Customizing MPEG Video Compression Algorithms to Specific Application Domains: The Case of Highway Monitoring*

N.Zingirian[1], P.Baglietto[2], M.Maresca[1] and M.Migliardi[2]

[1] Dipartimento di Elettronica ed Informatica – University of Padua, Italy
[2] Dipartimento di Informatica Sistemistica e Telematica – University of Genoa, Italy

Abstract. *This paper describes the design of a MPEG coding algorithm targeted to compress video sequences generated by highway surveillance cameras. The physical constraints of vehicles as well as the geometry of the highway environment projected into the camera determine a relevant base of a-priori knowledge, which allows obtaining considerable processing speed up as well as higher compression ratio. This paper models the highway environment in terms of rules and shows how a a customized MPEG coding algorithm can take advantage of such rules to improve processing speed and compression ratio.*

1 Introduction

This paper presents a MPEG coding technique which runs on low cost general purpose platforms in real time and achieves a compression efficiency higher than regular MPEG taking advantage of the known characteristics of the application domain.

The starting point of our work is the fact that video sequences generated in a specific application domain represent objects which are frequently constrained in their shapes, sizes, motions because of the nature of the specific application domain. The constraints can be translated into specific rules; the design of video processing algorithms can take advantage of these rules in order to reduce the amount of produced information.

The specific application domain we selected as a case study for this paper is highway monitoring [MB94] and more in particular the compression of video sequences generated by highway surveillance cameras[YH93].

The paper is organized as follows. Section 2 describes the methodology adopted; Section 3 describes the details of the customized MPEG coding algorithm; Section 4 gives some results as concluding remarks.

2 Methodology and Model

The methodology adopted to design this algorithm is based on the following guidelines :

* This work was supported by European Community through *ESPRIT BRA project 8849-SM-IMP*

- The algorithm is not aimed at generating a new compression standard form scratch. On the contrary it is aimed at producing a customization of the MPEG coding algorithm which *i)* absolutely preserves the standard format of the MPEG stream and *ii)* leaves untouched the maximum number of modules of the existing general purpose MPEG coders.
- Our MPEG coder is targeted to real-time coding constraints in low bandwidth environments.
- The control of the compression ratio delivered by the coder cannot be based on the quantization parameters. This technique is in fact already operating in the standard MPEG coders and causes a reduction of video quality which is uniform in more and less significant areas of the frames.
- The computing power reduction and the bandwidth reduction are obtained by analyzing the constraints of the object appearing in the specific video sequences, by modeling them in terms of rules and by designing some customizations of the MPEG coding in compliance with these rules. We devised the following rules:
  - Vehicle motion is bound in speed, acceleration and direction
  - Vehicles can appear and move only in limited regions of frames
  - Vehicle representation is ruled by perspective effects

3 The Algorithm

In this section we illustrate how our compression algorithm can exploit the *a priori* knowledge base both to save computations and to reduce the output information per time unit (i.e. output bandwidth).

The main idea is to partition each frame into three different types of regions, namely:

Vehicle-reachable region : This region corresponds to the frame area representing the highway gangways for both vehicle directions.
Vehicle entry region : This region corresponds to the frame area where vehicles appear for the first time.
Background : This region corresponds to the frame areas which represent the background; this region does not contain any information about the status of the highway.

These regions are represented in Fig. 1 and are respectively denoted by letters A, B and C.

Our MPEG compression algorithm processes each of these regions in a specific way. This section is organized in three subsections each of which respectively describes the processing of a region.

3.1 Processing of the Vehicle-reachable Region

Vehicles which have been already identified at most once are the only moving objects which are expected to appear in the Vehicle-reachable region. This is due to the fact that *i)* no vehicle can appear in the middle of the highway without having been previously identified within the Vehicle entry region and *ii)* no moving objects except for vehicles usually occupy highways.

