International Journal of Image and Graphics

SEMANTIC VIDEO TRANSCODING USING CLASSES OF RELEVANCE

RITA CUCCHIARA, COSTANTINO GRANA, ANDREA PRATI
Dipartimento di Ingegneria dell'Informazione, University of Modena and Reggio Emilia
Via Vignolese, 905, Modena, 41100, Italy
E-mail: {cucchiara.rita,grana.costantino,prati.andrea}@unimo.it

Received (July 31, 2002)
Revised (September 13, 2002)

In this work we present a framework for on-the-fly video transcoding that exploits computer vision-based techniques to adapt the Web access to the user requirements. The proposed transcoding approach aims at coping both with user bandwidth and resources capabilities, and with user interests in the video's content. We propose an *object-based semantic* transcoding that, according to user-defined *classes of relevance*, applies different transcoding techniques to the objects segmented in a scene. Object extraction is provided by on-the-fly video processing, without manual annotation. Multiple transcoding policies are reviewed and a performance evaluation metric based on the *Weighted Mean Square Error* (and corresponding PSNR), that takes into account the perceptual user requirements by means of classes of relevance, is defined. Results are analyzed varying transcoding techniques, bandwidth requirements and video types (with indoor and outdoor scenes), showing that the use of semantic can dramatically improve the bandwidth to distortion ratio.

Keywords: Transcoding; video semantic; compression; motion segmentation; performance evaluation, multimedia Web access

1. Introduction

Video publishing is becoming a more and more spreading reality on the Web. The main motivations for this are that videos increase the amount and the quality of the information provided with respect to still images and that they call for a more "real" interaction, with the meaning of a more similar reproduction of the way in which we typically look at a scene. Moreover, the improvement of the multimedia technology permits now to afford the huge amount of computation, bandwidth and storage required by videos.

Nevertheless, in the video accessibility through the net two important problems arise: the first is the unavailability to all the users of new technologies such as large bandwidth connections to the Internet; the second is the constant growth and diffusion of a diversity of devices to access the Internet. For example, PDAs (personal digital assistant) are devices with limited display and resolution resources and have typically a wireless network card with limited bandwidth. With these premises, video delivery and reproduction can be unaffordable tasks. In this con-

text, often called UMA (Universal Multimedia Access) [1], we assist to a tremendous variety of multimedia contents, and, at the same time, to a different panorama of possible clients characterized by specific network bandwidth constraint and display, processing and storage capabilities. *Transcoding* is consequently now one of the key process to adapt the multimedia content to user requirements: the major effort is often devoted to a bandwidth reduction, especially for new terminal types (PDAs, HCCs, smart phones, ...).

Many works are present in the literature addressing the problem of the *transcoding of information*.[1,2,3,4,5] Transcoding is a term currently associated with the process of changing a multimedia object format into another: it is referred either as an *intramedia* transcoding when the media nature does not change (e.g. varying the video compression rate, the color description or transforming a video from a MJPEG to a MPEG4 code) or as an *intermedia* transcoding when also the media nature changes (for instance transforming audio into text).

Focusing on intramedia video transcoding only, the selection among the large plethora of multimedia formats must be done in terms of tradeoff costs/benefits: the costs rely on the hardware resources to provide transcoding (processing capabilities especially for on-the-fly transcoding and storing capabilities if many transcoded videos are available at the same time); the benefits should be measured both in terms of bandwidth saving and video fidelity.

The final goal of transcoding remains a suitable adaptation to the client resources, maintaining an acceptable *QoS* whose model definition and performance analysis still remain open problems. This is particularly true in the case of videos where the user satisfaction should be measured in terms of perceptive cues and video fidelity is hard to define, unless the application and its purposes are well known. For this reason, expertise of computer vision community in image and scene understanding could be essential in order to extract semantics from videos, handle the transcoding process, and enhance the perception of meaningful data, even if the quality of the accessed video is limited by the bandwidth. In this context, within video transcoding, we want to adopt the term *semantic* or *content-based transcoding* with the twofold meaning that the transcoding process is guided by the video's semantic and at the same time the transformation may change the video perception and possibly its appearance, while preserving the semantic. The capability of preserving semantic allows the effective scalability of the video on almost every existing device. For example, in video-surveillance applications you may want to access the camera installed inside your factory, but you may have only a cell phone to do it. In this case, our claim is that a system with semantic transcoding of the video will be able to analyze the video and to send the information of possible intruders only, in a format visible on your cell phone's display.

Several approaches have been proposed addressing semantic transcoding, mostly dealing with stored videos. They are often associated to a process of *annotation* that takes care of video content, annotated in the video database.[6] The standardization work of MPEG-7 with Multimedia Content Description Interface has de-

fined the meta-data description coupled with stored videos, to support transcoding applications.[7] For instance, the Video Semantic Summarization Systems described in [8] exploits MPEG-7 for semantic transcoding. A good survey of transcoding products is presented in [1], where the idea of preserving multimedia content in Web access is well underlined. Transcoding can be provided at level of server, proxy or client; the authors of [1] claim that there are some advantages in designing transcoding capability in multimedia servers especially because the provider keeps the control of distributed data. Therefore the authors provide a general framework, called InfoPyramid, to store annotated information and transcoded versions of the same multimedia content in the server. A similar approach is proposed in Columbia's video on demand testbed.[9] An alternative solution to storing multimedia data already transcoded is to provide transcoding directly on compressed data.[9,10] It is exploited especially to downscale the compression rate or the video resolution. [11]

In this paper, we propose a video server architecture featured with a suitable transcoding of videos deriving from live cameras. In this case, the annotation cannot be provided (at least for real-time users) and only a on-the-fly transcoding is allowed. Thus, as Smith et al. in [4] propose image analysis processes for content-based image transcoding, we adopt computer vision techniques to handle semantic video transcoding. We define a transcoding approach in order to cope with both user bandwidth and resources capabilities, and user interests in the video's content. In the paper we analyze different techniques of video transcoding: as well as standard approaches (e.g. color or size downscaling) we propose some new techniques that exploit computer vision for extracting objects of interests. Objects are not defined a-priori but are segmented according to their visual features. We use a robust moving object segmentation approach called Sakbot [12] followed by an object classification (e.g, vehicle, people...). Objects extracted are described by means of their texture and alpha plane (as in MPEG-4 specifications). Then objects are grouped in classes of relevance depending on user requests. The video is coded as a sequence of objects for each frame; a mixed transcoding policy exploiting different JPEG compression and temporal transcoding is provided, depending on the coupled class of relevance.

In addition, we propose a model of performance evaluation that takes into account the object distortion weighted by the classes of relevance. If no semantic is associated to the video the performance model is the standard PSNR. Conversely, a *Weighted Mean Square Error* adjusts the value of distortion to the perceptual user requirements by means of classes of relevance.

In section 2 we define the framework, presenting the evaluated transcoding policies, the new defined object-transcoding, and the reference architecture. In section 3 we define the performance evaluation models and results are discussed in section 4. Conclusions end the paper. Results are very encouraging since we can reduce the bit-rate dramatically (e.g., from 27.74Mb/s uncompressed, or 1282Kb/s JPEG video to 203.91 Kb/s only) although keeping the same JPEG fidelity for objects of interest. Moreover, we are able to satisfy severe bandwidth requirements (e.g. 128Kb/s or 56Kb/s), with a compression of transcoded video that preserves the

video semantic.

