MVA2005 IAPR Conference on Machine VIsion Applications, May 16-18, 2005 Tsukuba Science City, Japan
8-29

Adaptive Vector Quantization of Sequences of Local Blocks for Video Surveillance System

Mickael Pic and Takio Kurita
Neuroscience Research Institute
National Institute of Advanced Industrial Science and Technology
Tsukuba AIST Central 2, Umezono 1-1-1, Tsukuba 305-8561, Japan

1 Introduction

To keep security, video surveillance cameras have been installed in public space such as convenience stores, streets, ATM, etc. The number of such cameras is increasing rapidly. When an incident happens, the circumstances of incidents can be confirmed by reviewing the recorded scene. However, many of the current surveillance systems record continuously everything with low frame rate (for example 4 frames/sec.) even when nothing happens. Because of lack of retrieval function, the current user of the surveillance system has to review every bite of information to find the parts where the incident is recorded. Since a static camera is used for video surveillance in most cases, a lot of the information recorded is redundant such as the background of the scene and when nobody is in the scene. The reviewing of such redundant information can create a loss of time with dramatic consequences. Also the system often fails to record very important information such as frontal face because of its low frame rate.

To overcome such drawbacks of the current surveillance system, we should develop a method to retrieve scenes where some moving objects are recorded. Also an efficient video compression algorithm has to be developed to increase its frame rate and reduce the required storage.

For a case of static camera, a typical approach to detect moving objects is background subtraction. If there are some moving objects in the current image, the differences at the points corresponding to the moving objects become large. The moving objects can be easily detected by thresholding the differences. In realistic situation, however, the background may gradually or suddenly change because of the day light changes or movements of the equipments, etc. To treat such changes in the background, the adaptive background estimation method has been proposed by one of authors [1, 2]. In [3] a method is proposed to use Gaussian Mixture Models with pixel values to modelize the background.

In this paper we address the problem of recording and retrieving scenes of interest from a static camera for a video surveillance system. Here we consider sequences of local blocks instead of sequences of pixel values in the case of background subtraction. To model a sequence of a local block, we use an adaptive vector quantizer (AVQ). By applying a vector quantizer to a sequence of a local block, we can represent the sequence by some of code vectors (codebook). This corresponds to modeling multiple backgrounds appearing in the sequence. Then novelty of a new block can be easily detected by checking the distances between the new block and the code vectors. This makes moving object detection possible. Also by representing the sequence using a codebook the required storage is reduced and approximation of all blocks in the sequence can be regenerated from the stored codebook. To maintain information on the current background, the code vectors are stored in a stack in the order of the occurrences. Also a small memory of its history is assigned to each code vector of the codebook allowing a forgetting factor of the vector when it has not been used for a long time.

We also present a method to remove noise from the scene such as flickering by monitoring the codebook evolution through time.

2 Adaptive Vector Quantization of Sequences of Local Blocks

A frame in video can be divided into small local blocks $M = B \times B$ pixels (Usually $B = 4$ or 8). Then a sequence of frames in video can be viewed as a set of sequences of local blocks. We assign a vector quantizer for each block and each of the sequences of local blocks is modeled by using the vector quantizer. By applying a vector quantizer to a sequence of a local block, we can represent the sequence by some of code vectors (codebook). This means that we can model multiple backgrounds appearing in the sequence by its code vectors. Then novelty of a new block can be easily detected by checking the distances between the new block and the code vectors. This makes moving object detection possible. Also by representing the sequence using a codebook the required storage is reduced and approximation of all blocks in the sequence can be regenerated from the stored codebook.

To maintain information on the current background, the code vectors are stored in a stack in the order of the occurrences. Also a small memory of its history is assigned to each code vector of the codebook allowing

a forgetting factor of the vector when it has not been used for a long time. These mechanisms make the vector quantizer adaptive.

2.1 Adaptive Vector Quantization

Vector quantization (VQ) is used mainly for data compression [4, 5]. Let T be the set of all possible vectors for a task. Assume that T is separated into K distinct regions (clusters) that exhaust T and each of the regions is assigned a vector called *code vector* $\boldsymbol{\theta}_k, (k = 1, \ldots, K)$. Then, given a $\boldsymbol{x} \in T$, we can determine the region where it belongs to and we can adopt the corresponding code vector instead of the original vector $\boldsymbol{x}$. If the code vectors do not change during the task, they can be specified by some codes. This reduces the amount of storage required to represent the original contents.

