GUO, ZHONG, M.S., AUG 2006 COMPUTER SCIENCE

AN ALGORITHM FOR FAST PERCEPTUAL OBJECT TRACKING IN A CODED STREAM BY ANALYSIS OF COMPOSITE SCENE MOTION'S REFLECTION ON THE MOTION VECTORS (60 PP.)

Director of Thesis: Javed I. Khan

In this research we developed an object tracking algorithm suitable for perceptual encoding. Object based bit allocation can result in significant improvement in the perceptual quality of relatively low bit-rate video. However, a particularly computationally challenging aspect in the process is the video object tracking. This is because traditional algorithms are based on classical object detection techniques. In this thesis we present an algorithm particularly designed for perceptual video coding. The algorithm is optimized for minimum decoding and uses only motion information and detail case analysis that accounts for variety of cases that arise in motion vector coding due to relative scene and camera movements. The result is a fast yet highly effective perceptual region tracking algorithm that can operate in stream rate and track regions of perceptually significant object despite camera movements such as zoom, panning and translation. We have implemented this algorithm into a live MPEG-2 perceptual transcoder. We also share the performance of its MPEG-2 compliant real implementation. This fast object aware video rate transcoder is particularly suitable for live streaming and can convert a regular coded video into an object-based perceptually coded video stream.

AN ALGORITHM FOR FAST PERCEPTUAL OBJECT TRACKING IN A CODED STREAM BY ANALYSIS OF COMPOSITE SCENE MOTION'S REFLECTION ON THE MOTION VECTORS

A thesis submitted
To Kent State University in partial
Fulfillment of the requirements for the
degree of Master of Arts

by

Zhong Guo

August, 2006

Thesis written by
Zhong Guo
M.S., Kent State University, 2006

Approved by

_____________________________________, Advisor

_____________________________________, Chair, Department of Computer Science

_____________________________________, Dean, College of Arts and Sciences

TABLE OF CONTENTS

Approval…………………………………………………………………………ii

Table of Contents……………………………………………………………..iii

List of Figures…………………………………………………………………...vi

List of Tables…………………………………………………………………vii

Acknowledgement………………………………………………………….....viii

Chapter 1

Introduction

1.1 Dynamic video transcoding – solution to highly asymmetric Internet for video servers…………………………………………………………..………1

1.2 Object-based perceptual transcoding – low bit-rate but still good QoS………..2

1.3 Related works in object detection and tracking…………………………..…….3

1.3.1 Feature-based object detection……………………………………..….…4

1.3.2 Template-based object detection…………………………………..…..6

1.3.3 Motion detection……..…………………………………………….....7

1.3.4 Object tracking using motion information……………………………….9

1.4 Real-time object tracking – challenge for in-line video transcoding…………..12

1.5 Distinctions: transcoding vs. encoding………………………...……………14

1.6 Our approach……………………………………………...……………16

Chapter 2

System Model………………………………………………………………..19

Chapter 3

Perceptual Object Detection and Tracking Model

3.1 Prediction overview……………………………………………………………22

3.2 Update overview……………………………………………………………….24

3.3 Tracking algorithm…………………………………………………………..27

3.3.1 Initialization: automatic birth of an object………………………………27

3.3.2 Tracking window prediction…………………………………………..…..29

3.3.3 Tracking window update……………………………………………...……31

3.3.4 Shape integrity adjustment………………………………………...….34

3.3.5 Tracking termination……………………………………………………….35

Chapter 4

Rate Control

4.1 Transcoder rate control mechanism……………………………………………36

4.2 Quantization factor determination……………………………………………..37

4.3 Object based activity factor determination…………………………………….37

Chapter 5

Experiments

5.1 Tracking effectiveness………………………………………………………...39

5.2 Object-based quality transcoding……………………………………...……46

5.3 Algorithm complexity…………………………………………………………50

5.4 Observed speed…………………………………………………………………...51

Chapter 6

Conclusions…………………………………………………………………52

References……………………………………………………………………………55

LIST OF FIGURES

Figure 2.1: Object aware rate transcoding…………………………………………20

Figure 3.1: The Motion Case Based iterative cycle of dynamic tracking window…...22

Figure 3.2: The four layers of macroblocks of the tracking model…………………..24

Figure 3.3: Pseudo Code…………………………………………………………..26

Figure 5.1: Calculation of object coverage and tracking window mis-coverage…….41

Figure 5.2: Results on tracking effectiveness evaluation on sample videos without camera motion…………………………………………………………………...44

Figure 5.3: Results on tracking effectiveness evaluation on sample videos with camera motion…………………………………………………………………………45

Figure 5.4: Bit distribution comparison between object and background region MBs after object-based vs non-object-based quality transcoding………………………………..48

Figure 5.5: Comparison of SNR between object and background regions after object-based vs non-object-based quality transcoding……………………………………….48

Figure 5.6: Perceptual quality comparison between non-object-based and object-based quality transcoding………………………………………………………………49

LIST OF TABLES

Table 5.1: Sample Video Sequences……………………………………………...42

Table 5.2: Video Content Description and Their Tracking…………………………43

Table 5.3: Time Costs for Object Tracking……………………………………….51

ACKNOWLEDGEMENT

I would like to express my deep gratitude and appreciation to Professor Javed Khan, who continually guides and supports me for this thesis. He has not only been an amazing advisor for this thesis but has also been of help on academic and personal matters.

I would also like to thank faculties and stuffs in Department of Computer Science, Kent State University. They provided all kinds of valuable help during the time I study here. Without their generous support, I would not be able to complete my MS degree.

I am also thankful to my friends ---classmates, lab mates, and officemates --- who make my stay at Kent State University so enjoyable and memorable.

Last but not the least, special thanks to my wife without whose love, support and encouragement this would not have been possible. I dedicate this thesis to my wife.

CHAPTER 1

INTRODUCTION

1.1 Dynamic video transcoding – solution to highly asymmetric Internet for video servers

The asymmetry in the Internet capacity- particularly at the egress networks is growing dramatically. With the adoption of digital video standards, deployment of broadband, and the availability of low cost cameras there is now an enormous flux of video content. However, at the same time the relatively slower increase of bandwidth at network edges and simultaneous flooding of small and mobile devices (such as Personal Digital Assistant) seems to indicate that the bandwidth asymmetry is also on the rise. The Internet servers are now increasingly facing severe streaming asymmetry. In one hand high performance networks such as Grid offers the opportunity to share live programs at unprecedented quality while in other hand the ubiquitous deployment of myriad of mobile devices has already created huge user base who would like to login to the same session from limited devices, even from mobile phones.

Unfortunately, the current video server technology will not be able to meet such challenges of asymmetric access in the emerging new era of ubiquitous video. Video transport scalability profiles [1] designed in pre-web era also failed to provide effective solution to the problem. They are quite inflexible and support videos only in a few predetermined classes. With the exception of some specific rate combinations, their coding efficiency also diminishes. Therefore they are hardly used. Consequently, almost all practical servers now prefer to use the naive method of pure duplication. Completely separated videos files are prepared and stored ahead of time in a few popular rate classes. Then viewers are asked to select the class they want. However, this strategy is static, wasteful and will not scale too long.

Therefore it is necessary to streamline again the video server and streaming infrastructure in the face of this new reality. A new generation of dynamic video transcoding scheme has to be integrated into the video communication infrastructure in the coming era.

1.2 Object-based perceptual transcoding – low bit-rate but still good QoS

Transcoding is an important step to adapt the video to user requirements. The major effort is devoted to a bandwidth reduction, especially for new terminal types (PDAs, HCCs, smart phones, etc). There are various ways to classify transcoding. Vetro et al [2] classify transcoding into bit-rate conversion or scaling, resolution conversion and syntactic conversion. The first type copes with bandwidth limitation. The second type is used for device limitation, as well as for bandwidth limitation. The third type deals with syntactic conversion for protocol layer.

The current video transcoding techniques are principally based on re-quantization [3-9]. Unfortunately, re-quantization alone does not have enough capability to down scale a video to the very low bit-rate required by the emerging internet scenarios, while still be able to provide acceptable quality.

Psychological research on human vision has revealed that different regions of a video frame contribute differently to the overall perceptual quality. Human eyes are more sensitive to details of interested moving objects, while still background quality does not affect perception much. Research has shown that object based encoding can be an important tool for creating perceptually pleasant video at lower rates [10-13]. However, the conversion of a regular video to an object-based stream remains computationally very challenging. Some research on object-based video encoding has been conducted, but little or no research has been conducted on real-time object-based video transcoding.

1.3 Related works in object detection and tracking

Object detection in videos involves verifying the presence of an object in video frames - a sequence of images taken at closely spaced time intervals, and possibly locating it precisely for recognition. Object tracking monitors an object's spatial and temporal changes during a video sequence, including its presence, position, size, shape, etc. This is done through solving the temporal correspondence problem - the problem of matching the target region in successive frames. These two processes are closely related because tracking usually starts with detecting objects, while detecting an object repeatedly in subsequent frames is often necessary to help and verify tracking.

Object detection and tracking have a wide variety of applications in computer vision, such as video compression, video surveillance, vision-based control, human-computer interfaces, medical imaging, augmented reality, and robotics. Additionally, it provides input to higher level vision tasks, such as 3D reconstruction and 3D representation. It also plays an important role in video database, such as content-based indexing and retrieval.

Although it has been studied for dozen of years, object detection and tracking remain an open research problem. A robust, accurate and high performance approach is still needed.

A large number of approaches have been proposed in the literatures. All of those efforts focus on several different research areas, each of which deals with one aspect of the object detection and tracking problems, or a specific scenario. Most of them use multiple techniques and there are combinations and intersections among different methods. The diversity of techniques makes it difficult to have a uniform classification of existing approaches. However, these approaches can be classified into three groups: feature-based, template-based and motion-based.

1.3.1 Feature-based object detection

In feature-based object detection, standardization of image features and registration (alignment) of reference points are important. The images may need to be transformed to another space for handling changes in illumination, size and orientation. One or more features are extracted and the objects of interest are modeled in terms of these features. Object detection and recognition then can be transformed into a graph-matching problem.

1.3.1.1 Shape-based approaches

Shape-based object detection is one of the hardest problems to tackle due to the difficulty of segmenting objects of interest in the images. In order to detect and determine the border of an object, an image may need to be preprocessed. The preprocessing algorithm or filter depends on the application. Different object types, such as persons, flowers, and airplanes may require different algorithms. For more complex scenes, noise removal and transformations invariant to scale and rotation may be needed. Once an object is detected and located, its boundary can be found using edge detection and boundary-following algorithms. The detection and shape characterization of the objects becomes more difficult for complex scenes where there are many objects with occlusions and shading [14, 15].

1.3.1.2 Color-based approaches

Unlike many other image features (e.g. shape), color is relatively constant under viewpoint changes, and it is easy to be acquired. Although color is not always appropriate as the sole mean of detecting and tracking objects, the low computational cost of the algorithms proposed makes color a desirable feature to exploit when appropriate [16]. Grove et al [17] developed an algorithm to detect and track vehicles or pedestrians in real-time using color histogram based technique. They created a Gaussian Mixture Model to describe the color distribution within the sequence of images and to segment the image into background and objects. Object occlusion was handled using an occlusion buffer. Fieguth [18] achieved tracking multiple faces in real time at full frame size and rate using color cues. This simple tracking method is based on tracking regions of similar normalized color from frame to frame. These regions are defined within the extent of the object to be tracked with fixed size and relative positions. Each region is characterized by a color vector computed by sub-sampling the pixels within the region, which represents the averaged color of pixels within this region. They even achieved some degree of robustness to occlusion by explicitly modeling the occlusion process.

1.3.2 Template-based object detection

If a template describing a specific object is available, object detection becomes a process of matching features between the template and the image sequence under analysis. Object detection with an exact match is generally computationally expensive and the quality of matching depends on the details and the degree of precision provided by the object template. There are two types of object template matching, fixed and deformable template matching.