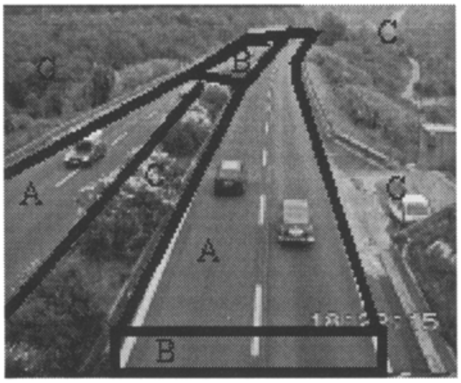

Fig. 1. The regions of frame

The structure of the Vehicle-reachable region processing is reported below. The key statements of this algorithm are written in italics and will be described in depth in this subsection.

```
for each frame $i$
    for each block $b$ inside the Vehicle-reachable region of frame $i$
        if block $b$ is *to be updated*
            estimate motion vector of block $b$ *using predicted vector of b*
            *determine blocks to be updated* in frame $i + 1$
            *determine predicted vectors of blocks to be update* in frame $i+1$
        else do nothing
    end for
end for
```

This sketch of algorithm shows that the concept of *predicted vector* (different from estimated vector) and the concept of *block to be updated* characterize the processing of the Vehicle reachable region with respect to the general purpose MPEG processing.

In this subsection we describe the Vehicle-entry region processing according to the following scheme:

1. we first assume to know whether a block *"is to be updated"* or not and we assume also to know the value of the predicted vector of b at the current frame i (henceforth denoted as $v_p^i(b)$). Under these assumptions we describe how our coder estimate motion vector of block b (henceforth denoted as $v^i(b)$) *"using predicted vector"* $v_p^i(b)$.

2. We then describe how our coder can *"determine blocks to be updated"* for frame $i+1$
3. We finally describe how our coder can *"determine predicted vector of blocks to be updated"* for frame $i+1$

Estimating Motion Vectors Using Predicted Vectors Our customized MPEG coder computes the value of the estimated motion vector of current block b by means of a block matching routine that processes a specific frame area, namely the *search area*, as all general purpose MPEG coders do. Our block matching routine stops when the search area has been completely spanned through a spiral pattern or when the matching function (usually the absolute difference) returns a value less than a determined *stop threshold*, as many general purpose MPEG coders do.

On the contrary, unlike general purpose decoders, our coder contains three customizations targeted to the specific highway monitoring application domain:

1. the center of the *search area* does not correspond to the center of the block b , but it corresponds to the pixel pointed by vector $v_p^i(b)$ (see Fig. 2 B)
2. the radius of the *search area* is variable according to the distance of the objects from the camera.
3. the *stop threshold* is variable according to the distance of the object from the camera.

The first customization is based on the fact that predicted vector $v_p^i(b)$ – which is assumed to be available – indicates the expected position within frame i of the vehicle contained in block b within frame $i-1$. If the expected vehicle position corresponds to the actual vehicle position then the motion estimation stops immediately, as the actual position coincides with the center of the search area, in this case. Otherwise, the more the expected position pointed by vector $v_p^i(b)$ is far from the actual position, the larger the search area is to be scanned by motion estimation routine. The criterion for evaluating predicted vectors is explained in future sections, and it is based on the assumption of the bounded vehicle speed variation.

The second customization is based on the fact that actual distances covered by vehicles are misrepresented in the frames due to perspective effects.

The third customization is based on the fact that the size of vehicles depend on the distance between vehicles and camera due to perspective effects again. As a consequence the stop threshold has to be larger when blocks under estimation represent far objects and vice-versa, since motion estimation of small objects requires a more precise matching evaluation than motion estimation of large objects.

