VIDEO OBJECT SEGMENTATION FOR INTERACTIVE MULTIMEDIA

A THESIS
SUBMITTED TO THE DEPARTMENT OF ELECTRICAL AND
ELECTRONICS ENGINEERING
AND THE INSTITUTE OF ENGINEERING AND SCIENCE
OF BILKENT UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
MASTER OF SCIENCE

By
Tolga Ekmekçi
November 1998

VIDEO OBJECT SEGMENTATION FOR INTERACTIVE MULTIMEDIA

A THESIS

SUBMITTED TO THE DEPARTMENT OF ELECTRICAL AND

ELECTRONICS ENGINEERING

AND THE INSTITUTE OF ENGINEERING AND SCIENCES

OF BILKENT UNIVERSITY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

MASTER OF SCIENCE

By

Tolga Ekmekçi

November 1998

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

[illegible]

Prof. Dr. Levent Onural (Supervisor)

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

[illegible]

Assist. Prof. Dr. Orhan Arıkan

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

[illegible]

Assoc. Prof. Dr. Gözde Bozdağı

Approved for the Institute of Engineering and Sciences:

[illegible]

Prof. Dr. Mehmet Baray
Director of Institute of Engineering and Sciences

ABSTRACT

VIDEO OBJECT SEGMENTATION FOR INTERACTIVE MULTIMEDIA

Tolga Ekmekçi
M.S. in Electrical and Electronics Engineering
Supervisor: Prof. Dr. Levent Onural
November 1998

Recently, trends in video processing research have shifted from video compression to video analysis, due to the emerging standards MPEG-4 and MPEG-7. These standards will enable the users to interact with the objects in the audiovisual scene generated at the user's end. However, neither of them prescribes how to obtain the objects. Many methods have been proposed for segmentation of video objects. One of the approaches is the "Analysis Model" (AM) of European COST-211 project. It is a modular approach to video object segmentation problem. Although AM performs acceptably in some cases, the results in many other cases are not good enough to be considered as semantic objects. In this thesis, a new tool is integrated and some modules are replaced by improved versions. One of the tools uses a block-based motion estimation technique to analyze the motion content within a scene, computes a motion activity parameter, and skips frames accordingly. Also introduced is a powerful motion estimation method which uses *maximum a posteriori probability* (MAP) criterion and Gibbs energies to obtain more reliable motion vectors and to calculate temporally unpredictable areas. To handle more complex motion in the scene, the 2-D affine motion model is added to the motion segmentation module, which employs only the translational model. The observed results indicate that the AM performance is improved substantially. The objects in the scene and their boundaries are detected more accurately, compared to the previous results.

Keywords: Video processing, video object segmentation, data fusion, object tracking, interactive multimedia, MPEG-4, content-based search, MPEG-7

ÖZET

ETKİLEŞİMLİ ÇOĞULORTAMLILIK İÇİN VİDEO NESNE BÖLÜTLEMESİ

Tolga Ekmekçi
Elektrik ve Elektronik Mühendisliği Bölümü Yüksek Lisans
Tez Yöneticisi: Prof. Dr. Levent Onural
Kasım 1998

Geliştirilmekte olan MPEG-4 ve MPEG-7 standartları, verilerin nesneler halinde saklanmasını ve yapılacak işlerin bu nesneler üzerinde yürütülmesini öngörmektedir. Ama bu standartlar bu nesnelerin nasıl elde edileceğini tanımlamamaktadır. Video nesnelerinin bölütlenmesi için bir çok metod önerilmiştir. Avrupa Topluluğu tarafından organize edilen projelerden COST-211ter çerçevesinde geliştirilen "Analiz Modeli" de bunlardan biridir. Analiz Modeli'nin performansı bazı durumlarda kabul edilebilir olmakla birlikte diğer birçok durumda elde edilen sonuçları "anlamlı nesneler" olarak değerlendirmek mümkün değildir. Bu çalışmada Analiz Modeli'nin modüler tasarımı sayesinde modele yeni bir modül eklenmiş, eski modüller daha iyi çalışan yenileriyle değiştirilmiş ve sonuç olarak modelin daha başarılı olması sağlanmıştır. Yeni eklenen modül içinde hareket kestirimi modülü kullanılarak bir parametre hesaplanmış, ve bu parametre video karelerinin atlanmasında kullanılmıştır. Varolan hareket kestirimi modülüne en büyük sonsal olasılık kriteri *(MAP)* ve Gibbs enerjilerine dayanan yeni bir metod eklenmiş, böylece daha doğru hareket vektörleri elde edilmiştir. 3-B düzlemsel nesnelerin hareketlerini açıklayan modelin hareket bölütlemesi modülüne ilave edilmesiyle Analiz Modeli'nde daha karmaşık hareketleri inceleyebilmek mümkün olmuştur. Elde edilen sonuçlar Analiz Modeli'nin performansının önemli ölçüde iyileştiğini göstermektedir.

Anahtar kelimeler: Video işleme, video nesne bölütlemesi, veri tümleşimi, nesne takibi, etkileşimli çoğulortamlılık, MPEG-4, içeriğe dayalı arama, MPEG-7

ACKNOWLEDGMENTS

I feel indebted to my supervisor Dr. Levent Onural. I appreciate his supervision and suggestions throughout the development of this thesis. I have also enjoyed his guidance about many “off-topic” talks, which I will miss a lot. :)

I would like to express my gratitude to the other members of my committee, Dr. Orhan Arıkan and Dr. Gözde Bozdağı, for taking their valuable time to read this thesis and commenting.

The thesis bears only my name as the creator, but I feel this is unfair to many friends who provided support during the creation of this work. The “Image Processing Lab” people, Aydın Alatan, Ertem Tuncel, Tunç Bostancı, and Serkan Kıranyaz did not leave me alone at any stage of the thesis work. In fact, Aydın and Ertem deserve special mention since this thesis would not see the sunlight without their never-ending contributions.

Friends in Bilkent also deserve to be mentioned here, since they will be the ones to remember after I leave the university. I am thankful to –in alphabetical order– Arçın Bozkurt, Ayhan Bozkurt, Deniz Başkent, Deniz Gürkan, Güçlü Köprülü, Gülbin Akgün, Gün Akkor, Lütfiye Durak, Tolga Kartaloğlu, and Yamaç Dikmelik and to the ones that I have forgotten to mention. They have suffered a lot during my thesis work. I am grateful they have chosen not to leave me alone in spite of all the nuisance I have caused.

Nothing I can do to acknowledge my family is sufficient to describe what they have experienced during the development of this thesis. They felt pain as I did, they felt joy as I did; I am sure they are now sharing my all-mixed-up feelings.

This thesis is devoted to my family. It nowhere compensates what my mom, sister, grandmom have given me so far; it is just a little reimbursement. If only dad could see these days... I am sure he would be proud.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation and Aim

Research on video processing commences in many fields [5]. As a result, different standards aiming different functionalities have been developed. With the progress in the area, new standards are still being developed and some others are being planned. Among these standards, major ones can be named as MPEG-1, MPEG-2, MPEG-4, MPEG-7, H.261, H.263.

Also, other parties' work affected these standards: ITU Standard H.261 (a standard for videoconferencing) is basically a result of the project COST-211bis. Similarly, COST-211ter project recommendation formed the basis for the ITU standard H.263 (videotelephony over regular phone lines).

Recent trends in multimedia research led to standardization activities MPEG-4 and MPEG-7. MPEG-4 has been developed with interactive multimedia in mind, while MPEG-7 targets "content-based indexing and querying". Both of these standards assume their input data is composed of "objects" in some form.

MPEG-4 and MPEG-7 work on an "object basis", i.e., the unit of action will be an object defined according to some criteria. However, neither of the standards defines how to obtain these objects. In case of video, if the source data is not already in the form of objects, "object segmentation" is a must to extract the objects in the scene. When the source is not already in the form of objects, neither MPEG-4 nor MPEG-7 can work without object segmentation.

COST-211 group has focused on the standardization activities of MPEG-4 and MPEG-7. The group has developed an "Analysis Model" (AM), which contains "a full description of tools and algorithms for automatic and semi-automatic image sequence segmentation (object detection, extraction and tracking)" [2]. The main idea is fusion of information from various sources by a set of rules yields better segmentation results.

Results of AM are acceptable in some cases. However, in many other cases, what is obtained is not good enough to be classified as "semantically meaningful objects" in the scene (i.e., boundaries of detected objects do not coincide well with the objects in the scene). For MPEG-4 and MPEG-7 to work satisfactorily, the objects should be identified properly (suitable for the purposes of the application).

In short, the model needs improvement to obtain "better" segmentation results. Fortunately, the model is modular; improving the performance simply requires removal of the old modules and insertion of better performing ones. This thesis involves replacement of several modules of the current AM to achieve "better segmentation".

1.2 Outline of the thesis

Chapter 2 gives information about the multimedia standards mentioned in this chapter, with an emphasis on the recent activities MPEG-4 and MPEG-7. Also, issues regarding objects and object segmentation will be discussed. Chapter 3 discusses COST-211 AM, with detailed information about the modules and their functions. Chapter 4 is a survey about various methods and algorithms in the literature about video object segmentation. Chapter 5 explains the work done to improve the COST-211 AM. Detailed information about the new modules are presented here. With these improvements, AM is upgraded to Version 4.0. Chapter 6 concludes the thesis, with comments on the obtained results and future work.

Chapter 2

Standardization Activities in Video Communication

2.1 Completed Standards: H.261, MPEG-1, MPEG-2, H.263

2.1.1 H.261

Activities on ITU standard H.261 were completed in 1990. H.261 is a standard for videoconferencing over ISDN ($p\times$ 64 kbs, $1 \leq p \leq 30$) [6]. H.261 follows mainly from COST-211bis proposal "Redundancy Reduction Techniques for Coding of Broadband Video Signals" [7].

The spatial block resolution in H.261 is either 8×8 pixels (for INTRA coded frames – frames coded directly, without a reference to the previous frame) or 16×16 pixels (for INTER coded frames, – frames coded with reference to the previous frame). In INTER coding, a *prediction error* is calculated between a *macroblock* (a region of size 16×16 pixels) in the current frame and the corresponding macroblock in the previous frame. Both INTRA frames and prediction errors are coded using the discrete cosine transform (DCT). Next step is the quantization of DCT coefficients and coding of these coefficients using entropy coding (Huffman coding) to achieve further compression.

For INTER frames, motion compensation is utilized for compression, i.e., the current frame is predicted using previous frame and motion information obtained from current and previous frames.

ISO standards MPEG-1 and MPEG-2 followed H.261 with minor modifications.

2.1.2 MPEG-1

MPEG-1 studies were completed in 1992. It is related to "coding of moving pictures and associated audio for digital storage media up to 1.5 Mb/s" [8] [9]. The media mentioned here is mainly CDROM. Basically MPEG-1 deals with storing/retrieving audio/video to/from CD-ROM. The limit 1.5 Mbs is the limit of the CD-ROM technology of those years.

Block-based motion estimation algorithms (at a suitably chosen spatial resolution) are utilized for motion estimation. The so obtained motion vectors are used in motion compensation to obtain a prediction of the frame to be coded. The prediction error is coded using DCT. This compensation is performed in three ways: by using a previous frame (and related motion information) to estimate current frame, by using a future frame to estimate the current frame, or by using both approaches in the estimation of current frame. Final bitstream containing information from motion estimation and DCT is coded using variable length codes.

2.1.3 MPEG-2

Work on MPEG-2 was completed in 1994. It is related to "generic coding of moving pictures and associated audio information" [10] [11]. The standard deals with bit rates up to 20 Mbs, aiming Digital TV and HDTV applications. Another intention is the transfer of digital audio/video content between production studios.

MPEG-2 builds on the coding tools of MPEG-1 for video and audio compression. These are grouped in different "profiles" to offer different functionalities. The main improvement here is "scalability", such as "SNR scalability" (the ability to play with the bit rate) or "spatial scalability" (the ability to change the spatial resolution).

2.1.4 H.263

H.263 related activities were completed in 1994. This ITU standard has been designed for low bit rate communication (i.e., videotelephony over regular phone lines); early drafts specified bit rates less than 64 Kbits/s (p $\times$ 8 kbs, p$\leq$8) [12]. Later this limitation has been removed and H.263 is used in many areas, not just low bit rate communications.

H.263 improves upon H.261 in many ways. Half-pixel accurate motion vectors are utilized in motion compensation (H.261 uses full-pixel accurate motion vectors). Some parts of the (hierarchical) bitstream structure are now optional, so H.263 can be configured flexibly for a lower bit rate (or better error recovery). Also implemented is the optional forward and backward frame prediction similar to that of MPEG. These prediction algorithms help achieving better quality than is provided in H.261, at the same bit rate. In addition to QCIF (176 $\times$ 144) and CIF (352 $\times$ 288) that have been supported in H.261, the formats SQCIF (128 $\times$ 96), 4CIF (704 $\times$ 576), and 16CIF (1408 $\times$ 1152) have been introduced.

2.2 Standard in development: MPEG-4

Research activities on MPEG-4 are completed. It will be an international standard in December 1998 [13].

Algorithms such as H.261 deal with coding (compression) of video frames without any semantic content analysis, which makes these algorithms unsuitable for interactive multimedia. They further assume that the video is composed of moving blocks. This is not always true, since objects in real life may have arbitrary shapes.

The spirit of MPEG-4, however, is "objects". Any information (audio, video etc) to be sent from one place to another is required to be in the form of an object (or a compound object, which is composed of "simple" objects). MPEG-4 is a standard on how these objects are represented, how compound objects are to be formed, and how these objects (along with their composition information) are to be transmitted.

In case of video, MPEG-4 defines "video objects" which correspond to distinct objects in the scene and "video object planes" which are the instances of these objects at a given time. Each object (plane) is coded separately and at the destination, the video is recomposed using the information from the objects and the composition information sent along with these objects.

2.3 Standard being planned: MPEG-7

Activities on MPEG-7 have formally started with a "call for proposals" in October 1998. It is formally named as "Multimedia Content Description Interface". MPEG-7 will be a standardized description of various types of multimedia information. This description will be associated with the content itself, to allow fast and efficient searching for material that is of interest to the user [14].

In other words, MPEG-7 will deal with "labeling multimedia information", in order to be able to search that multimedia information like text search of today.

2.4 Relationship of Object Segmentation to MPEG-4 and MPEG-7

Both MPEG-4 and MPEG-7 work with objects, but neither has a prescription on how to obtain the objects. This issue is very important for video part of MPEG-7 since the amount of data that exist in the "usual" form (not in the form of objects) is much larger than the data obtained as a composition of objects (i.e., by using a technology such as "blue screen"). In order to utilize this vast amount of data, a method which will decompose a given video into its objects is required. It is vital for MPEG-4 and MPEG-7 that such a segmentation algorithm feeds the objects to the MPEG-4 (-7). Without such an algorithm, both standards are useless.

The solution proposed by COST-211 is shown in Figure 2.1 [1]. The structure KANT (Kernel of Analysis for New multimedia Technologies) is the abstract layer, from which the particular solution, the COST-211 Analysis Model is developed. In particular, the shaded region is the Analysis Model that feeds the objects to the MPEG-4 coder. Although an MPEG-4 Coder is depicted as the coding block, any block that takes objects as input can benefit from the KANT approach (in particular, MPEG-7).

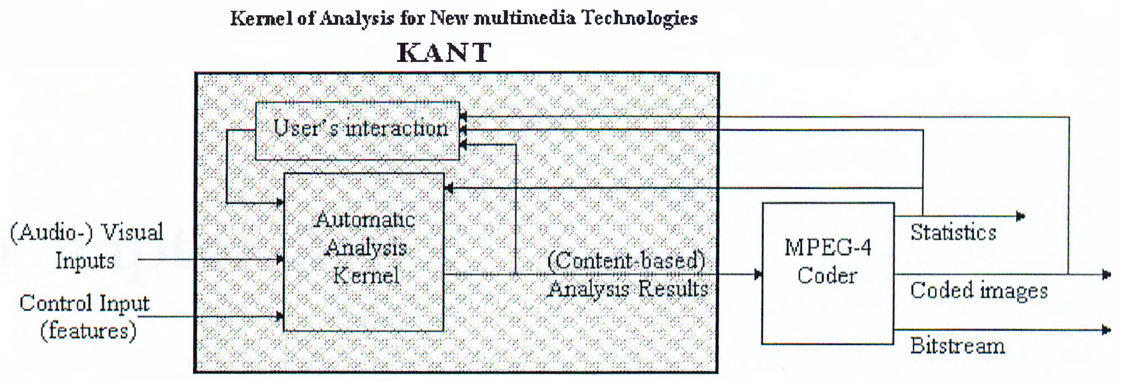

Figure 2.1. COST-211 Analysis Model: KANT – Broad Overview. (Reprinted as a courtesy of Alatan *et al.* [1])

The standards do not dictate any particular algorithm for object segmentation, however their performance strongly depends on the performance of those algorithms. Various studies on video object segmentation are discussed in Chapter 4.

Chapter 3

COST-211 Analysis Model

As indicated in Chapter 1, the aim of COST-211 Analysis Model (AM) is the fusion of information from various sources by a set of rules for a better segmentation result [2]. Currently motion information, color information, intensity changes and results from previous frames are fused by the rules.

The block diagram is in Figure 3.1 [1].

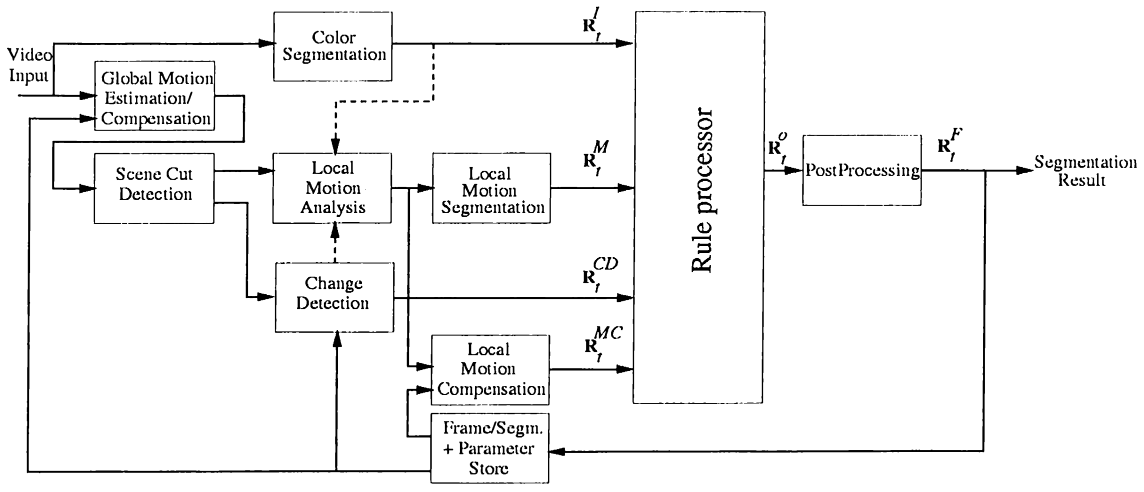

Figure 3.1. COST-211 Analysis Model. (Reprinted as a courtesy of Alatan *et al.* [1])

COST-211 Analysis Model has two modes of operation:

- Mode 1 gives a binary mask which distinguishes between foreground (moving) and background (stationary) objects.
- Mode 2 can differentiate between moving objects.