1.3.2.1 Fixed template matching

Fixed templates are useful when object shapes do not change with respect to the viewing angle of the camera. Two major techniques have been used in fixed template matching.

1.3.2.1.1 Image subtraction

In this technique, the template position is determined from minimizing the distance function between the template and various positions in the image. Although image subtraction techniques require less computation time than the correlation techniques described below, they perform well in restricted environments where imaging conditions, such as image intensity and viewing angles between the template and images containing this template are the same.

1.3.2.1.2 Correlation

Matching by correlation utilizes the position of the normalized cross-correlation peak between a template and an image to locate the best match. This technique is generally immune to noise and illumination effects in the images, but suffers from high computational complexity caused by summations over the entire template. Point correlation can reduce the computational complexity to a small set of carefully chosen points for the summations [19].

1.3.2.2 Deformable template matching

Deformable template matching approaches are more suitable for cases where objects vary due to rigid and non-rigid deformations. These variations can be caused by either the deformation of the object per se or just by different object pose relative to the camera. Because of the deformable nature of objects in most video, deformable models are more appealing in tracking tasks.

In this approach, a template is represented as a bitmap describing the characteristic contour/edges of an object shape. A probabilistic transformation on the prototype contour is applied to deform the template to fit salient edges in the input image. An objective function with transformation parameters that alter the shape of the template is formulated reflecting the cost of such transformations. The objective function is minimized by iteratively updating the transformation parameters to best match the object [20]. The most important application of deformable template matching techniques is motion detection in video, which we will review in the following section [21, 22].

1.3.3 Motion detection

Detecting moving objects, or motion detection, obviously has very important significance in video object detection and tracking. A large proportion of the research efforts of object detection and tracking focused on this problem in last decade. The motion detection algorithms proposed can be classified into the following groups.

1.3.3.1 Threshold technique over the interframe difference

These approaches [23] rely on the detection of temporal changes either at pixel or block level. The difference map is usually binarized using a predefined threshold value to obtain the motion/no-motion classification.

1.3.3.2 Statistical tests constrained to pixel-wise independent decisions

These tests assume intrinsically that the detection of temporal changes is equivalent to the motion detection [24]. However, this assumption is valid when either a large displacement appears or the object projections are sufficiently textured, but fails in the case of moving objects that preserve uniform regions. To avoid this limitation, temporal change detection masks and filters have also been considered. The use of these masks improves the efficiency of the change detection algorithms, especially in the case where some a priori knowledge about the size of the moving objects is available, since it can be used to determine the type and the size of the masks. On the other hand, these masks have limited applicability because they cannot provide an invariant change detection model (with respect to size, illumination) and cannot be used without an a priori context-based knowledge.

1.3.3.3 Global energy frameworks

The motion detection problem is formulated to minimize a global objective function and is usually performed using stochastic (Mean-field, Simulated Annealing) or deterministic relaxation algorithms (Iterated Conditional Modes, Highest Confidence First). In that direction, the spatial Markov Random Fields [25] have been widely used and motion detection has been considered as a statistical estimation problem. Although this estimation is a very powerful, usually it is very time consuming.

1.3.4 Object tracking using motion information

Motion detection provides useful information for object tracking. Tracking requires extra segmentation of the corresponding motion parameters. Existing approaches can be classified into two categories: motion-based and model-based approaches [26]. Motion-based approaches rely on robust methods for grouping visual motion consistencies over time. These methods are relatively fast but have considerable difficulties in dealing with non-rigid movements and objects. Model-based approaches also explore the usage of high-level semantics and knowledge of the objects. These methods are more reliable compared to the motion-based ones, but they suffer from high computational costs for complex models due to the need for coping with scaling, translation, rotation, and deformation of the objects.

Tracking is performed through analyzing geometrical or region-based properties of the tracked object. Depending on the information sources, existing approaches can be classified into boundary-based and region-based approaches.

1.3.4.1 Boundary-based approaches

Also referred to as edge-based, these types of approaches rely on the information provided by the object boundaries. It has been widely adopted in object tracking because the boundary-based features (edges) provide reliable information independent of the motion type, or object shape. Usually, the boundary-based tracking algorithms employ active contour models, like snakes [27] and geodesic active contours. These models are energy-based or geometric-based minimization approaches that evolve an initial curve under the influence of external potentials, while it is being constrained by internal energies.

1.3.4.1.1 Snakes

Snakes are deformable active contours used for boundary tracking that was originally introduced by Terzopoulos et al [28]. Snakes move under the influence of image-intensity forces subjected to certain internal deformation constraints. In segmentation and boundary tracking problems, these forces relate to the gradient of image intensity and the positions of image features. One advantage of the force-driven snake model is that it can easily incorporate the dynamics derived from time-varying images. The snakes are usually parameterized and the solution space is constrained to have a predefined shape. So these methods require an accurate initialization step since the initial contour converges iteratively toward the solution of a partial differential equation [29-31].

Considerable work has been done to overcome the numerical problems associated with the solution of the equations of motion and to improve robustness to image clutter and occlusions. Curwen et al [32] proposed a B-spline representation of active contours. Dubuisson et al [33] employed polygonal representation in vehicle tracking problems, and Metaxas et al [34] proposed a deformable superquadric model for modeling of shape and motion of 3D non-rigid objects.

1.3.4.1.2 Geodesic active contour models

These models are not parameterized and can be used to track objects that undergo non-rigid motion. In Caselles et al[35], a three-step approach is proposed which start by detecting the contours of the objects to be tracked. An estimation of the velocity vector field along the detected contours is then performed. At this step, very unstable measurements can be obtained. Following this, a partial differential equation is designed to move the contours to the boundary of the moving objects. These contours are then used as initial estimates of the contours in the next image and the process iterates. More recently, Bertalmio et al described a front propagation approach in [36] that couples two partial differential equations to deal with the problems of object tracking and sequential segmentation. Additionally, Goldenberg et al reported [37] a new, efficient numerical implementation of the geodesic active contour model which was applied to track objects in movies.

1.3.4.2 Region-based approaches

These approaches rely on information provided by the entire region, such as texture and motion-based properties using a motion estimation/segmentation technique. In this case, the estimation of the target's velocity is based on the correspondence between the associated target regions at different time instants. This operation is usually time consuming (a point-to-point correspondence is required within the whole region) and is accelerated by the use of parametric motion models that describe the target motion with a small set of parameters. The use of these models introduces the difficulty of tracking the real object boundaries in cases with non-rigid movements/objects, but increases robustness due to the fact that information provided by the whole region is exploited.

Optical flow [38, 39] is one of the widely used methods in this category. In this method, the apparent velocity and direction of every pixel in the frame have to be computed. It is an effective method but time consuming. Background motion model can be calculated using optic flow, which serves to stabilize the image of the background plane. Then, independent motion is detected as residual flow, the flow in the direction of the image gradient that is not predicted by the background plane motion. Although slightly more costly to compute, this measure has a more direct geometric significance than using background subtraction on a stabilized image. This method is very attractive in detecting and tracking objects in video with moving background or shot by a moving camera.

1.4 Real-time object tracking – challenge for in-line video transcoding

In all, object detection algorithms are generally quite computation intensive [40]. Several of the image processing based techniques have been generalized and applied for video encoding [41, 42]. Among the recent methods, Ngo et al [41] described object detection based on motion and color features using histogram analysis. This technique could process less than 2 frames in one second. Other techniques are based on even more involved image processing methods, including active contour model. There are also reports on using motion vectors for object tracking in compressed domain similar as in the transcoding scenario [43-45]. They usually achieve lower computational cost by predicting object's position using motion vectors, and maintain tracking accuracy and stability through intermittent updating and adjustment on the object image by image processing techniques. Unfortunately, most of the other techniques described have provided little if any evaluation of their method's time performance or tracking efficiency.

The faster systems for moving objects extraction system are compressed domain technique, such as those described by Wang et al [43]. This system derives texture, spatial, temporal, and directional confidence measures from incoming stream based on DCT coefficients, motion vectors and spatial/temporal continuity of motion and buffering three adjacent frames. Based on the combined confidence score, they performed a hard cut on low confident macro-blocks that are very likely to be mismatched by encoders. To identify multiple objects, they performed k-means and/or EM clustering based on spatial and motion features. They then tracked the objects by their location and motion. The eventual goal of the system was to generate the description of objects in a video including appearance time, length, velocity, and object shape characteristics. This system too achieved about 0.5 sec/frame for the CIF size on a Pentium III 450 MHz. Because of the speed, these techniques are adequate for first stage object-based encoding of video. The reported performance of several other compressed domain approaches is also similar.

Perceptual re-encoding for video communication offers a set of new challenges compared to classical video coding. Despite the availability of many algorithms for object detection, not all can be applied. In stream transcoding the object detection has to be performed extremely fast at the rate of the stream. Also, in a streaming scenario, the entire bit stream is not available at any time (thus, techniques such as histogram can not be used). Transcoding the original pixel level frame images are no longer explicitly available. Instead, the information is organized in an encoded transform space. A transcoder generally may receive some refined information (such as motion vectors). Techniques for transcoding can generally be used in first stage encoding but the reverse in not always true.

However, there are also few advantages. Although perceptual regions of importance are highly correlated to objects, the eye generally tracks areas surrounding objects beyond the exact object boundaries. We have confirmed this by direct eye-tracker analysis [46]. Thus, the problem of object detection and tracking is not identical to the problem of perceptual object tracking. Hence, it might be possible to design perceptual object detection algorithms that are much faster and thus suitable for perceptual transcoding.

1.5 Distinctions: transcoding vs. encoding

Notably most of the approaches investigated for video object detection use scene analysis. Even the compressed domain approaches mix multilevel information in filtering phases. Consequently, it becomes very difficult for these approaches to be fast enough to be applicable in live transcoding scenarios.

The transcoding scenario has some notable differences from first stage encoding. Most first stage encoding scenarios (except for live video) allow off-line processing. In contrast, in transcoding, the original frame images are no longer explicitly available. The transcoder receives an encoded video stream. Many algorithms presented in classical video tracking research have been designed from image understanding days and has no awareness about coding depth of the elements used. Many for example use lavish frame-wide pixel analysis- where pixels have to be mined from deepest coding levels. This becomes overwhelming for coded stream analysis. Most modern streams are highly coded. Consideration to the cost of coding level can greatly leverage performance. Transform domain techniques such as DCT filtering do provide some relief. However, even the DCT coefficients generally reside at two levels deeper than motion vectors in a coded stream due to differential block coding and subsequent Huffman coding.

Secondly, the object detection has to be performed extremely fast at the rate of the stream. A third distinguishing aspect is that generally the unprocessed input stream seldom contains pixels level information but rather contains highly structured coded and refined information such as motion vectors. Techniques for transcoding can generally be used in first stage encoding but the reverse in not always possible. There have been very little work that explores techniques for object based perceptual transcoder for streaming.

In this study we focus on this particular problem and describe a low computation based object tracking algorithm particularly suitable for focus region based perceptual video coded stream recompression. The algorithm relies on motion vector analysis rather than on pixel analysis to gain speed. This algorithm optimizes against coding depth of element of information accessed. To our knowledge, this is the first algorithm that tries to optimize against coding depth sensitive stream access. However, to avoid sacrificing the efficacy of the object tracking, the algorithm uses complex case analysis that can accounts for various types of object movements, camera motion, and their resultant effect of motion vector coding. The result is a fast yet highly effective object-tracking algorithm that can operate in stream rate and detect object despite camera movements, such as zoom, panning and translation. The scheme is applicable for any content-based transcoding, such as MPEG-2 to /MPEG-2 perceptual transcoding or rate transcoding or MPEG-2 to MPEG-4 transcoding.

1.6 Our approach

In our approach we have restricted the problem so that no pixel level decoding of DCT or image components are allowed. This is to conform to the constraints of the transcoding scenario. Based on this scope, we showed that two characteristics of transcoding could be utilized to reduce the computational cost dramatically.