2. The transcoding framework

Commercial video servers for live camera (as [13]) provide a user interface in which the image quality can be modified. Given the fixed compression quality the streaming process adapts the frame rate to the bandwidth. This is often unacceptable for users that need high details only in particular regions of the image; in this case users are forced to choose between frame rate and quality. For this reason, we want to propose a framework in which the transcoding is dynamically and on-the-fly adapted upon the requirements of the clients.

2.1. *Transcoding policies*

Transcoding has been classified in various ways: Vetro et al[14] distinguish among *bit-rate conversion* (or *scaling*), *resolution conversion* and *syntactic conversion*: the first copes with bandwidth limitation; the second is used for device limitation, as well as for bandwidth limitation; the third deals with syntactic conversion for protocol layer. By focusing on bit-rate scaling only, the authors propose two solutions: a conservative transcoding varying temporal and spatial quality of multimedia objects and an aggressive model that accepts dropping less relevant object on the scene (based on MPEG-7 specifications on stored videos) .

According with Vetro et al. we accept an aggressive model if it is guided by the semantic: thus, in addition to some downscaling techniques used for reducing bit-rate without relationships with the video content, we propose a semantic transcoding of the video, that maintains unchanged interesting objects' content. We classify transcoding as:

- *spatial* transcoding (`spat_tr`);
- *temporal* transcoding (`temp_tr`);
- *code* transcoding (`code_tr`);
- *color* transcoding (`color_tr`);
- *object* transcoding (`object_tr`).

Spatial transcoding is the standard frame size downscaling, from standard formats (as CIF 352x288, QCIF 176x144, etc.). This is necessary for some specific clients with limited display resources. This approach allows also bandwidth reduction (since the uncompressed amount of data decreases) in most of the cases. Many examples are reported in [15] for still images, especially focusing on the differences between codings such as JPEG and GIF.

Temporal transcoding copes with a reduction of number of frames: this is automatically provided by the streaming process that downscales the number of transferred frames. In other researches dynamic frame skipping techniques have been developed to choose when frames can be eliminated according with the changes in the motion vectors.[16] In [17] the composition problem (i.e., how to merge/compose

transmitted objetc with the reference/background image in an effective way) is examined, associated with varying the temporal transcoding of multimedia objects. In this work we use as a reference a simple temporal transcoding obtained by a static selection of the number of frames to skip. A dynamic temporal transcoding is further integrated in our proposal of object-based transcoding, as will be discussed in next sections.

Color transcoding, like spatial transcoding, is sometime requested for specific clients (like gray level PDAs). A color downscaling is automatically performed by all JPEG, MPEG standard with the 4:2:0 YUV code. Using less bits for pixel, chrominance suppression (adopting 8 bits gray level) and a more aggressive binarization (1 bit B/W code) are possible transcoding policies that can reduce bandwidth but also modify the perception of images. It can be accepted by human users but sometimes should be avoided if the transferred videos must be processed by computer vision algorithms that typically make a large use of colors.

Code transcoding, i.e. the change of (standard) coding, has been widely analyzed: increasing the level of compression saves bandwidth and sometimes could be acceptable for the video QoS standard too; however, a too aggressive compression could be unacceptable for many applications due to the lost details.

Finally, the class *object* or *semantic* transcoding comprises some different techniques to tract differently multimedia objects in the video. [14,1] We propose to manage moving objects computed with computer vision processes. Basically the goal is to extract semantically valuable objects from the scene and transfer them with the lower amount of compression in order to preserve both details and speed.

2.2. *Computer vision based object transcoding*

Our proposal aims to extract objects from frames on-the-fly, without manual annotation. Unfortunately a unique approach for segmenting frames into objects does not exists, being dependent on the application and the user requirements. We define *object* a set of (connected) pixels sharing some visual features. With *visual features* we mean some characteristics, such as color, texture, optical motion field, etc., that can be computed by means of image analysis and computer vision. The techniques we used to segment the scene into objects will be described in the next section. Moreover, once objects have been segmented, processes of model-based vision are used to classify them into the different classes. For example, among moving objects people are detected by vehicles by using simple rules based on the total area, the profile and on the ratio between the height and the width of the bounding box (people are proportionally taller, while vehicles are wider). To detect faces we use an approach described in [18] and based on the analysis of the histogram obtained by the projection of the shape of the object on the horizontal axis: the rationale is that the head of a person will produce a mode in the histogram. Although very trivial, these techniques work well in most of the cases we considered. With these techniques, we are able to group objects in *classes of relevance*.

In particular, for each type of video and application the computer vision system can apply some operators able of extracting visual features, segmenting objects and classifying them. Thus, a set of classes of relevance $C = \{C_i\}_{i=1...c}$ is *a-priori* defined. For instance the class of `people` corresponds to all the objects that can be classified as "people". A special class is the class `background` with the meaning of "background scene": when it is applicable, i.e., when the camera is not in motion and acquires a scene with a fixed background, the class is not associated with segmented objects but corresponds to the background model that has been defined (see next section for details). Moreover, another special class `other` is used to label all the objects that cannot be classified in other classes. The classes used can be defined statically or selected by the user (that can choose in the set C which classes are interesting). For instance, let's suppose that the system is able to extract people, vehicles, stopped vehicles and background; in a traffic scene a user may want to select two classes only, `background` and `vehicles` (in this case people and stopped vehicles are automatically included in the class `other`), while another user can be interested in vehicles, stopped vehicles and background (e.g. in the case of the surveillance of a parking area). Classes are ordered with a weight (for sake of simplicity we normalize the sum of weights to 1) representing the degree of interest of the class. An example used in this work is the following: `[People, Background]=[0.9, 0.1]` means that the user is strongly interested in people (at 90%) and does not care much of the background; this last class could be even distorted, whether the bandwidth limitation require it. Instead, `[Face,People,Background]=[0.8,0.1,0.1]` means that the faces of people are the most semantically valuable parts, while the people's body and the background are less meaningful. Thus we propose to segment frames into objects, classify them by visual features and mix standard transcoding techniques, adopting different methods (or simply, different compression rates) with respect to the classes of relevance.

2.3. *Detecting objects for object transcoding*

In this paper we assume that the main visual feature for segmentation is *motion*. The typical application contexts could be video-surveillance, traffic control, remote people or environment control, in which the semantic we are interested in is basically the moving objects perceived in the scene. Thus we need a computer vision process which extracts shapes characterized by a not null motion.

We assume that videos are acquired with a fixed camera, and the background, although not fixed, is almost stationary; slow background changes are due to luminance variation and objects that, changing their motion status, are included in or exit from the background. We used an approach developed as a general-purpose system for moving object segmentation in the scene, called Sakbot[19]. It is currently tested for indoor and outdoor video-surveillance, in traffic control and domotics applications. As shown in Fig. 1, the Sakbot input is the current frame; if the live video comes encoded from a network camera, video must initially be

