Real-Time Counting of moving objects in H.264 video flows

G. Flores-Chapa, L.M. Rojas-Cardenas, Hossam Afifi
Department of Electrical Engineering, UAM-Iztapalapa, México D.F., México
Departament RSM, Telecom SudParis, Evry, France
Phone (52) 5558044635 Fax(53) 5558044628 Email:lmrc@xanum.uam.mx

Abstract—This paper proposes a real-time and low cost system for automatic tracking and counting of moving objects in H.264 video flows. In contrast to most of approaches proposed in the literature, which normally process raw images/pictures with costly image processing methods, our approach employs the motion information already contained in H.264 video sequences. So, our proposal is based on three main elements: 1) H.264 moving vectors, 2) a morphological description of the objects to be counted and 3) a forward object position estimator for reducing counting errors. A number of tests has been achieved over High-Definition TeleVision (HDTV) video sequences (1440x1080 pixels at 25 frames per second) and results show that the proposed approach is capable of achieving real-time object counting.

Index Terms—Counting Object, Computer Vision, Object Tracking

I. INTRODUCTION

In computer vision, counting objects is an active research area where the problem has been analyzed for a variety of objects densities, for distinct camera positions and a number of several different techniques. Additionally, those approaches can be classified, in general, as: Counting objects in the pixel domain and counting objects in the video compress domain. In the former domain, it is possible to develop algorithms for recognizing and counting objects with great accuracy. Unfortunately, they are very costly in computational resources, specially when they are integrated into conventional tele-surveillance systems. Indeed, these counting objects systems must consider not only the processing time for their algorithms but also the decoding time of video sequences. In practice, most of approaches in this domain are not able to deal with real-time video flows, specially if state-of-the-art cameras are considered, which normally generate H.264 video sequences that currently go from VGA (640x480 pixels) to HDTV (1920x1080 pixels) at 25/30 frames per second. On the other hand, counting objects in the video compress domain offers a number of advantages: 1) Information about how objects move in the scene is ready to use and it can be easily recovered in the form of moving vectors (MV), 2). Additional information, such as textures and colors, aid to improve the accuracy of recognition and counting process. This information is directly available too, 3) The obtaining of these information requires a partial decoding only, specifically, a VLC decoding. These elements allow to create low cost algorithms. Actually, there are several research works based on this approach. In the domain of MPEG II , in [2] MV are used to track objects and additional information such as DCT coefficients allow to improve accuracy. They do not inform about the temporal performance of their approach nor about the picture size and frame rate. In [8], MV are grouped by means of a k-means clustering algorithm. This algorithm reduces tracking errors by eliminating MV related to covered regions. They implement their approach in MATLAB and C language, which allows to process from 4 to 10 frames per second depending on the size of the object to be tracked. The characteristics of the video are not mentioned. Moreover, to cope with real-time applications they propose a multiprocessor implementation. For object tracking in H.264/AVC, [9] employs neural networks. This approach is based on information directly available on the H.264 data flow, specifically MV and textures. The utilization of textures requires areas related to MV to be completely decoded, which is very costly. Nevertheless, they show that it is possible to track objects without considering information contained in Intra-frames. Indeed, only P and B frames are taken into account, even if they show some deformations. Our proposal works similarly. The tests has been achieved over video sequences with pictures having 352x288 pixels and they report a processing capacity of about 240 frames per second. [8] proposes a object tracking algorithm based on contours extracted from motion vector information. A technique called “area-region” is used to reduce errors in case of occlusions. A post-processing phase is required for establishing the object's area. On the other hand, camera position is an important aspect. The most straightforward strategies use top-view cameras in order to avoid occlusions. Nevertheless, the main disadvantage of this solution is that it makes difficult people identification. More sophisticated approaches place cameras in front of the people flows. It has been shown that [7] the best camera's position for tracking objects is when objects flow is parallel with respect to the camera's axis. In this document, we propose a system for object counting based on moving vectors. On the other hand, our system can be classified as zenithal, that is, the camera is placed at the top, with a view angle that is perpendicular to the plan where objects move. This avoids occlusions between objects. Moreover, we employs video sequences obtained from conventional IP cameras.