In [6], the authors used Adaptive Vector Quantization for video compression. A frame is divided in L blocks of $M = B \times B$ pixels (Usually 4 or 8). A global codebook is created for the full frame. If a good match is found, then the codebook is not changed, and the block from the codebook is used for approximation. If no good match exists, then a new vector is created from the block and added to the codebook while another one is removed. This method works well for video compression or tele-conference system.

In this paper, instead of using a codebook for the whole frame, we prepare L independent vector quantizers and one vector quantizer is assigned to each block. Their codebooks (code vectors) are maintained independently.

Let $\{\boldsymbol{x}_i^{(l)}\}_{i=1}^{t}$ be a sequence of local block l ($l = 1, \ldots, L$). Then a set of code vectors $\{\boldsymbol{\theta}_k^{(l)}\}_{k=1}^{K^{(l)}}$ is prepared for the block l, where $K^{(l)}$ is the number of clusters in the vector quantizer assigned to the block l. When we use the mean squared errors

$$\boldsymbol{E}^{(l)}(\boldsymbol{\theta}_1^{(l)}, \ldots, \boldsymbol{\theta}_k^{(l)}) = \sum_{i \in C_k^{(l)}} \|\boldsymbol{x}_i^{(l)} - \boldsymbol{\theta}_k^{(l)}\|^2 \qquad (1)$$

as distortion measure to design a vector quantizer, then the optimum code vectors can be obtained as

$$\boldsymbol{\theta}_k^{(l)} = \frac{1}{N_k^{(l)}} \sum_{i \in C_k^{(l)}} \boldsymbol{x}_i^{(l)} \qquad (2)$$

where $C_k^{(l)}$ is the $k^{(l)}$-th cluster and $N_k^{(l)}$ is the number of vectors assigned to the cluster $C_k^{(l)}$.

When a new frame is captured, novelty of each block in the frame is checked and the corresponding vector quantizer is updated. Let $\boldsymbol{x}_{new}^{(l)}$ the block l in the new frame. The distances between the block $\boldsymbol{x}_{new}^{(l)}$ and the code vectors $\boldsymbol{\theta}_k^{(l)}$ are computed. If the minimum distance is smaller than a pre-specified threshold Th, the block is similar with one of the previously appeared block in the sequence. Assume that the minimum distance is obtained to the cluster $C_k^{(l)}$. Then the corresponding code vector is updated by

$$\boldsymbol{\theta}_k^{(l)\prime} = \frac{1}{N_k^{(l)} + 1}\{N_k^{(l)}\boldsymbol{\theta}_k^{(l)} + \boldsymbol{x}_{new}^{(l)}\} \qquad (3)$$

If the minimum distance is larger than the threshold $\boldsymbol{Th}$, the block is considered as new and a new code vector is created in the vector quantizer.

2.2 Codebook and stack

To maintain order of occurrences in a sequence of blocks, code vectors of each vector quantizer are maintained in a stack. Figure 1 outline the typical behavior of a codebook stack. At first the stack is initialized with the code vector obtained from the first frame. Then using the Adaptive Vector Quantizer algorithm explained in the previous section, at each frame, novelty of the block $\boldsymbol{x}_{new}^{(l)}$ is checked. If the block is new, then the new code vector $\boldsymbol{x}_{new}^{(l)}$ is added to the top of the stack. This is shown at $t + 1$ in in Figure 1. The code vector D is added to the top of the stack. If the block $\boldsymbol{x}_{new}^{(l)}$ is similar with one of the code vectors, the code vector is put on the top of the stack and it's values are updated by using the equation (3). This is shown at $t + 2$ in Figure 1. In this figure, the code vector A is moved on the top of the stack.

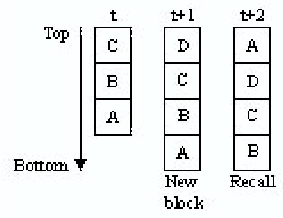

Figure 1: AVQ stack

2.3 Forgetting factor and history

To maintain frequencies of occurrences of the similar blocks, we also associated a counter to each code vector in a vector quantizer. Each time a vector is updated, the counter value is increased by one. Otherwise, its value is decrease by one.

The purpose of this counter is mainly to provide a forgetting mechanism to the stack. This system is introduced to allow to forget old code vectors that have not been updated for a long time. For example, if someone moved a chair in a room, the blocks located at the previous position of the chair have to be removed from the stacks if the chair is not put back to it's previous position.