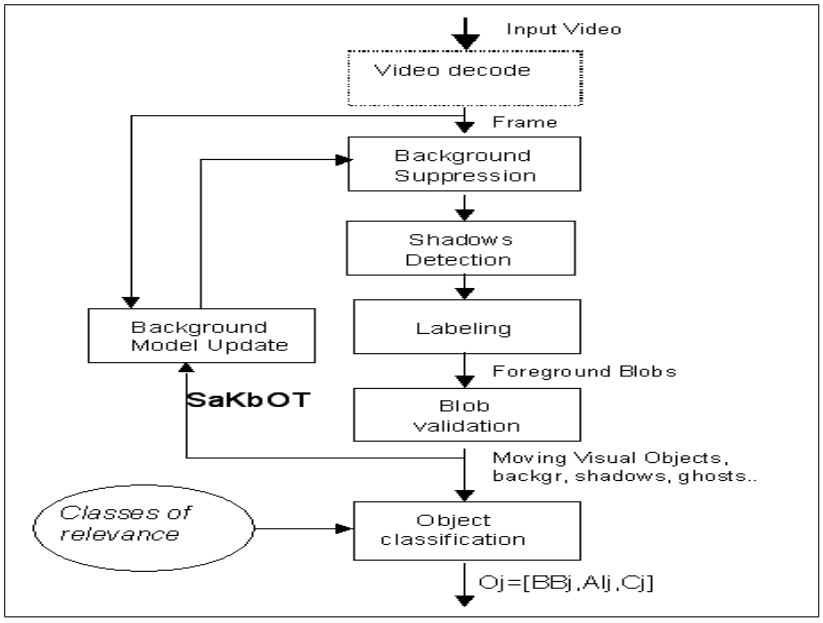

Figure 1: The Sakbot system for object segmentation

decoded into frames. In our prototype, for instance, input videos are in MJPEG format with the lower compression rate possible. Objects are extracted by means of background suppression: the Sakbot acronym (Statistical And Knowledge Based ObjecT detector) derives from the model we use for background update. It is based on the concurrent use of a *statistical* process on previously sampled frames and the *knowledge-based* feedback from previously detected moving objects. For details on the moving object detection in Sakbot, refer to [19,20]. The system provides a suitable shadow detection[21], so that shadows are not confused with objects, if it is necessary: for tracking applications, for instance, shadows are a problem since causes under-segmentation, while for other purposes like human-interactive surveillance, shadows should be linked with objects to provide the best scene fidelity possible. Pixels are grouped into blobs; blobs are further validated and are labeled as real moving objects, ghosts (objects with a only apparent motion), stopped objects, shadows [20,12] and can be classified according with classes of relevance. For instance, we provide a simple classification of objects in vehicles, stopped vehicles, people, people's face and body and other, based on area, motion and some geometrical constraints. The Sakbot system allows a fast object segmentation based on motion. In our prototype with last generation PC, we are able to segment objects at 10 frame/sec, that is the frame rate of the input images coming from the network camera. Then a further tracking phase follows, able to correlate objects among frames. Tracking is necessary as soon as a temporal compression at object level (similarly to MPEG-4) is provided. In this work, dealing with video streaming and calling for a very fast cod-

ing for on-the-fly transcoding we do not address temporal compression and do not need a sophisticated tracking. A tracking module is used only in order to determine when an object remains still (stopped) enough to be included in the background.[12] The output of the module, for each frame k is a list of objects $O_j^k = [BB_j^k, Al_j^k, C_j^k]$. BB_j^k is the textured bounding box of the object, i.e. the smallest rectangle of pixels containing the object, with the reference of the co-ordinate w.r.t. the frame. Al_j^k is the binary alpha plane, as defined by MPEG-4 standard; it represents the exact points segmented as belonging to the object. C_j^k is the class of relevance of the object that gives information to transcoding. A special class is the background class, which has the bounding box and alpha plane as large as the frame. It corresponds to the model of background that is computed and updated by Sakbot. For instance, if a point is not covered by moving objects along time, the color values are computed statistically over a number of previous sample to limit small variations due to noise, otherwise the background value derives from previous background values computed in that point[19]. Finally, Sakbot is able to deal with objects that change their status from moving to stopped and viceversa, by classifying them and updating the background model accordingly. Moreover, Sakbot's classes of relevance provide the capability to selectively choose how to behave in updating the background in the case of interesting or not interesting classes. If an object detected as moving does not belong to a class selected as relevant by the user or it belongs to the `other` class, it is not sent to the client and thus removed from the scene. For instance, if a people moves a chair, after a defined number of frame and according with the statistical model, the chair is included in the background model in the new position (and also the old position is updated). In fact the objects in the `background` class have a status flag ($Changed/NotChanged$) that indicates if the current background results to be different enough from the background model, updated by Sakbot.

For instance in the image of Fig. 2 a chair is moved in the upper part of the image. In this figure we report three examples of behaviour in dependence of the policy adopted for the background update and the classes of relevance. If the background is not correctly updated by using Sakbot approach and there is no distinction between classes of object, we will have two chairs instead of one in the upper part of the image (first image in the upper left corner of Fig. 2): in fact the chair is not removed from the background in its old position and is sent as moving object (at higher quality) in its new position. By using the knowledge of the classes of relevance is possible to distinguish between useful classes (the person) and useless ones (the chair): thus, the chair is not sent in its current position. Finally, if the background is correctly updated, the chair disappears also from its old position, since it is removed from the background (lower left image). In the lower right image of Fig. 2 the situation after some frames is reported: in this case the system includes the chair in the background in its current (still) position and thus the chair is transmitted at lower quality.

2.4. *The video server architecture for object-based transcoding*

Figure 2: Results obtained b y the system when a non-interesting object (a chair) is moved; from the upper left: result when the system does not use Sakbot to update the background and does not have any knowledge of the classes of relevance; result when it does not update the background with Sakbot but uses classes of relevance; result using both backgroudn updating and classes of relevance; situation some frames ahead when the chair stops moving and is included in the background.

The reference architecture is designed for a "smart" video server that as well as having camera interface and Web interface has also the processing power to process videos frame by frame. Fig. 3 shows the architecture of our prototype, the ImageLab Video Server.

The system architecture is structured in three modules: a *Web server module* will deal with the client requests and with their capabilities; a *Transcoding Policy Resolver* (TPR) module will choose the best transcoding policy, trying to match the requested video characteristics and the client capabilities; a *Video Transcoder* module will apply the selected transcoding policy. The communication between the different modules is designed by means of XML coded messages to allow easy integration of new modules in the future.

The Web server module uses a standard cookie-based system to identify the

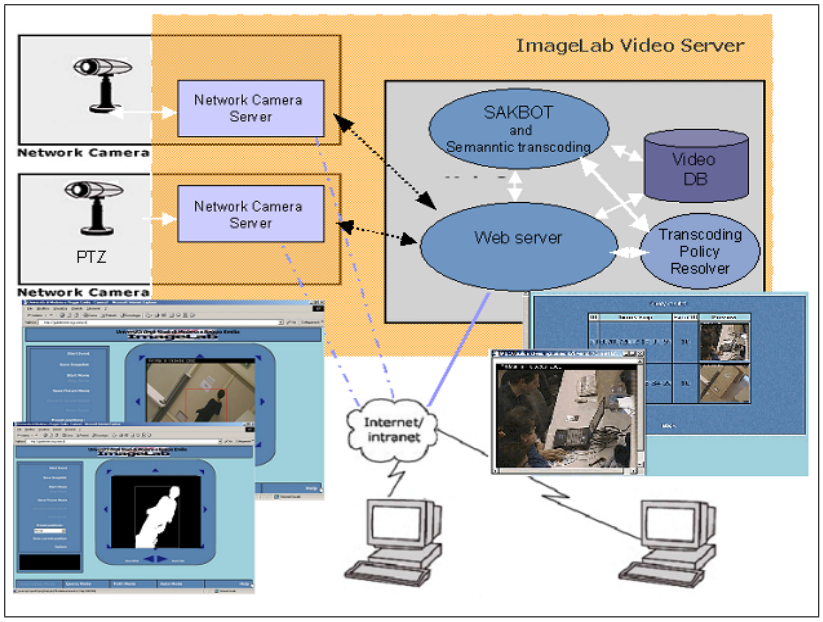

Figure 3: The architecture of the video server for semantic transcoding

client so that, after a first description of the system, each request can be associated with its capabilities. The client data are its capabilities (maximum bandwidth, display size and color or black and white display) and its preferences including the type of video source (live or from a video database) and the weights of the classes of relevance, if semantic transcoding can be exploited. Conversely, all scene objects will be considered without semantics. The module creates "Capabilities" and "Request" XML messages, that are then passed to the TPR.