First, we observed that region-based recompression usually does not need to perform precise boundary detection of the object boundary. Indeed, such approximation is rather more desirable. Because, in region based perceptual encoding, the human eye in effect scans both inside and outside areas near boundaries as opposed to only inside boundary. Notably, many of the previous approaches have been derived from image based object detection or scene interpretation applications, and thus perform rigorous computation in perfecting the boundaries and shapes. This cost is justifiable in application that targets complex scene understanding, but it seems that an object detection technique specifically for perceptual compression should not have to bear this cost [46].

Second, some useful information (such as motion vectors) has been made available in the stream by the original encoder's computation. In this research, we therefore particularly focused on how much precise object tracking is possible from motion vector analysis. It is important to note that motion vectors are not simply codified information about block's motion, as it may appear at first glance. Rather, motion vectors also contain highly refined color, texture and shape information. As we shall show, that through case by case analysis of this compressed domain information the algorithm can avoid a great amount of processing delay in raw image data, and yet remain very effective.

Also, this frugal algorithm does not involved indiscriminate full frame scene analysis, and performs detailed tracking analysis in the vicinity of the object boundary. For initial detection of germinating object regions, it optionally accepts logical description of the expected target video objects in terms of high-level descriptors, such as approximate initial position, size, and shape. It then automatically detects and tracks the region covered by these objects for subsequent perceptual encoding.

Our fast perceptual transcoding algorithm presented here has been implemented as an MPEG-2 based perceptual rate transcoder. We first describes the system architecture in

Chapter 2, and then present the object-tracking algorithm in Chapter 3. Chapter 4 describes how the object information is combined with MPEG-2 rate control. Finally in Chapter 5, we present the performance of this system from the real MPEG-2 transcoder experiments.

CHAPTER 2

SYSTEM MODEL

The overall architecture of the transcoding video stream (TVSS) shaper is shown in Figure 2.1. For this experiment we have implemented a Video Streamer (VS) that accepts and produces MPEG-2 ISO-13818 [1] video stream. Thus, it is fully transparent and does not require the switch of player or server while the rate transcoding is in effect. The rate transcoder has a full-logic decoder and re-encoder embedded in it. In between the BOF (Bird-of-Flock tracking unit) unit performs the perceptual object detection tracking and provides object specific information to the re-encoder. The re-encoder is capable of dynamically adjusting the incoming bit-rate to an outgoing piece-wise constant bit-rate (pCBR) [47, 48]. The rate controller is a double loop feedback mechanism, which is similar to the MPEG-2 TM-5 algorithm. Besides the pCBR operation, the system can modulate the sample density both in temporal as well as spatial dimension based on the detected objects. More details of the rate control algorithm were previously described [47]. Here we will principally describe the BOF unit that is the research work directly related to this thesis, and will briefly describe the corresponding MPEG-2 recoder.

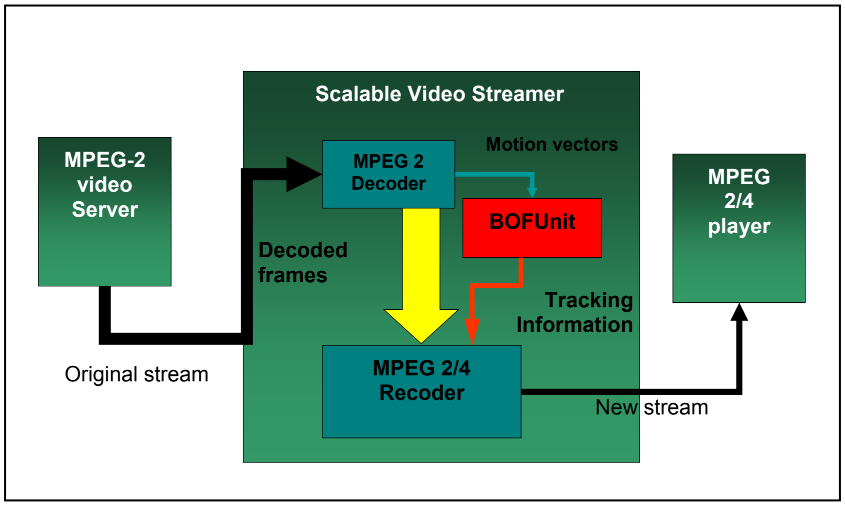

Figure 2.1 Object Aware Rate Transcoding

CHAPTER 3

PERCEPTUAL OBJECT DETECTION AND TRACKING MODEL

Since the tracking algorithm relies solely on motion vector (MV) analysis, it uses macroblocks (MBs) as the basic image unit instead of pixels. We view a video frame as a collection of MBs which are organized into a rectangle. An object is described by a subset of this MB collection which we call the representation set of this object. One MB is considered to belong to an object's representation set, if any pixels of this MB are part of this object's image. Under this description, the size of an object is actually the number of MBs belonging to this object, and the shape of the object becomes a region of 16 by 16 pixel squares that covers the object image. Given an object whose representation set is O in a frame, the purpose of tracking is to find a set of MBs W that will match O as closely as possible, i.e., the resulting set of the following equation will be minimal:

$$E = (O \cup W) - (O \cap W) \qquad \text{….(1)}$$

We call W the tracking window of this object. Of course, the size and shape description of a tracking window follows the same way as with object description using MBs. So the ultimate purpose of tracking an object in a video is to discover a series of tracking windows in the frame sequence.

The overall algorithm is a process composed of two alternative processes: prediction and update (Figure 3.1), which is repeated once for each frame. To generate W of an object in frame f, we first predict its initial MB composition from the motion properties of W' in the previous frame f'. Then, we update W through analyzing its MVs and those of surrounding MBs. Again, this updated W will be used to predict the initial tracking window in again next frame.

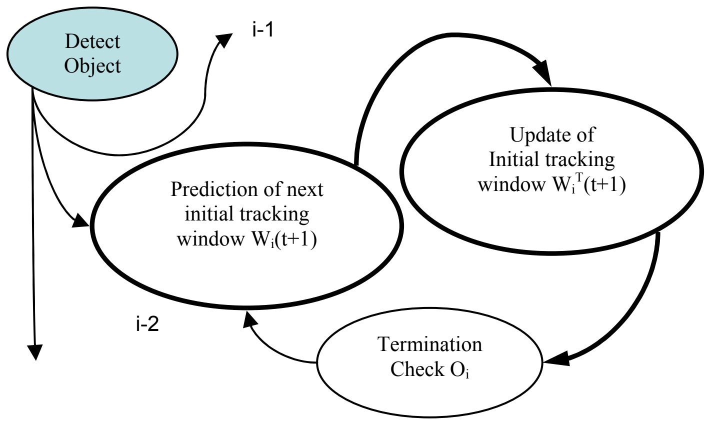
Detect Object
i-1
Update of Initial tracking window $W_i^T(t+1)$
Prediction of next initial tracking window $W_i(t+1)$
i-2
Termination Check O_i

Figure 3.1 The Motion Case Based (MCB) iterative cycle of dynamic tracking window

3.1 Prediction overview

The motion properties of a tracking window W are described by a tracking window speed vector $\overline{V}$, which has two components, $\overline{V_x}$ and $\overline{V_y}$, representing the horizontal and vertical speed of W, respectively. Once a real world object has been shot into a video, its actual movements are reflected by the position displacements of its image in consecutive frames. This position displacement can be quite different from the real object's speed in the real world. It is determined by several factors: the real object speed and direction, its distance and orientation with respect to the camera, and speed and direction of camera movement, and zooming. The $\overline{V}$ of a frame actually represents the position displacement of the tracking window W in pixels between the current frame and the previous frame. It is obtained through analyzing MVs of W. Assuming a real-time frame rate and gradual change of object and camera movement, we can predict that the position displacement of W between the current frame and next frame will be close to $\overline{V}$. So we can predict the approximate position and MB composition of W in next frame based on $\overline{V}$.

The prediction of initial W usually introduces some errors due to three reasons: (i) MB alignment, (ii) object image displacement speed change and (iii) object image changes. Although $\overline{V}$ describes position displacement in pixels per frame interval, one can only match a MB in current frame to a MB in next frame, which is aligned in the frame. So unless $\overline{V}$ is multiples of macroblock size (16 pixels in both horizontal and vertical directions for most coding scheme), some pixels of distance has to be truncated from the displacement to align MBs, which will result in a W prediction error. This error is very limited. It can only result in incorrect inclusion or exclusion of MBs along the W boundary within one MB distance. The object image displacement speed change depends on the video frame rate and the smoothness of the movements of object and camera. In most cases with real-time frame rate and reasonable camera stability, the speed change is also very limited. Object image changes are caused mainly by four factors: object size change, distance change between object and camera, object posing change, and camera zooming. The first three factors usually change gradually in most cases, but camera zooming can have greater effect on the object image.

3.2 Update overview

The update process is to remove the prediction errors as much as possible. Once the initial W is predicted, we will recalculate $\overline{V}$ based on MVs of current W. Then, we will examine the MBs in W and its surrounding region that may have been incorrectly classified into W or background, comparing their V 's distance from $\overline{V}$ with that from the background to re-determine their belongingness. The result is that some MBs may be removed from W and some may be taken into W. This update process is based on a four-layer MB model. We classify all MBs in a frame into four layers according to their relative position with the boundary of W: (i) core layer (L_{cor}), (ii) shell layer (L_{shl}), (iii) buffer layer (L_{buf}) and (iv) background layer (L_{bkg}) (Figure 3.2).

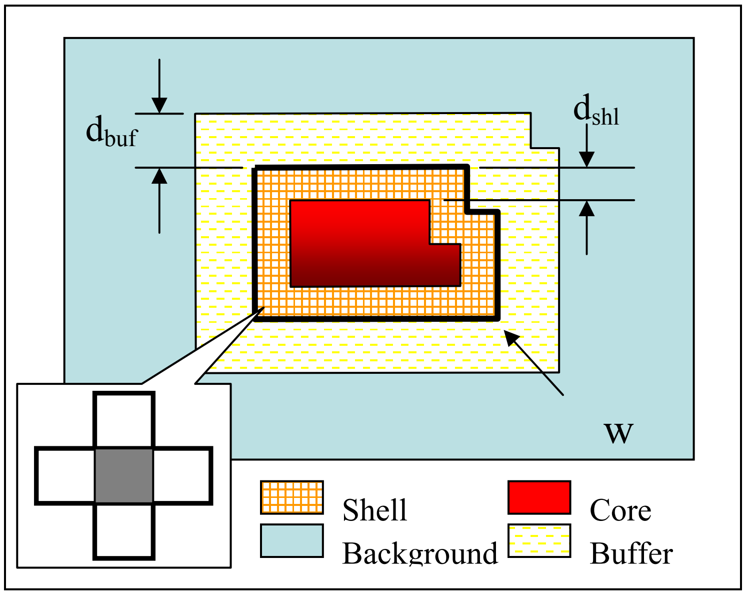

Figure 3.2 The four layers of macroblocks of the tracking model. Inset shows the neighborhood filter.

First, along the edge of W towards its center, we specify a layer of MBs of certain width in MBs (d_{shl}) to be L_{shl}. The rest MBs of W belong to L_{cor} . So W_j is actually the union of L_{cor} and L_{shl} . Then along the edge of W but away from its center, we specify a layer of MBs of certain width in MBs (d_{buf}) to be L_{buf} . All other MBs beyond L_{buf} consist of L_{bkg} . Each of these four layers plays a different role in our tracking algorithm. L_{cor} is stable during update process of W. That is we do not remove any of its MBs out of W. Only the MBs in L_{shl} will be subjected to a qualification test during update process. Failing of this test will result in being removed out of W. So, only L_{shl} is really dynamic for updating. Similarly, only MBs in L_{buf} are possible to be taken into W by meeting certain criteria during update process. MBs in L_{bkg} are also used in MV analysis, but they are never taken into W. In general, both L_{cor} and L_{bkg} stay the same for updating, dynamic changes occur only in L_{shl} and L_{buf}.