Evaluation of Blocks to Be Updated in the Next Frame Blocks to be updated are those blocks which are estimated to exhibit significant variations between current frame i and frame $i+1$. During the processing of the current

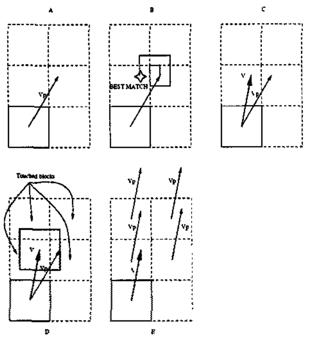

Fig. 2. The operations of the motion estimation

block b, our coder estimates a number of blocks to be updated in the next frame by following two steps:

1. It determines a set of blocks $b_1, \ldots, b_n$, defined as *touched blocks* depending on b
2. For each $b_k, 1 < k \leq n$, a corresponding finite state automa $A(b_k)$ receives an input values which specifies that b_k is a *touched block*. The output of automa $A(b_k)$ determines whether block b_k is *to be updated* or not.

In this section we first describe how our coder determines block dependent on b, and we present the finite state automa $A(bl)$ which determines whether the corresponding block bl is to be updated.

Touched blocks depending on b are identified by means of motion vector $v^{(i)}(b)$, which is yielded by motion estimation of block b, described in Sect 3.1. In order to determine touched blocks $b_1, \ldots, b_n$, our MPEG coder considers the 16×16-pixels area surrounding the pixel pointed by $v^{(i)}(b)$. This area shares its pixels with blocks $b_1, \ldots, b_n$, where $n \in \{1, 2, 4\}$ as shown in Fig. 2. These blocks are defined as *touched blocks* dependent on b. This definition is based on the fact that when a vehicle (or a piece of vehicle) included in block b is detected to move according to vector $v^{(i)}(b)$, touched blocks dependent on b will contain this vehicle within frame $i + 1$ with high probability.

Touched blocks are blocks expected to contain moving vehicles in the next frame. Unfortunately, touched blocks are not the only blocks *to be updated.* In fact, when the background is restored behind a moving vehicle, a number of blocks must be processed although none of these is touched by any vehicle. A finite state automa $A(bl)$ for each block bl determines whether the corresponding block bl is *to be updated* or not.

The input values of this machine are :

mark as touched (MAT): given to $A(bl)$ when block bl is identified as *touched.*
mark as untouched (MAU): given to $A(bl)$ when block bl is not identified as *touched*, after the whole current frame has been processed.

is static (ISS): given to $A(bl)$ when motion estimation of block bl returns a null motion vector.

The output values determine whether the block is to be updated in the next frame or not. The output values are:

to be updated (UPD): the block is estimated *to be updated* in the next frame
not to be updated (NUPD): the block is estimated not *not to be updated* in the next frame.

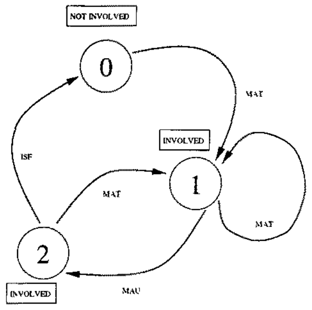

Fig. 3. Finite state machine of a generic block

The structure of finite state machine $A(bl)$ is defined in Fig. 3. Let us start its description from state 0. State 0 corresponds to the NUPD output . When a block of the frame i is NUPD, it will not be processed at the frame $i + 1$ (see algorithm at the beginning of this section).

If block bl is touched then MAT input enables transition from state 0 to state 1. State 1 generates the output of UPD. This output enables the processing of block bl in the next frame.

While input value is MAT, no transition takes place from state 1; it means that lock bl is processed at any frame, because new vehicles continuously touches it.

When no estimated vector points to block bl, then MAU input feeds the finite state machine and enables the transition to state 2. This state is still associated to UPD output and indicates that block bl must be processed to restore the background after the vehicle that touched it has left block bl.

Of course if MAT input occurs, transition back to state 2 takes place. If ISS input occurs, then transition to state 0 takes place. It means that block bl has restored the background and it is not *to be updated* in next frame.

Determining Predicted Vectors of Blocks to be Updated in next frame Each block bl estimated as *to be updated* is associated to a a predicted vector $v_p^{i+1}(bl)$. Each predicted vector is computed during the processing of frame i and is used by the motion estimation processing of frame $i+1$. The predicted vectors are computed as follows:

- If the block bl is not a *touched block* then the predicted motion vector $v_p^{i+1}(bl)$ is set to 0.
- If the block bl is a *touched block* dependent on current block b, then the corresponding predicted vector $v_p^{i+1}(bl)$ is set to $v^i(b)$ (see Fig. 2 E).