The TPR module gets the "Request" message and queries the video camera properties or the stored video database. The videos are characterized by frame rate, color depth and video class. Three classes of video are possible corresponding to different acquisition type: fixed camera, PTZ (pan tilt zoom) camera with constraint motion, PTZ with free motion. This is important since it affects the type of computer vision processes involved. In our prototype we have a fixed camera and a PTZ Axis[13] camera with a network camera server connected to the main video server by means of a link of large bandwidth, so that it is does not introduce bottleneck. In case of stored video a caching policy is also considered. The TPR then makes a match between the actual video characteristics and the client capabilities starting by fitting requested maximum size and color depth. Spatial and color transcoding do not give much advantages in terms of bandwidth (see the section reporting the results) and decrease considerably the object fidelity. Therefore, they are typically adopted only as a first step to fir the display requirement.

After fitting the requirements, the video is processed by Sakbot and the objects

and their relative classes are extracted. By using the weights associated by the user to each class, the different classes (including the special classes `background` and `other`) are JPEG compressed. The level of the compression is proportional to the assigned weight. We decided to not use a linear function to correlate the weights and the compression level, but to adopt a simple four-level decision rule. Thus, for the i-th class with associated the weight w_i, we use a JPEG compression with the compression factor CF as follows:

$$CF = \begin{cases} 20 & if \quad 0 \leq w_i < 0.2 \\ 40 & if \quad 0.2 \leq w_i < 0.5 \\ 60 & if \quad 0.5 \leq w_i < 0.7 \\ 80 & if \quad 0.7 \leq w_i \leq 1 \end{cases} \tag{1}$$

The Video Transcoder applies the conversion; if object-based transcoding policy is adopted, it exploits Sakbot for detecting objects; then feeds the data back to the Web server for transmission. Since the compression depends on the number of objects in the scene and on their size, the output dimension is checked and used to adjust the quality factor of the JPEG compression accordingly. Having the knowledge of the objects in the scene a large number of transcoding policies could be designed, by mixing the transcoding policies. As introduced, we now do not consider temporal compression with forward and backward prediction, as MPEG standards (some comparison will be done at the end of the paper), neither in the standard codes nor in our object-based ones. Nevertheless our work will be integrated in a MPEG-4 framework with streaming and object-based capability, whenever the MPEG-4 standard will be well assessed with MPEG-4 codec fast enough to be used on-the-fly. Currently the execution time for the encoding process is very high (and one order of magnitude higher than the decode time [22]).

In a general framework we should tract differently objects according to all the classes of relevance. Instead the background class is sent with both code transcoding and temporal transcoding: the latter is achieved dynamically, sending a new background only when it is significantly different from the reference background (with the *Changed* label). The composition (i.e., the merge between moving objects and the background model) is then performed in two different ways:

- `BB_obj_tr` (BB object transcoding, with bounding boxes)
- `Alpha_obj_tr` (Alpha object transcoding with alpha planes)

The difference between the two methods are in the object manipulation. In the (`BB_obj_tr`) transcoding for each frame we send a list of bounding boxes [BB_1^1, BB_2^1, .., BB_1^2, .., BB_j^k, ..] compressed in JPEG and containing the color values of the object's pixels. In the `Alpha_obj_tr` transcoding for each frame we send a list of bounding boxes compressed in JPEG and the alpha planes coded in RLE (run length encoding) that is more suitable for B/W templates [(BB_1^1, Al_1^1), (BB_2^1, Al_2^1), .., (BB_1^2, Al_1^2), .. (BB_j^k, Al_j^k)]. In the BB case, the bounding boxes are simply superimposed to the background image (with no transparency), starting from the

coordinates of the upper left corner encoded in the bounding box (see previous section). In the case of the alpha planes, the B/W masks are used to decide singularly which pixels to superimpose to the background.

The difference in bandwidth is almost negligible (slightly higher in the second case). For the perceptual point of view of the whole scene the second is better since allows a better mosaicing (bounding box borders are visible in the first case if the background has even small changes). In theory, the use of alpha planes should be the best approach with an ideal moving object segmentation. In practice, from the point of view of fidelity of moving objects the first is the best choice since it overcomes some unavoidable under-segmentation errors of the computer vision process: objects extracted by Sakbot are eroded in the border or can have some holes (if they are too similar to background) and these errors are compensated if the real scene contained inside the bounding box is sent.

3. Performance Evaluation Metrics

Evaluating the result of transcoding is not trivial. If the video has not an associated semantic, i.e. there is no distinction between important and useless information, the trade-off between the bandwidth reduction and the minimal distortion of the information is typically the best choice. On the other hand, in real applications the limited bandwidth of the connection is the key constraint and, therefore, the distortion should be minimized.

Whenever the user can notify its interest in the video content, the quality of transcoding should be associated to the satisfaction of the user, and a model to measure it should be defined. In [1] a performance measure V called *value* is introduced ranging from 1 and 0 (0 when the object is excluded) but the authors state that "the drawback is that we still do not have a computational mechanism for determining V". In [14] a concept of *value* of a video object is measured as $\eta = (\sigma + c)\frac{B}{\gamma}$, being σ the intensity of motion activity as the standard deviation of the motion vector in MPEG-7 [23,7], B the number of bits and c and γ two normalization parameters. Thus objects have a high value if large and with a high motion. In spite of the definition this model is not used in detail, and is accounted only for deciding if a multimedia object should be dropped or not.

Similarly, we define a model of performance analysis based on the user interests by means of classes of relevance to test the transcoding policies simulating different applications. In the context of semantic video transcoding the defined classes of relevance give a priority in the value of objects that are in the video. Think for instance to video-surveillance applications in which a video from live camera is transmitted remotely to a human operator. In these applications the operator can be interested in seeing only the moving people inside a room: the best transcoding policy in this case should be the one that sends the moving people without any compression and does not send the static part (background) of the scene at all. For this reason the distortion introduced in the background should not be considered

(weight equal to 0) or should have a very small weight. Another example can be biometric-based surveillance in which the face of moving people can be the more important region of the scene. Thus we use classes of relevance and their weight not only for deciding the transcoding policy but also for evaluating performance.

A common metric to measure the distortion/error in compressed/transcoded images is the *Peak Signal-to-Noise Ratio (PSNR)*[24], defined as:

$$PSNR = 10 \log_{10} \left(\frac{V_{MAX}^2}{MSE} \right) \tag{2}$$

where V_{MAX} is the maximum (peak-to-peak) value of the signal to be measured and MSE is the Mean Square Error, typically computed as:

$$MSE = \frac{1}{NM} \sum_{i=1}^{N} \sum_{j=i}^{M} d^2(i,j) \tag{3}$$

with $d(i,j)$ a properly defined distance to measure the error between original and distorted images. As distance, we used the *Euclidean distance* in the RGB color space, that is:

$$d(i,j) = \sqrt{(I_O^R(i,j) - I_D^R(i,j))^2 + (I_O^G(i,j) - I_D^G(i,j))^2 + (I_O^B(i,j) - I_D^B(i,j))^2} \tag{4}$$

being I_O the original image and I_D the distorted version of it. Consequently, V_{MAX} is equal to $\sqrt{3} \cdot 255$.