This four-layer MB model is based on the few assumptions we mentioned above, i.e., real-time frame rate, gradual change of object and camera movement, and gradual object image change. Obviously, this model can be adjusted by setting up two parameters, d_{shl} and d_{buf}, by the user. Based on our experience, $d_{shl} = d_{buf} = 1$ MB wide is good for most cases. They need to be increased to 2 or 3 MBs wide if the object image position displacement between two consecutive frames is very large and/or the object image size changes dramatically. Parameter values more than 3 are never useful because those cases are very rare.

```
1.  DETECT INITIAL TRACKING WINDOW(S) $W[0]$
        a.  Either the User specifies $W[0]$ or
        b.  Using the Automatic Germination Parameters ($G_{SCP}$, $G_{SZ}$, $G_{SPD}$, $G_{DIR}$)

2.  For each frame f iterate the process

3.  ESTIMATE $\overline{V}[f]$ FROM $W[f]$

        a.  First estimate $\overline{V}_x[f]$
        b.  $G0 = G0 + \{m_x\}$ if $|v_x| \leq 1$
        c.  $G+ = G+ + \{m_x\}$ if $v_x > 1$
        d.  $G- = G- + \{m_x\}$ if $v_x < 1$
        e.  If $|G0| < 0.8\,(|G0|+|G+|+|G-|)$ then $\overline{V}_x[f]=0$
        f.  If $|G+| > |G-|$ $\overline{V}_x[f]$ = median { $v_x$ of G+}
        g.  Else $\overline{V}_x[f]$ = median {$v_x$ of G-}

4.  Repeat the above steps 3.a-g to estimate $\overline{V}_y[f]$

5.  $W^T[f+1]$ = TRANSLATE($W[f]$, $\overline{V}[f]$).
6.  ESTIMATE THE FOUR LAYERS FOR $W'[f+1]$
        a.  Let $d_x$ is the distance from boundary of $W'[f+1]$. + for inward and – for outward.
        b.  If $d_{shl}<d_x<0$, $m_x \in L_{shl}[f+1]$
        c.  If $0<d_x<d_{buf}$, $m_x \in L_{buf}[f+1]$
        d.  $L_{cor}[f+1] = W^T[f+1] - L_{shl}[f+1]$
        e.  $L_{bkg}[f+1] = Frame[f+1] - L_{cor}[f+1] - L_{shl}[f+1] - L_{buf}[f+1]$

7.  DETECT BACKGROUND MOVEMENT

8.  If there is no background movement i.e. $|v_x| \leq 1$ and $|v_y| \leq 1$
        a.  If $|\overline{V}_x|=0$ and $|\overline{V}_y|=0$ then $W[f+1]= W^T[f+1]$.
        b.  If $|\overline{V}_x|>0$ or $|\overline{V}_y|>0$ check all $m_x \in L_{shl} \cup L_{buf}$.
                i.  If $|v_x|>1$ or $|v_y|> 1$ include $m_x$ in $W[f+1]$
                ii. Else exclude $m_x$ from $W[f+1]$.
        c.  Proceed to STEP-13.

9.  If there is background movement:
        a.  CALCULATE LOCALIZED BACKGROUND SPEED $\overline{U}[f+1]$
        b.  A = FILTER( $L_{shl}$, Pre-remove, $\overline{V}[f]$, $\overline{U}[f+1]$)
        c.  B = FILTER( $L_{buf}$, Pre-uptake, $\overline{V}[f]$, $\overline{U}[f+1]$)

10. $W[f+1]= W^T[f+1] - A + B$

11. CHECK THE EXTENT OF CHANGE IN WINDOW SIZE
        a.  $k = |B| - |A|$
        b.  if $|B| - |A| > S_{SZ}$
                i.  SORT-Rear (B, $\overline{V}$),
                ii. $W[f+1]=W[f+1]$ – SELECT (k- $S_{SZ}$, SORT-Rear (B))
        c.  if $|A| - |B| > S_{SZ}$
                i.  Sort-Front (A, $\overline{V}$)
                ii. $W[f+1]=W^T[f+1]$ + SELECT (k- $S_{SZ}$, SORT-Front (A))

12. CHECK SHAPE CONTIGUITY. Check W region.
        a.  If $m_x \in W$, but surrounded by macroblocks in W then add $m_x$ in $W[f+1]$.
        b.  If $m_x \in W$, but surrounded by macrblocks not in W then remove $m_x$ from $W[f+1]$.

13. END-OF-FRAME
14. VOD Termination Check ($T_{spd}$, $T_{dir}$, $T_{sz}$)
15. Proceed to next frame iteration in STEP-2.
```

Figure 3.3 Pseudo Code

3.3 Tracking algorithm

The system can track multiple objects simultaneously and independently. Tracking of each object is one iteration of the same algorithm but with different parameter data structure. Figure 3.3 above shows the pseudo code of this algorithm. The following is the description of the algorithm using one arbitrary object whose representation set is O and tracking window is W, as an example. We assumed a sequence of P frames only and the direction of MVs has been reversed for ease of explanation.

3.3.1 Initialization: automatic birth of an object

The algorithm starts with an initial tracking window W[0]. We allow two ways to start. The user can generate W either directly by specifying its MB set, or automatically by specifying some starting parameters. Since the analysis does not go down to pixel level, it relies on some motion features, such as speed, direction and features that can be represented by MB sets to identify an object.

The search process used to detect the birth or appearance of a perceptual object in the frame is modeled by four values called germination parameters. These are searching scope (G_{scp}), size (G_{sz}), speed (G_{spd}) and moving direction (G_{dir}). The algorithm searches automatically for such a continuous region of MBs in each frame that satisfies these parameters until W is established. G_{scp} is a rectangle region of MBs within which the algorithm will search for a potential object. The search scope not only reduces the number of MBs processed, but also helps to capture the interested object more easily by eliminating those objects with similar features outside of the searching scope. G_{sz} defines the size range of O set in MBs. G_{spd} gives the range of how many pixels an object moves in the picture from one frame to the next. G_{dir} describes object's moving direction in the picture. We defined nine values for G_{dir}: none, north, northeast, east, southeast, south, southwest, west and northwest.

The following are the default values of these parameters. The default value of G_{scp} is the entire frame. Default for size and speed is 4, 4. None is the default for direction which means that there is no preference about object's moving direction.

The algorithm searching for an object satisfying above start criteria works as follows:

1. Check each MB m_i within G_{scp} for candidate MBs which satisfies the following criteria:

    a. It does not belong to any existing tracking windows.

    b. The direction of its MV follows G_{dir} if specified.

    c. Its speed V calculated as following falls into G_{spd}.

    $$V = \sqrt{V_x^{\,2} + V_y^{\,2}}$$

    where V_x and V_y are the horizontal and vertical components of MV.

2. If the total number of candidate MBs is greater than the lower boundary of G_{sz}, do the following:

    a. Group all candidate MBs into continuous regions. The four continuous neighbor MBs of a specific MB are shown in Figure 3.2 inset.

b. Calculate the MB size of each continuous region.

c. Select the first continuous region whose MB size falls into G_{sz} as qualified region.

3. All MBs in this qualified continuous region compose the initial W.

There are two modes of user customization. User can modify the default values of the germination parameters to control the description of the perceptual object. Or the user can specify the W[0] set directly on any starting frame F[0] based on one's knowledge about the video image of the interested object in the stream. In either mode, the user approximation does not have to be accurate. User can specify the initial W using an approximate rectangle of MBs. Of course, most of the time, the O set of an object on the MB grid is not a rectangle shape. But the shape and size of W is updated automatically to be closer to O through the tracking process.

3.3.2 Tracking window prediction

3.3.2.1 Speed vector estimation

The tracking of an object is composed of two alternative processes: prediction and update. First we explain the prediction stage. If an object is found and the corresponding W is established, it calculates the speed vector $\overline{V}$ to describe its moving properties, which is used later to predict the future composition of W. This is also conducted after W is updated in each following frame.

Although these calculations rely on the analysis of MVs of W, the average or median values of all MVs can not be used. Instead, we developed the coded stream case based a slightly more complicated but much more accurate way to calculate object motion based on following observations about MVs:

1. If an object image is not moving in the scene, then the MVs of its corresponding MBs should be zero.

2. If an object is poorly textured, i.e., there are some uniform areas within the object, some MBs may have a zero MV value even if the object is actually moving.

3. Even if there is no movement in the scene, one can still has some random MBs whose MV magnitude is 1 occasionally. This might be due to the slight vibration of the camera imperceptible to human. So we treat the MV magnitude 1 the same as 0. This causes problem only when the object is really moving at the speed of 1 for a considerably long time, which is very rare.

4. Some MBs can have severely deviated MV values from the majority, especially when the object is not well textured. Part of these MBs can be recognized easily if its MVs have different direction from the majority. Since most of the MBs in a well textured object have very similar MVs that correctly reflect the movement, it is better to choose the median value as the object speed vector. Otherwise, those deviated MV values will lower the reliability of the speed vector if an average is taken.

The followings are our algorithm to calculate W's horizontal speed vector $\overline{V_x}$. The calculation of $\overline{V_y}$ follows exactly the same algorithm.

1. Classify V_x of all MBs in W into three groups: zero, positive and negative groups as follows:

zero group (G0) : $|V_x| <= 1$

positive group (G+): $V_x > 1$

negative group (G-): $V_x < 1$

2. When the number of MBs in the zero group is overwhelmingly large (a threshold factor of 0.8 was used) of the total number of MBs, set $\overline{V_x} = 0$. We assume the non-zero values are either from deviated MVs due to poor object texture, or from falsely included background MBs in W.

3. Otherwise, choose either the positive or negative group which has more MBs, then sort all V_x in that group into increasing order and set $\overline{V_x}$ to be the median value of that group.

3.3.2.2 Interframe translation

First we predict the initial W^T in the next frame f+1 from W of current frame f based on its $\overline{V}[f]$. Then we update W through analyzing its MVs and the MVs of those MBs surrounding the window.

If the speed vector of current tracking window W[f] with respect to previous tracking window W[f-1] is $\overline{V}[f]$, we will assume that this tracking window is going to move with that speed until the current frame and the window shape and size will not change significantly. So we can obtain the motion predicted window W^T[f+1] in the next frame by shifting W[f] horizontally and then vertically by respectively $\left\lfloor \frac{\overline{V_x}'}{w} \right\rfloor$ and $\left\lfloor \frac{\overline{V_y}'}{h} \right\rfloor$ MBs, where hxw is the macroblock dimension. This step deals with the major movement and defines the W^T[f+1] set that will be used for analysis in the following update phase.

3.3.3 Tracking window update

The purpose of this phase is the case-based update of the deviations of the prediction $W^T[f+1]$ from the O set and obtain the corrected window $W[f+1]$. These adjustments usually happen around the edge of W^T if it is large enough.

1. Group all the MBs in the frame into four layers as described in the tracking model section, core layer (L_{cor}), shell layer (L_{shl}), buffer layer (L_{buf}) and background layer (L_{bkg}).

2. Detect movement in background. Examine the MVs of the inner most layer of MBs belonging to L_{bkg}. Check for MBs whose MV component V_x and V_y satisfies the following condition:

   a. $|V_x| > 1$ or $|V_y| > 1$

   b. If such a MB is found, we will consider that there are some movements in the background. Otherwise, we consider the background is still.

   c. A still background occurs when the camera is fixed and there are no moving objects in the scene surrounding the tracked object, which is a fairly common video scene. Identifying of these cases can reduce its MV analysis cost and increase its tracking accuracy. A moving camera or a moving object surrounding the tracked object usually results in a moving background in which case a much more complicated motion case analysis would be needed.

