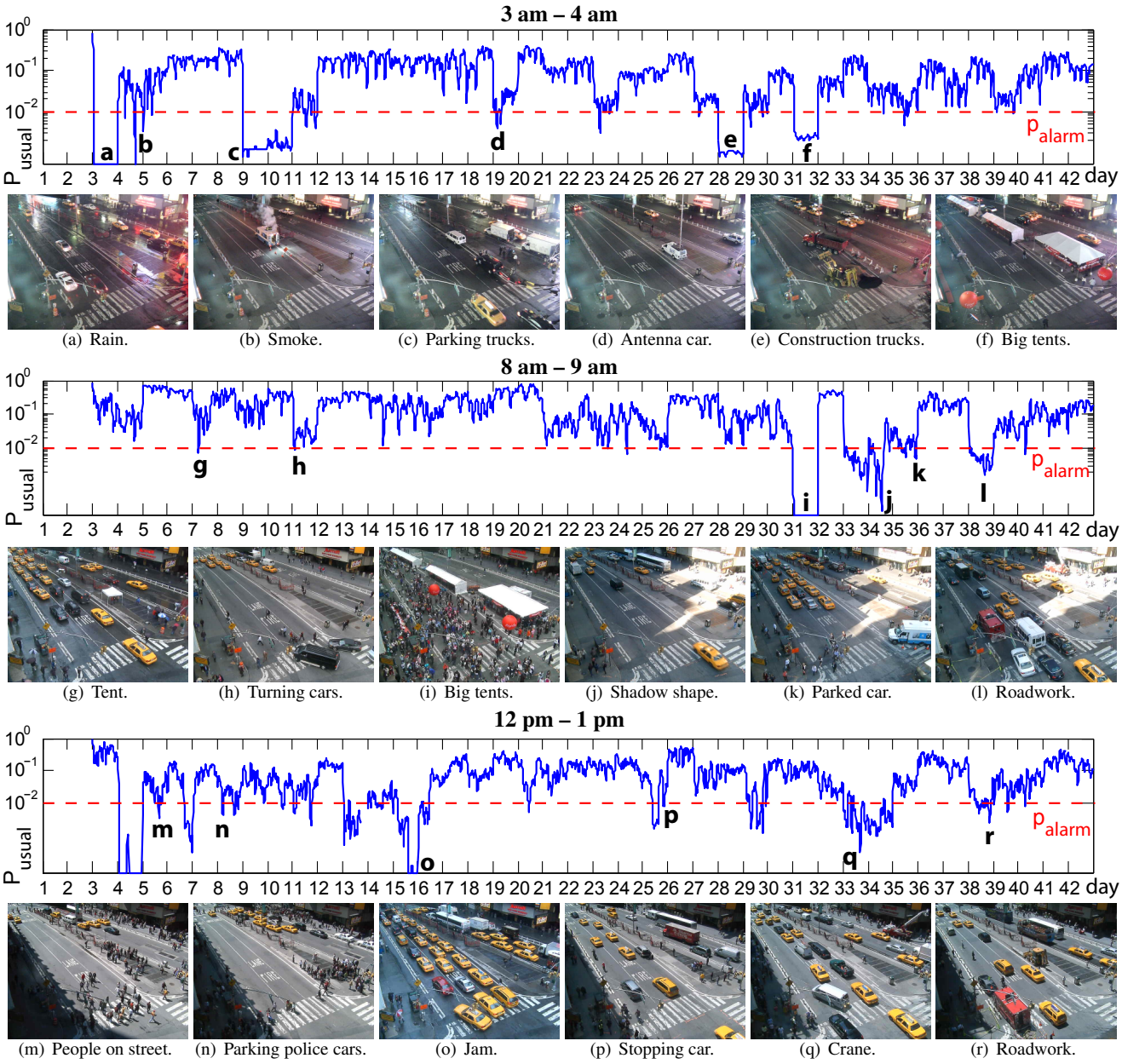
typical matches found for the input in Fig. 7(a). The first two scenes are correctly classified as usual. In the right image, a low-loading truck has stopped at the roadside, which intuitively should be an unusual scene. Because an exceptionally similar image is found (below), this scene is also classified as usual. However, by simply looking at this image, humans probably would not detect this event neither.

In Figs. 8 and 9, we plot P_{usual} on a logarithmic scale for the whole time period, and the resulting unusual scenes are shown. For *Time Square*, we select three representative hours of the day; *Loch Ness* is shown at once. 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 (see Fig. 8(a)). 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 (8(f), 8(i), 8(j), 8(o)), or (ii) *very atypical, local events* (8(b), 8(d), 8(k), 8(n), 8(q), 8(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 Fig. 8(p), because other cars are forced to cross the lane markings.

The unusual scenes in the first half of the *Loch Ness* dataset are mainly caused by weather conditions (Figs. 9(a)–9(e)). However, after the usual variety of conditions appeared in the data, the system becomes robust (see P_{usual} in Fig. 9). To demonstrate that our algorithm would be able to detect the monster *Nessie*[2], we manually inserted an artificial image [1] (Fig. 8(f)). Details are shown in Fig. 10. The algorithm finds the best matching cluster center and current feature representation (Fig. 10(b)). Furthermore, the monster is localized by computing the similarities of the individual feature histograms (Fig. 10(c)).