We define a performance evaluation model that accounts for classes of relevance. To this aim, we call $WMSE$ (*Weighted MSE*) the following measure:

$$WMSE = \sum_{k=1}^{N_{CL}} w_k \cdot MSE_k \tag{5}$$

where N_{CL} is the number of classes of relevance and MSE_k can be written as:

$$MSE_k = \frac{1}{|C_k|} \sum_{(i,j) \in C_k} d^2(i,j) \tag{6}$$

where C_k is the set of the points belonging to the class k and $|C_k|$ is its cardinality.

To decide whether a point (i,j) belongs to a certain class or not, we compare with a manually segmented ground truth for the tested videos. The weights w_k are chosen by user (or tuned a-priori) and with the following rules:

$$w_k \geq 0 \quad \forall k = 1, ..., N_{CL} \quad ; \quad \sum_{i=1}^{N_{CL}} w_k = 1 \tag{7}$$

Clearly, in the absence of semantic, $WMSE \equiv MSE$. We use PSNR as in Eq. 2, with WMSE in place of MSE.

As a complementary measure, we use the bandwidth B expressed in Kb/s. To outline the improvement introduced by the transcoding, the *bandwidth enhancement* ($\frac{B_O}{B_D}$) is also reported.

4. Results Analysis

We compare, both in terms of PSNR and bit rate, the five types of transcoding reported in Subsection 2.1. In the case of *spatial* transcoding, we reduced the size of the image from CIF (352x288) to QCIF (176x144). The *temporal* transcoding analyzed consists in taking one frame each seven (this number has been chosen in order to obtain a bandwidth similar to the other methods). The *color* transcoding uses grayscale and b/w images. As *code* transcoding we limited our study to frame by frame compression, and in particular to the MJPEG compression (JPEG with different compression values). A test with MPEG is then described , even our techniques have not any temporal compression as MPEG standard includes. All the transcoded versions except for the binary image have been compressed with JPEG (CF=20). In the case of binary images (1-bit images) the JPEG compression (that does not allow 1-bit compression) results in worst bandwidth performance. The references videos are initially in an uncompressed format (after having decoded the MJPEG format from network camera), with true color and CIF size, at 10 fps (frames per second). As benchmark, we used two very different videos: the first is a typical video-surveillance sequence acquired by the live network camera installed in our laboratory (see `http:\\guilderstern.ing.unimo.it`) with people moving; the second one has been acquired in a parking lot and stored in our database and consists in a car approaching to the lot and parking in a free space.

We envisioned two Web applications: accessing to the first video, users may have interest mainly in seeing the people present in the scene. On the other hand, the second video's purpose can be twofold: user may want either to see moving vehicles approaching to the parking lot or to count parked cars. In this latter case, background is far more relevant than moving vehicles. To this aim, we simulated some tuples of weights of classes of relevance.

The novelty of this work is the use of *semantic-based* performance analysis with four case studies of user interest: one application without semantic, one with two classes of relevance `[People,Background]` or `[Vehicle,Background]` and two with three classes `[Face,People,Background]` (people is only people body without face). In the latter cases, we simulated a surveillance application (in which faces are the most important information) and a "landscape-view" application (in which background is the essential information to be transmitted).

4.1. *Indoor sequence*

Table 1 reports the value of WMSE and PSNR for the different transcoding policies for the four cases in the indoor benchmark. The values reported in the first column between square brackets are the weights w_k applied to the different

classes (P=people, F=faces, B=background). The first row has performance figures without semantic, with the standard MSE and PSNR definition. In the second row we simulate users searching for people only, in the third for face while in the last row users are more interested in the background Lab. An example is reported in Fig. 4 with three people in the Lab: one is hidden by the desks, one is walking around and one is sitting on the left. Fig. 5 shows details of the same frame.

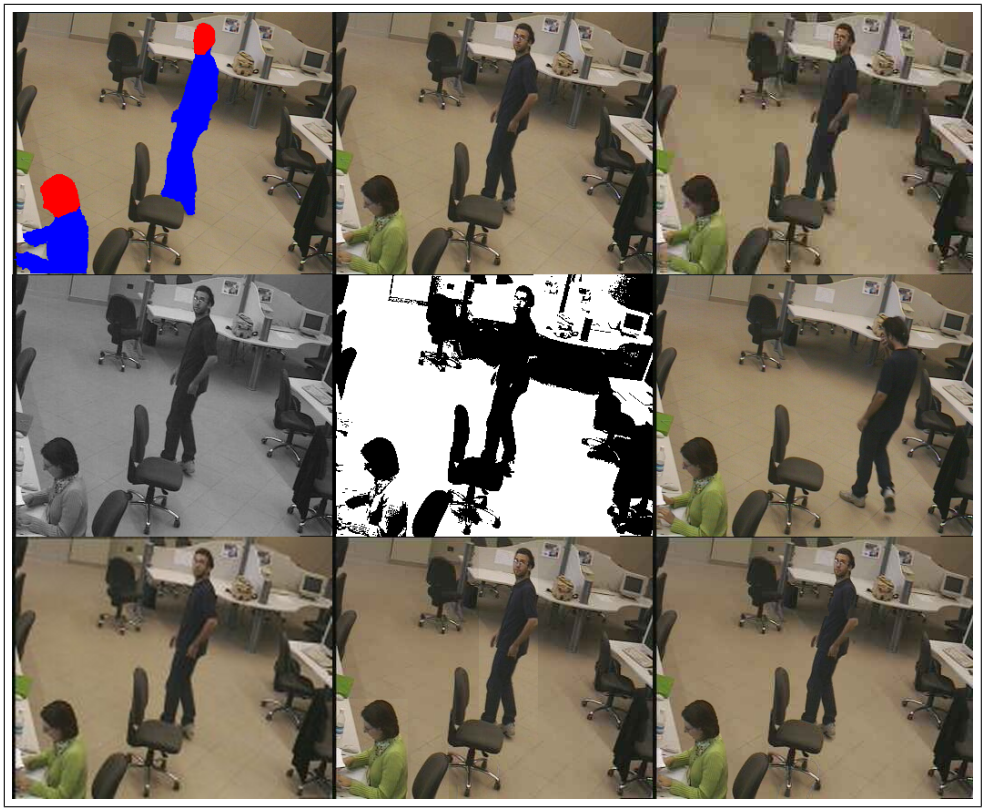

Figure 4: Indoor benchmark comparison; from top left: the ground truth, `code_tr` JPEG (CF=20), `code_tr` JPEG (CF=80), `color_tr` gray levels, `color_tr` binary, `temp_tr` 1 of 7, `spat_tr`, `obj_tr` BB, and `obj_tr` Alpha

As foreseeable, the best PSNR (that is the lowest distortion, measured as WMSE) is achieved by using code transcoding with low compression (JPEG CF=20). The other methods add to the code transcoding other processes and generate a not negligible distortion. Color transcoding, both grayscale and B/W, changes a lot scene perception. Temporal transcoding too: for instance if you are interested in the people, temporal transcoding sends people data at time steps and pixels of the people are, in average, distorted considerably. Nevertheless, not only "pure" JPEG, but also our object transcoding preserves data from distortion. Obviously, in this case

Figure 5: Indoor benchmark comparison: details; same order of Fig.4

the PSNR depends on the weights. Background is sent with temporal transcoding, and is not always correct (we moved a chair in the scene to change the background); thus the performance is low in the case of no semantic or background's relevance. Instead, if users are interested in people or faces, these parts are sent with the same detail of JPEG (adding only some over- and under- segmentation errors due to the computer vision process). For example the weighted PSNR when people are more relevant than background is similar: 35.31 dB and 34.54 dB for JPEG and object transcoding (BB), respectively. The Fig. 6 describes the PSNR comparison graphically.