3. If there is no movement in the background, we will need to discriminate two cases:

a. If $|\overline{V_x}| = 0$ and $|\overline{V_y}| = 0$, which means the tracked object is not moving either, no adjustment to the initial tracking window will be needed.

b. If $|\overline{V_x}| > 0$ or $|\overline{V_y}| > 0$, we will examine the MVs of all MBs in L$_{shl}$ and L$_{buf}$. Take the MB into W set if its MV satisfies $|V_x| > 1$ or $|V_y| > 1$. Otherwise, move the MB out of W set.

4. If there is any movement in the background, more complicated motion case analysis is needed to modify the tracking window. As discussed in the tracking model section, only MBs in L$_{shl}$ can be removed from W, and only MBs in L$_{buf}$ can be taken into W. This is decided by comparing the distance of the MV of observed MB to $\overline{V}$, and its distance to the MVs of its surrounding background MBs.

a. Calculate a 'localized' background speed vector $\overline{U}\,(\overline{U_x}\,,\ \overline{U_y}\,)$ for each MB belonging to either L$_{shl}$ or L$_{buf}$. First, starting from the MB under consideration, examine all MBs which are one MB away from it, then all MBs two MBs away from it, etc. Identify those MBs that belong to L$_{bkg}$ as candidate local background MBs, until the total number of these MBs are more than 5 after a around. Then apply the algorithm of section 1.3 on this collection of local background MBs to determine $\overline{U}$.

b. Pre-removal of MBs from L$_{shl}$. If a MB satisfies at least one of the following conditions, it will be pre-removed from L$_{shl}$ and from W.

i) $\overline{V_x} \times V_x < 0$ or $\overline{V_y} \times V_y < 0$

ii) if $\overline{V_x} > \overline{V_y}$, then $|\overline{V_x} - V_x| > |\overline{U_x} - V_x|$

iii) if $\overline{V_x} \le \overline{V_y}$, then $|\overline{V_y} - V_y| > |\overline{U_y} - V_y|$

c. Pre-uptake of MBs from L_{buf} into W. If a MB satisfies all of the following conditions, it will be pre-uptake from L_{buf} into the W.

i) $\overline{V_x} \times V_x \geq 0$ and $\overline{V_y} \times V_y \geq 0$

ii) if $\overline{V_x} > \overline{V_y}$, then $|\overline{V_x} - V_x| \leq |\overline{U_x} - V_x|$

iii) if $\overline{V_x} \leq \overline{V_y}$, then $|\overline{V_y} - V_y| \leq |\overline{U_y} - V_y|$

d. If the net size change of W after pre-removal and pre-uptake is less than a size change sensitivity parameter (S_{SZ}) specified by the user, accept the modification made in step b and c. Otherwise, do either step e or f. S_{SZ} is a percentage of W with a default value 20%. This guards against drastic change to W caused by false MVs.

e. If net size increase exceeded S_{SZ}, then remove some of the MBs pre-uptaken in step c, starting from the rear side of the moving tracking window, until the size increase is within the S_{SZ} limit.

f. If net size decrease exceeded S_{SZ}, then recover some of the MBs pre-removed in step b, starting from the front side of the moving tracking window, until the size decrease is within the S_{SZ} limit.

3.3.4 Shape integrity adjustment

Check W region, if any MBs not belonging to W but are surrounded by MBs belonging to W is found, then take them into W. Remove any isolated MBs from W. The purpose is to make the tracking window into one continuous and solid region of MBs.

3.3.5 Tracking termination

Tracking of an object can be terminated unconditionally or conditionally.

3.3.5.1 Unconditional tracking termination

This is signaled by the user through the control interface. The tracking window is dissolved and tracking of this object is stopped.

3.3.5.2 Automatic death of object detection

The conditions under which tracking of an object should be terminated is also defined using a set of termination parameters. The parameters set we implemented is very similar to those tracking window generation parameters. These include speed (T_{spd}), direction (T_{dir}) and size (T_{sz}). Their meanings are same as defined in the automatic tracking window generation. After tracking window W is updated in each frame, it is checked against the following conditions:

1. The moving speed of W calculated as $\sqrt{\overline{V_x}^2 + \overline{V_y}^2}$ falls into the range of T_{spd}.

2. The direction of speed vector $\overline{V}$ follows T_{dir}.

3. The size of W set falls into the range of T_{sz}.

If any of the above criteria is satisfied, the W will be dissolved and the tracking of this object is stopped. These parameters can be set by users as well.

CHAPTER 4

RATE CONTROL

4.1 Transcoder rate control mechanism

Once the objects are detected, the active sets are fed into the following spatial rate control mechanism of the re-encoder. MPEG-2 TM-5 has already defined a quantity called activity factor for taking into account of perceptual significance of the macroblocks. Instead of reinventing, we refine this handle to perform the spatial varying coding. The proposed mechanism is also a double-loop feedback control mechanism where the output bit-rate is continually sensed to determine overall piecewise constant rate, with appropriate accounting for variations in frame/picture type like TM-5. A second internal feedback loop further tracks the efficacy of key conversion factors/constants for additional stability. Here, the perceptual content and activity in a particular picture area dictates the inherent amount of bits that may be required to encode it. Also the bit requirement per macro-block is dependant on the picture type (I, B or P) as well other subjective factors. Like TM-5 the bit-rate is controlled by the requantization-step of the DCT coefficients. The quantized output for intra-and non-intra frames are respectively given by:

$$y = \frac{\mathrm{f(x,\ quant\_step\ \ )} + .75 \times \mathrm{mquant}}{2 \times \mathrm{mquant}} \qquad \ldots\ldots(2)$$

$$y = \frac{16 \times f(x, quant\_step)}{mquant}$$

Here x is the DCT coefficient, y=f(x, quant_step) is determined from ISO/IEC 13818-2 tables [ISO96]. As mquant increases, the effective quantization steps become larger, more information is lost, the encoding requires lower bits, and the quality of the picture degrades, and vice verse.

4.2 Quantization factor determination

To account for few of these factors, in the topmost level the value of mquant for each macroblock is calculated as a product of two primary factors (a) the buffer fullness and (b) the macroblock activity. The mquant for the jth frame is computed as a product of two parameters:.

$$mquant_j = Q_j \times N\_act_j \quad \ldots.(3)$$

The final value of $mquant_j$ is coded either in the slice or in the macroblock. Qj is determined based on the frame and macroblock type, and uses standard TM-5 model header [ISO96]. The part that is relevant for this experiment is the N_act_j. The motivation behind the original TM5 activity factor is that human visual perception is less sensitive to distortions in noisier textured areas and more sensitive to distortion in image areas with uniform texture. We used enhanced region based activity assignment algorithm for estimation of N_actj, based on the object tracking results. It allows spatial distribution of the bits to be controlled for a given allocation of frame bits.

4.3 Object based activity factor determination

The relative bit allocation factor of the various objects and backgrounds detected is specified as an object resolution parameter α_i for each macroblock i. We maintain total per frame bit-allocation fixed. Thus:

$$\sum_i \log N\_act^{\mathrm{mod}\,ified}{}_i \approx \sum_i \log N\_act_i \qquad \ldots\ldots(4)$$

The macorblocks in the boundary set is assigned a macroblock resolution factor α_i (-8,0,+8). Based on the overall distribution of the α_i over a frame the N_activity(i) of a blocks is then calculated as:

$$N\_act^{\mathrm{mod}\,ified}{}_i = N\_act_i \cdot \alpha_i \cdot 2^{\sum \frac{-\log \alpha_i}{n}} \qquad \ldots\ldots(5)$$

The log normalized value ensures that the bit distribution over the frame remains close to the original allocation of the TM-5 model.

CHAPTER 5

EXPERIMENTS

We have performed extensive evaluation of the algorithm under various scenarios. In this section we provide some results about the algorithms characteristics. We have evaluated both the time efficiency and tracking effectiveness of the algorithm. For time efficiency we also provide the analytical complexity of the algorithm for better appreciation of its scalability.

5.1 Tracking effectiveness

We define two criteria for the evaluation of the motion-tracking algorithm. That is to see if the tracking window W covers the object representation set O completely and nothing else.

Coverage: This is the zonal fraction of the actual visual object successfully covered by the active set. Object coverage factor is the ratio of O covered by W.

Best case obviously should be 100%.

$$\operatorname{cov}erage = \frac{O \cap W}{O} \qquad \text{.....(6)}$$

Mis-coverage: This is the factor of W that is not covering O but the background compared to the actual window size. The ideal situation of course is 0%.

$$mis - coverage = \frac{W - (O \cap W)}{W} \qquad \ldots..(7)$$

These two criteria are illustrated in Figure 5.1.

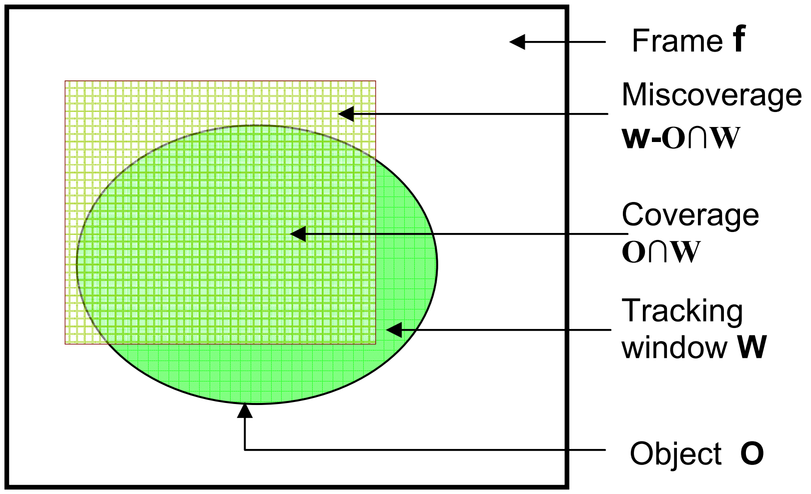

Figure 5.1 Calculation of object coverage and tracking window mis-coverage.

There is, however, no automatic method to determine the reference actual object O. Therefore the following manual method was used. In the output stream the transcoder drew a visible grid to indicate the MB boundaries and the tracking window boundaries on every frame. The algorithm also provided the size of the tracking windows. Then we played back these new MPEG-2 videos on regular media player. Once every second we paused and evaluated the object size in MBs manually and observe their overlap with the tracking window. Then we calculated coverage and mis-coverage.

Perceptual encoding is highly dependent on the video content. Therefore, we avoided providing any ‘average’ performance. Rather we have used case videos each carefully selected to offer a special challenge to the system interms of number of objects, camera motion, etc. Here we present analysis of eight representative cases. These sequences represent several groups with various types of objects and motion combinations, which can impact the effectiveness of our object tracking algorithms. First Table 5.1 provides the format parameters of these sequences. Column two of Table 5.2 describes their motion properties. Figure 5.2 and Figure 5.3 plot the coverage and miscoverge factors as a function of frame sequences for all of them.

The first four described in Figure 5.2 (a)-(d) were shot with a fixed camera. Under this condition, only absolute object movements can result in non-zero MVs. As visible in the results the MCB algorithm can track objects with high effectiveness, regardless whether they are well textured or not. The coverage remains more than 90%, and the miscoverage is 2-8%. Two-tractors and Walking_people have multiple objects moving in the scene. Two objects can be discriminated even if their speeds are close such as in the Two_tractors sequence. It can even track objects with deformable shapes like people.

The last four sequences shown in Figure 5.3 (a)-(e) were shot with a moving camera, where the irregular MVs in the background complicated the tracking a great deal. In these situations, the tracking effectiveness relies heavily on the object’s texture and the regularity of camera movements. For example, the object in tractor_with_moving_camera sequence is well textured and the camera moved smoothly towards one direction for a short time, the tracking is as good as in the first four videos.