Forgetting mechanism implemented in the current system is very simple. When a counter reach zero, its vector is removed from the stack. When a new vector is added to a stack, the counter should not be initialized at 1, because often noise happen, and a block initialized at 1 would probably disappear at the next frame. A value of 10 has proven to be enough to overcome such problem. We have also defined an upper bound

for the counter. This is to prevent an 'over-learning' of a vector. For example if an object in the scene stay for a very long time, say 10000 frames, it would take the same amount of frames to forget it when it would have moved. A high value allows to keep the value of the background even when an object stay stationary hide the background for a long time.

When using the counter with the stack, it is possible to recall the history of a block. Compared to traditional background extraction methods such as [1, 2], this mechanism gives an improvement toward efficient tracking of multiple backgrounds. Because traditional background extraction can only keep one background that slowly adapt with time. Our method can instantly adapt to any change and also can keep in memory the previous value of the background. If the change is only temporary, we can recover the recent background from the stack. With time the old background is forgotten and only new backgrounds are kept in the stack. We can also backtrack the moving objects in the frames by accessing the vectors in the codebooks of each block.

3 Experimental result

We tested the algorithm on various stream of images. Table 1 lists the information from the two main videos used in the experiments. Pokemon is the record of a scene where an object (pokemon figure) is put at the center of the frame and then the camera is suddenly paned. This video was used to test the behavior of the algorithm in case of sudden change of the scene background. TeaRoom is a one hour-video extracted from a 24-hour video monitoring of a meeting room. The TeamRoom is an uncontrolled environment with day, night, and everyday situations (ex: people entering, leaving and staying at the same spot for long time). Every video has been recorded at 30 frames-seconds.

Table 1: Video information

Video	*Size*	*Frames*	*Duration*
Pokemon	320x240	3001	1 mins 30
TeaRoom	640x480	103950	60 mins

Figure 2 shows some frames sampled from the TeaRoom video. Someone come inside the room, take a book, read a few pages of it, and then leave.

Figure 2: Samples frames

3.1 Scene detection

By monitoring the activity of the vector quantizers of the frame, scenes of interest with moving objects can be detected. When something is moving inside the scene, new vectors are created or old ones are recalled. A simple thresholding allow to start, and stop, the recording of frames while moving objects are detected. We define the measure of activity (MA) of the vector quantizers for a frame at time t has

$$MA = \sum_{l \in L} CV + UV, \tag{4}$$

where CV is the number of created vectors, and UV is the number of updated vectors.

Figure 3 shows the typical behavior of the measure of activity MA when someone enters the room (frame 137), stays for sometimes, and leaves (frame 240). After that no one enter the room for sometimes. If the measure of activity of the vector quantizers of the frame is higher than 0, then the system records the frames. A higher threshold could be used to record

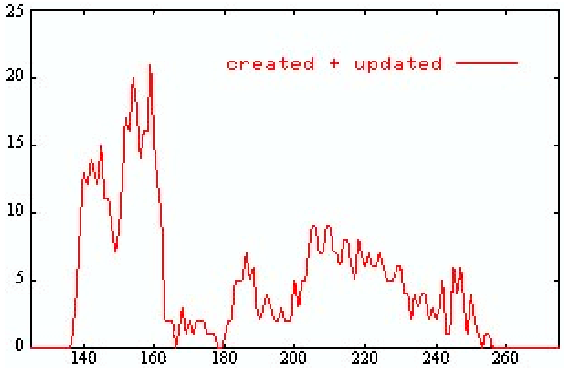

Figure 3: Measure of activity

only large activity, such as recording heavy traffic on road while discarding light traffic, or people when they are close enough to the camera. The system is working in real-time on a Pentium 4 2Ghz processor.

3.2 Fast access

Previously we have shown how to record only scene of interest for the user. This fulfills one of our goal, but another goal is to allow a fast access for the user to those scenes of interest. While MPEG video format allow for quick access using I-frames as reference ([7]), it does not provide information about the relevance of the scene recorded, or entry points to them. Using MPEG, the user would have once again to review all the recorded sequences one after another to find the information he is looking for.

We designed a simple video coding-decoding format based on the vector quantizers. Our coding algorithm record the time lap between two recorded frames, then for each recorded frames, we only saved the activity of the vector quantizers. When a new vector is added, we save its information with the codec. When a vector is updated, only its code is transmitted. The decoder reads the information from the file, and updates its vector quantizers according to it.