II. SYSTEM OVERVIEW

The proposed system achieves, above all, the extraction of MVs. This task is accomplished by means of a Huffman decoding procedure (see Figure 1). Then, a filter is applied to
the recovered MVs in order to keep MV having a given magnitude; they are candidates for grouping them. Actually, this grouping represents the cornerstone of our proposal. Grouping choice MV by tracking into account their magnitude and direction. After grouping, a morphological filter determines which groups of MV must be considered to counted; The objects that do not conform this criteria are discarded. On the other hand, for our system, an object is counted only once: when the object appears in the scene. For guarantee that this occurs only once, the system tracks the object anywhere in the scene and a forward position predictor is used to reduce errors. Finally, the system allows a counting zone to be established, which can be tuned in order to satisfy the application requirements.

Fig. 1. System overview

1.1 Filtering procedure

For each GOP, a VLC decoding procedure is applied to P and B frames only. The information they offer can be represented as:

\[ VM_i = (mb_i, x_i, y_i) \]  

(1)

Where \( VM_i \) represents the \( i \) – esimo moving vector, \( mb_i \) represents the block number related to the moving vector, \( x_i \) is the horizontal component of the moving vector and \( y_i \) is the vertical component. The vector's magnitude is obtained by means of the Euclidean distance. That requires \( mb_i \) to be clipped:

\[ mbx_i = mb_i \mod sh \]

\[ mby_i = mb_i \div sh \]

\[ \sqrt{(x_i - mbx_i)^2 + (y_i - mby_i)^2} \geq \text{umbral} \]  

(4)

Where \( sh \) is the size of pictures.

Fig. 2. Orden de decodificación de macrobloques para la extracción de vectores de movimiento.

1.2 VM Clustering

The type of objects considered in this system corresponds to those solid objects moving at a speed more or less regular and its morphology is relatively static. A cloud of smoke, the foliage of certain trees, a blanket moved by the wind to produce irregular shapes are not considered by our system. On this condition, it is expected that the VM-related objects have a set of vectors that share the same magnitude, direction and spatial proximity. Defined as:

\[ CVM = \{ VM_1, ..., VM_n \} \]

\[ GVM = \{ GVM_1, ..., GVM_k \} \]

\[ VM_i \in GVM_j \]  

(7)

where \( CVM \) is the set of vectors in the current frame that match the defined criterion in (4), \( GVM \) is the set of clusters of motion vectors in the current frame and \( VM_i \in GVM_j \) defines the relationship “corresponds to”, which implies that the spatial distribution (location within the video scene) of \( VM_i \) wholly or parcially belong to the spatial distribution of \( GVM_j \), otherwise \( VM_i \notin GVM_j \) (see Figure 3). At the beginning of the decoding of the current framework, the cardinality of the set \( CVM \) and \( GVM \) is zero.

VM clustering is done by tracking data within the borders of a frame following the sequential storage order defined by the standard. This indicates that while the VM is stored in sequence, these correspond to an order from left to right and from top to bottom with respect to a decoded image (see Figure 2).

The process of filtering and grouping of vectors are performed concurrently, which means that the tracing of vectors is done once, as shown:

1. Be \( VM_{n+1} \) the following vector decoded
2. Be \( VM_{n+1} \) add to \( CVM = \{ VM_1, ..., VM_n \} \)
3. If \( \exists GVM_j \) within \( GVM = \{ GVM_1, ..., GVM_k \} \), such that \( VM_{n+1} \in GVM_j \)

\[ GVM_{n+1} \]

Resets the area \( GVM_j \), if necessary, for \( VM_{n+1} \) corresponds to totally within the spatial distribution of \( GVM_j \) (see Figure 4.b).

4. Otherwise

Is defined \( GVM_{k+1} \) using as the spatial distribution of \( VM_{n+1} \) add a tolerance, this tolerance will allows next spatial vectors can be clustered. It adds \( GVM_{k+1} \) in \( GVM \) (see Figure 4.a).
1.3 Morphological Filter

The morphological filter receives as input the current frame set GVM, which contains clusters of vectors candidates for moving objects MO. The procedure that determines when a GVM is regarded as a MO uses two basic elements: the number of VM for this GVM and spatial distribution (area covered) of GVM. Then, each GVMi who meets these criteria define a filtering MO. Define:

\[
CMO = \{MO_1, \ldots, MO_k\} \tag{8}
\]

where CMO is the set of MO for the current frame, and \( MO_i \) is the i-th MO in CMO for the current frame. It is important to note that the cardinality of CMO is less than or equal to the cardinality of GVM. This set is the input data for tracking moving objects.