The functions of the blocks in Figure 3.1 are as follows:

3.1 Color Segmentation

This module handles the segmentation of the current frame into a predefined number of regions using only color information. For this purpose, a recursive-shortest-spanning-tree (RSST) based method [15] is used. The advantage of RSST is that the only input it requires is the final number of regions in the segmented image. This lets the user set the amount of detail in the resulting segmented image ("segmentation mask"). Results on the performance of the algorithm can be found in [16].

RSST initially maps input image into a weighted graph. Nodes of the graph forms the regions and links between the nodes denote the "distance" between two neighboring regions. Initially each pixel is a node.

After initialization, RSST checks all links, and merges the two regions of the link which minimizes the distance measure. Merging continues until desired number of regions is reached.

Distance measure is as follows [2], [17]: For regions R_1 and R_2

$$d(R_1, R_2) = \|\mu_{R_1} - \mu_{R_2}\|^2 \frac{N_{R_1} \times N_{R_2}}{N_{R_1} + N_{R_2}}, \quad \mu = \begin{bmatrix} Y_{avg} \\ U_{avg} \\ V_{avg} \end{bmatrix} \tag{3.1}$$

Here, N denotes the number of pixels in each region. μ is the "feature vector" for each region. In color segmentation, it consists of region averages of Y (luminance), U, and V (chrominance) values. First term in the expression forces RSST to join "similar" regions, and second term inhibits joining of large regions. Joining large regions is undesirable since it may lead to loss of object boundaries.

An example to segmentation with RSST is given in Figure 3.2. Here, the number of regions is set to 256.

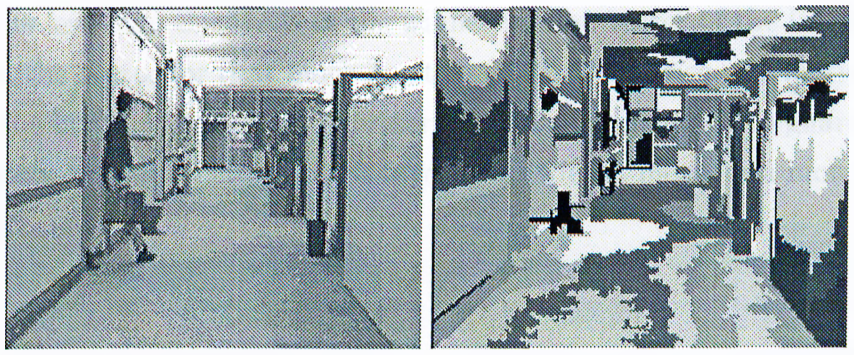

Figure 3.2. An example to segmentation using the color information

3.2 Local Motion Analysis

Motion between two consecutive (previous and current) frames is estimated. For this purpose, an estimation algorithm based on block matching, "Hierarchical Block Matching (HBM)" [18], has been used.

In HBM, the estimation is performed in three levels [2], [3]. In all these three levels, a sparse version of the exhaustive search is performed on current and previous frames.

One motion vector is found for each 4×4 block. The estimated block motion vectors are interpolated using 0^{th}-order interpolation in order to obtain a dense motion field.

Table 3.1 shows the hierarchical search parameters used at each level:

Hierarchy Level	**1**	**2**	**3**
Measurement window size	32	16	4
Search Range	16	8	2
Search Step Size	2	2	1
Spatial Resolution of measured vector field	4	4	4

Table 3.1. HBM parameters used in each level

The error criterion is the Mean of Absolute Differences (MAD) between the measurement window on the current Y-frame and the displaced measurement window in the previous Y-frame. If the measurement window goes out of borders, MAD is calculated only for the part that is inside the frame.

Finally, in order to force the resultant motion vector for a 4×4 block to $(0, 0)$, the following is applied: The 4×4 block MAD for the $(0, 0)$ vector is calculated. Then a predetermined constant (currently set to 1.0) is subtracted from it. If the result is less than the MAD of the winner motion vector of the last hierarchy level, then the vector for that block is set to $(0, 0)$. Otherwise, the motion vector found at the last hierarchy level is preserved. This procedure is to guard against noisy motion vectors. Vector $(0, 0)$ is favored a little bit more for a more uniform vector field.

3.3 Local Motion Segmentation

As in the color segmentation block, RSST is used for motion segmentation. The only difference from color segmentation is the use of estimated motion vector field components as input during the segmentation. The distance measure for this block is defined as follows [2] [17]:

$$d(R_1, R_2) = \|\mu_{R_1} - \mu_{R_2}\|^2 \frac{N_{R_1} \times N_{R_2}}{N_{R_1} + N_{R_2}}, \ \mu = \begin{bmatrix} M_{x,avg} \\ M_{y,avg} \end{bmatrix} \tag{3.2}$$

Here, $M_{x,avg}$ and $M_{y,avg}$ denote the averages of horizontal and vertical components of the motion vectors for a region, respectively.

The number of regions in the final segmentation ($\mathbf{R}_t^M$ in Figure 3.1) is set to four. This number is arbitrary, and determines the maximum number of objects that can be observed in the same frame. The number four is found to be appropriate for a number of sequences, but it may be changed to accomodate more/less number of objects in a sequence.

The boundaries of the regions are coarse, due to the matching errors inherent in the motion estimation. However, the object locations are found correctly.

An example to motion segmentation with RSST is given in Figure 3.3.

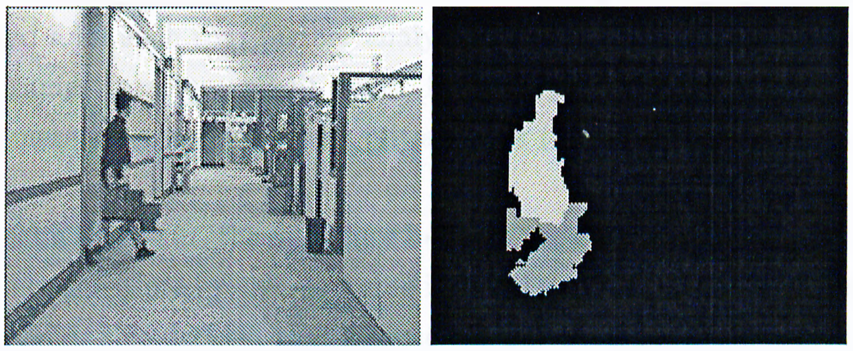

Figure 3.3. An example to segmentation using the motion information

3.4 Local Motion Compensation

Using the previous segmentation results and the motion information, it is possible to predict the locations of the objects in the current frame. Using results from previous segmentation masks is necessary in order to be able to track the objects throughout the sequence. Many methods in the literature work only on two frames at a time, therefore they can not guarantee the temporal coherency of the detected objects in the rest of the sequence. In AM, this information is used in determining the status of the objects in the current frame (for example, "the object is still moving", or "a new object has appeared"). Motion compensated result mask is denoted as $\mathbf{R}_t^{MC}$ in Figure 3.1.

3.5 Global Motion Estimation/Compensation

Given the current and previous frames I_t and I_{t-1}, a possible camera motion is estimated and compensated, as explained in [19] – [21]. Camera motion is modeled by the perspective motion field model by 8 parameters.

Next step is a postprocessing step which finds the regions where the model has failed. In such regions, motion vector field accuracy is improved by performing a full search within an area of a predetermined size.

In current version of AM, this module is not utilized by Mode 2.

3.6 Scene Cut Detection

Scene cut detector tries to detect the frames in a sequence where scene content has changed a lot such that further analysis based on information obtained from previous frames (previous result mask, for example) is meaningless. In such a case, parameters are reset to their initial value (values at the beginning of the execution).

To detect a scene-cut, the difference between the current frame I_t and the camera motion compensated previous frame is calculated. If this difference exceeds a given threshold, it is decided that a scene-cut has occured.

In current version of AM, this module is not utilized by Mode 2.

3.7 Change Detection

The change detection mask (CDM) between two successive frames is estimated. In this mask, pixels for which the image luminance has changed due to a moving object are labeled as changed.

The algorithm for estimation of the CDM [20] - [22] can be subdivided into several steps which are described in the following subsections. The final CDM is simplified and small regions are eliminated.

The steps to obtain the final CDM are explained below.

3.7.1 Computation of the initial CDM

The initial CDM (CDMi) is calculated from the camera motion compensated previous frame and current frame, by a thresholding operation on the squared luminance difference image [23].

3.7.2 Relaxation of initial CDM

Boundaries in the CDMi are smoothed by a relaxation technique as explained in [23] and [24]. Here, every border pixel is decided whether it will be in the *changed* area or *unchanged* area. For this purpose, a local threshold for each border pixel is calculated, taking into account the neighborhood of that pixel. The relaxation is processed iteratively, until only a small number of pixels are changed by relaxation or the maximal number of iteration steps N is reached. Final CDM is denoted as CDMs.

3.7.3 Temporal coherency of the object shapes

In order to finally get temporally stable object regions, the previous object masks are taken into account. In the CDMs, additionally all pixels which belong to the *changed* area in the *pixel memory* are set to *changed.* This memory keeps information about the last L CDM's. This dynamic memory is updated according to Equation 3.3 [22]:

$$MEM_{(t)}(x,y) = \left\{ \begin{array}{lll} L & , & if \;\; CDMs_{(t)}(x,y) = 1 \\ max(0, MEM_{(t-1)}(x,y) - 1) & , & if \;\; CDMs_{(t)}(x,y) = 0 \end{array} \right\} \tag{3.3}$$

The current CDMs is then updated by a logical OR operation (Equation 3.4) between CDMs and the previous output mask, taking into account the memory MEM. This CDM is denoted as $\mathbf{R}_t^{CD}$ in Figure 3.1.

$$CDM_{(t)}(x,y) = CDMs_{(t)}(x,y) \vee \left\{ \begin{array}{lll} \mathbf{R}_t^{O}(x,y) & , & if \;\; MEM_{(t)}(x,y) > 0 \\ 0 & , & if \;\; MEM_{(t)}(x,y) = 0 \end{array} \right\} \tag{3.4}$$

An example change detection mask is given in Figure 3.4.

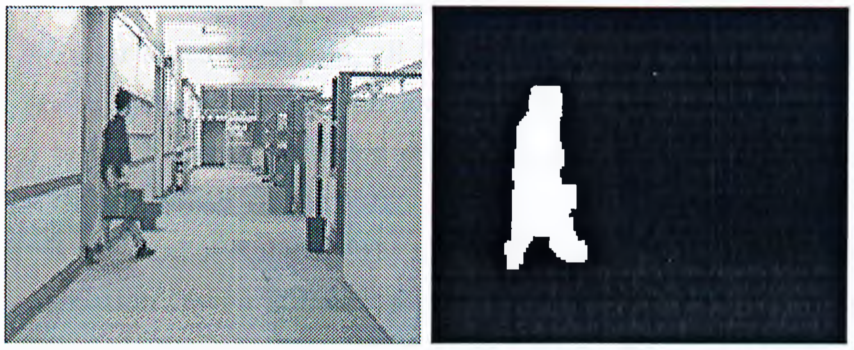

Figure 3.4. An example to change detection mask (CDM)

In current version of AM, this module is not utilized by Mode 2.

3.8 Rule Processor

The information from four sources ($\mathbf{R}_t^I$, $\mathbf{R}_t^M$, $\mathbf{R}_t^{MC}$, and $\mathbf{R}_t^{CD}$) are fused in this module, to obtain the object segmentation mask, $\mathbf{R}_t^O$ [1] – [4].

As mentioned before, AM incorporates two modes of operation: The first mode segments the scene into foreground (moving) and background (stationary) areas. Mode 2 distinguishes different objects in a scene.

3.8.1 Mode 1: Detection of moving objects and background regions [2]

In this mode, the results from the change detection, color segmentation and local motion analysis are used in order to distinguish foreground and background areas.

Initially, the uncovered background areas are eliminated from $\mathbf{R}_t^{CD}$ as in [22], [25], resulting in a preresult mask. A pixel is set to *foreground* if both the starting and ending points of the corresponding displacement vector are in the *changed* area of the change detection mask. If not, the pixel is set to *background*.

The color segmentation mask $\mathbf{R}_t^I$ has accurate boundary information; therefore color segmentation boundaries are utilized as object boundaries whenever appropriate. The color segmentation mask is projected onto the preresult mask and the following decision rules are applied to obtain the resulting object mask [2]:

Rule 1: Foreground detection

If the number of *foreground* pixels mapped onto a color segmentation region is above a predetermined threshold, all *foreground* pixels mapped onto this color region are set to *foreground.* In addition, all pixels within a range of N pixels with respect to the boundary of the preresult mask are set to *foreground.*

Rule 2: Background detection

This rule is the dual of Rule 1: if the number of *foreground* pixels mapped onto a color segmentation region is below a predetermined threshold, all *background* pixels mapped onto this color region are set to *background.* In addition, all pixels within a range of M pixels with respect to the boundary of the preresult mask are set to *background.*

Currently, the threshold is taken to be 80% of the color region's area, and M and N are set to 2.

3.8.2 Mode 2: Extraction of moving objects [1], [3], [4]

Mode 2 execution starts with mapping of each color segmentation ($\mathbf{R}_t^I$) region onto one region in $\mathbf{R}_t^M$ and one region in $\mathbf{R}_t^{MC}$. The rule for mapping is as in the following:

- Map a color region onto the $\mathbf{R}_t^M$ region (and onto the $\mathbf{R}_t^{MC}$ region) with maximum intersection area.

In Figure 3.5 [1], an example is given to mapping of color regions onto $\mathbf{R}_t^{MC}$ regions.

Second step is labeling of each region in $\mathbf{R}_t^M$ as "moving" or "stationary" by comparing its average motion with a given threshold. Each color region has the label of the motion region it is projected onto, and each $\mathbf{R}_t^{MC}$ has its label from previous segmentation mask.

The following rules are applied to obtain the mask at the output of the rule processor $\mathbf{R}_t^O$ mask [3], [4]:

Rule 1: Tracking of Objects

- If all color regions mapped onto the same $\mathbf{R}_t^{MC}$ region have the same label, merge those color regions.

This rule indicates the existence of a previous object in current scene (object tracking): the color regions that belonged to some object in the previous mask

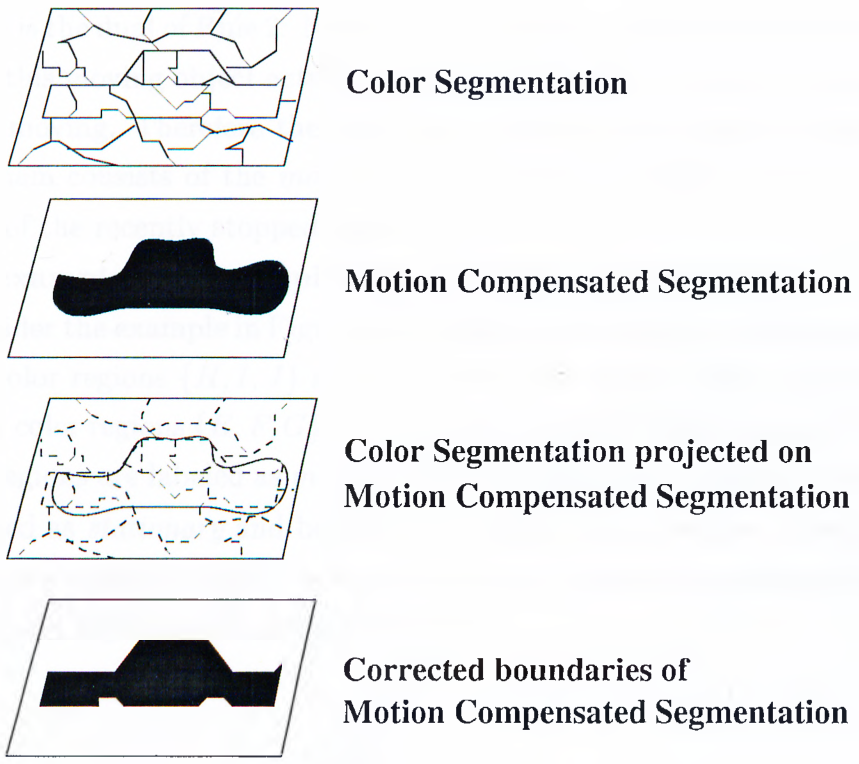
Color Segmentation
Motion Compensated Segmentation
Color Segmentation projected on Motion Compensated Segmentation
Corrected boundaries of Motion Compensated Segmentation

Figure 3.5. The projection of color regions onto $\mathbf{R}_t^{MC}$ (and correction of boundaries). (Reprinted as a courtesy of Alatan *et al.* [1])

also show up here, still as part of that particular object. No new objects have appeared in the scene.

Rule 2: Newly Exposed Objects

- Else if a $\mathbf{R}_t^{MC}$ region is *stationary*
  - Merge the *stationary* color regions mapped onto this $\mathbf{R}_t^{MC}$ region
  - Merge the *moving* color regions mapped onto same $\mathbf{R}_t^{M}$ region

This rule indicates a *stationary* object has changed its label, possibly due to the fact that a smaller still object (inside this larger *stationary* object) has started moving. Thus the old object is now split into two: one of them is the remnant of the old *stationary* object, and the others are the new (moving) objects.

Rule 3: Articulated Motion of Objects

- Else if a $\mathbf{R}_t^{MC}$ region is *moving*
  - Merge the *stationary* color regions mapped onto this $\mathbf{R}_t^{MC}$
  - Merge the *moving* color regions mapped onto this $\mathbf{R}_t^{MC}$

This rule is the dual of Rule 2: Now a moving region has changed its label, possibly because this moving object actually consisted of many objects, and some of them stopped moving. Therefore the old moving object is now split into two objects: one of them consists of the *moving* parts of the old object, and the second one consists of the recently stopped parts.

An example for the application of the proposed rules

Consider the example in Figure 3.6, which is taken from [1] (courtesy of Alatan *et al*). Color regions $\{H, I, J\}$ are mapped on one motion ($\mathbf{R}_t^M$) region (one object) and color regions $\{E, F, G\}$ are mapped to another motion region. These two motion regions are labeled as *moving*. The remaining color regions, $\{A, B, C, D\}$, are labeled as *stationary* and belong to the third motion region. However, there are only two regions in $\mathbf{R}_t^{MC}$: one region which contains color regions $\{E, F, G\}$, and a second region which contains the rest.