The tracking of a plane in the plane video was also successful because the uniform blue sky background cancelled out the camera movements. But tracking was interrupted more frequently in the last two videos which were taken with much more irregular camera movements. Between these two, the tracking of the mower was relatively more stable than the tracking of van in the shaking_camera video because of its good texture. Despite the interruptions the system was able to get back to tracking. Overall in stable state the coverage was also 80-90% and miscoverage was 5-10%.

Table 5.1 Sample Video Sequences

Name	bit-rate (mbps)	resolution	Duration (frames)
Toycar	10	704x480	300
Mycar_in_parkinglot	5	704x480	663
Two_tractors	5	704x480	472
Walking_people	4	704x480	153
Tractor_with_moving_camer	4	704x480	483
Plane	4	352x240	300
Mower	5	352x240	2850
Shaking_camera	5	704x480	1350

Table 5.2 Video Content Description and Their Tracking

video name	Content description	Tracking result
Toycar	Fixed camera, still background, one object tracked, well-textured, fast-moving, heavy shadow	Tracked through entire sequence
Mycar_in _parkinglot	Fixed camera, some movements in background, one object tracked, poor-textured, slow-moving	Tracked through entire sequence
Two_tractors	Fixed camera, some movements in background, two objects tracked, well-textured, slow-moving, partial exclusion	Tracked through entire sequence
Walking_people	Fixed camera, still background, three objects tracked, well-textured, deformable shape, slow-moving, illumination change	Tracked through entire sequence
Tractor_with_ moving_camera	Fast smooth camera movement, well-textured background, one object tracked, well-textured, slow-moving, partial exclusion	Tracked through entire sequence
Plane	Smooth camera movement, poor-textured background, one object tracked, poor-textured, irregular-moving	Tracked until object is too small
Mower	Smooth camera movement, well-textured background, one object tracked, well-textured, slow-moving, exclusion	Lost tracking only at complete or near complete exclusion
Shaking_camera	Irregular camera movement, poor-textured background, one object tracked, poor-textured, fast-moving	Lost tracking multiple times due to sudden camera pulsation

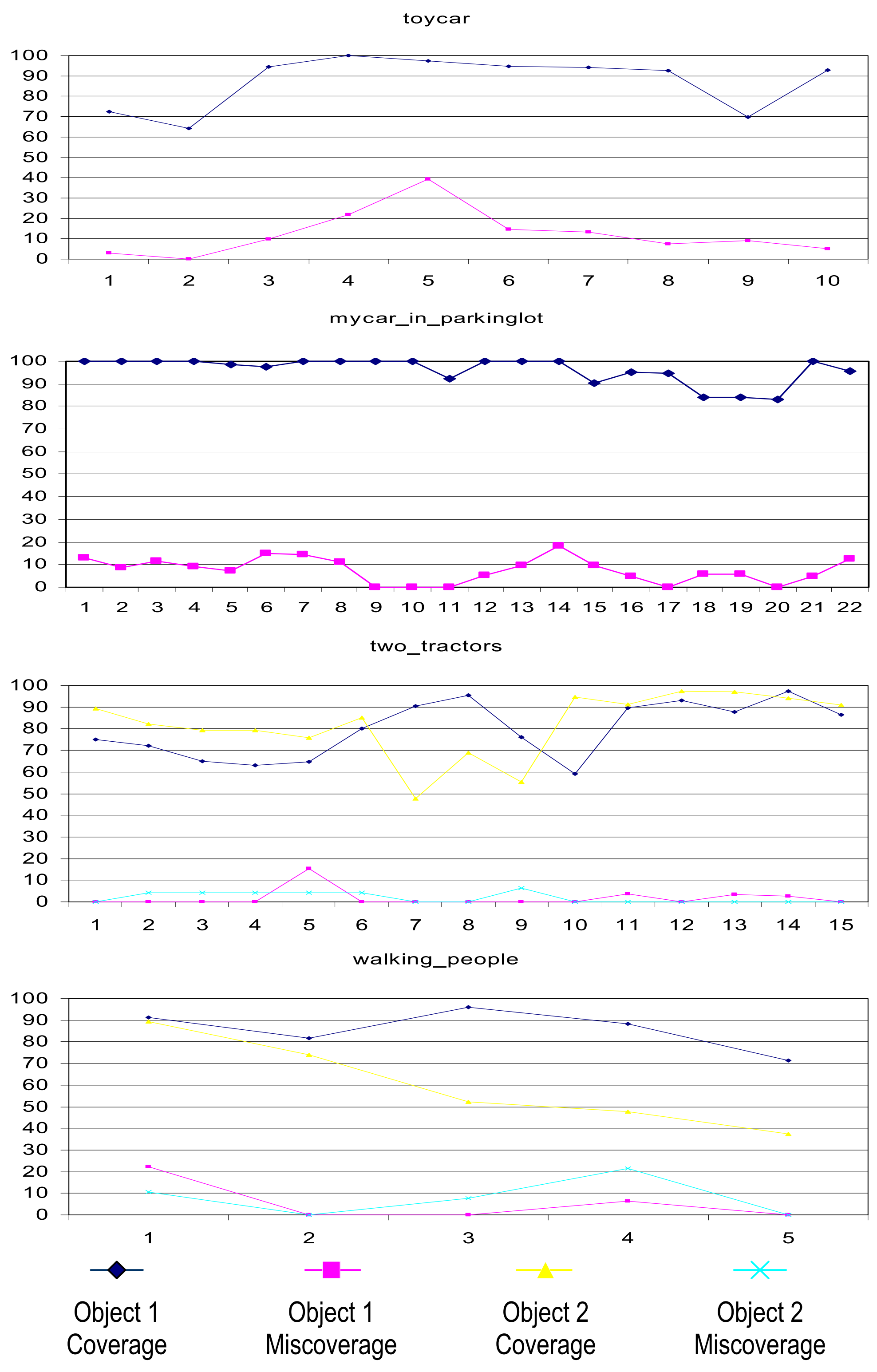

Figure 5.2 Results on tracking effectiveness evaluation on sample videos without camera motion.

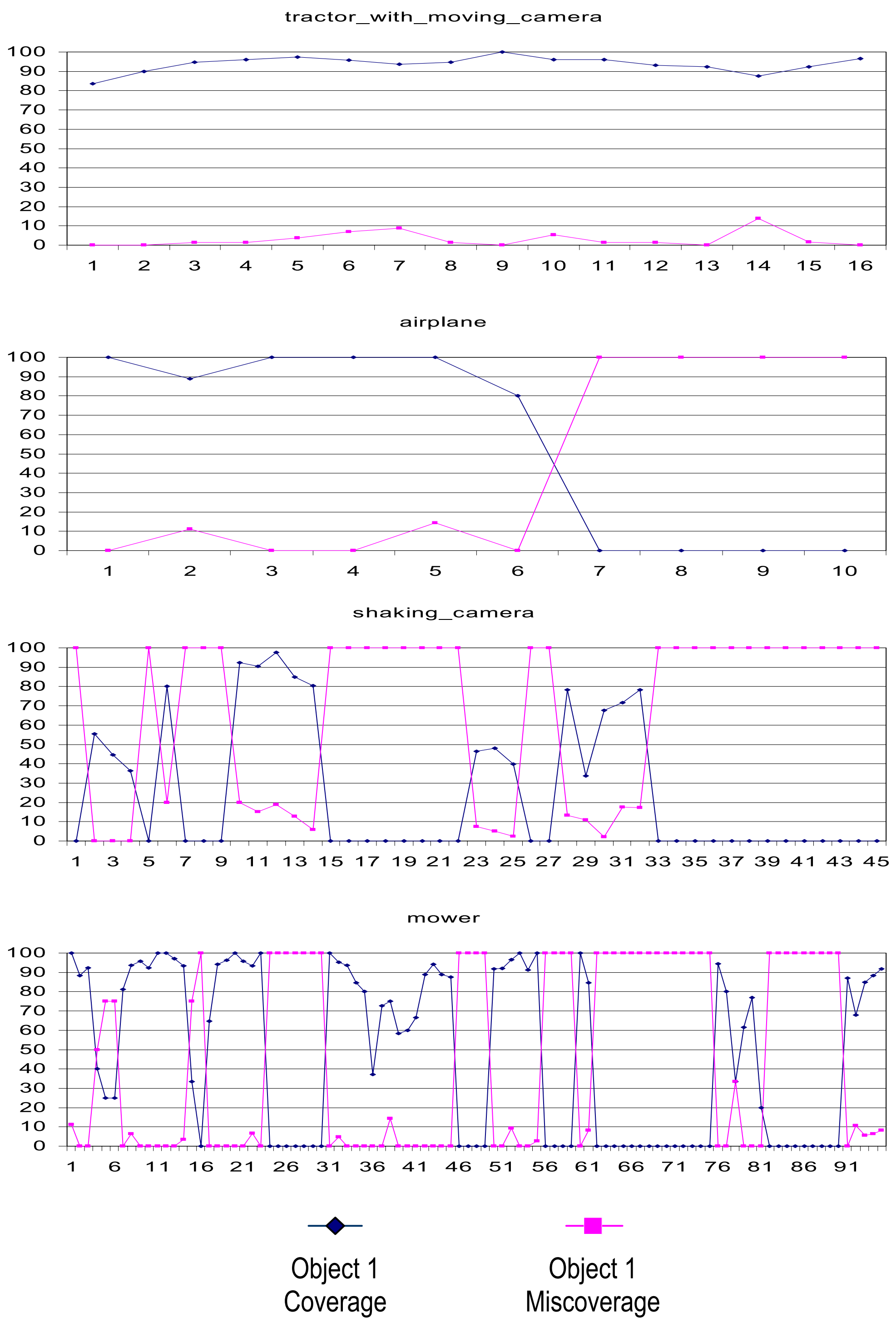

Figure 5.3 Results on tracking effectiveness evaluation on sample videos with camera motion.

5.2 Object-based quality transcoding

The purpose of object tracking in video transcoding is to provide objects' position information to the transcoder so that the region covering the objects and background can be processed separately in transcoding. These processing can be versatile limited only by the video format and the functions available in the transcoder. For our case, we have used rate/quality transcoding as per region based rate allocation technique given in chapter 4. Here we provide a sample case of rate/quality improvement by the technique for the Two_tractors video sequence as processed by our object-based quality MPEG-2 transcoder. The original MPEG-2 stream has a bit-rate of 5mbps and the file size is 9844KB. It is downscaled to 500kbps by our transcoder using both object-based and non-object-based quality transcoding, resulted in a file size of 1001KB and 1012KB, respectively. We present the result in all three forms.

First, Figure 5.4 provides the bit distribution of the transcoder stream. It shows the bit distribution per macroblock separately for the object and background regions for each frame. For comparison it also shows the corresponding bit-distributions when object unaware regular transcoding was used. The I frames has usual larger bit allocation. However, the non-object based transcoding indiscriminately reduces bits everywhere. Indeed it is interesting to note that unaware scheme in places took away more bits from regions of object resulting in severe perceptual quality loss. In contrast the object-aware scheme has been able to allocate more bits in the object area. However, it should be noticed that although relatively large amount of bits can be provided into the object region, the overall bit at the background is slightly smaller than that of unaware scheme.

This has been possible for typical small size of objects of perceptual significance. The resulting drastic improvement of SNR quality is shown in Figure 5.5. Figure 5.5 plots the average macroblock SNR quality for these same four cases. As can be seen the object region's SNR drastically improved in the coding. This diagram also shows how a 'unaware' transcoding scheme can severely damage video quality by inadvertently reversing the perceptual quality of background and object by classically activity analysis. We also provided two screen shots from two transcoded streams in Figure 5.6 for visual demonstration of the result of perceptual encoding. Obviously, the one obtained from object-based transcoding is more perceptual pleasant than the one from non-object-based transcoding. These improvements are much more perceptually pronounced in the actual movie.