The most important measure is the bandwidth occupation of the original video and of those transcoded reported in Table 2. The proposed object transcoding can

reduce the bandwidth from 23 Mb/s to 204 Kb/s or 238 Kb/s by maintaining most of the useful information, while JPEG applied indifferently over all pixels has a 1283 Kb/s bandwidth occupation. It is possible to note that the best bandwidth enhancement is obtained by performing temporal transcoding, but this degrades heavily the data (25 dB of PSNR in the best case).

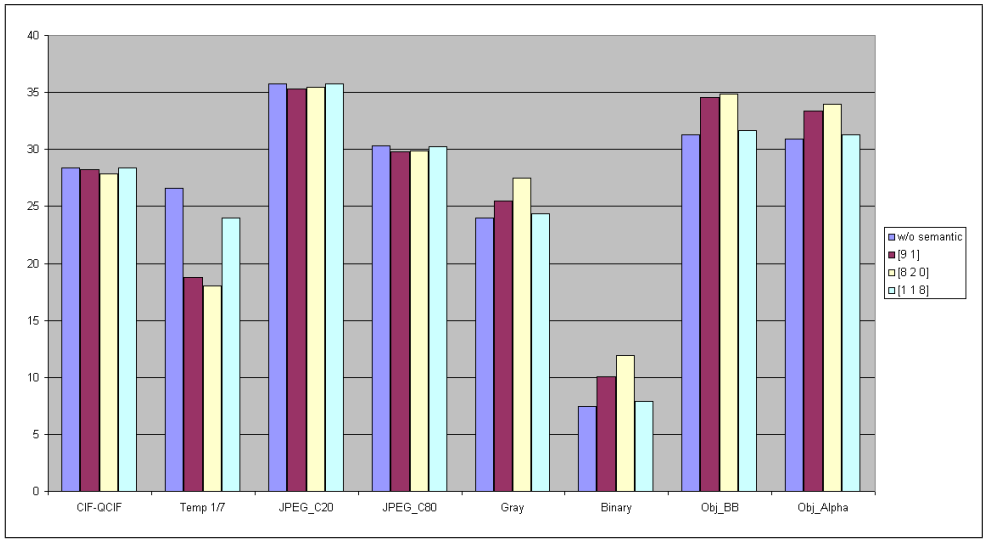

Figure 6: PSNR comparison of indoor sequence; four cases are considered: without semantic, with `[People,Background]=[0.9,0.1]`, `[People,Face,Background]=[0.8,0.2,0]`, `[People,Face,Background]=[0.1,0.1,0.8]`

4.2. *Outdoor sequence*

The same tests have been carried out with an outdoor benchmark; a sample frame is in Fig. 7 (and in Fig. 8 in detail). Vehicles are detected because their motion and area; when a vehicles stopped for a period higher than a defined timeout it becomes part of background. Performance analysis confirms the previous considerations. In this video the original bandwidth is slightly higher since the input image size is 360x288, larger than a CIF standard. From the point of view of bandwidth enhancement by means of transcoding, the object transcoding outperforms other methods (two order of magnitude w.r.t. JPEG with the same compression rate in the moving objects). This is due to the fact the objects of interested have a very small size (the large shadows, segmented by Sakbot, are included in the class of relevance of the background). On the other hand, background is distorted since there is a little latency between the time when the car stops and when the background is updated. It is described in Table 3, showing the WMSE. Without object semantic in performance analysis, the distorted background forces a low PSNR (29.27 of

Table 1: Indoor benchmark - Weighted Mean Square Error (WMSE) for the transcoding policies with different weights. The numbers in brackets are the Peak Signal-to-Noise Ratio (PSNR).

Parameters	spat_tr	temp_tr	code_tr	
	Resize CIF-QCIF	**1 frame each 7**	**JPEG (CF=20)**	**JPEG (CF=80)**
w/o semantic	281.63	428.10	51.65	180.38
(*PSNR* in dB)	(28.41)	(26.59)	(35.77)	(30.34)
P/B [0.9 0.1]	291.15	2590.68	57.37	204.70
(*PSNR* in dB)	(28.26)	(18.77)	(35.31)	(29.79)
F/P/B [0.8 0.2 0]	320.17	3091.40	55.31	201.81
(*PSNR* in dB)	(27.85)	(18.00)	(35.47)	(29.85)
F/P/B [0.1 0.1 0.8]	284.63	785.59	52.08	182.97
(*PSNR* in dB)	(28.36)	(23.95)	(35.74)	(30.28)

Parameters	color_tr		object_tr	
	Grayscale Image	**Binary Image**	**Object BB**	**Object Alpha**
w/o semantic	781.05	34983.52	145.80	157.24
(*PSNR* in dB)	(23.98)	(7.46)	(31.26)	(30.94)
P/B [0.9 0.1]	548.39	19399.87	68.52	89.05
(*PSNR* in dB)	(25.51)	(10.02)	(34.54)	(33.41)
F/P/B [0.8 0.2 0]	349.86	12469.84	63.73	78.84
(*PSNR* in dB)	(27.46)	(11.94)	(34.86)	(33.93)
F/P/B [0.1 0.1 0.8]	719.19	31582.43	133.12	145.46
(*PSNR* in dB)	(24.33)	(7.91)	(31.66)	(31.27)

Table 2: Indoor benchmark - Bandwidth requirement in Kb/s and enhancement introduced by the transcoding policies.

	Original video	spat_tr	temp_tr	code_tr	
		Resize CIF-QCIF	**1 frame each 7**	**JPEG (CF=20)**	**JPEG (CF=80)**
Bandw. (Kb/s)	23762.33	438.14	185.50	1282.89	440.78
Bandw. enhanc.	1.00	54.23	128.10	18.52	53.91

	Original video	color_tr		object_tr	
		Grayscale Image	**Binary Image**	**Object BB**	**Object Alpha**
Bandw. (Kb/s)	23762.33	1228.34	280.72	203.91	237.70
Bandw. enhanc.	1.00	19.35	84.65	116.53	99.97

BB_obj_tr versus 37.13 of JPEG). The differences are almost flattened if user is not interested in the landscape (31.66 of BB_obj_tr versus 33.36 of JPEG). A difference still exists since the computer vision task that segments moving object automatically makes little errors in some object parts (e.g., the car window). Nevertheless, the perceptual results with a so limited bit rate (43.02 Kbps) are very good.

4.3. *Comparison with fixed bandwidth*

In addition, we simulated the behaviour of the system with clients with very limited bandwidth resources and test our method (BB_obj_tr) against normal coding compression. Results are visually presented for comparison in Table 5. In this case a constant bit rate (CBR) has been imposed by considering three standard bit rate

Table 3: Outdoor benchmark - WMSE and PSNR for the transcoding policies with different weights (F=Foreground, B=Background).