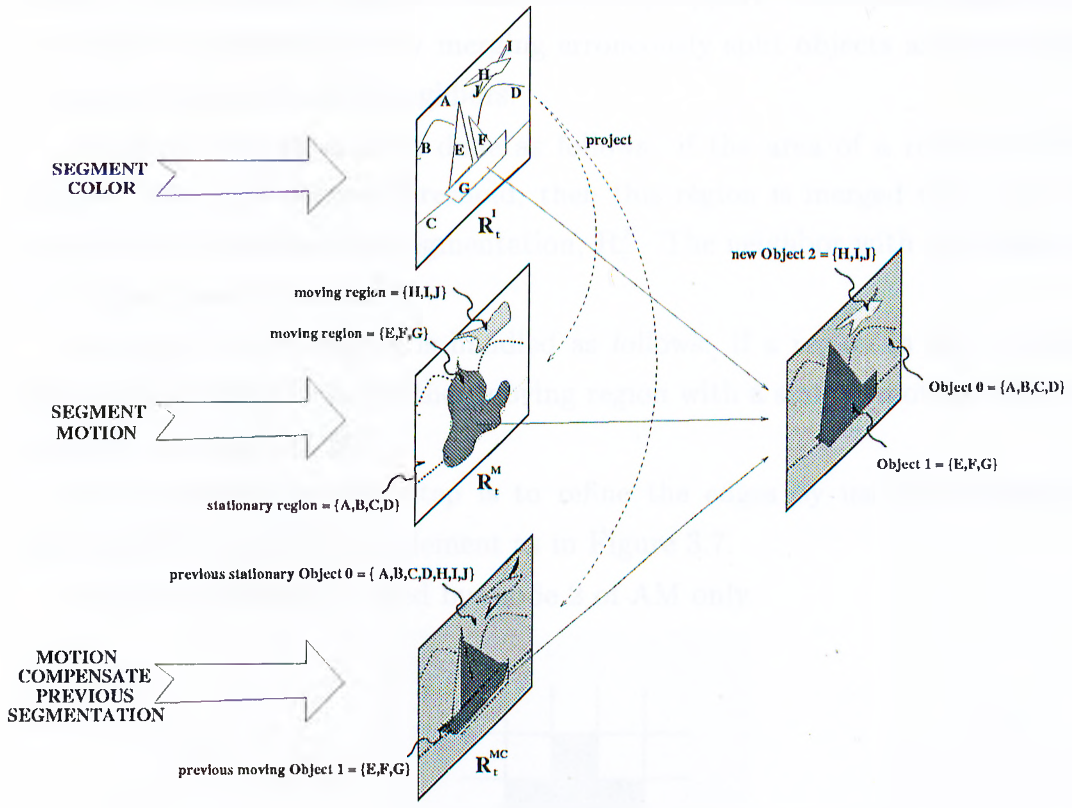

Figure 3.6. A simple example for Mode 2. (Reprinted as a courtesy of Alatan *et al.* [1])

Color regions $\{E, F, G\}$, which belong to a *moving* region in $\mathbf{R}_t^{MC}$ also belong to a *moving* region in $\mathbf{R}_t^M$. Rule 1 says that this object is not new, it has been tracked from previous frame. So, these color regions are merged and labeled as part of object-1 in current object mask $\mathbf{R}_t^O$.

Rule 2 is in action in the rest of the scene. The color regions $\{A, B, C, D, H, I, J\}$ belonged to a *stationary* region in the previous object mask. Currently, some of these color regions are still labeled as *stationary*, while some of them are labeled as *moving*. These moving color regions ($\{H, I, J\}$) are merged to form the new (*moving*) object in the scene (object-2), while the regions $\{A, B, C, D\}$ make up the stationary region (object-0).

3.9 Post Processing

Since the rule processor splits regions, it is likely that one semantic object is broken up into multiple objects. Also, some very small regions (which are not likely to be semantic objects themselves) may appear. Post processing tries to improve the segmentation by merging erroneously split objects and by merging the small regions to their neighbors.

Merging of small areas is done as follows: if the area of a region in $\mathbf{R}_t^O$ is smaller than a predefined threshold, then this region is merged with one of its neighbors to form the final segmentation, $\mathbf{R}_t^F$. The neighbor with the same label and largest area is chosen.

Merging of split objects is handled as follows: if a region in $\mathbf{R}_t^O$ is moving and if it is a neighbor to another moving region with a similar motion, these two regions are merged in $\mathbf{R}_t^F$.

Final operation in this step is to refine the edges by using morphological opening with a structuring element as in Figure 3.7.

The post-processor is used by mode 2 of AM only.

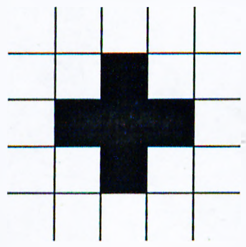

Figure 3.7. The structuring element

Chapter 4

Video Object Segmentation

The term "object segmentation" can be defined as "extracting objects from some source". Two immediate questions follow:

- What is an object?
- What can be the source?

A quick answer to second question could include a single image, an image sequence (a sequence of frames, or video), an audio stream (a song, some mutter), a movie etc. However, there is no easy, or "correct" answer to the first question. The "object" may have different meanings under different conditions: Source may be different (audio vs video); moreover, within the same source, what might constitute an object (i.e., object features) may be different.

Consider Figure 4.1. Which sections should we classify as objects? The woman's mouth, eyes, face, head, the entire woman, or the screens behind the woman?

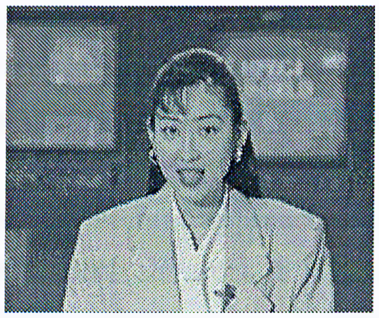

Figure 4.1. An example to demonstrate difficulties in object definition: What should be considered as objects in this scene?

This simple example shows that even human beings may not agree on what should be classified in a scene. A rule of thumb could be "anything that has a name can be an object" but this rule is too abstract for a computer to process. Therefore, attempts in object segmentation have been concentrated on what kind of low-level information can be obtained from a given scene, and how this information is related to high-level, semantic objects.

When the source for object segmentation is a single image, extracting semantically meaningful objects becomes harder (compared to extracting objects from video, since video provides extra information in the form of more frames which are temporally related). Therefore, methods which try to extract objects may bring in extra constraints, or make prior assumptions, some of which may be due to the nature of the desired application. In [26] and [27], a template-based approach is utilized. The statistics (model parameters) of the template to be recognized is obtained and compared to that of the image to be analyzed. An application suitable for such an algorithm is recognition of traffic signs. In another application, where airborne fiberglass particles are to be analyzed in a scanning electron microscopy image [28], the description of objects is generated using a polygonal approximation of their boundary.

Sometimes it is necessary to define new features, based on the immediate observable features like color, intensity. In [29], for example, where a multi-resolution color clustering algorithm is applied to images for indexing and retrieval (in context of MPEG-7), a new color feature based on octree data structure is introduced. Any input to this querying mechanism is an image. The newly defined feature of input image is calculated and compared to the ones in the database.

The type of information contained in an image may also be coming from an application specific source. In [30], for example, a 3-D image segmentation technique is described, where the input image is a range image (an image which contains range information about an object when viewed at a particular distance and angle).

Another issue is that many different approaches may be utilized to work on the same type of information. Both the work in [31] and [32] rely on the texture properties of the image. In [31], the image is segmented into regions using the texture information by 2-D Wold decomposition. [32] also deals with segmentation based on textures, but utilizes a hierarchical Markov Random Field (MRF) to model the textures.

Video segmentation is a very different issue, however. Now source is much broader: there are many frames to consider, and those frames are closely related to each other. In other words, temporal relation gets into the scene and supplies more information than individual frames do. The term "video segmentation" actually refers to two different type of operations. First one is "extracting semantically meaningful objects from a given video", and second is "dividing video into temporal segments where each segment can be described in a compact way." For example, in an MPEG-7 context, a "compact" description will be the one which allows indexing and querying to be done fast and efficiently.

Video Segmentation: dividing video into temporal segments

In [33], for example, video is assumed to be in the format that MPEG-4 can decode, and that video is analyzed for "Decision Support Representatives (DSR's)" which properly represent each shot (video segment). Then queries to that video will be processed using the DSR's of its shots. A similar study is presented in [34]. The aim is to extract effective discriminating features from reduced sets (shots) and use them in indexing. The key frames are selected using a discriminant function (based on eigenvectors obtained from the image). After key frames are selected, another discriminant function is used to group similar key frames, which allow each group to be treated as a unit.

The approach in [35] discusses a system for indexing of video using motion information. The system, mainly developed for surveillance applications (in which motion is assumed to have long trajectories and is mainly translational) expects video as an MPEG-1 stream, and segments it to determine the "correspondence of objects" between frames (object tracking). The data belonging to the center of objects is utilized (one (x, y) pair for each object in each frame), and for the video segment that object exists, two vectors from x and y positions are formed. In the database, first eight coefficients of wavelet transform of these vectors are kept. When the user sends a query (by drawing trajectory of desired object using a mouse), these coefficients and coefficients from user entry are compared for a match, and segments closest to user's entry are returned. If desired, search may be supported with extra information regarding object's color or size.

Another approach, [36], which tries to segment the video based on camera cuts (scene cuts, places where a substantial amount of change occurs in the scene) uses intensity histograms. Based on the information from histogram, features such as dissolve, fade in/out, wipe etc. can properly be detected and labeled.

The study "VideoBook" [37] is another framework for content based query and retrieval from video databases. Inspired by the human eye (which has motion sensors, spatial orientation and color detectors), a set of measures to be used in characterization of video segments are proposed. The measures are obtained from motion, texture and colorimetry data, as well as entropy. For each shot, an 8-parameter vector is constructed (3 from motion, 2 from texture, and 3 from color). Similarity is measured by using the mean-square-error between vectors in the database and vector belonging to the input shot. Since the amount of data to be processed is small, the system can work in real-time.

Video Segmentation: extracting semantic objects from video

In this case, the use of temporal and spatial information will differ from the methods mentioned above. Now, a common approach to object definition based on low-level information is "a region with uniform color properties and coherent motion" and with this understanding, extracting objects will reduce to "finding whereabouts of objects using motion information and incorporate spatial information while deciding on object boundaries." The idea behind this approach is that temporal (motion) information will yield coarse boundaries (due to ill-posedness of motion estimation problem [5] and -to a degree- the methods used in motion estimation) but help in locating the object in a frame. Spatial (color) information, which gives sharp boundaries, will be used to determine object shape.

Object segmentation algorithms, however, do not always treat objects in this manner. Rather, they may be investigated under three groups, according to the information they make use of to find the objects (actually the approach "uniform color properties and coherent motion" is only one of them). This classification is rough, since in each group, the algorithms may incorporate extra features as an aid, or they may differ in the ways they utilize the "main" source of information.

- Algorithms that utilize motion information
- Algorithms that utilize color (intensity) information
- Algorithms that use both of these information.

Algorithms that utilize motion information for object detection

Algorithms in this group use motion estimation results in detecting and segmenting video objects. If the motion estimation results are correct, such algorithms yield good results. But motion estimates near the boundaries of objects can only be reliable if the segmentation mask is known beforehand. Hence, boundaries obtained from such methods are generally incorrect.

The general approach is as follows: Initially, motion between two frames is estimated, to obtain a dense motion field. Afterwards this field is segmented using any distance measure and motion model. The model may be translational [38], [39], affine [40], quadratic [41], or any other motion model. In the extreme case, [42] presents a method which incorporates all these three for maximum performance. The distance measure also varies from one method to another. In [40], the residue between estimated motion field and the motion field calculated using model parameters are utilized. On the other hand, the residue between motion compensated first frame and second frame (i.e., displaced frame difference, DFD) are used in [38]. Another approach, [39], checks the region averages and merges two regions if they are close enough (distance between them is below some threshold).

The approach in [41] is a bit different, since it also aims to code the extracted objects efficiently (using minimum number of bits). Here, the image is divided into blocks, and for each block, motion parameters are calculated. Merging is based on a region growing algorithm, where seeds (initial regions) are the well compensated blocks (blocks for which DFD is less than some threshold). The ultimate aim is to label each region as either temporally unchanged, model compliance, or model failure. Model compliance regions are the regions which can be described by their shape information and motion parameters. Similarly, when this information is insufficient to describe a region, that region is flagged as a model failure region, and coded by other means.

The method in [42] is also different in that three motion models are used in the segmentation process. The initial dense motion field is segmented using the translational motion model, with a distance measure similar to [39]. Region merging proceeds until the distance between any two neighboring regions is greater than some threshold. Next stage is the merging of these regions using affine model. At each step, affine motion parameters are extracted from two neighboring regions (which are candidates for merging) and the standard deviation of the residue between estimated field and the field calculated from motion parameters is checked. The two candidates are merged if this value is below some threshold. Final step is similar to this one, except a quadratic model (as in [41]) is employed.

The algorithms may also bring in additional constraints or assumptions. In [40], motion estimation is done around each pixel's neighborhood, implying the motion is assumed to be small. If the displacement happens to be outside that neighborhood, the estimated motion will be incorrect. Another assumption is that the error will be high if the search window during motion estimation is placed across object boundaries and vice versa. Depending on the window size and position, this may turn out to be untrue in case of closely located objects, which again leads to incorrect estimates. Another example to such (implicit) assumption is the work in [42], in which a static background is assumed. The motion estimation and segmentation is only considered on the areas which are labeled as *moving* by the change detection mask obtained from two frames, however, there is no global motion estimator/compensator employed. In case of camera motion, the *moving* areas will be labeled incorrectly and this will fail the whole method.

As shown in [5], motion estimation based on two frames only is an ill-posed problem. The occluded areas has to be treated separately in order to get correct estimates. However, among the algorithms discussed until now, the only one that employs occlusion detection is [38]. Here, in addition to motion estimation between current and previous frame, motion between current and next frame is also estimated, and using both of these estimates, the spatial order of the objects is determined. Otherwise, in case of occlusion, the motion estimates will be incorrect.

Another issue is related to the "memory" utilized in these methods. None of the algorithms discussed above keep track of the status of the objects they find. After the objects are found using a pair of frames from the sequence, the algorithms start over with another set of frames (the use of "model compliance" areas in [41] may be considered as a kind of memory, but not the kind that is being mentioned here, and it only depends on previous frame). For proper tracking of an object throughout the sequence, one needs to consider issues such as whether that object exists in previous frame(s); if so what is its status (i.e., stopped, moving, new object) etc. Therefore, none of the algorithms presented here guarantee the continuity of the objects they have found.

Algorithms that utilize color (intensity) information

These methods use only spatio-temporal intensity information instead of estimating the motion. The result is a change detection mask which marks moving and stationary regions; therefore multiple objects can not be identified. In addition, the methods' performance will degrade if some pre- and post-processing is not applied to handle a possible camera motion, illumination change, or noise in the input sequence.

A common approach to obtain a change detection mask starts with the detection and removal of (possible) camera motion. Then the difference image is obtained and thresholded: If difference for any pixel is greater than some threshold, that pixel is marked as *changed*, otherwise *unchanged*. How this "raw" detection mask will be used depends on the specifics of an algorithm or application. In most cases, this mask is further processed to obtain better results (using constraints such as smooth contours, or elimination of isolated points).

As in the previous case, many algorithms bring in other assumptions, or utilize application-specific constraints. Stationary background is one of these constraints which is employed in [43], [44], and [45]. In [43] and [45], image sequences obtained by a fixed camera are analyzed for objects. Specifically, [43] deals with road sequences in real-time, while [45] takes its input from a surveillance camera fixed to some location. In addition, the background is known *a priori* in these cases. The initial detection mask can simply be formed by comparing the known background with the acquired image.

In [44], input is assumed to be a "head & shoulders" sequence. Stationary background is also inherent here, however, unlike [43] and [45], no *a priori* knowledge of background is available.

The detection masks obtained in [43] and [44] are not used directly, but further processed: polygons are fitted to the boundaries of changed areas in [43], while [44] extracts the edges and fits them to the regions in the detection mask. In contrast, [45] does not do any more processing, but searches for an ellipse template (representing head) in the changed region. If such a template is found, a face template is searched for facial features such as eyes and mouth.

Although camera motion is not a concern in these cases, illumination changes and occlusions still plague these methods, especially [43] and [45]. Under different conditions (day/night in case of [43] and different lightning in [45]), illumination is different, which will lead to an incorrect detection mask, unless the algorithm is trained somehow beforehand for such situations. In case of an occlusion, there is no way to differentiate between two objects and these algorithms which try to build an object model according to object shape will fail.

The assumption of small motion which was incorporated into [40] also exists in [44], albeit in a more restricted way: The motion is assumed to be little but it should be sufficient for detecting facial features. This is necessary since the algorithm checks overlap between previously found mask and currently obtained mask for object correspondence. In the sequences where this assumption does not hold, incorrect correspondences will be established, which will propagate throughout the sequence.

The methods in [46] and [22] are general purpose algorithms, with no specific application in mind[1]. Both algorithms obtain an initial change detection mask as described earlier, and impose MRF-based smoothness constraints on the detection mask. Another point is that both of them have "memory". In [22], processed detection mask is improved with previously obtained masks, while in [46], only last mask is used. Both of these algorithms perform well, as long as no shading changes or occlusions occur.

Algorithms that utilize both information

The algorithms in this group may further be grouped according to how they utilize motion and color information. This is not conclusive, but it gives an idea about the approaches in this class of algorithms:

1. Algorithms which utilize a single metric that both takes into account motion and color information [47] – [50],
2. Algorithms which utilize motion and color information separately [51] – [53],
3. Algorithms that perform simultaneous estimation and segmentation [54].

The algorithms that go into first group employ a distance measure (in merging two neighboring regions) as in Equation 4.1:

$$D = \sum_i \omega_i f_i \tag{4.1}$$

Here, f_i denote the value of a specific feature (like color, motion), and ω_i denote the weight assigned to that particular feature.

Specifically, consider [47], where the similarity measure used for joining pixels to an existing region R is defined as:

[1]In fact, [22] forms the basis for Rule Processor Mode 1 in AM. Therefore, the explanations here will be a brief summary of the mentioned study. More detailed information is presented in Chapter 3 and related references.

$$S(x,y;R) = \alpha S_m(x,y;R) + (1-\alpha)S_i(x,y;R) \tag{4.2}$$

The S_m term for any pixel is the displaced frame difference

$$S_m(x,y) = I_k(x,y) - I_{k-1}(x - d_x(x,y), y - d_y(x,y)) \tag{4.3}$$

Here, I_k denotes intensity values in frame k, d_x and d_y are the horizontal and vertical components of the motion vectors, respectively. The S_i term for any pixel is simply the intensity difference between current pixel and the region under consideration.