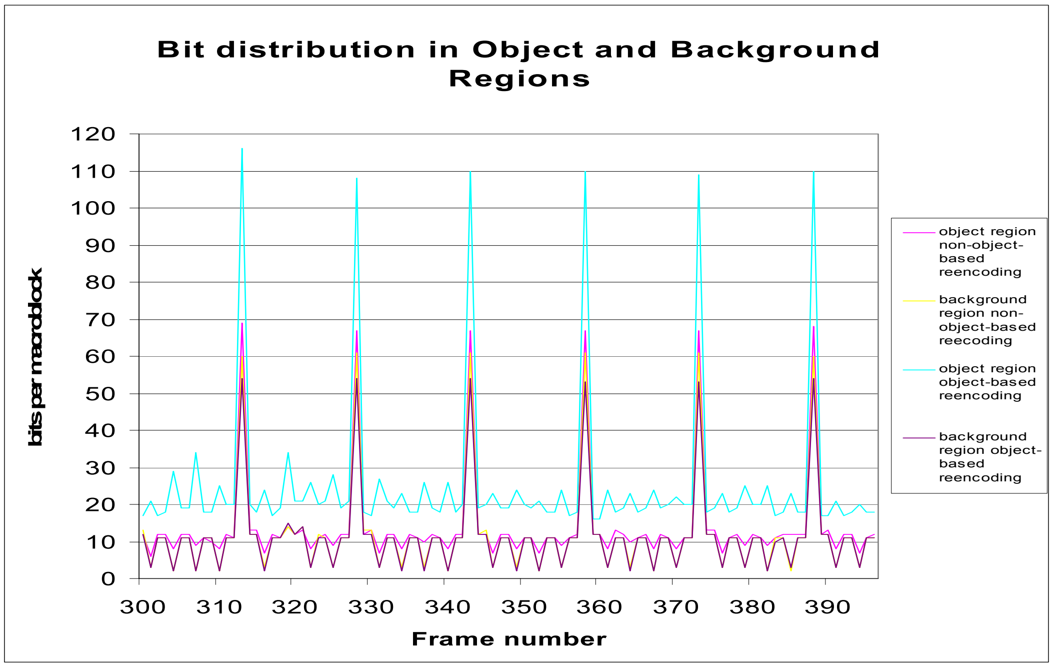

Figure 5.4 Bit distribution comparison between object and background region MBs after object-based vs non-object-based quality transcoding.

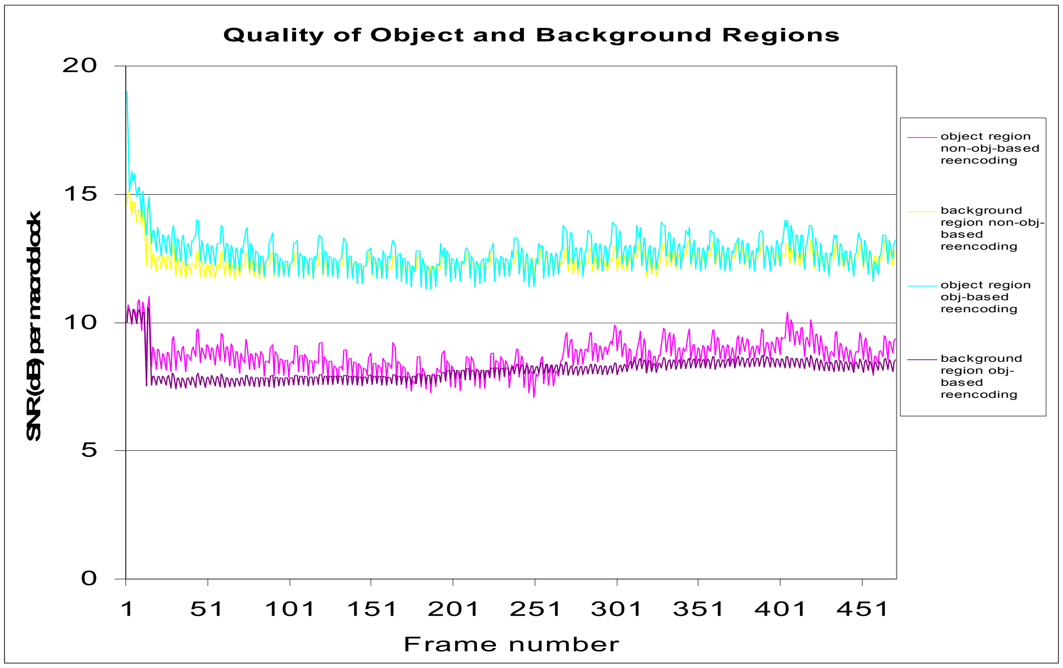

Figure 5.5 Comparison of SNR between object and background regions after object-based vs non-object-based quality transcoding.

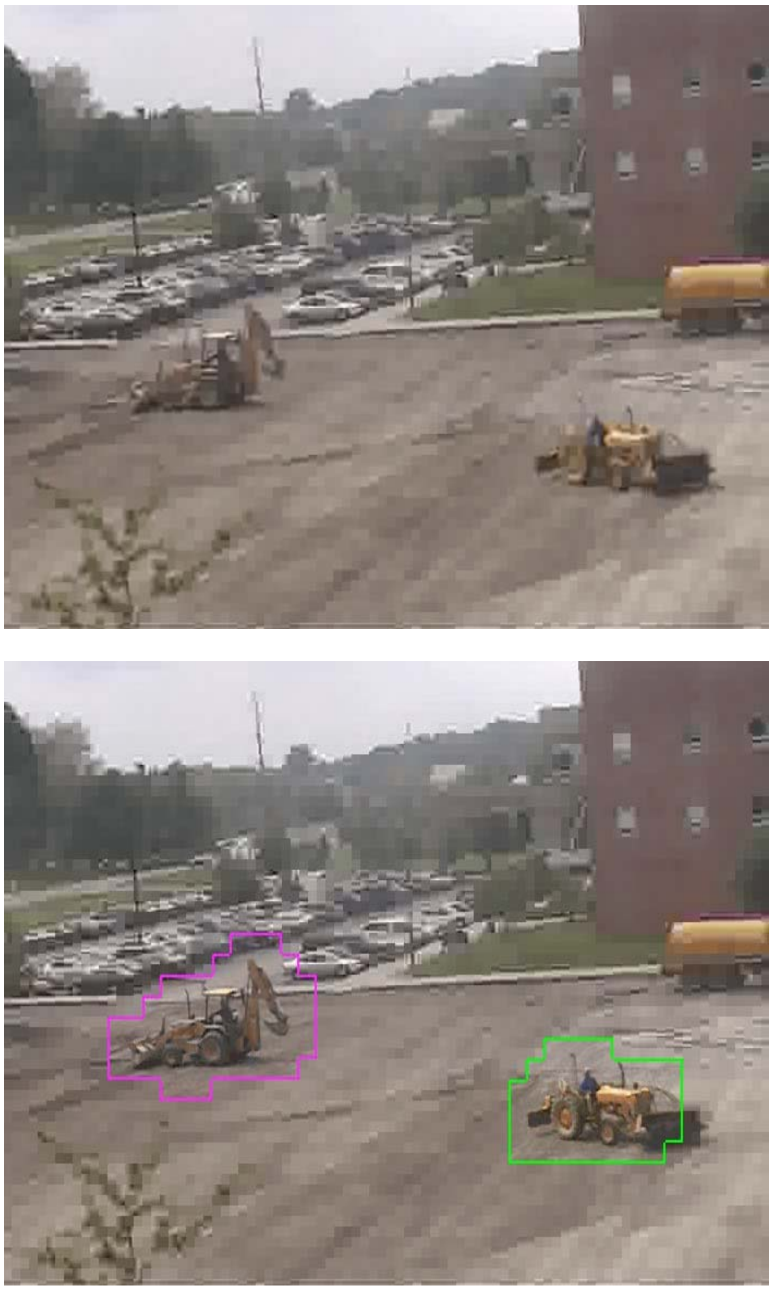

Figure 5.6 Perceptual quality comparison between non-object-based (upper) and object-based quality transcoding. In object-based transcoding, the two objects indicated by the color window have higher quality than in the non-object-based transcoding.

5.3 Algorithm complexity

The algorithm can track multiple objects in parallel. Each object repeats the major part of the algorithm once. Since the main task of the algorithm is to analyze MVs, the computing cost certainly is related with the number of MBs to be processed. Here two numbers matter, the number of MBs in a frame or the frame size, the number of MBs in each object or the object size. We use S_{frm} to indicate frame and s_i. to indicated the side of object i both in MBs. If the number of objects simultaneously under tracking at any point is N_{obj}, the algorithm complexity grows linearly with N_{obj} The overall complexity of our tracking algorithm is as following:

$$[C_1 \cdot S_{frm} + \sum^{N_{obj}} (C_2 \cdot s_i + C_3 \cdot s_i \cdot \log s_i)] \qquad \text{.....(8)}$$

C_1 and C_3 are small and less than 10. C_2 is at the order of tens. The only non-linear item s_i lg(s_i) comes from the quick sorting algorithm which used to sort MVs of an object's MBs. However, the complexity of the algorithm does not grow indeterminately with the increase of the number of objects or the size of the objects. In either case the product $s_{i.}$ .N $_{obj}$ is limited by S_{frm} which is only 8100 even for HDTV format, lg(s_i) actually is no more than a small constant around 10.

$$S_{frm}(C_1 + C_2 + C_3 \cdot \log S_{frm}) = O(S_{frm}.\log S_{frm}) \qquad \text{.....(9)}$$

For example, when the sizes of objects increase, although the cost of processing individual object increases but the number of objects that could exist simultaneously N_{obj} decreases towards 1. So in general, the complexity of the algorithm is linear with the object size for most scenes, but in the limit grows linearly along with the frame size.

5.4 Observed speed

We ran our object tracking system on a single AMD 1.3 GHz processor Linux machine with several sample MPEG-2 videos and collect data about CPU time spent on each functions using the gprof utility. The processing time for tracking after MVs extraction for each frame and video sequence information are given in Table 5.3.

These results show that the time used for tracking one object is only about 0.47 milliseconds per frame for 704x480 video resolution. With normal frame rate of 30fps, only 14.1 milliseconds is spent on tracking an object for every one second of MPEG-2 video. This surpasses the real-time requirement with a large margin. According to our algorithm complexity analysis, the estimated tracking time for HDTV resolution (1920x1080) is 2.88 ms/frame, which is also far beyond real-time.

Table 5.3 Time Costs for Object Tracking

Sequence name	Bitrate (mbps)	resolution	Object size (MB)	Tracking time (ms/frame)
Toycar	10	704x480	82	0.69
Mycar_in_parkinglot	5	704x480	35	0.42
Two_tractors	5	704x480	31	0.24
Walking_people	4	704x480	24	0.60
Tractor_with_moving_camera	4	704x480	69	0.42

CHAPTER 6

CONCLUSIONS

In this research we have implemented a perceptual region detection algorithm specifically designed for fast perceptual encoding. The algorithms presented in contemporary research are generally based on classical object detection that use varied mix of input information from various coding levels. As we move into the era of massive video stream processing, it is now important to look into the fundamental cost or raw information used as ingredient by various algorithms.

In a dynamic rate transcoding scenario, the cost of raw information used is remarkable. It is most expensive to obtain back the pixel level information. Even the DCT coefficients used by transform domain techniques also reside at two levels deeper than motion vectors in a coded stream due to differential block coding and subsequent Huffman coding. The proposed technique can avoid a great deal of raw image data processing, and yet remain very effective. Our steady-state system is able to operate with 80-100% coverage and less than 5% mis-coverage. A fundamental question is whether additional DCT or pixel level information used by other algorithms can provide additional tracking efficiency? Unfortunately, no qualitative measure of tracking efficiency was given by other comparative techniques in the literatures. It would be interesting to determine actually how much detection and tracking accuracy can be gained (or lost), if any, by the additional computational complexities involved in those methods? Motion vectors indeed contain implicit information about color and shape besides the motion. Thus, motion vectors are possibly the richest single coding element in terms entropy of perceptual information. Additionally, it is important to note in this context that for perceptual coding, the use of detail object boundary sought by the extra computations by several other techniques may not be very useful and may actually reduce the perceptual quality, if it is not again repealed by wide boundary approximation [46]. Thus, there are strong reasons to believe that the use of deeper information might bring only diminishing return.