Parameters	spat_tr	temp_tr	code_tr	
	Resize CIF-QCIF	**1 frame each 7**	**JPEG (CF=20)**	**JPEG (CF=80)**
w/o semantic	489.35	175.88	37.73	284.45
(*PSNR* in dB)	(26.01)	(30.45)	(37.13)	(28.36)
F/B[0.9 0.1]	542.40	3388.48	84.68	384.66
(*PSNR* in dB)	(25.56)	(17.60)	(33.62)	(27.05)
F/B[0.1 0.9]	495.05	471.10	42.25	294.29
(*PSNR* in dB)	(25.96)	(26.17)	(36.64)	(28.21)
F/B[1 0]	548.32	3753.15	89.99	395.96
(*PSNR* in dB)	(25.51)	(17.16)	(33.36)	(26.93)
F/B[0 1]	489.13	106.43	36.94	282.99
(*PSNR* in dB)	(26.01)	(32.63)	(37.23)	(28.38)

Parameters	color_tr		object_tr	
	Grayscale Image	**Binary Image**	**Object BB**	**Object Alpha**
w/o semantic	465.52	25869.73	230.97	232.20
(*PSNR* in dB)	(26.22)	(8.77)	(29.27)	(29.24)
F/B[0.9 0.1]	735.34	18647.45	143.16	144.37
(*PSNR* in dB)	(24.24)	(10.20)	(31.34)	(31.31)
F/B[0.1 0.9]	491.40	25206.04	222.92	224.15
(*PSNR* in dB)	(25.99)	(8.89)	(29.42)	(29.40)
F/B[1 0]	765.83	17827.62	133.18	134.40
(*PSNR* in dB)	(24.06)	(10.39)	(31.66)	(31.62)
F/B[0 1]	460.90	26025.86	232.89	234.12
(*PSNR* in dB)	(26.27)	(8.75)	(29.23)	(29.21)

Table 4: Outdoor benchmark - Bandwidth requirement in Kb/s and enhancement introduced by the various transcoding policies.

	Original video	spat_tr	temp_tr	code_tr	
		Resize CIF-QCIF	**1 frame each 7**	**JPEG (CF=20)**	**JPEG (CF=80)**
Bandw. (Kb/s)	24302.83	625.54	306.46	2077.68	648.95
Bandw. enhanc.	1.00	38.85	79.30	11.70	37.45

	Original video	color_tr		object_tr	
		Grayscale Image	**Binary Image**	**Object BB**	**Object Alpha**
Bandw. (Kb/s)	24302.83	1878.78	389.49	43.02	46.67
Bandw. enhanc.	1.00	12.94	62.40	564.97	520.70

(56, 128 and 256 Kbps).The PSNR in the case of no semantic is reported. Note that in the case of JPEG compression video at 56 Kbps the actual bandwidth obtained is 72 Kbps. Additional transcoding policies have been used to be able to fit the bandwidth requirements. Even reducing the size from QCIF to CIF and using the greater compression available, JPEG in not able to meet the 56 Kb/s requirement. Temporal downscaling is mandatory in this case. Finally, a result worthy of note is that our method can operate at full resolution when the bandwidth is 256 Kb/s or greater.

As previously introduced, our semantic transcoding does not exploits any temporal compression between frame. Instead it is well known that motion prediction, as

Figure 7: Outdoor benchmark comparison; same order of Fig.4

it used in MPEG-2 and following standard is a powerful method to save bandwidth for still image's parts, sending only differences. This will be included in our approach using the same MPEG-4 issues. Although the comparison is now unfair, we compare in the Fig. 10 performance of the indoor sequence with object transcoding and an off-line MPEG-2 compression. We transcoded the video in order to achieve a 128 Kbps bit rate. MPEG 2 compression, thanks to prediction, does not need resize. Differently from the case of Table 5, we achieved 128 Kbps without resize but with a strong JPEG compression. In fact PSNR without semantic is 27.66 dB and 29.39 dB for our object transcoding and MPEG-2, respectively. Instead, if the user is more interested in moving objects, they result less distorted, as is shown in the inner columns of the graph of Fig. 10.

5. Conclusions

In this paper we reported a framework for video on-the-fly transcoding. The main goal was to propose the adoption of general-purpose computer vision techniques in order to allow a semantic transcoding that takes into account the objects moving

Figure 8: Outdoor benchmark comparison:details; same order of Fig.4

in the scene. More worthy, we have described as it is potentially possible to adapt the transcoding policies not only to the bandwidth and resource capability but also to the interests of the user, by means of classes of relevance. In the future, we will improve this idea, by increasing the number of class of relevance and designing a framework that provides autoamtic association to each of them the most profitable transcoding.

We classified existing transcoding policies and compared them with a novel object-based transcoding. It allows to transmit 10 fps videos at very low bit rate with an acceptable distortion of the meaningful data. It could be very useful for applications such as video conference, Web access by clients with limited resources, remote video-surveillance...

We proposed a performance evaluation metric based on the WMSE and that takes into account different weights in accordance with the semantic of the scene and the relevance of the classes of the objects for that application. By using this metric we demonstrated that a mixed transcoding where objects are compressed or downscaled differently is able to maximize the bandwidth reduction and to minimize the error introduced. Since we are interested in fast transcoding for live video server with available on the market technology, we do not adopted temporal prediction in the object streaming: the improvement with the use of semantic will be more valuable with a transcoding with objects and prediction as the MPEG-4 standard.

In fact, our first future direction will be to include temporal prediction in our

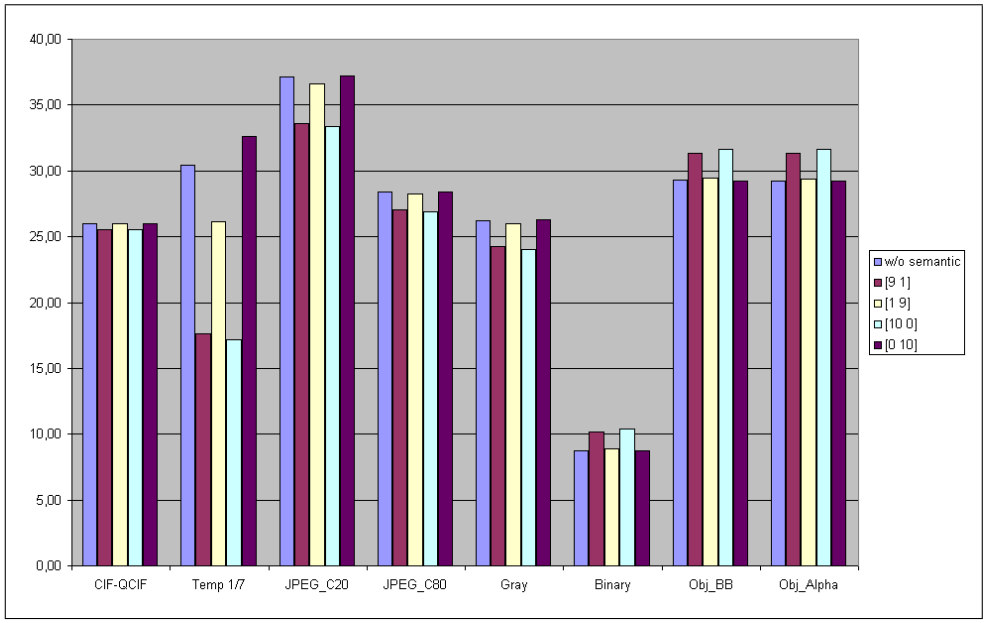

Figure 9: PSNR comparison of outdoor sequence; five cases are considered: without semantic, with `[Vehicle,Background] = [0.9, 0.1]`, `[Vehicle,Background] = [0.1, 0.9]`, `[Vehicle,Background] = [1, 0]`, `[Vehicle,Background] = [1, 0]`

system. This will require some compromises, since live applications do not allow forward predictions: this should be overcome by taking some delay into account. We certainly want to compare our approach with off-the-shelf techniques, such as MPEG-2 and MPEG-4.