The method in [49], which is based on the similarity tests in [55], also brings in a similar metric:

$$F_{AB} = T_{AB} - k\, T_{AB}\, (M - S_{AB}) \tag{4.4}$$

F_{AB} is the spatio-temporal similarity between regions A and B. T_{AB} is the temporal similarity, S_{AB} is the spatial similarity. M is the maximum value of the similarity between region A and its neighbors.

All three methods calculate their features using different algorithms but they all end up using a similarity measure which incorporates some "unknown" coefficient (k in [49], α in [47], and ω_i in [48]). The idea of incorporating temporal and spatial information into a similarity measure in this manner may severely affect algorithm performance. The methods do not indicate any clue about how to select the weight parameters, yet these parameters directly affect segmentation performance. There is no simple way of setting the weights which will work "best" on all sequences, therefore this approach may turn out to be inferior.

Methods in second group treat temporal and spatial information separately. The advantage is that color segmentation yields sharp boundaries, which coincides with object boundaries better than motion boundaries. In [51], for example, the regions obtained after color segmentation are merged if they are similar in motion. Measures such as DFD and maximum likelihood tests are used to evaluate the similarity.

In [52], initial step is forming motion regions which will form the basis of the objects. An iterative k-means algorithm is utilized to get the motion regions. Merging continues, until distance between two candidates is greater than some predetermined threshold. Remaining pixels are joined to existing regions using luminance information.

Another work, [50], also attempts object segmentation using motion and color (luminance) segmentation. Initially, any possible camera motion is estimated and removed. Local motion is estimated by a matching technique [56]. Next step is the segmentation of current image based on luminance values using a k-means algorithm [57]. For each region obtained by luminance segmentation, motion parameters are estimated. The regions which are similar in motion are joined using a k-medoid clustering algorithm [57] performed on motion parameters of each region.

Processing the video in pairwise frames makes the algorithm lack the temporal coherency in terms of object continuity. To cover up this last "deficiency", the study in [58] (which is a tracking algorithm added to [50] and an improved merging algorithm) is proposed. First step is the prediction of locations of objects (of previous frames) in the current frame. Then, mean-square-error (MSE) after this motion compensation is compared to a threshold. If MSE is less than the indicated threshold, the object is classified as valid. For such valid objects (in a way similar to Rule Processor Mode 2), it is checked if they correspond to any previous object. Second step is a kind of rule processor which checks a number of hypotheses (based on motion similarity and spatial similarity tests) and decides the "correspondence" of objects in current frame with the ones in previous frames. With this algorithm, tracking of objects throughout the sequence is established.

Coding of the objects is an additional issue in [53]. The object segmentation methods are not drastically different from that of the methods described above, but additional assumptions are utilized. An example to such assumptions is based on the idea that human eye is less sensitive to chrominance component of a color. Therefore, one object is coded only using one chrominance value (only one UV pair), instead of using all its pixels' UV values, saving significant bandwidth.

Human supervision is another important factor in object segmentation algorithms. Since a human being is the one who can know best what an object is, a little help can drastically improve the segmentation result, compared to fully automatic segmentation algorithms.

This is the idea behind the studies in [59] and [60]. In [59], the user outlines the object's interior and using morphological dilation, this "interior" outline is extended to form the "exterior" outline. Using these outlines, interior and exterior cluster centers are calculated. Interior outline should be close to the boundary and exterior outline should lie out of the object in order for the algorithm to work properly.

Then the pixels in this intermediate area assigned either to the object or not, depending on a distance measure which is as follows:

$$D_i = \omega_{color}(|r - r_i| + |g - g_i| + |b - b_i|) + \omega_{coord}(|x - x_i| + |y - y_i|) \qquad (4.5)$$

where ω_{color} and ω_{coord} are weights for color and coordinate information, r, g, b are color component values for cluster centers (either interior or exterior), x and y are cluster center coordinates.

Next step is motion estimation and warping of previously obtained object to current frame, assuming that the motion is little (therefore the shape does not change significantly from frame to frame). This tracking continues until the end of the sequence with a similar dilation operation.

This algorithm also suffers from the unknown coefficient issue like [49], [47], and [48]. Moreover, the algorithm is limited in its use, since it does not allow new objects, it does not allow objects to disappear. However, it demonstrates what can be achieved using supervision. Objects can be tracked very accurately using this algorithm.

The other example which utilizes human assistance, [60], is similar to [59] in that user assistance is in the form of "drawing". However, this time, the only requirement is that the drawing stay inside the object, not necessarily close to the boundary. It constructs feature vectors for each pixel, and assumes that distribution of a particular feature in each region, which will be obtained from user input, can be approximated by sum of Gaussian PDF's. Number of PDF's (*modes*) is limited to 5 and an expectation-maximization algorithm is used to estimate the parameters that fit best to the user input. Then the remaining pixels are assigned to the previously formed regions, by calculating the *a posteriori* probabilities. Used features are color, motion, texture, position, luminance, and any available *a priori* background information).

The supervised mesh-based approach introduced in [61] is different from other algorithms in its class in that object segmentation is performed only in the initial frame, by human assistance. Then the objects are tracked throughout the sequence. New objects are detected by the use of uncovered background regions.

In this approach, regions which will form the objects in the scene are obtained using color and motion segmentation results, as described in [62]. These regions are joined interactively to form semantic objects, with object boundaries approximated by polygons. Then nodes are placed in each region using the algorithm described in [63], and with constrained Delaunay triangularization, meshes are formed (constraint = object boundaries). The same segmentation algorithm is applied to uncovered background areas. If such an area is found to be similar (in terms of motion, color, texture properties) to any existing region, they are merged. If not, such uncovered background areas form new regions.

The primary drawback of this algorithm is its new object detection mechanism. The uncovered background areas are generally small, therefore motion segmentation can not be reliably applied upon them. Another problem rising due to the size of such areas is that reliable texture, color, or motion information can not be obtained from these areas. Moreover, similarity in terms of such features does not mean the two regions belong to same object (such incorrect merging is corrected interactively). One final point is related to the temporal coherency of the objects detected: this method does not make use of the previously found objects (similar to the many methods discussed so far, it works on two frames at a time) and does not consider the temporal evolution of one object (object stopped, disappeared etc). This means the object continuity is not guaranteed throughout time.

It has been noted earlier that for reliable motion estimates, boundaries of regions in the segmentation mask should be known. However, to obtain correct boundaries, good motion estimates are needed. To solve this "chicken & egg" problem, some algorithms propose to simultaneously perform motion estimation and object segmentation. These algorithms iteratively update the motion field and the segmentation mask. This approach is a powerful one, but the computational complexity of the methods utilizing it is too high; this is the main drawback of such methods.

Such an algorithm is presented in [54] [64]. It is based on the *maximum a posteriori* (MAP) criterion, and the interdependence of constraints for motion estimation and object segmentation are expressed by a Gibbs distribution. Three energy functions are constructed to express the constraints for motion estimation and object segmentation. Weighted sum of these functions constitutes the actual energy function to be minimized. *Highest Confidence First* [65] and *Iterated Conditional Mode* [66] algorithm as explained in [67] are utilized to minimize the energy function. As mentioned before, the main drawback of the algorithm is the computational power it requires. Another issue is the determination of the weights associated with each energy expression. They give user the ability to control the relative emphasis on various terms, but they have to be determined in an *ad-hoc* manner, since there is no way of knowing in advance what kind of a distribution will result in good estimates and segmentation.

Chapter 5

Improved COST-211 Analysis Model: Version 4

As mentioned in introduction, COST-211 Analysis model performs well to a degree but there is a lot of room for improvement. The improvements mentioned in this chapter are integrated into COST-211 Analysis Model, and with these, the Analysis Model is improved to Version 4.

Consider Figure 3.1, which is replicated here for convenience.

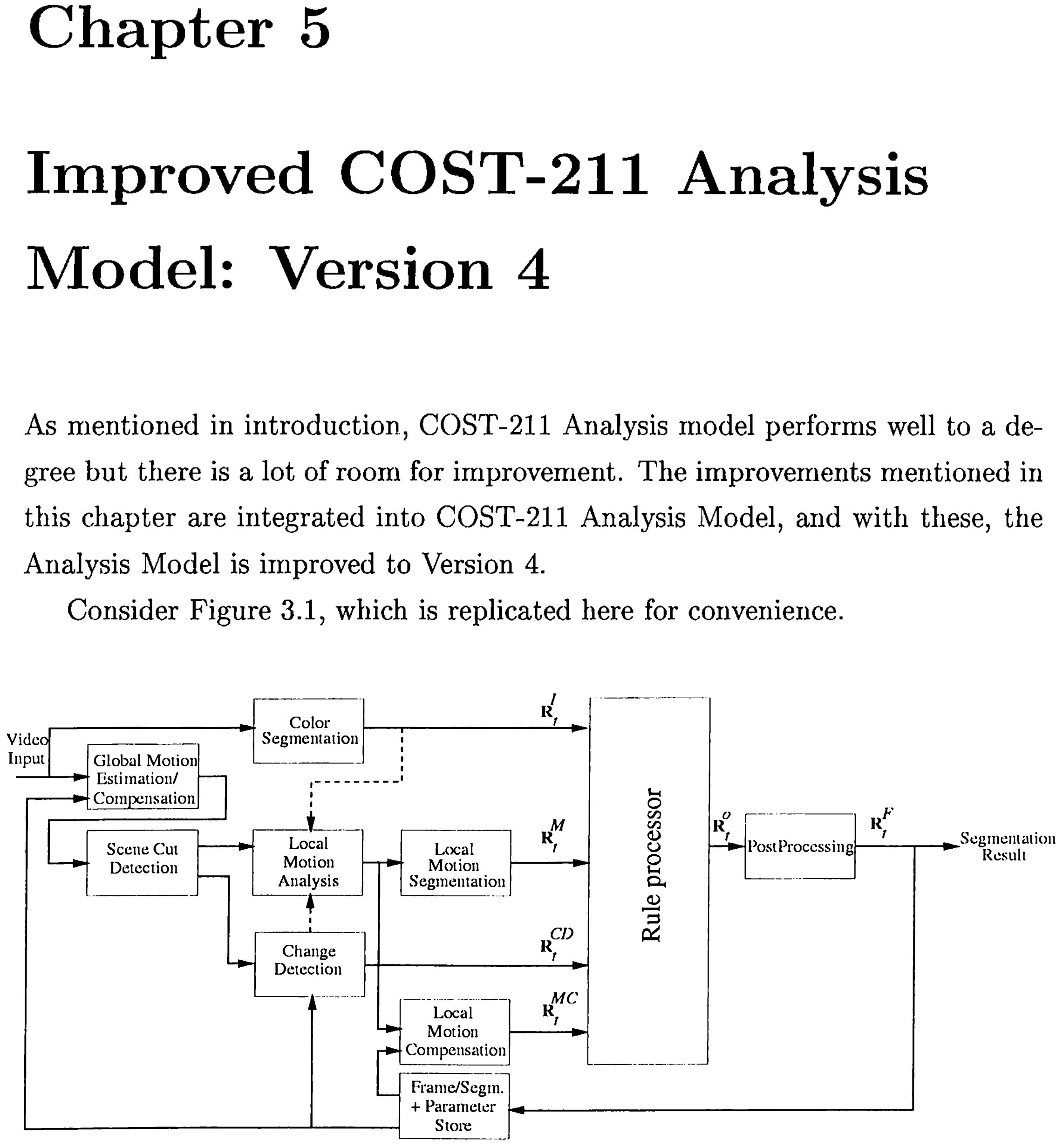

Figure 5.1. COST-211 Analysis Model. (Reprinted as a courtesy of Alatan *et al.* [1])

The improvements that will be discussed in this section come in the form of addition of new blocks, improving existing blocks, and addition of new functionalities to existing blocks.

The first improvement, which can be considered as a new block, is related to the part denoted as "Video input" in Figure 5.1.

5.1 Adaptive Frame Skip and Interpolation

The input to the Analysis Model is a sequence (video). At a time, two frames are fed into the Model. The selection of those frames is done using three parameters: START_FRAME, MAX_FRAME, and FRAME_SKIP[1]. If, for example, the values for these are 0, 299, and 3, respectively, this indicates that the analysis will start at frame 0 and will end at 299, skipping two frames (to get the third) at each step. The situation is depicted in Figure 5.2.

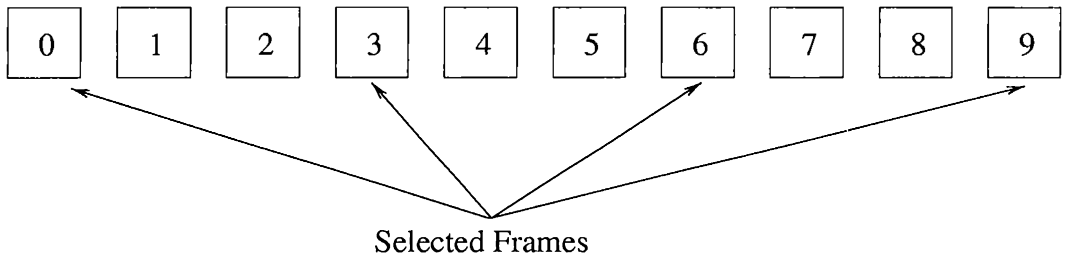

Figure 5.2. Frame selection algorithm currently employed in AM: constant frame skip

This approach may have a major impact on AM performance (especially Mode 2 which relies on motion information to detect objects) in that AM will be less successful in the parts of the sequence with little information (due to lack of sufficient motion information) and it will miss some important frames in the fast-motion parts. To overcome this problem, a new adaptive approach, which will take into account the amount of motion between frames and decide which ones to feed into the AM, is proposed [68] [69]. See Figure 5.3 for a possible scenario.

Initially, the motion between frame 0 and 1 is estimated. If it is found to be below a user-determined threshold, estimation is performed between 0 and 2. Procedure is repeated until the motion exceeds the threshold. AM is invoked with the two frames obtained by this method. The second ("current") frame of this step becomes the first ("reference") frame for next step, and frames subsequent to this new reference frame are checked for motion content. The process continues until all the source frames are considered.

[1]Please refer to Appendix A for a detailed description of the employed parameters.

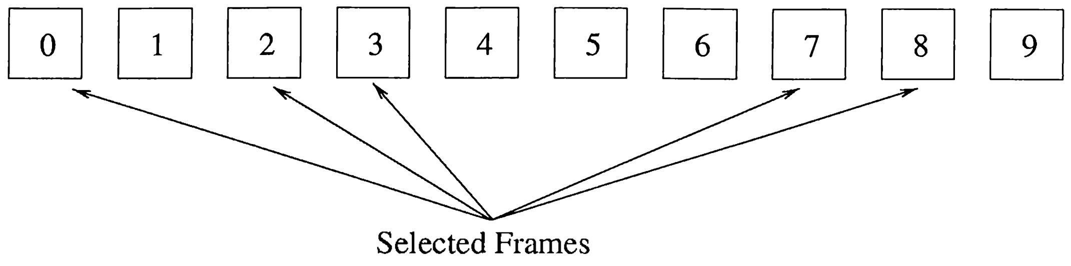

Figure 5.3. Proposed frame selection algorithm: Adaptive frame skip

The motion content can be calculated using any appropriate method. Initially, the motion content was calculated over all image, using an average as indicated in Equation 5.1. The motion estimation is performed using the HBM algorithm in the Model.

$$M^2 = \frac{1}{WH} \sum_{(x,y)\in I} |M_x(x,y)|^2 + |M_y(x,y)|^2 \tag{5.1}$$

where W is the width, H is the height of the image I, M_x and M_y are the horizontal (x) and vertical (y) components of the motion vector at location (x, y). There is no prescribed algorithm to determine the threshold for motion; various M values ranging from 0.125 to 0.5 pixels have been tried (it is observed that values outside this range have caused too many skips or too few skips in the sequences, therefore, experiments have been performed with motion threshold only in this range).

This method proved to be useful in some cases, but its main drawback is that it does not detect frames which contain small (compared to image size) but fast-moving objects, due to the averaging process. To partially handle this situation, another measure, which is based on the number of vectors with magnitude greater than some predetermined threshold is proposed. Motion vectors whose magnitudes are greater than the threshold (which is 2.5 pixels at the moment) are counted and frames are skipped until number of such vectors in the estimated motion field exceeds a threshold (currently 600 for QCIF sequences). This measure has proved to be more reliable, since it ensures that a certain amount of motion exists between two candidate frames; therefore it indicates that there are some objects in the scene.

The skipped frames are interpolated using the 0^{th}-order interpolation. In other words, if we consider the example in Figure 5.3, the masks for frame 1 is the same as the mask for frame 0. Similarly, the mask found for frame 3 is valid for frames

4, 5, 6, and 7. Although this may seem to be inappropriate, it works in practice, since frame skips occur only if there is not sufficient motion. Therefore, using such an interpolation algorithm is not a great deviation from true masks. The only case that the effects of this algorithm will be noticeable is the sequences where motion is small in terms of the quantitative measure (not necessarily measured by the proposed method), but not in terms of the qualitative description. For example, in the sequence "Hall Monitor", one man enters a corridor, drops his bag, and exits from another door. While he drops his bag, he bends down, and up. The motion during this action is small in quantitative terms, therefore a lot of frames are skipped here, and misalignments are observed in the resulting segmentation mask. However, there are no such visible effects in the part of the sequence where he enters the corridor and exits, since the motion is sufficient to detect the man properly. It is clear that the problem is not with the interpolation algorithm, but with the motion metric employed. With a proper motion metric, such effects will not be observed.

Some examples to the improvements achieved by using AFS are demonstrated below using standard MPEG sequences.

Results from "Hall Monitor", "Akiyo" and "Container Ship"

Frames from a total of 300 are selected to display the effects of proposed algorithm. The first set of frames come from the run without adaptive frame rate. Here, a constant frame skip rate of 3 is applied, and masks for skipped frames are again filled with 0^{th}-order interpolation. Therefore, only 100 of the masks are generated by the software. Second set of frames come from the run with adaptive frame skip enabled, with motion vector magnitude threshold 2.5 pixels and count threshold 800. In this selection, only 63 of the frames show up at the output in Hall Monitor. The frames displayed in Figures 5.4 and 5.5 for Hall Monitor are frames 22, 89, 109, 117, 121, 134, 148, 189, 204, 214, 264, respectively. Frames for Akiyo in Figures 5.6 and 5.7 are 58, 61, 64, 103, 115, 127, 139, 180, 194, 207, 220, and for Container Ship (in Figures 5.8 and 5.9) 82, 164, 247, 298. The sequences are all QCIF size (width = 176 pixels, height = 144 pixels).