1.4 Database Objects

This database contains the information of the MO found. For each MO the following data is stored:

1. Spatial distribution
2. Orientation
3. Last reference frame

Define:

\[
DBMO = \{DMO_1, \ldots, DMO_k\} \tag{9}
\]

where DBMO which defines a set of MO they have been tracked in previous frames, these MO are different elements of the set denoted as CMO and denotes DMO as the ith element in DBMO.

1.5 Tracking moving objects

This module has the responsibility to track any moving object that appears in the video scene until it disappears. This type of monitoring prevents count an MO more than one time, moreover, not only uses the spatial position obtained from the decoded current frame but also the previous position recorded in the database DBMO.

MO tracing found at the scene vide takes elements in CMO that correspond to the MO found in the current frame. The following procedure is performed for all \( MO_i \) in CMO founded in the current frame:

1. If \( \exists DMO_j \in DBMO = \{DMO_1, \ldots, DMO_k\} \), such that \( DMO_j \) follow to \( MO_i \)

Then it is considered that \( DMO_j \) continues in the video sequence, so it is necessary to update your details in the DBMO.

2. Otherwise

It verifies if the distribution of \( MO_i \) this near to the spatial limits of video scene, in which case the object is appended to the DBMO.

We define the relationship “follow to” as

\[
dsMO_i = dsDMO_j + \Delta ds \tag{10}
\]

where \( dsMO_i \) is spatial distribution of \( MO_i \) in CMO, \( dsDMO_j \) is the spatial distribution of \( DMO_j \) contained in the database DBMO and \( \Delta ds \) is a prediction of spatial distribution of \( DMO_j \) using as parameters its orientation and last frame of reference.

This process not only uses the spatial position obtained from the decoded current frame but also the previous position recorded in the database objects. The predicted position is determined by the front position estimator. It is necessary to compare the current position of the movement of the object with the estimated position because sometimes a moving object "disappears" from the scene, particularly when there is no motion vectors in the current framework that is related to the object in question. In this case, the estimated position helps establish a relationship between the object disappeared and the object appeared in the center of the scene. Of course, this is not possible in a video taken in the real world. Therefore it is necessary to have evidence that these two objects are related. Also, in case of "missing objects" and "no recurrence", each moving object recorded in the data base has an associated TTL. This timer allows an object to be eliminated from the database after a given period. We have defined the counter TTL to 5 frames.

III. EXPERIMENTAL RESULTS

As established at the beginning of this document, this system has been implemented for a fixed top-view high resolution camera. One of the goals of this project is to obtain a very fast algorithm that enables a single desktop computer to manage video flows coming from several video cameras. We measure a mean processing time of around 400 µs per frame, that without considering the inverse VLC process.

In our experimental tests we use video sequences with moving objects (people) as shown in Figure 4. Our
Experimental results in low density flows showed an average processing time of 425.93 microseconds per image, this result is as expected due to the simplicity of our proposal. The average time for processing depends largely on the number of objects in the video scene.

Our system has a high reliability when the video stream is low, however, the reliability decreases as the special distribution of objects is very close, so that in future work is expected to characterize the movement of an object vector in order to qualify.

As far as the cases where our system offers good results, these cases are:

a) Identification and tracking of objects without bringing big objects such as trolleys, pushchairs, etc. This task is accomplished with accuracy and satisfactory performance.

This task is accomplished very quickly because the processes to be achieved are simple. They only are: the definition of the object area, the morphological filtering and the database updating.

In the case of multiple motion objects at the same rhythm and very close, an estimation of the size allows to determine the number of object moving under such conditions.

b) Removing of vector cluster no corresponding to objects. Our system is intelligent enough to identify vector cluster whose morphology differs considerably from the object morphology.

IV. CONCLUSION AND FUTURE WORK

In this work we propose a lightweight method for automatic counting object system that process H.264 video sequences and, in particular, it process motion vectors already contained in the bitstream. Tacking this kind of information offers two main advantages: The H.264 coder solve the problem of obtain motion information from the scene and then, it allows achieve a low cost counting process in terms of computing resources, which enable a single computer to handle the video coming from several video cameras. Here, our system is capable of processing a high-resolution frame (1440x1080 pixels) at around 400 μs. That does not consider the time of the inverse VLC process.

A tracking system allows object to be followed during their presence in the scene as well as it avoids counting more than once a object. This system relies on two important procedures: a morphological filter and a forward position estimator.