Several extensions of this work can be contemplated. Currently, we have implemented an MPEG-2/MPEG-2 transcoder - a fully transparent transcoding system for a streaming video. The advantage of this specific system is that it will not require any other video distribution component - player or server to be aware of the rate adaptation. However, the algorithm can also be applied for other stream combination/interchange such as MPEG-2/MPEG-4 transcoding.

Another particularly interesting usage of the algorithm is video screening. The logical POP descriptors can be dynamically configured to detect and track specific perceptual objects and events. The descriptors can be further used as a basis set for a high level language. In more automated implementation, a descriptor can be added into standards' private stream (MPEG-2 and MPEG-4 already has the mechanism). Or adhoc mechanisms are possible where XML type object schema's can be downloaded transparently. Such extension can help in resolving the subjective ambiguity inherent in video object detection problem.

Yet another interesting are of application area of such fast perceptual object detection is the human augmented target detection. It is extremely difficult to surpass the target detection ability of human in quality and perceptual noise/blunder avoidance. Recently, there are interesting researches being conducted in man machine coupled target detection. The coupling has been attempted both at neurological and motor control level. Roughly, any human augmented target detection system precludes the use of any algorithm that takes more than milliseconds to process a frame. We are currently using this algorithm in a human augmented eye-tracking coupling based target detection system that requires machine processing in sub-millisecond range [49].

REFERENCES

[1] Information Technology- Generic Coding of Moving Pictures and Associated Audio Information: Video, ISO/IEC International Standard 13818-2, June 1996.

[2] 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.

[3] Keesman, Gertjan; Hellinghuizen, Robert; Hoeksema, Fokke; Heideman, Geert, Transcoding of MPEG bitstreams Signal Processing: Image Communication, Volume: 8, Issue: 6, pp. 481-500, September 1996.

[4] Youn, J, M.T. Sun, and J. Xin, "Video Transcoder Architectures for Bit Rate Scaling of H.263 Bit Streams," ACM Multimedia 1999', Nov., 1999. pp243-250.

[5] U. Chong and S. P. Kim, Wavelet Trancoding of block DCT-based images through block transform domain processing, SPIE Vol. 2825, 1996, pp901-908.

[6] Niklas Björk and Charilaos Christopoulos, Video transcoding for universal multimedia access; Proceedings on ACM multimedia 2000 workshops, 2000, Pages 75 – 79.

[7] J. Youn, M.T. Sun, and C.W. Lin, "Motion Vector Refinement for High Performance Transcoding," IEEE, Transactions on Multimedia, Vol. 1, No. 1, pp.30-40, March 1999.

[8] P. Assuncao and M. Ghanbari, "A frequency-domain video transcoder for dynamic bit rate reduction of MPEG-2 bit streams," Trans. On Circuits Syst. Video Technol., vol. 8, no. 8, pp. 953-967, 1998.

[9] Seo, Kwang-Deok; Lee, Sang-Hee; Kim, Jae-Kyoon; Koh, Jong S., Efficient rate-control algorithm for fast transcoding, ,SPIE Vol. 3528, 1998.

[10] Casas, J. R., & Torres, L, Coding of details in very Low Bit-rate Video Systems, IEEE Transactions CSVT, vol. 4, June, 1994, pp. 317-327.

[11] Haskell B. G., Atul Puri and Arun Netravali, Digital Video: An Introduction to MPEG-2, Chapman and Hall, NY, 1997.

[12] Khan, Javed I. & D. Yun, Multi-resolution Perceptual Encoding for Interactive Image Sharing in Remote Tele-Diagnostics, Proc. of the Int. Conference on Human Aspects of Advanced Manufacturing: Agility & Hybrid Automation, HAAMAHA'96, Maui, Hawaii, Aug. 1996, pp183-187.

[13] Minami. T. et. al, Knowledge-based Coding of facial Images, Picture Coding Symposium, Cambridge, MA, pp. 202-209.

[14] C. Faloutsos, R. Barber, M. Flickner, J. Hafner, W. Niblack, D. Petkovic, and W. Equitz, Efficient and Effective Querying by Image Con-tent. J. Intelligent Information Systems, vol. 3, no. 1, pp. 231-262, 1994.

[15] A. Pentland, R. Picard, and S. Sclaroff,Photobook: Tools for Content-Based Manipulation of Image Databases.Storage and Retrieval of Image and Video Databases II, Paper No. 2185-05, San Jose, Calif., pp. 34-47, SPIE, Feb. 1994.

[16] V. V. Vinod and Hiroshi Murase,Video Shot Analysis using Efficient Multiple Object Tracking.O-8186-7819-4/97, 1997 IEEE.

[17] T.D.Grove,K.D.Baker,and T. N. TAN, Color based object tracking.14th International Conference on Pattern Recognition (CV41).

[18] Paul Fieguth, Color-Based Tracking of Heads and Other Mobile Objects at Video Frame Rates.Proceedings of the 1997 Conference on Computer Vision and Pattern Recognition (CVPR '97).

[19] W. Krattenthaler, K.J. Mayer, and M. Zeiller,Point Correlation: A Reduced-Cost Template Matching Technique.Proc. ICIP, pp. 208 - 212, 1994.

[20] A.K. Jain, Y. Zhong, and S. Lakshmanan,Object Matching Using Deformable Templates.IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 18, no. 3, pp. 267-278, Mar. 1996.

[21] Stan Sclaroff and Lifeng Liu,Deformable Shape Detection and Description
via Model-Based Region Grouping.IEEE transactions on pattern analysis and machine intelligence, vol. 23, no. 5, pp. 475-489, May 2001.

[22] Yu Zhong, Anil K. Jain, M.-P. Dubuisson-Jolly,Object Tracking Using Deformable Templates.IEEE transactions on pattern analysis and machine intelligence, vol. 22, no. 5, pp544-549, May 2000.

[23] N. Diehl,Object-Oriented Motion Estimation and Segmentation in Image Sequences.IEEE Trans. Image Processing, vol. 3, pp. 1,901-1,904, Feb. 1990.

[24] H.H. Nagel, G. Socher, H. Kollnig, and M. Otte,Motion Boundary Detection in Image Sequences by Local Stochastic Tests.Proc. European Conf. Computer Vision, vol. II, pp. 305-315, 1994.

[25] N. Paragios and G. Tziritas,Adaptive Detection and Localization of Moving Objects in Image Sequences.Signal Processing: Image Comm., vol. 14, pp. 277-296, 1999.

[26] Nikos Paragios and Rachid Deriche,Geodesic Active Contours and Level Sets for the Detection and Tracking of Moving Objects.IEEE transactions on pattern analysis and machine intelligence, vol. 22, no. 3, pp. 266-280, march 2000.

[27] Natan Peterfreund,Robust Tracking of Position and Velocity With Kalman Snakes.IEEE transactions on pattern analysis and machine intelligence, vol. 21, no. 6, June 1999.

[28] M. Kass, A. Witkin, and D. Terzopoulos,Snakes: Active Contour Models. Int'l J. Computer Vision, vol. 1, pp. 321-332, 1988.

[29] Shoichi Arakil, Takashi Matsuoka, Haruo Takemura, and Naokazu Yokoya,Real-time Tracking of Multiple Moving Objects in Moving Camera Image Sequences Using Robust Statistics.1051-4651/98, 1998 IEEE.

[30] Yun-Ting Lin and Yuh-Lin Chang,Tracking Deformable Objects with the Active Contour Model.O-8186-7819-4/97, 1997 IEEE.

[31] Lili Qiu, Li Li,Contour Extraction of Moving Objects.

[32] R. Curwen and A. Blake,Dynamic Contours: Real-Time Active Splines.Active Vision, A. Blake and A. Yuille, eds., pp. 39-58. MIT Press, 1992.

[33] M.P. Dubuisson, S. Lakshmanan, and A.K. Jain,Vehicle Segmentation and Classification using Deformable Templates.IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 18, no. 3, pp. 293-308, 1996.

[34] D. Metaxas and D. Terzopoulos,Shape and Nonrigid Motion Estimation Through Physics-Based Synthesis.IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 15, no. 6, pp. 580-591, 1993.

[35] V. Caselles and B. Coll,Snakes in Movement.SIAM J. Numerical Analysis, vol. 33, pp. 2,445-2,456, 1996.

[36] M. Bertalmio, G. Sapiro, and G. Randall,Morphing Active Contours.Proc. Int'l Conf. Scale-Space Theories in Computer Vision, pp. 46-57, 1999.

[37] R. Goldenberg, R. Kimmel, E. Rivlin, and M. Rudzsky,Fast Geodesic Active Contours.Proc. Int'l Conf. Scale-Space Theories in Computer Vision, pp. 34-45, 1999.

[38] L. Wixson,Detecting Salient Motion by Accumulating Directionally-Consistent Flow.IEEE transactions on pattern analysis and machine intelligence, vol. 22, no. 8, August 2000.

[39] Robert Pless, Tomas Brodsky, and Yiannis Aloimonos,Detecting Independent Motion: The Statistics of Temporal Continuity.IEEE transactions on pattern analysis and machine intelligence, vol. 22, no. 8, August 2000.

[40] Hotter, M., & R. Thoma, Image Segmentation based on Object-Oriented Mapping Parameter Estimation, Sinal Process., v. 15, 1998, pp.315-334.

[41] Ngo, Chong-Wah, Ting-Chuen Pong and Hong-Jiang Zhang, “On clustering and retrieval of video shots”, ACM Multimedia 2001, Oct., 2001. pp51-60.

[42] Kuehne, Gerald, Stephan Richter and Mark Beier, “Motion-based segmentation and contour-based classification of veideo objects”, ACM Multimedia 2001, Oct., 2001. pp41-50.

[43] Wang R., H.J. Zhang and Y.Q. Zhang. A confidence measure based moving object extraction system built for compressed domain. 2000.

[44] Achanta, R., Kankanhalli, M. and Mulhem, P. Compressed domain object tracking for automatic indexing of objects in MPEG home video. Multimedia and Expo, 2002. ICME '02. Proceedings. 2002 IEEE International Conference on Volume 2, 26-29 Aug. 2002 Page(s):61 - 64 vol.2.

[45] Hariharakrishnan, K., Schonfeld, D., Raffy, P. and Yassa, F. Object tracking using adaptive block matching. Multimedia and Expo, 2003. ICME '03. Proceedings. 2003 International Conference on Volume 3, 6-9 July 2003 Page(s):III - 65-8 vol.3.

[46] Oleg Komogortsev and Javed I. Khan, Contour Approximation for Faster Object Based Transcoding with Higher Perceptual Quality, Proceedings of the International Conference on Computer Graphics and Imaging, CGIM 2004, Kauai, Hawaii, USA - August 2004, pp441-446.

[47] Khan, Javed I., Darsan Patel, Wansik Oh, Seung-su Yang, Oleg Komogortsev, and Qiong Gu, Architectural Overview of Motion Vector Reuse Mechanism in MPEG-2 Transcoding, Technical Report TR2001-01-01, Kent State University, [available at URL http://medianet.kent.edu/technicalreports.html, also mirrored at http://bristi.facnet.mcs.kent.edu/medianet] January, 2001].

[48] Khan, Javed I. & S. S. Yang, Resource Adaptive Nomadic MPEG-2 Transcoding on Active Network, International Conference of Applied Informatics, AI 2001, February 19-22, 2001, Insbruck, Austria. Pp.655-660.

[49] Oleg Komogortsev and Javed I. Khan, Predictive Perceptual Compression for Real Time Video Communication , ACM International Conference on Multimedia, ACM MM 04, New York, October 2004, best student paper contender, pp-220-227.