It is clear from the figures that when proper frames are selected for motion estimation (by AFS algorithm), the objects are detected better than the usual constant frame rate approach. Consider the Hall Monitor sequence in Figures 5.4 and 5.5. Selecting input frames with sufficient motion content yields more reliable

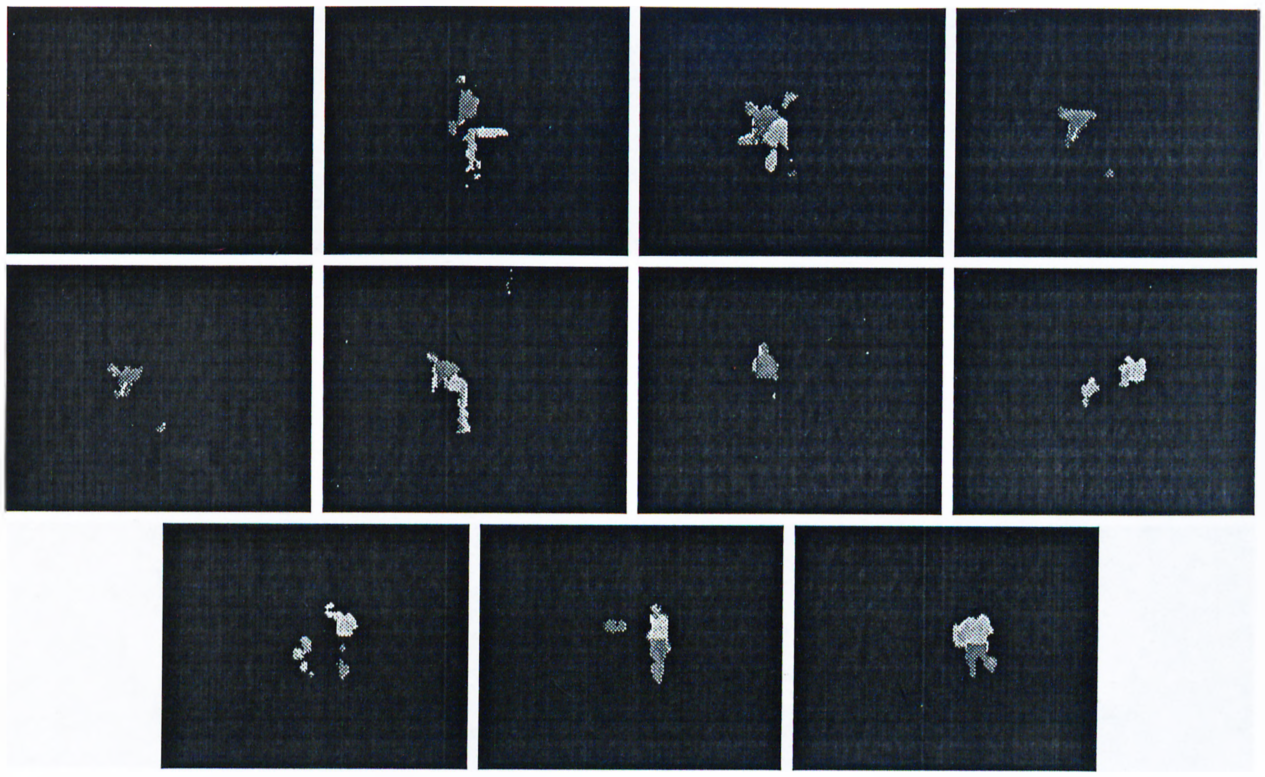
Figure 5.4. Results from AM run without AFS, Hall Monitor

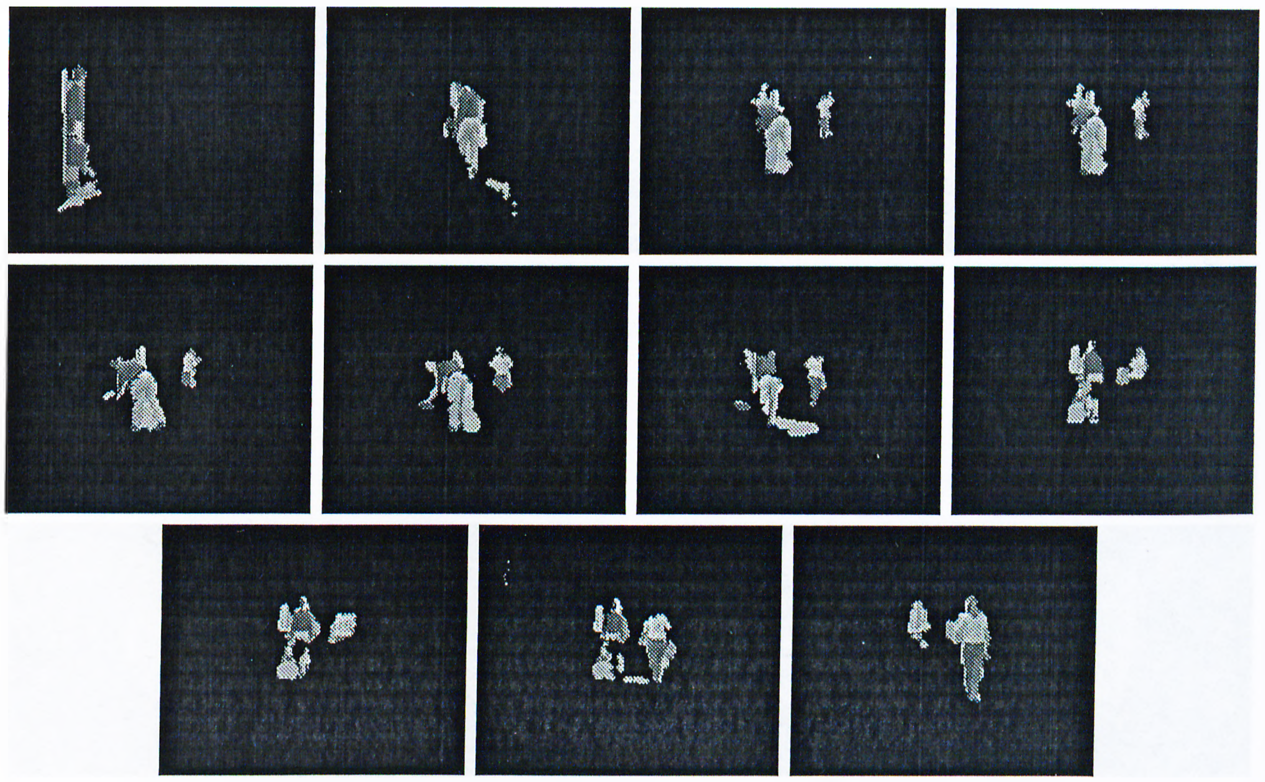
Figure 5.5. Results from AM run with AFS, Hall Monitor

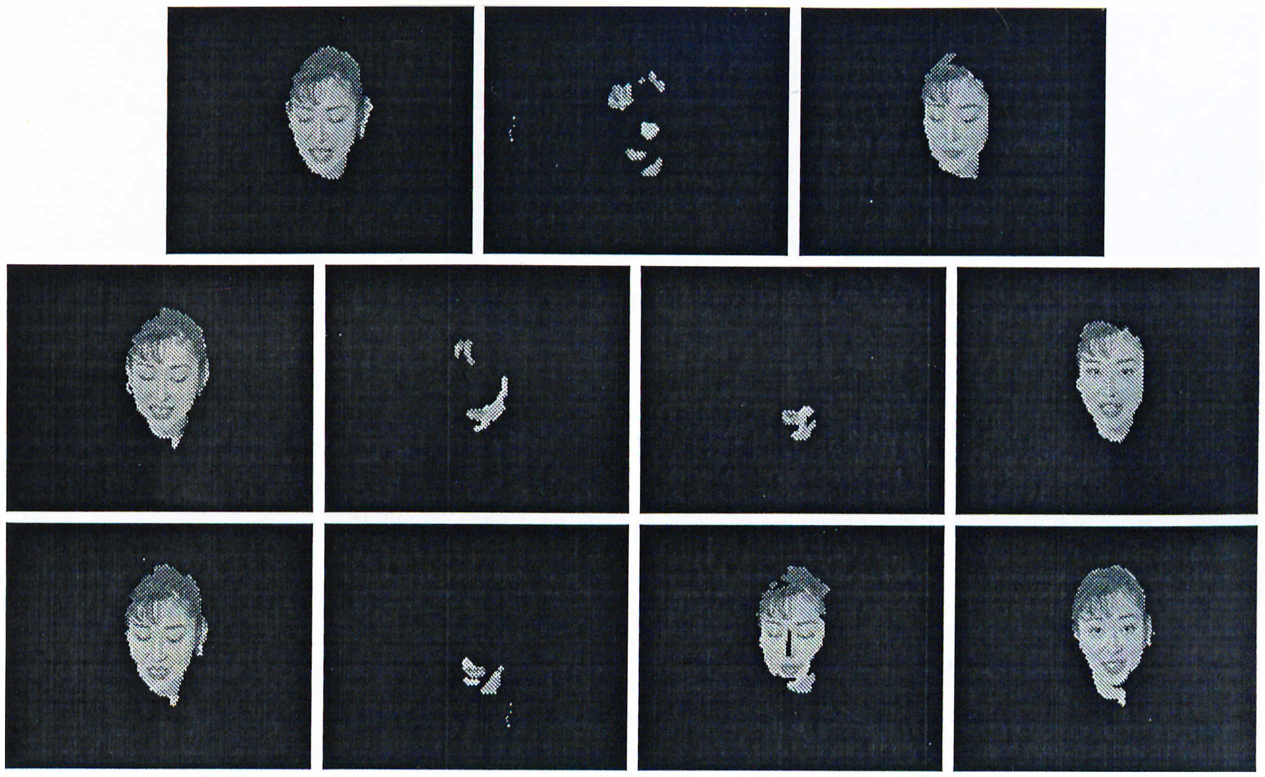
Figure 5.6. Results from AM run without AFS, Akiyo

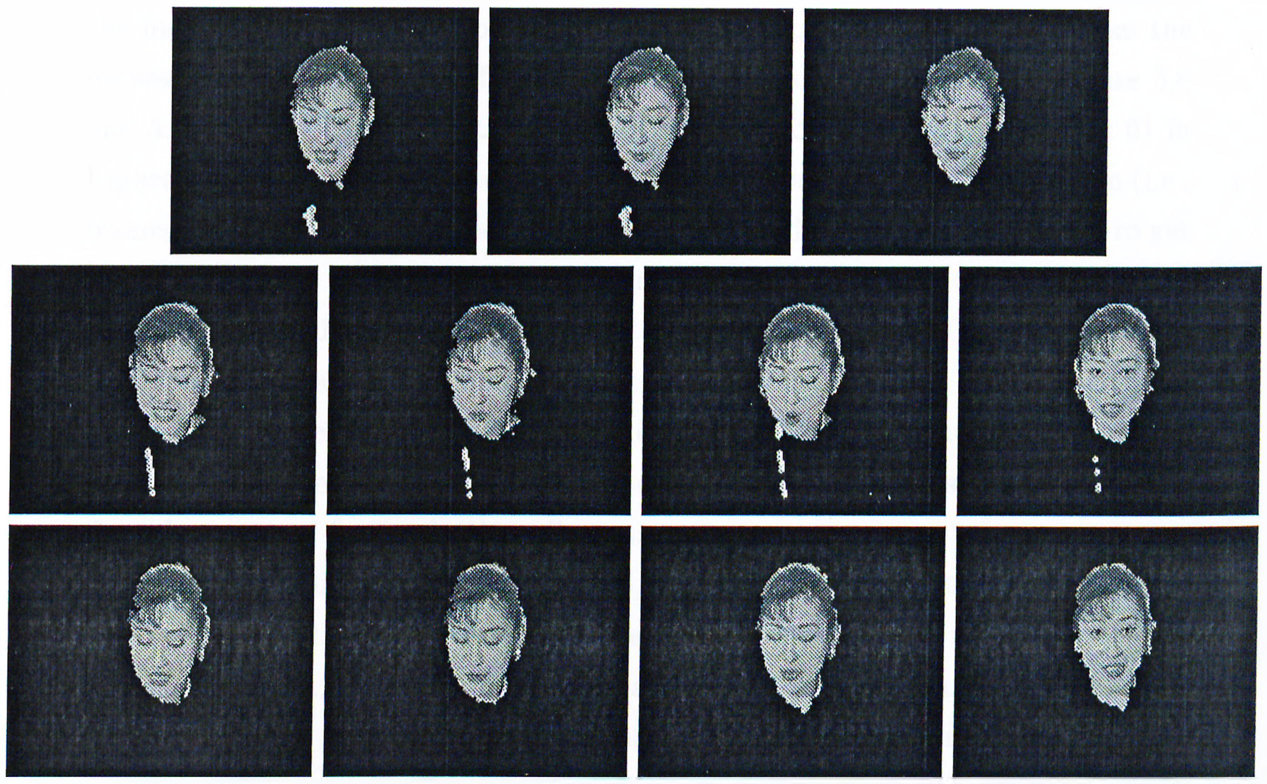
Figure 5.7. Results from AM run with AFS, Akiyo

Figure 5.8. Results from AM run without AFS, Container Ship

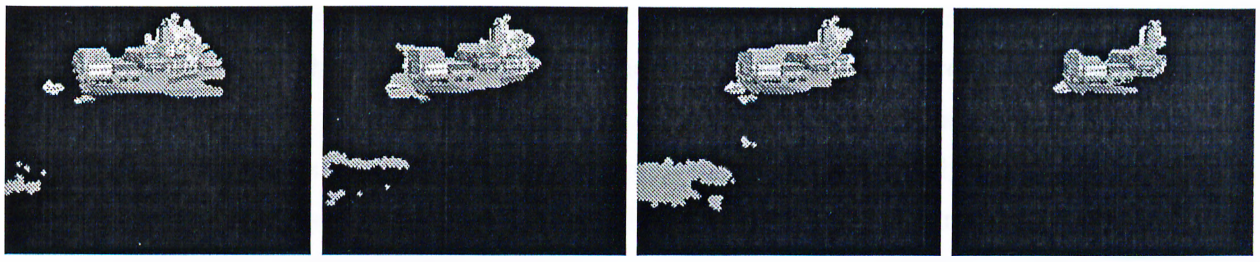
Figure 5.9. Results from AM run with AFS, Container Ship

motion vectors, which, in turn, yield a better motion segmentation mask. Using AFS, both men are extracted properly (Figure 5.5). On the other hand, by using constant frame approach, we end up with segmentation masks in which the men are extracted only partially. The Akiyo sequence also demonstrates the necessity of selecting input frames according to motion content. In Figure 5.6 and 5.7, top row shows masks for frames 58, 61, and 64. To generate mask 61 in Figure 5.6, frames 58 and 61 are utilized in the constant frame skip algorithm (i.e., frames from the input sequence are selected uniformly, skipping two frames to get the third), while in the AFS case (Figure 5.7), frames 48 and 64 are utilized to generate[2] mask 61. Since motion content obtained from 58 and 61 is not sufficient to describe the face properly, the segmentation mask contains only a little part of the face. However, frames 48 and 64 contain sufficient information for proper segmentation, the face is extracted properly. Similar arguments can be made for the other rows of frames in these figures.

One other issue apparent in Figure 5.5 (frame 117, top right) is the misalignment effect mentioned a while ago. Here, the adaptive skip algorithm selects frames 109 and 121 as suitable frames, according to their (quantitative) motion content, and due to 0^{th}-order interpolation, all the segmentation masks from 109 to 120 are the same (in 109, the man has bent down). However, the man bends

[2]Remember, due to 0^{th}-order interpolation, all segmentation masks from 48 to 63 are the same.

up from frame 109 to 120, and the proposed measure fails to notice this change in scene content. This is, as mentioned before, due to the fact that quantitative motion content measured by this approach does not reflect the qualitative content in all cases.

It is not by mistake that Container Ship masks without AFS are empty. Because, by using the constant frame skip algorithm (frame skip is set to 3), nothing is detected in this sequence, due to the insufficiency of motion content of these frames. By experimenting, it has been found out that minimum value of FRAME_SKIP should be 6 for Container Ship sequence in order to detect the objects properly. However, this practice (setting frame skip to a proper value for each sequence) is not possible for two reasons: it is not possible to try various values for every sequence and find which parameter suits a particular sequence, and moreover, such a trial is not desired, even if it were possible. In a fully automatic segmentation algorithm like AM, algorithm parameters should not depend on the inputs. However, parameters are allowed to depend on some *properties* of the input sequence (size[3], for example).

5.2 Sub-Pixel Accurate Motion Vectors

Previous versions of Analysis Model only uses motion vectors with full-pixel accuracy. Version 4 incorporates the use of sub-pixel accurate motion vectors. If sub-pixel accurate motion vectors are requested, input intensity images will be upsampled by a factor of two and linearly interpolated before estimation. Then, resulting motion field will be downsampled by two in order to get the sub-pixel accurate vectors.

Impulse response of the filter used for interpolation is as follows:

$$I = \begin{bmatrix} 0.25 & 0.50 & 0.25 \\ 0.50 & 1.00 & 0.50 \\ 0.25 & 0.50 & 0.25 \end{bmatrix}$$

The effects of using sub-pixel accurate motion vectors are shown in Figure 5.11 (Container Ship) and Figure 5.13 (Hall Monitor). As seen in Figure 5.11 that the contribution of half-pixel accurate vectors is detecting one more segment of the

[3]It is currently allowed in Analysis Model that parameters can be specified differently for different sequence sizes (QCIF or CIF). See Appendix A for further details.

ship. An (unwanted) addition is detection of more of the waves, which were not detected previously.