References

1. Rakesh Mohan, John R. Smith, and Shung-Sheng Li. Adapting multimedia internat content for universal access. *IEEE Transactions on Multimedia*, 1(1):104–114, March 1999.
2. A.W. Huang and N. Sundaresan. A semantic transcoding system to adapt web services for users with disabilities. In *Proceedings of the ACM SIGCAPH Conference on Assistive Technologies*, pages 156–163, 2000.
3. N. Jayant, J. Johnston, and R. Safranek. Signal compression based on models of human perception. *Proceedings of the IEEE*, 81(10):1385–1422, October 1993.
4. John R. Smith, Rakesh Mohan, and Chung-Sheng Li. Content-based transcoding of images in the internet. In *Proceedings of IEEE Int'l Conference on Image Processing*, volume 3, pages 7–11, October 1998.
5. Y. Yu and Chen C.W. Snr scalable transcoding for video over wireless channels. In *Proceedings of the Wireless Communications and Networking Conference (WCNC)*, volume 3, pages 1396–1402, 2000.

Table 5: Visual comparison of JPEG Compression vs. object transcoding policy.

	56 Kb/s	**128 Kb/s**	**256 Kb/s**
JPEG Compr.			
Transcoding policy	Resize QCIF + Temp.	Resize QCIF	Resize QCIF
PSNR w/o semantic	19.74	25.27	27.30
VO+Backgr.			
Transcoding policy	Resize QCIF	Resize 264x216	none
PSNR w/o semantic	26.63	28.91	31.65

6. Katashi Nagao, Yoshinari Shirai, and Kevin Squire. Semantic annotation and transcoding: Making web content more accessible. *IEEE Multimedia*, 8(2):69–81, April-June 2001.
7. ISO/IEC 15938-3:CD. *Information Technology - Multimedia Content Description Interface. Part 3: Visual*.
8. IBM research. *http://www.research.ibm.com/MediaStar/VideoSystem.html*.
9. Shih-Fu Chang, Dimitris Anastassiou, Alexandros Eleftheriadis, Jianhao Meng, Seungyup Paek, Sassan Pajhan, and John R. Smith. Development of advanced image/video servers in a video on demand testbed. In *Proceedings of the IEEE Visual Signal Processing and Communications Workshop*, September 1994.
10. Jeongnam Youn, Ming-Ting Sun, and Chia-Wen Lin. Motion vector refinement for high-performance transcoding. *IEEE Transactions on Multimedia*, 1(1):30–40, March 1999.
11. N. Bjork and C. Christopoulos. Video transcoding for universal multimedia access. In *Proceedings of ACM Multimedia MM2000*, Januart 2000.
12. R. Cucchiara, C. Grana, and A. Prati. Detecting moving objects and their shadows: an evaluation with the pets2002 dataset. In *Proceedings of Third IEEE International Workshop on Performance Evaluation of Tracking and Surveillance (PETS 2002) in conj. with ECCV 2002*, pages 18–25, May 2002.
13. Axis Communication : Web site http://www.axis.com.
14. Anthony Vetro, Huifang Sun, and Yao Wang. Object-based transcoding for adaptable video content delivery. *IEEE Transactions on Circuits and Systems for Video Technology*, 11(3):387–401, March 2001.
15. Surendar Chandra, Ashish Gehani, Carla Schlatter Ellis, and Amin Vahdat. Transcoding characteristics of web images. In *Proceedings of the SPIE Multimedia Computing and Networking Conference*, January 2001.
16. Jenq-Neng Hwang, Tzong-Der Wu, and Chia-Wen Lin. Dynamic frame-skipping in

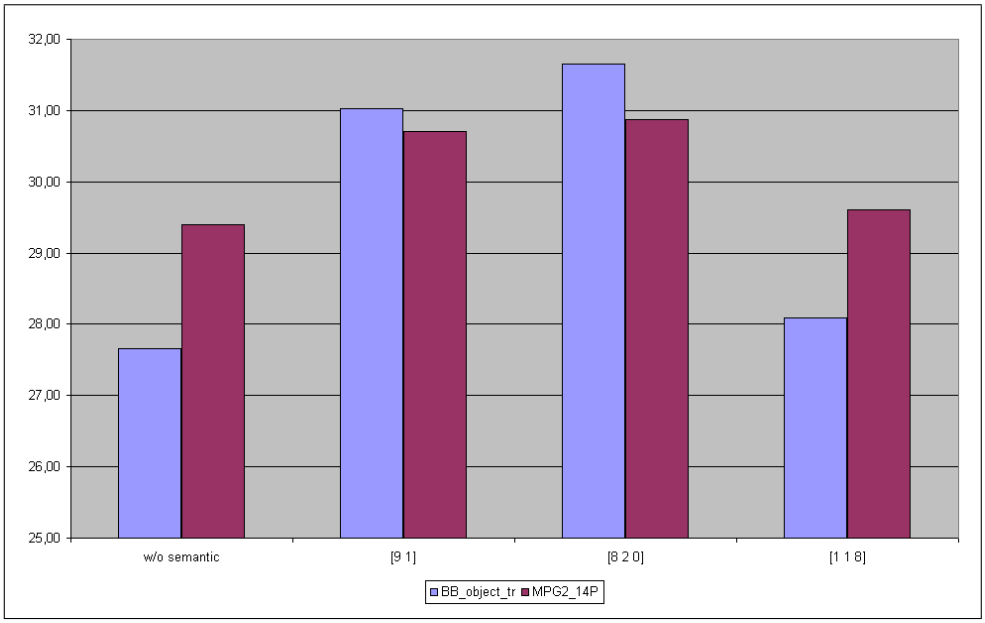

Figure 10: PSNR comparison between object transcoding and MPEG-2 of indoor benchmark at 128Kb/s; same classes of Fig. 6

video transcoding. In *Proceedings of the IEEE Second Workshop on Multimedia Signal Processing*, pages 616–621, December 1998.

17. Anthony Vetro and Huifang Sunt. Encoding and transcoding multiple video-objects with variable temporal resolution. In *Proceedings of Intern. Symposium of Circuit and Systems*, May 2001.
18. I. Haritaoglu, D. Harwood, and L.S. Davis. W4: real-time surveillance of people and their activities. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):809–830, August 2000.
19. R. Cucchiara, C. Grana, G. Neri, M. Piccardi, and A. Prati. The sakbot system for moving object detection and tracking. In *Video-based Surveillance Systems - Computer Vision and Distributed Processing*. Kluwer Academic, 2001.
20. R. Cucchiara, C. Grana, M. Piccardi, and A. Prati. Detecting objects, shadows and ghosts in video streams by exploiting color and motion information. In *Proceedings of International Conference on Image Analysis and Processing (ICIAP 2001)*, pages 360–365, Sep 2001.
21. A. Prati, I. Mikic, C. Grana, and M.M. Trivedi. Shadow detection algorithms for traffic flow analysis: a comparative study. In *Proceedings of IEEE Intelligent Transportation System Conference (ITSC 2001)*, pages 340–345, Aug 2001.
22. F. Cavalli, R. Cucchiara, M. Piccardi, and A. Prati. Performance analysis of mpeg-4 decoder and encoder. In *Proceedings of International Symposium on Video/Image Processing and Multimedia Communications (VIPromCom-2002)*, pages 227–231, Jun 2002.
23. A. Divakaran and H. Sun. A descriptor for spatial distribution of motion activity. In *Proceedings of Storage and Retrieval from Image and Video Databases*, January

2000.
24. Jerome M. Shapiro. Embedded image coding using zerotrees of wavelet coefficients. *IEEE Transactions on Signal Processing*, 41(12):3445–3462, December 1993.