[2]Nessie is a legendary monster living in a lake in the UK. [3]



(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 8. Detected unusual scenes from the *Time Square* dataset (see text; best viewed magnified and in color). Videos are available on: `http://www.vision.ee.ethz.ch/~bremicha`.

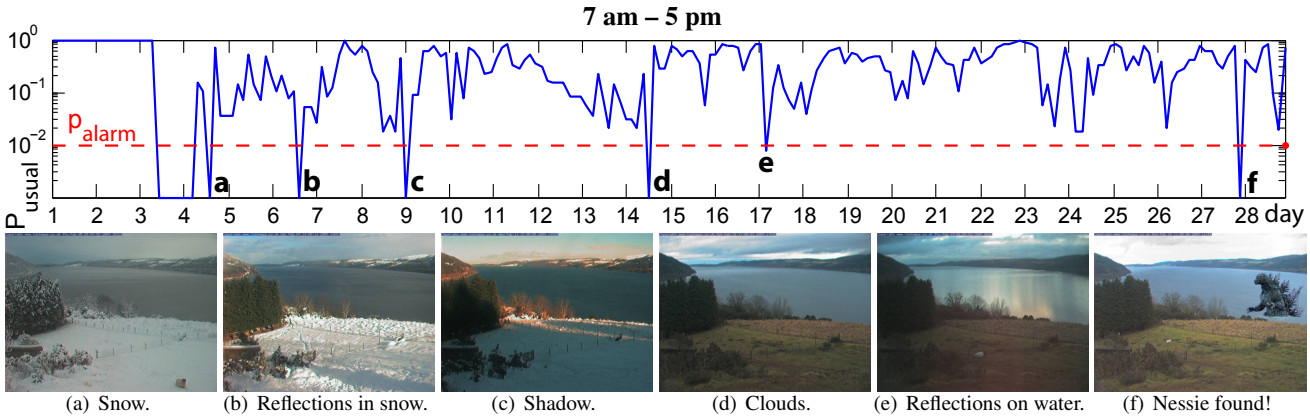
6. Conclusion

We presented a novel approach for real-time abnormality detection from continuous streams of webcam footage. The main idea is that everything observed in the past is usual. Therefore, our method classifies unusual scenes using the concept of meaningful nearest neighbours, which has several advantages. In order to focus on informative parts of the scene, we automatically adapt the data representation. To handle concept drift, the algorithm incrementally integrates new observations. Our experiments demonstrate that our method finds plausible unusual scenes in several months of recorded webcam data. In the future, we will further exploit statistics of cluster centers. For example, typical time-dependencies between scenes can be learned and specific scenes can be recognized by incorporating additional information *e.g.*, from cross-media content or weak annotations.

Acknowledgments: We gratefully acknowledge support by the EU project HERMES (IST-027110) and Google.

References

[1] Loch Ness Monster, ©Dan Blake. `http://www.lochness.co.uk/livecam/lochnessmonster.html`, 2009/01/17.



(a) Snow. (b) Reflections in snow. (c) Shadow. (d) Clouds. (e) Reflections on water. (f) Nessie found!

Figure 9. Detected unusual scenes from the *Loch Ness* dataset. After having observed different weather and illumination conditions, the system is stable and detects the Loch Ness monster!

(a) (b) (c)

Figure 10. The artificial input of *Nessie* (a) and the matching cluster center (b), with automatically selected feature representation (b). The monster is located by computing the similarities of the individual feature histograms (c) (higher intensity means lower similarity score).

[2] Loch Ness Webcam. `http://www.lochness.co.uk/livecam/img/lochness.jpg`, 2009/02/02 – 2009/03/02.

[3] Nessie, the Loch Ness Monster. `http://en.wikipedia.org/wiki/Loch_Ness_Monster`.

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

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

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

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

[8] H. Dee and S. Velastin. How close are we to solving the problem of automated visual surveilla @nce? *Machine Vision and Applications*, 19(5-6):329–343, 2008.

[9] A. Frome, Y. Singer, F. Sha, and J. Malik. Learning globally-consistent local distance functions for shape-based image retrieval and classification. In *ICCV*, 2007.

[10] L. V. Gool, M. D. Breitenstein, S. Gammeter, H. Grabner, and T. Quack. Mining from large image sets. In *CIVR*, 2009.

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

[12] J. Hays and A. A. Efros. Scene completion using millions of photographs. *SIGGRAPH*, 2007.

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

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

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

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

[17] S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *CVPR*, 2006.

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

[19] 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.

[20] E. Nowak and F. Jurie. Learning visual similarity measures for comparing never seen objects. In *CVPR*, 2007.

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

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

[23] L. Parsons, E. Haque, and H. Liu. Subspace clustering for high dimensional data: A review. *SIGKDD Explorations*, 6(1):90, 2004.

[24] 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.

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

[26] S. Santini and R. Jain. Similarity measures. *PAMI*, 21(9):871–883, 1999.

[27] E. Spinosa, A. Carvalho, and J. Gama. An online learning technique for coping with novelty detection and concept drift in data streams. In *Int. Workshop on Knowledge Discovery from Data Streams*, 2006.

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

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

[30] A. Torralba, R. Fergus, and W. T. Freeman. 80 million tiny images: a large database for non-parametric object and scene recognition. *PAMI*, 30(11):1958–1970, 2008.

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

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

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