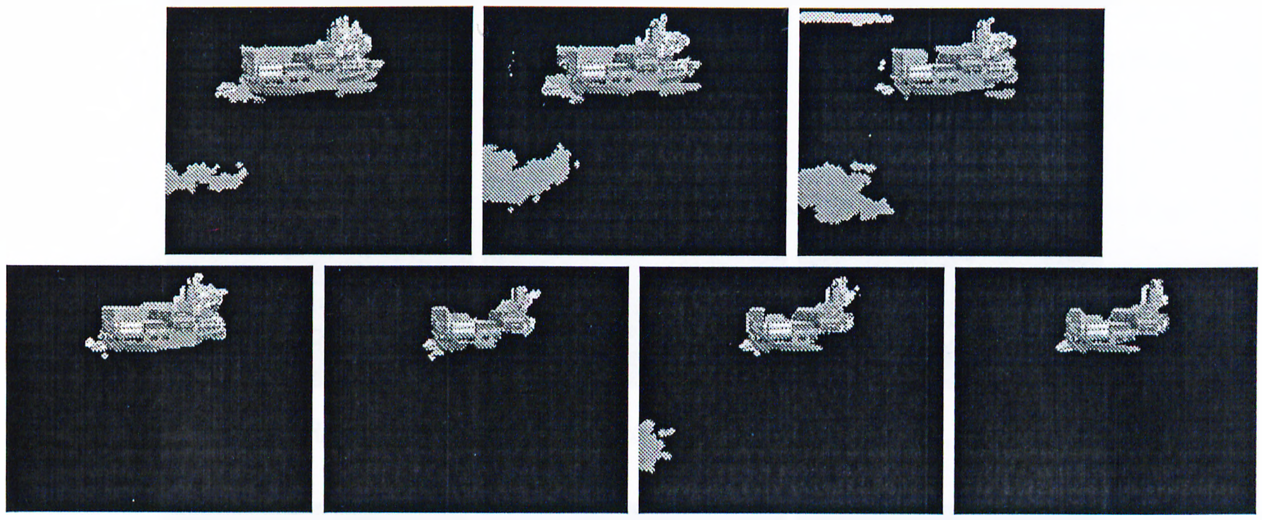
Figure 5.10. Results with full-pixel accurate motion vectors, Container Ship

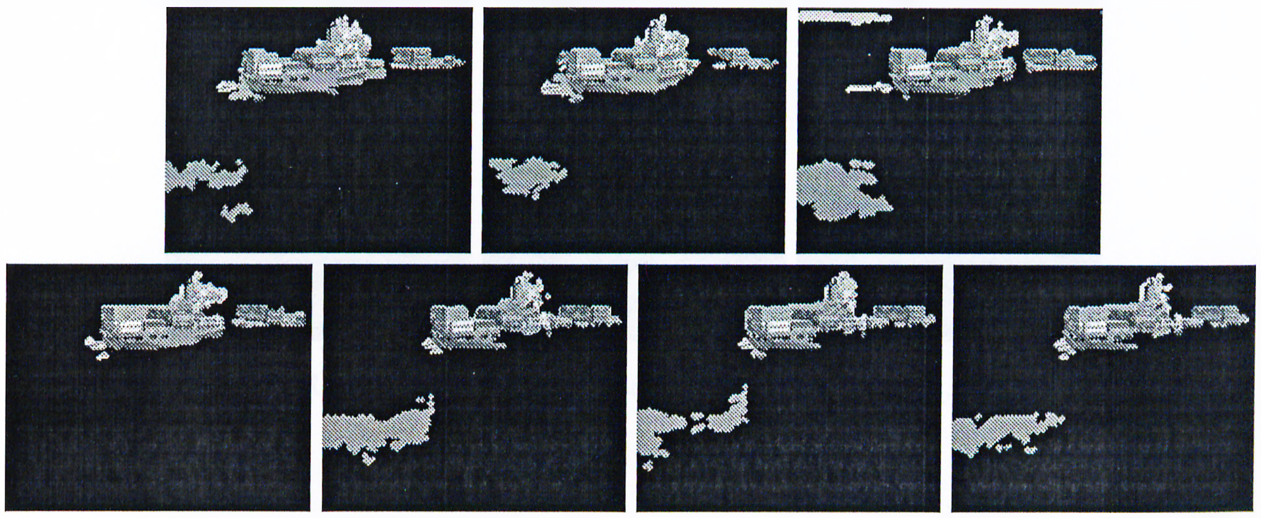
Figure 5.11. Results with sub-pixel accurate motion vectors, Container Ship

A similar improvement is observed in Hall Monitor. The persons' shape are more accurate compared to the case of full-pixel accurate motion vectors.

For Container Ship, frames 121, 152, 200, 215, 281, 290, and 298 are displayed. Hall Monitor frames are 86, 91, 182, 203, 213, and 222. Adaptive frame skip was also utilized in obtaining the results shown in Figures 5.10, 5.11, 5.12, and 5.13. We do not consider the cases without the adaptive frame skip, due to the problems encountered in selecting the frame skip parameter properly. Therefore, all of the results demonstrated from this point on will be the results obtained through the

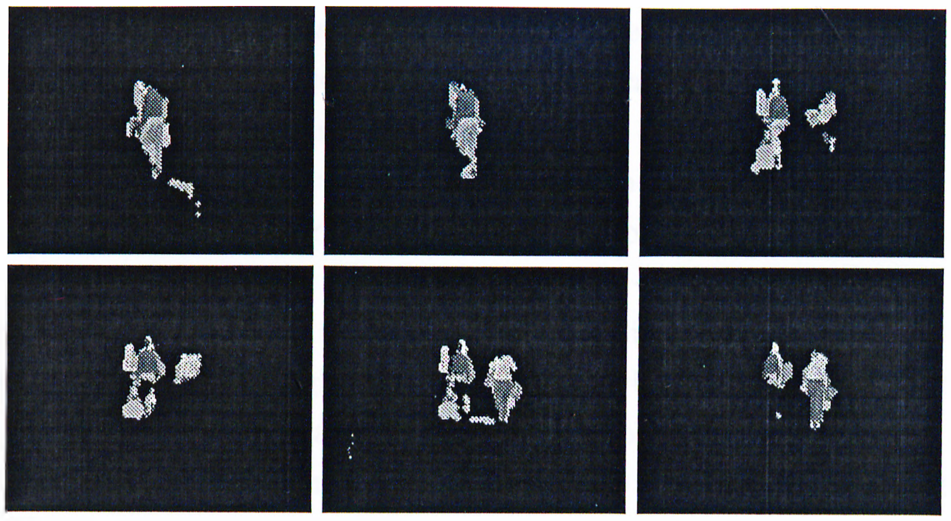
Figure 5.12. Results with full-pixel accurate motion vectors, Hall Monitor

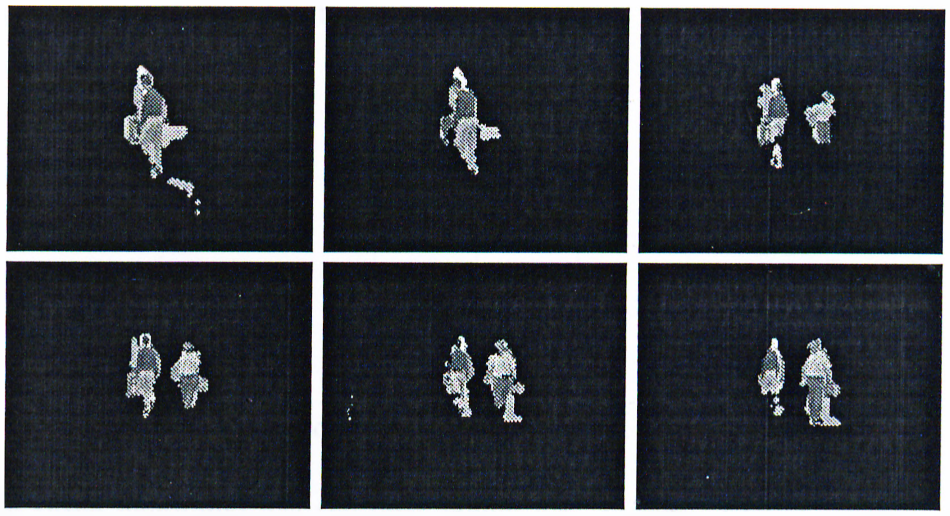
Figure 5.13. Results with sub-pixel accurate motion vectors, Hall Monitor

use of AFS.

5.3 Local Motion Compensation

AM Version 4 has sub-pixel accurate motion vectors, therefore the motion compensation routine which takes only full-pixel accurate motion vectors as input (along with the previous segmentation mask) has to be upgraded to accept motion vectors with sub-pixel accuracy.

Consider an example in which the pixel we try to compensate is at location (1,3) and the motion vector[4] corresponding to it is (1.25,–0.75), i.e., horizontal

[4]Although only half-pixel accurate motion vectors are employed in AM Version 4, the module

displacement is 1.25 pixels, and vertical displacement is 0.75 pixels. If this displacement were integer valued, such as (1, −1), there would be no need for a new rule for motion compensation, since the value of the compensated mask at (1,3) would simply be the value of the pixel at location (2,2) in the resulting segmentation mask. With non-integer valued motion vectors, a new rule should be devised.

The proposed method is a non-linear interpolation technique and it involves taking into account the neighboring pixel values while deciding on a particular pixel's value in the compensated mask. The neighbor selection algorithm works as follows: The pixels are considered to consist of 9 "regions", as in Figure 5.14. According to the region the motion vector is pointing to, three neighbors, which effects the value of the compensated pixel are chosen (in addition to the "main" pixel the motion vector points).

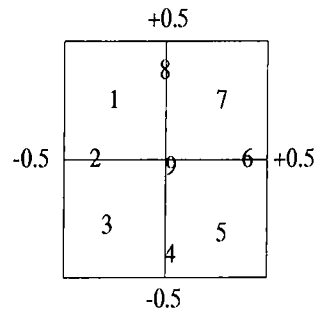

Figure 5.14. A "zoomed" view of a pixel and its "regions"

The three neighbors are the ones which are "closest" to the "region" under consideration: if, for example, motion vector points to region 7 (which is the case in our example), then neighbors on the right, top right, and the top neighbor are utilized in the decision process. (or, if the vector pointed to region 2, the neighbors on bottom left, left, and top left would be utilized).

After the neighbors are selected, following rules are applied, to get the "votes" from the neighbors and the main pixel:

- Each neighbor vote counts as one, main pixel vote counts as two.
- If the motion vector points to region 9, then the neighbors are considered to be the main pixel.

is developed such that it will handle motion vectors with more accuracy, keeping in mind such motion vectors might be implemented in a future version of AM.

- The value that has the largest count (i.e., the one that gets the maximum number of votes) becomes the value of the compensated pixel.

A corollary of rule 2 is that if the motion vectors are integer-valued, the algorithm reduces to the original motion compensation algorithm.

Now consider Figure 5.15 which contains an example to the ideas developed so far. With the pixel under discussion being (1,3) and its corresponding motion vector (1.25, -0.75), the pixels (3,2), (3,3), and (2,3) are selected as neighbors, and (2,2) becomes the main pixel. Assume further that pixels (2,2) and (3,2) have a value of 2 (denoting this pixel belongs to object 2), pixels (2,3) has a value of 1, and (3,3) has a value of 4. This indicates that the value 1 gets 1 vote, value 2 gets 3 votes, and value 4 gets 1 vote. Therefore it is decided that the value of the pixel (1,3) in the compensated mask is 2. This procedure may be called "weighted mode filtering."

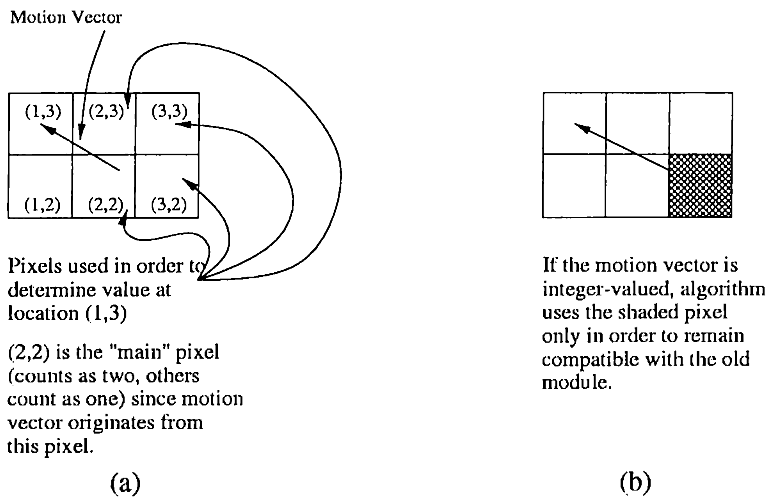
Motion Vector
(1,3)
(2,3)
(3,3)
(1,2)
(2,2)
(3,2)
Pixels used in order to determine value at location (1,3)
(2,2) is the "main" pixel (counts as two, others count as one) since motion vector originates from this pixel.
(a)
If the motion vector is integer-valued, algorithm uses the shaded pixel only in order to remain compatible with the old module.
(b)

Figure 5.15. (a) How neighbors are selected. (b) If integer valued vectors are used, algorithm executes in a manner compatible with the old module.

5.4 Local Motion Analysis

A new motion estimation algorithm is introduced as an alternative to the block-matching algorithm already used in AM. Block-matching algorithms generally give acceptable results with quite low computational demand. The new algorithm yields much better results, but is computationally more demanding.

The underlying theory for this method is similar to that of [54]: MRF's (Gibbs energies) are utilized for modeling the constraints for motion field and segmentation labels [70] [71] [72]. One difference is that to reduce the computational complexity, a segmentation estimate is input to the algorithm, and is not changed throughout the iterations. This approach is not optimal, however, it greatly reduces the execution time. This initial estimate comes from the color segmentation, with number of regions decreased. Another difference is that, *temporally unpredictable (TU)* regions are also incorporated into the Gibbs energy formulation [70] [73]. *TU* areas are regions which can not be predicted through motion compensation.

The constructed Gibbs energy function is formulated as follows [70]:

$$U(D, S|I_t, I_{t-1}) = U_n + \lambda_m U_m + \lambda_s U_s \tag{5.2}$$

where

$$\begin{aligned} U_n &= \sum_{\mathbf{x}} (I_t(\mathbf{x}) - I_{t-1}(\mathbf{x} - D(\mathbf{x})))^2 (1 - S(\mathbf{x})) + S(\mathbf{x}) T_s & (5.3)\\ U_m &= \sum_{\mathbf{x}} \sum_{\mathbf{x_c} \in \eta_{\mathbf{x}}} ||(D(\mathbf{x}) - D(\mathbf{x_c})||^2 \, \delta(R(\mathbf{x}) - R(\mathbf{x_c})) & (5.4)\\ U_s &= \sum_{\mathbf{x}} \sum_{\mathbf{x_c} \in \eta_{\mathbf{x}}} [1 - \delta(S(\mathbf{x}) - S(\mathbf{x_c}))] & (5.5) \end{aligned}$$

$\mathbf{x}$ denotes a group of pixels (or a single pixel, according to the resolution). $\mathbf{x_c}$ is a neighbor of $\mathbf{x}$, which lies in the neighborhood $\eta_{\mathbf{x}}$. $D(\mathbf{x})$ is the motion field defined at each $\mathbf{x}$ on frame I_t and show the displacement from its corresponding point on frame I_{t-1}. The U_n term forces motion vectors to be selected as to minimize intensity differences between corresponding $\mathbf{x}$'s in I_t and I_{t-1}. T_s term does not allow pixels in *TU* areas to be considered for a match. The U_m term penalizes the dissimilarity of motion vectors in the same region (i.e., in the same object). This region map is denoted by R and is initialized with a color segmentation result containing less number of regions (currently 25). It does not change throughout the iterations. S is the binary field which shows the TU regions. U_s term forces S field to consist of regions instead of individual points. Single points and unwanted shapes (cross, for example) are eliminated through this term.

Minimization of this energy function U is the maximization of the corresponding *a posteriori* probability. To minimize this non-convex energy function, *Iterated Conditional Modes* [66] have been used. In each iteration, the algorithm tries to find the values for motion field components and TU area labels which decrease the total energy U. Therefore there exists the probability of getting stuck at a local minimum. Some good initial estimates are required to initialize the algorithm.

The good initial estimates come inherently from the implementation of the method: the method has been designed to work in a multi-resolution manner. The iterations start with the coarsest analysis and progress on a finer scale at each level, passing the results of coarser level as initial estimates to the finer level. Currently, the coarsest level employed in AM is 3, while finest level[5] is 0. The resolution, which is determined according to the level, is calculated using the following equation:

$$Resolution = 2^{Level} \tag{5.6}$$

So at level 3, the resolution is 8×8, while at level 0, it becomes 1×1 (i.e., pixel level resolution). Actual minimization is done by the iterations at the finest level, the rest are "the initializations" to prevent getting stuck at a local minimum.

Search range for the iterations at each level is determined according to following equation:

$$Search\ Space = Level * VecCons_1 + VecCons_2 \tag{5.7}$$

The value for $VecCons_1$ and $VecCons_2$ are both set to 3. So at level 3, *Search Space* =12, therefore, in each step of iteration, for each $\mathbf{x}$, 25 different values (from $-$*Search Space* to *Search Space*) are searched for the value that minimizes the respective energy function. At level 0, *Search Space* =3, so at pixel level, 7 different values are checked for minimum.

The coefficients (relative weights) of individual energy terms are set to following values: $\lambda_m = 100$, $\lambda_s = 100$, $T_s = 20$. These coefficients have been determined in an *ad-hoc* manner, as in the case of [54]. It has been observed, however, that with these coefficients, satisfactory results are obtained for a wide range of sequences.

The number of iterations is adaptively determined: the iterations continue until either the limit is reached or the difference between two consecutive energy values fall below a user-determined threshold (currently it is 0.25%).

Achieved improvements are displayed in sequences Hall Monitor and Container Ship. Frames 96, 102, 128, and 138 of Hall Monitor are displayed. For Container Ship, the frames are 25, 50, 82, 103, 136, 154, 174, 228, and 257.

[5]When sub-pixel accurate vectors are required, the upsampled frames are analyzed at the coarsest level of 4; accordingly finest level becomes 1.

In Hall Monitor, the second man entering the corridor is properly detected. In Container Ship, the ship is found correctly most of the time.