For demonstration purpose, we designed a simple interface using the decoder to provide quick access to the interesting sequences where some moving objects were recorded. The system read the file, and each time a step between two frames is higher than a threshold S a new sequence start. For each sequence, a thumbnail

of the first frame is extracted, and can be used as a fast access point. Figure 4 show the interface, with

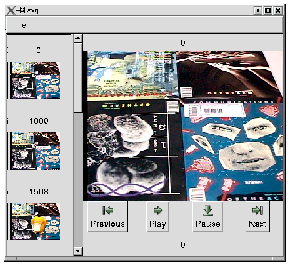

Figure 4: Quick access interface

the quick access points on the left side, and the main display window on the right side. A user can click on the thumbnail on the right side, and instantaneously the video will start to play on the left side starting from the frame clicked by the user.

3.3 Treatment of flickering

In real situation, flickering introduces a lot of noise in the frames and may creates changes strong enough to trigger the recording of frames even when there is no object of interest. To prevent this, a simple mechanism to monitor the activities in the vector quantizers is implemented in the current system. After monitoring the codebooks activity, it has been found that flickering usually appear in the form of two vectors rotating in the codebook. In case of flickering, at time t, a vector A is activated. At $t+1$, another vector B is activated. At time $t+2$, the vector A is once again activated, and so on. This is the simplest case of flickering. To remove such effects, activated vectors in the stack through time are recorded to define a pattern of a predefined size. Then we search if this

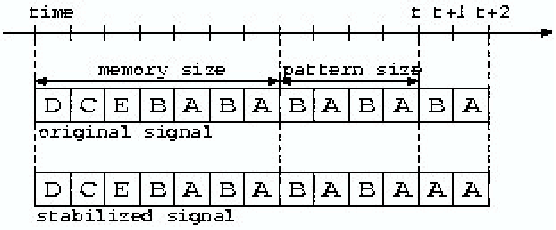

Figure 5: Noise Removal

pattern is already observed in the immediate past of the sequence. If such pattern is found, then we decide that there is flickering, and force the vector quantizer to keep the same vector always activated until something break the pattern, like someone passing in front of it. Figure 5 shows the typical behavior of the noise removal algorithm.

3.4 Results of the compression

Our goal is to provide an algorithm that can help the user to quickly find what he is looking for, while keeping all important information in a minimum of storage. Table 2 show the results of the proposed Adaptive Vector Quantization compared to traditionnal continual recording of all frames.

Table 2: Comparison of performance

Video	*A*	*B*	*C*	*D*
Pokemon	3001	375	39 Mb	5.7 Mb
TeaRoom	103950	26375	1322 Mb	42 Mb

In this table, A is the original number of frames, B is the number of frames recorded by our method. C is the original MPEG file size, D is the final size using our encoder. In the current method, the original block is stored as it is. Although there is no optimization for compression, the required storage of the proposed method is less than the standard video compression method. The reason is that our method recordes only the information of the moving objects in the scene of interest. We used no compression algorithm at all, only the raw data of the blocks are stored. With the help of a simple interface, using the AVQ compression algorithm to create access points to the recorded scenes, instantaneous retrieval speed is achieve. Our proposed method offer a practical and fast way to search for scenes of interest recorded using the AVQ algorithm, while providing high compression rate.

References

[1] Shimai H, Mishima T, Kurita T, and Umeyama S. Adaptive background estimation from image sequence by on-line m-estimation and its application to detection of moving objects. *Proc. of Workshop on Real-Time Image Sequence Analysis (RISA2000)*, pages 99–108, 2000.

[2] Pic M, Berthouze L, and Kurita T. Adaptive background estimation: Computing a pixel-wise learning rate from local confidence and global correlation values. *IEICE Trans. Inf and Syst*, E87-D(1):50–57, Jan 2004.

[3] Stauffer C and Grimzon E. Adaptive background mixture models for real-time tracking. *Proc. of CVPR99*, 2:246–252, 1999.

[4] Gray R M. Vector quantization. *IEEE ASSP Mag.*, 1:4–29, Apr 1984.

[5] Linde Y, Buzo A, and Gray R M. An algorithm for vector quantizer design. *IEEE Trans. Commun.*, COM-28(1):84–95, Jan 1980.

[6] Dietmar S and Bernd B. Real-time very low bit rate video coding with adaptive mean-removed vector quantization. *IEEE International Conference on Image Processing (ICIP'97), Santa Barbara*, Oct 1997.

[7] ISO. Information technology – coding of moving pictures and associated audio signal for digital storage median at up to about 1,5 mbit/s – part 2: Video. *ISO/IEC 11172-2*, 1993.