The first prediction is based on the fact that blocks which are *to be updated* but not touched are going to be processed to restore the background.

The second prediction is based on the assumption that vehicles tend to preserve their speed within one frame period.

In case that the same block is touched more than once within the processing of the same frame, its associated *predicted motion vector* is set equal to the average vector among all the predicted motion vectors destined to be associated to it.

3.2 Processing of the Vehicle-entry Region

The vehicle-entry region corresponds to the only areas of the image in which vehicles can suddenly appear. In this case the algorithm cannot rely on any previous prediction. The vehicle entry region processing consists of

- detecting the entry of new vehicles in the video, and
- producing the predicted vectors to be used in the processing of blocks belonging to the Vehicle-reachable region

The processing of the vehicle entry region is similar to the processing of the Vehicle-reachable region except for two main differences:

1. Blocks of the vehicle entry region are always considered *"to be updated"* and never considered *"touched"*;
2. Predicted vectors are computed in a different way with respect to the predicted vectors computed in the Vehicle-reachable region.

These two differences with respect to the processing of the Vehicle-reachable region are both consequences of the lack of information on vehicle motion extractable from previous frames.

Identification of predicted vectors and blocks *to be updated* is not based on vector yielded by motion estimation, but on a different vector named *reference vector*. The direction of the *reference vector* is fixed while the modulus can be computed in two different ways: through the *fixed speed assumption* or through the *dynamic speed estimation*.

The first way is quite trivial but allows saving computation time. It consists of assuming the average vehicle speed off-line. In this way each vehicle is assumed to appear in the video running at the same speed: the real speed will be detected in the vehicle-reachable area by means of the motion estimation. It is clear that for vehicles running at very high or very low speeds, the first motion estimation requires more computation time.

The second way is more sophisticated but requires more computations. It consists of assuming that traffic flow makes speeds of near vehicles mutually correlated.

This mechanism allows accomodating the estimation to the traffic conditions of the highway, however it can yield worse estimation than the fixed assumed speed in case of low traffic density.

3.3 Processing of the Background Region

The highway monitoring has little interest on what happens in the background region which is expected to be almost fixed. From the monitoring point of view the only events which must be monitored in this area are the incoming of some object through the highway borders (e.g. parked cars which begin to move again, people etc.). The main characteristic of such objects is their low speed with respect to the cars in the highway. From the video quality point of view the background is required to provide a realistic reproduction of the scene. Taking into account these features null motion vectors with no associated error are forced for the blocks belonging to the background. In our MPEG customization we implement a slow frequency refresh by coding only one row of blocks in a frame according to the standard MPEG algorithm.

4 Concluding remarks

The algorithm presented in this paper allowed us to modify an implementation of the standard MPEG algorithm[3]. The modified algorithm delivered a noticeably improved performance both in terms of bit rate and in terms of coding time[MM96].

In more details, the original software package running on a HP725/100 workstation takes 744 ms to perform the motion estimation of one frame; our algorithm requires only 44 ms to do it. In addition our algorithm concentrates the information loss necessary to obtain a 64 Kbps output bit rate in the background, while standard MPEG algorithm spreads the loss uniformly all over the frame.

References

[MB94] D. Murray and A. Basu. "Motion Tracking with an Active Camera". *"IEEE trans. on Pattern Analysis and machine Intelligence"*, 16(5), 1994.

[MM96] Zingirian Baglietto Maresca and Migliardi. "MPEG coding of higway monitoring video sequences". *"Internal Report DIST, Univ. of Genoa Italy "*, 1996.

[YH93] Y.Liu and T.S. Huang. "Vehicle Type Motion estimation from Multi-frame Images". *"IEEE trans. on Pattern Analysis and machine Intelligence"*, 15(8), 1993.

[3] MPEG de/coder by PVRG of Stanford University