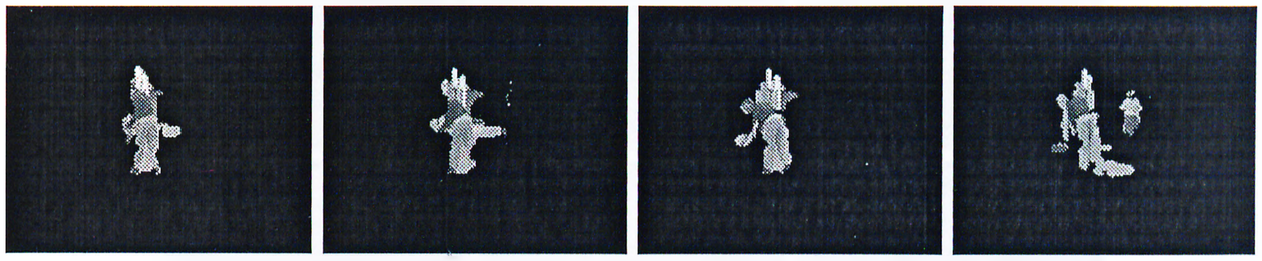
Figure 5.16. Results obtained by using HBM, Hall Monitor

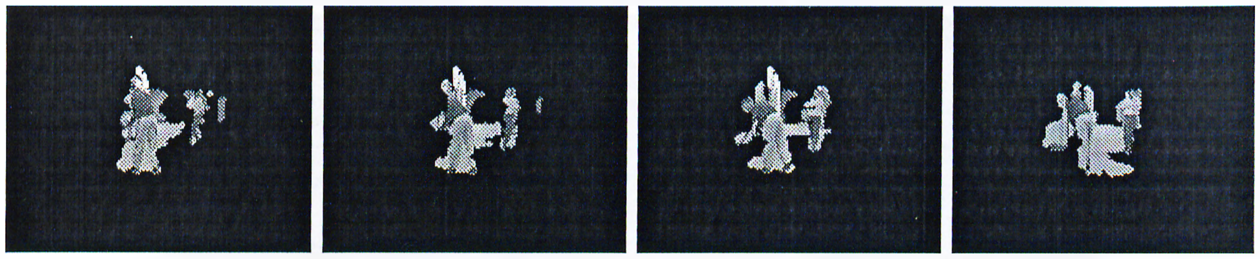
Figure 5.17. Results obtained by using Gibbs-based algorithm, Hall Monitor

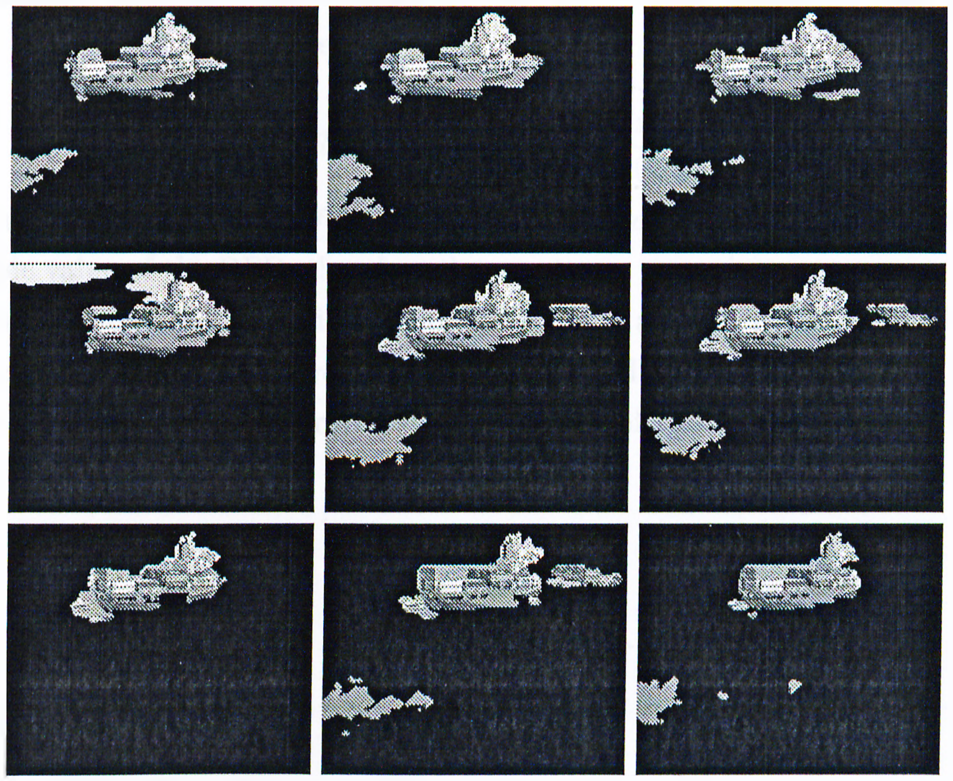
Figure 5.18. Results obtained by using HBM, Container Ship

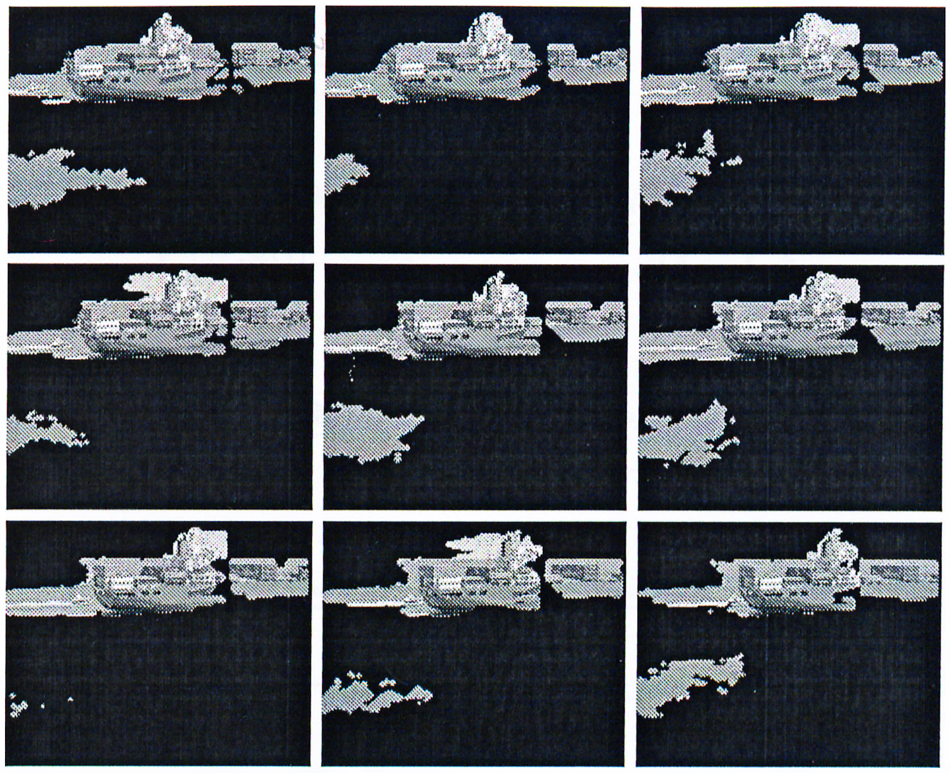
Figure 5.19. Results obtained by using Gibbs-based algorithm, Container Ship

Another measure of performance is the PSNR value, based on mean square error. The PSNR is calculated as

$$PSNR = 10\ log_{10} \frac{\sum_{x,y} |Ref(x,y)|^2}{\sum_{x,y} |DFD(x,y)|^2} \tag{5.8}$$

The image indicated by *Ref* is the *reference image.* It indicates the maximum possible value of the DFD, therefore the worst case scenario.

Hall Monitor and Container Ship PSNR plots are in Figure 5.20 and 5.21, respectively. Gibbs plots are, on the average, 3 dB above compared to HBM plots; this indicates an improvement by a factor of two in the displaced frame difference.

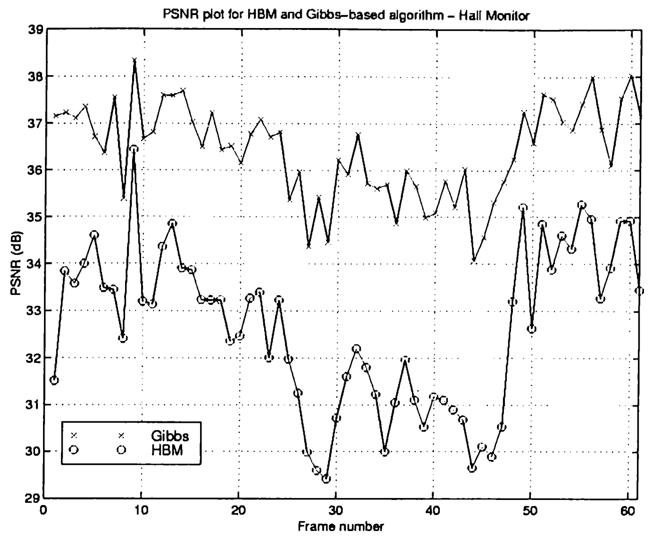

Figure 5.20. PSNR plot of Hall Monitor

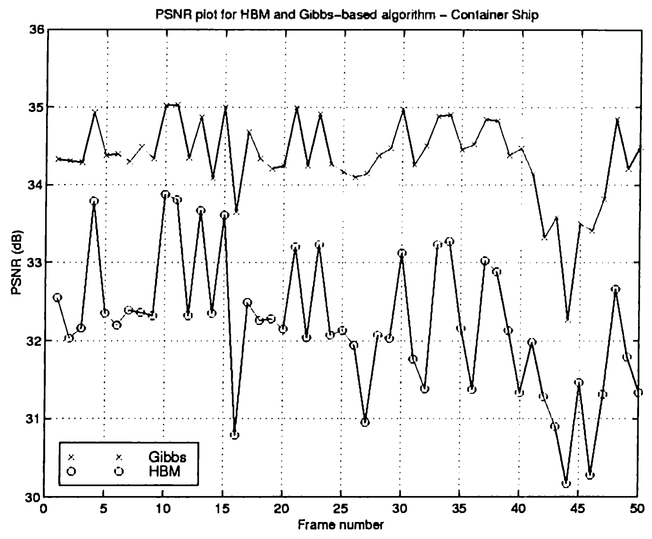

Figure 5.21. PSNR plot of Container Ship

5.5 Local Motion Segmentation

A segmentation algorithm based on the affine motion model is added to the existing "Local Motion Segmentation" block in Figure 5.1. Currently, the motion model employed in the Analysis Model is the "translational" motion model.

Model based motion segmentation can be viewed as a *surface-fitting* process: the surfaces are generated by the horizontal and vertical component of the estimated dense motion field. Each component is a grey-scale image and can be considered as a piece-wise smooth 3-D surface. Segmentation of the motion field is then extraction of these smooth regions.

The objects in the real world generally make rigid 3-D motion and projection of this motion onto 2-D plane yields the observed motion which can be described by various parametric models. Therefore segmentation of motion field actually corresponds to finding regions for which we can find a set of parameters to explain the observed motion using a particular model.

In translational model, motion vector components for each pixel in each region can be described as follows:

$$v_x = a_1 \tag{5.9}$$
$$v_y = a_2 \tag{5.10}$$

where a_1 and a_2 are constants within a region. This model is sufficient if the only motion in the scene is translation. If the motion in a region is translation, the motion vectors in that region will be similar to each other, and using this model, such regions can be extracted. However, when the scene contains other types of motion (such as rotation), this model becomes insufficient.

The newly introduced affine (6-parameter) motion model describes a 2-D motion field due to the rigid 3-D motion of a planar surface. This model reduces to the translational model when four of the parameters are zero, so translational model can be considered as a special case of the proposed model. Through our surface-fitting viewpoint, we can say that translational model fits "constants" to the estimated motion vector field, while affine model fits "planes".

Using affine motion model, motion vector components (v_x, v_y) at each pixel (x, y) in a region are given by

$$\begin{aligned} v_x &= a_1 x + a_2 y + a_3 \\ v_y &= a_4 x + a_5 y + a_6 \end{aligned} \tag{5.11}$$

Here, a_i's denote the motion parameters of pixel at location (x, y). Since it is impossible to find a unique set of parameters for each individual pixel (6 unknowns vs 2 equations), the algorithm initially "segments" the motion field into 2×2 regions, and for each region it calculates the mentioned 6 parameters. Since this is an overdetermined system of equations (6 unknowns and 8 equations), the calculations are done in a least-squares sense. The parameters which minimize the "distortion" D for a given region R are assigned as that region's parameters:

$$D = \sum_{(x,y)\in R} ||\mathbf{v}(x,y) - \hat{\mathbf{v}}_R(x,y)||^2 \qquad (5.12)$$

Here, $\mathbf{v}$ is the estimated motion field, while $\hat{\mathbf{v}}_R$ is the motion field generated using the affine motion model parameters of region R, using Equation 5.11.

This segmented image is accepted as the starting point for the actual segmentation. The method used to segment is again RSST, however, the distance measure has been modified [17]. Now, RSST tries to select the two neighboring regions which result in smallest *increase* in distortion. That is, for all neighboring region pairs (R_i and R_j), the non-negative quantity ΔD is calculated [17]:

$$\begin{aligned} D_{ij} &= \sum_{(x,y)\in R_{ij}} ||\mathbf{v}(x,y) - \hat{\mathbf{v}}_{R_{ij}}(x,y)||^2 && (5.13) \\ D_i &= \sum_{(x,y)\in R_i} ||\mathbf{v}(x,y) - \hat{\mathbf{v}}_{R_i}(x,y)||^2 && (5.14) \\ D_j &= \sum_{(x,y)\in R_j} ||\mathbf{v}(x,y) - \hat{\mathbf{v}}_{R_j}(x,y)||^2 && (5.15) \\ \Delta D &= D_{ij} - D_i - D_j && (5.16) \end{aligned}$$

The two regions which result in smallest ΔD are merged. In other words, in each step, the algorithm tries to make the increase in D as small as possible, so it joins the two regions which yield the smallest ΔD. Here, $R_{ij} = R_i \bigcup R_j$.

For this new block, results will not be demonstrated on standard MPEG sequences, but on a specific sequence which has planar rotations and various non-translational motion in it. "Ertem" sequence consists of 100 frames and is of QCIF size. The displayed results will not be regular video frames, but segmentation masks which show the detected objects. In these masks, different colors indicate different objects. The aim is to demonstrate that translational motion model can explain some motion by a few regions, but with affine motion model, this motion can be described by a single region, which means that object is correctly detected.

The frames 33, 60, 62, 68, 70, 77, 90, and 94 are displayed. Notice the oversegmentation in frames in the Figure 5.22. However, in Figure 5.23, those regions make up one single region, which is the actual object.

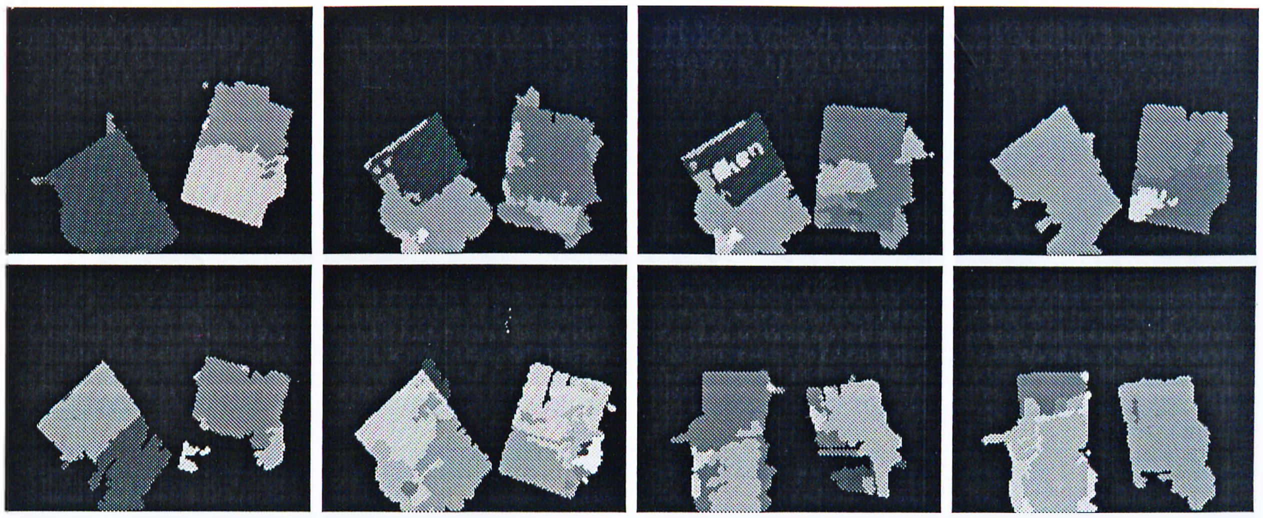
Figure 5.22. Results with translational motion model, Ertem sequence

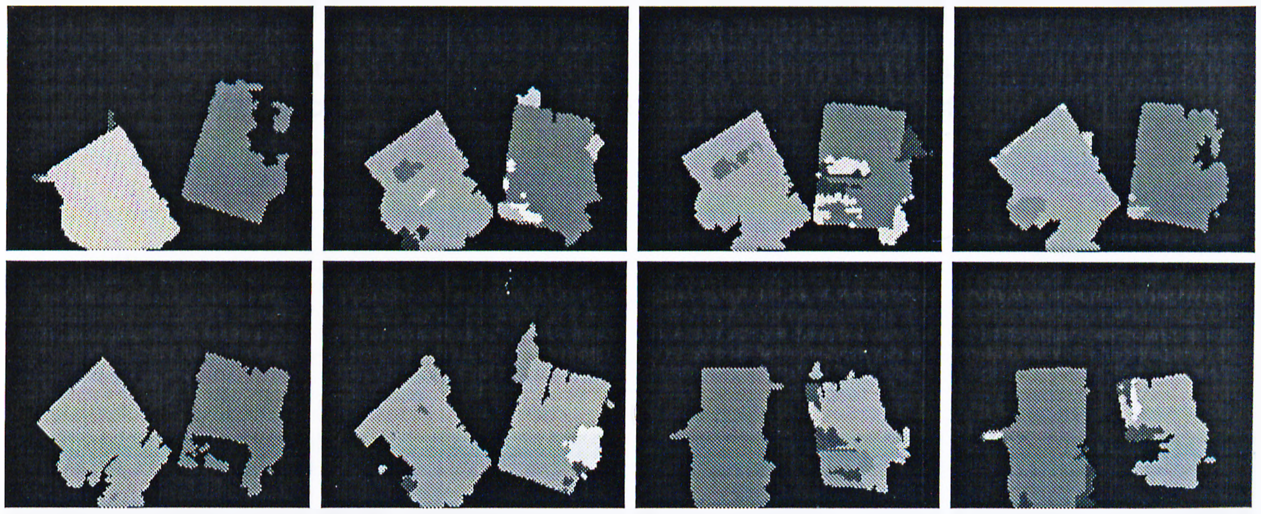
Figure 5.23. Results with affine motion model, Ertem sequence

The results in Figures 5.22 and 5.23 may be regarded as "test results" because they do not utilize any of the mentioned improvements (AFS, sub-pixel accurate vectors, or the new motion estimation method). They only demonstrate what the affine motion model can achieve, even if other improvements are not utilized. The comparative results in Figures 5.24 and 5.25 include the AFS, sub-pixel accurate motion vectors, and the new motion estimation algorithm; therefore better results are obtained. In Figure 5.24, we see the improvements from AFS, sub-pixel accurate vectors, and the new motion estimation method, without the affine model. The affine model shows itself in Figure 5.25, where object boundaries are detected more accurately.

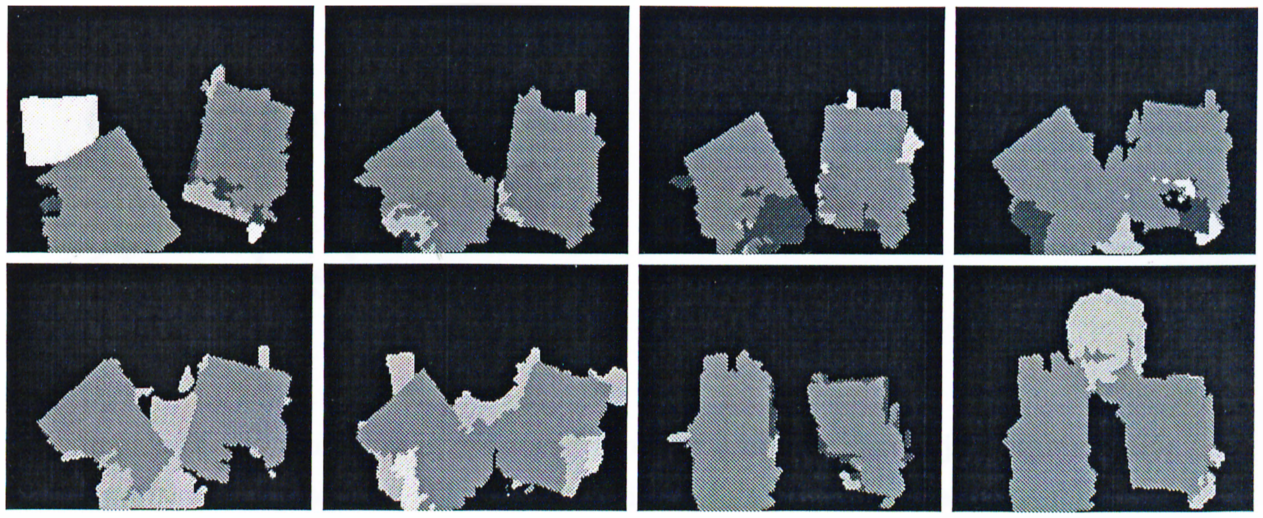
Figure 5.24. Results with translational motion model, Ertem sequence (using AFS, sub-pixel accurate vectors, and the Gibbs-based motion estimation method)

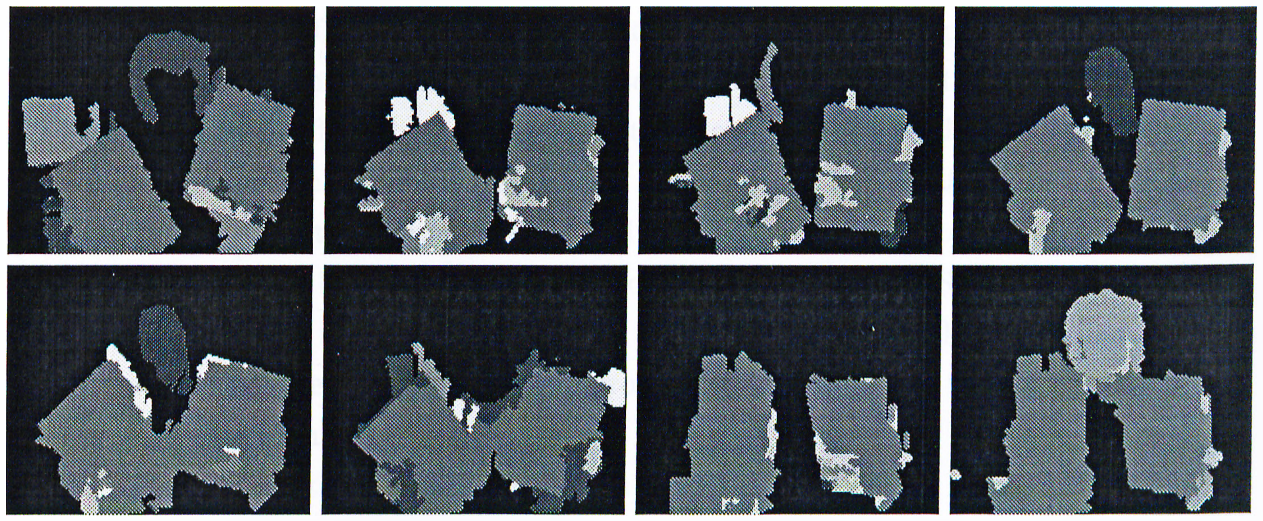
Figure 5.25. Results with affine motion model, Ertem sequence (using AFS, sub-pixel accurate vectors, and the Gibbs-based motion estimation method)

Chapter 6

Conclusions

In this thesis, a number of improved tools for a novel video object segmentation algorithm are proposed. The algorithm is modular, so it has been possible to replace a module by another one with similar functionality but better performance.

The proposed improvements include 1) a new frame rate approach (variable frame rate) which enables the algorithm to select its input frames adaptively according to motion content for better segmentation results, 2) the introduction of a powerful motion estimation algorithm for more reliable motion vectors, 3) incorporation of a new motion model which can explain more complex motion types than the translational model can do, and 4) the addition of sub-pixel accurate motion vectors for more precise motion field.

The variable frame rate approach is the most important improvement among the proposed improvements. If the motion content implied by the selected input frames is not sufficient, the objects in the scene will not be detected properly. When the motion content is insufficient, the use of the sub-pixel accurate motion vectors, or the other improvements will not help a lot in detecting the objects accurately. The best example for this case is the Container Ship Sequence, in which no object is detected if variable frame rate is not employed. When the frames are selected properly, each improvement contributes to the segmentation process. See Figure 6.1.

The contributions from other improvements depend on the scene. Not every sequence will benefit from every improvement. For example, sub-pixel accuracy will contribute in the parts of the scene where the motion content can not be fully described by integer-valued motion vectors. The Analysis Model will benefit from new motion estimation algorithm in the cases where the motion in the scene can not be described correctly by blocks. The affine motion model will only be useful in the sequence "Ertem" or alike that contain non-translational planar motion. Consider the examples in Figures 6.1 and 6.2. See Figure 6.3 for an example to affine motion model discussion.

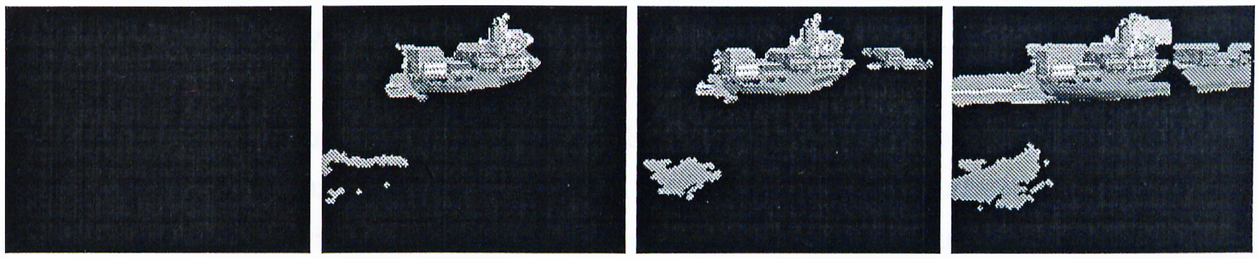
Figure 6.1. Results from Container Ship Sequence runs. Left to Right: original, with AFS, with AFS + sub-pixel, with AFS + sub-pixel + Gibbs

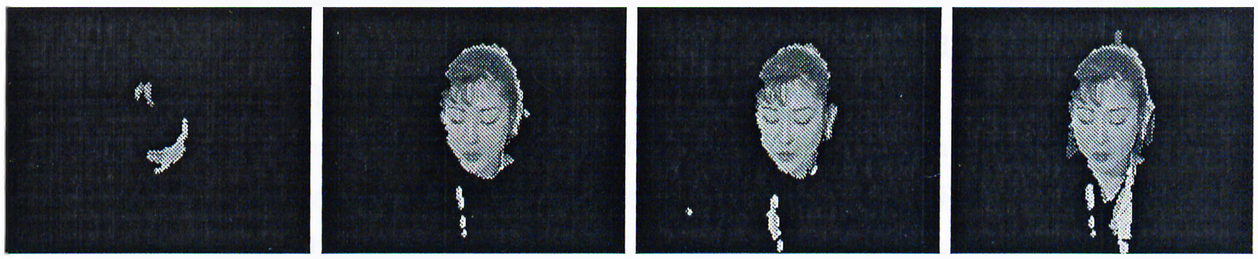
Figure 6.2. Results from Akiyo runs. Left to Right: original, with AFS, with AFS + sub-pixel, with AFS + sub-pixel + Gibbs

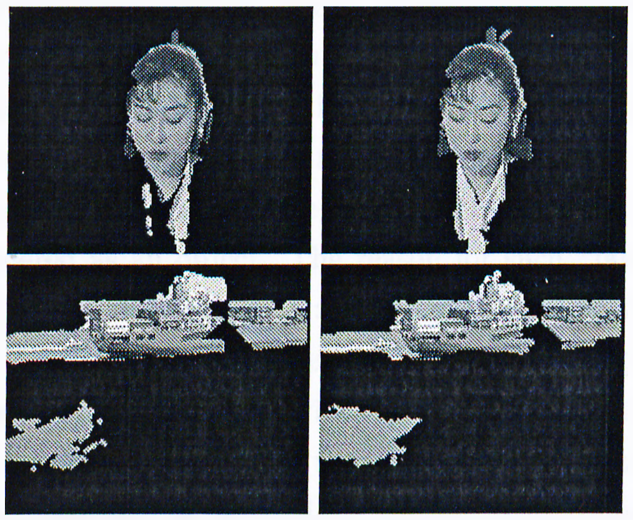
Figure 6.3. Results from Akiyo and Container Ship. Left: results obtained by using AFS, sub-pixel, and Gibbs. Right: results obtained by using AFS, sub-pixel, Gibbs, and affine model.

In Figure 6.1, we see that the use of sub-pixel accurate vectors leads to the detection of areas where the motion is less than full pixel. Also observed in this figure is the existence of areas for which the block-matching algorithm failed to find any motion[1]. The new motion estimation algorithm, which gives a dense vector field, correctly finds the motion vectors in such areas, and the ship is detected properly.

In Figure 6.2, we see an example in which the improvements do not contribute significantly. It may be concluded for this sequence that sub-pixel accuracy did not contribute much because the motion content in the scene is already described well by the full pixel accurate motion vectors, and the new estimation algorithm did not contribute a lot because the block motion vectors were mostly sufficient to describe the motion content in the scene.

The results in Figure 6.3 show that when the sequence does not contain non-translational planar motion, the contribution from affine model is not significant. The results on the left column and on the right can be considered the same as far as semantic objects are concerned.

A further future improvement could be determining the motion content for each object separately. This idea is similar to the Video Object Planes (VOP's) in MPEG-4: each object in the scene is a VOP and is coded separately. Final image is generated by laying the VOP's onto each other (and onto the background). In AM, calculating the motion content separately for each individual object ensures proper detection and tracking. One important prerequisite for this approach to work as intended is the detection of objects correctly in the previous frame.

Another observation related to the rule processing is that the lack of occlusion area analysis leads to false object detection. Consider the example in Figure 6.4.

In the first frame of Figure 6.4, the man has been detected as an object. In subsequent frames, the man walks to the right but the uncovered area behind the man is still considered as an object. The reason for such "false objects" is the errors in the motion estimation stage [5]: although there is no true match for the uncovered areas of current frame in the previous frame, those uncovered areas are matched to some other parts of previous frame incorrectly and this results in non-zero motion vectors for these areas; therefore they are treated as if they were real objects. In the next step of the analysis, motion estimation results with zero

[1]In those areas, representing the motion in a 4×4 block with (0,0) vector yields less error compared to the cases with a non-zero vector. Therefore, that block's motion vector is decided to be (0,0) and as a consequence, no object is detected in those areas.

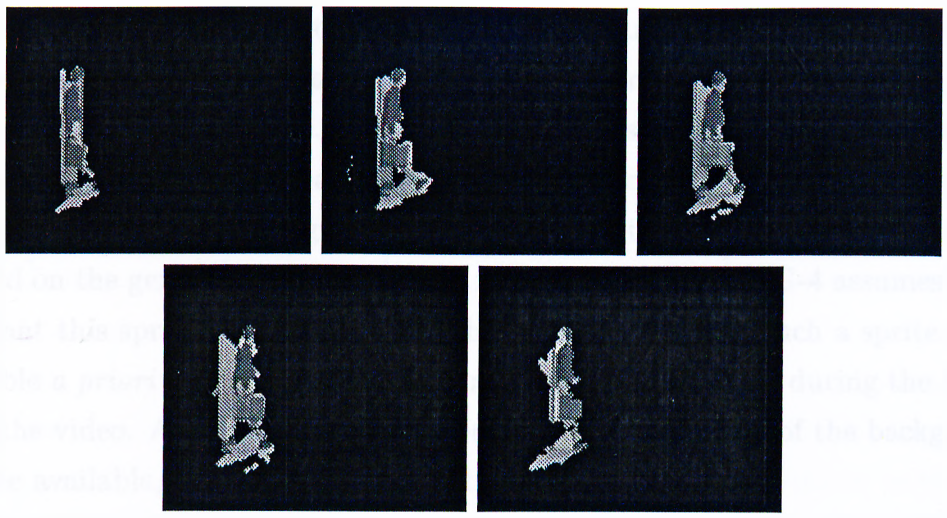

Figure 6.4. An example of occlusion regions, Hall Monitor

vector for these areas, and these areas are now marked as *stationary*. By Rule 1, it is assumed that a *moving* object has changed its status to *stationary*. And these *stationary* objects will be treated as objects for three frames[2] before being merged to background (being treated as a disappeared object).

A by-product of the newly introduced motion estimation module is the detection of the occluded areas. However, the information provided by that module can not be used directly. See Figure 6.5.

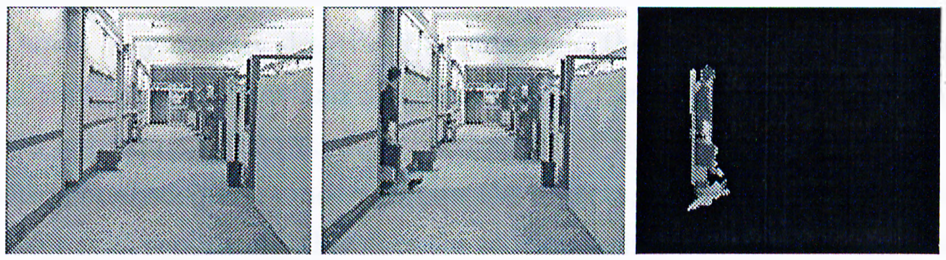

Figure 6.5. Frames 1 and 22 of Hall Monitor, and the segmentation mask

In this figure, two frames input to AM and the resulting segmentation mask is shown. The detected object (the man) is the occlusion region in this case, since there is no match for him in the previous frame. When he continues walking, the occlusion regions will be the areas which are covered and uncovered by him, but not himself. With the occlusion information from this module only, it is not possible to decide whether the occlusion area belongs to an object (the man in the first case) or background (in the subsequent frames). Therefore further analyses are required to distinguish where the detected occlusion areas belong to.

[2]See Appendix A for detailed explanation of AM parameters.

The "sprite" approach utilized in MPEG-4 is an elegant solution to the occlusion problem. A sprite is a large still image, describing panoramic background for all the frames in a video. For each frame in a sequence, the camera motion is calculated and only the parameters of this motion are utilized in subsequent frames to calculate the background. Then the objects which are coded separately are laid on the generated background to form the frame. MPEG-4 assumes, however that this sprite is extracted from the sequence. In AM, such a sprite is not available *a priori* but it is possible to construct it "on the fly", during the analysis of the video. As more frames are processed, different parts of the background become available, and the sprite estimate improves gradually.

Currently the Analysis Model is an unsupervised approach to video object segmentation. Many recent approaches, however, are based on "human supervision" in the video segmentation process. The idea behind this approach is "combining the best of two sides": Only a human can know what constitutes a semantic object in a scene, and given this information, computer can extract the objects much faster than a human can do.

The supervision can be either in the form of supplying an initial object mask, or telling the computer what kind of features a particular object possess in the current scene, as in [59] and [60]. When available (for example, through manual extraction), initial object masks provide a lot of valuable information to the algorithm, since they exactly describe the objects in the scene. Further information such as texture properties and shape will aid in getting a better color segmentation in subsequent frames. Another benefit of the Analysis Model can be in the motion analysis part: since we know the objects, we can calculate the motion content for each separately, therefore the motion estimation results (therefore segmentation results) will be much more reliable. This yields better object detection and tracking throughout the sequence.

If a manual segmentation is not available and such a segmentation is costly (both in time and effort), the "hint" type of supervision is more appropriate: the user "hints" the computer where to find the object in the scene (for example by drawing a few sketches with the mouse in the current frame). Then the computer attempts to extract the properties in those areas (color, texture etc) and seek the objects with similar features in the image. The approach in [60] is an example to this type of supervision. The study in [59] also employs sketching, but for the algorithm to work properly, the object's interior (the parts close to the actual border) should be outlined completely. This operation is not very desirable, since it takes time and effort to draw a border with certain accuracy.

Although human supervision is important, a good underlying unsupervised "segmentation engine" is necessary to make use of the extra information from the supervision. The improved model produces good results and its underlying modular unsupervised segmentation structure is open to the improvements in this form of "weak" supervision. Exploiting color and motion information separately enables the supervised segmentation results to improve compared to many other supervised approaches in the literature. The new motion estimation algorithm helps in getting better segmentation results when color segmentation works better, since it takes a color segmentation result (with reduced number of regions) as an estimate of object shapes. With improved color segmentation, these estimates improve, and more reliable motion vectors are obtained. This, in turn, leads to a contribution from AFS module: when color and motion segmentation modules perform better, it becomes possible to implement the separate motion content analysis for each object; therefore more accurate object shapes are obtained.

Appendix A

Issues about the Analysis Model Software, Version 4.0

The Analysis Model software is a combination of various modules, driven by a main program. It is written in "C" language, and works under Unix-like operating systems (SunOS, Solaris, Linux).

The directory structure of the software is as follows:

COPYRIGHT	Document to explain various copyright issues
Makefile	Together with Slave_Makefile, this is used in building the executable
README	Various information on Analysis Model
README_AM4	Information specific to Version 4
Slave_Makefile	Used with Makefile
am_main	Program code which drives other modules reside in this directory
am_motion	Code which handles motion estimation/compensation
am_post	Postprocessing code
am_rsst	Code in this directory handles segmentation based on RSST
am_rules	RuleProcessor Mode 2 code
bin	Executable code appears here after compilation
cproto	Utility to extract function prototypes
lib	Object code libraries for each directory
machines	Operating system dependent options

mom_baselib	Basic image processing tools
mom_extlib	Extended image processing tools
testb	Example parameter files
tools	Tools to extract comments from source code for documentation purposes
uh_cdet	Tools related to Change Detection Mask
uh_evaluate	Evaluation tools
uh_gmec	Tools for Global Motion Estimation/Compensation
uh_rules	Rule Processor Mode 1 code
uh_scd	Scene-cut Detection Module

GNU make utility is required for compilation. After compiling the code, the executable shows up in "bin" directory. Then, the software is executed as

```
am_linux.exe params.dat
```

provided the binary name is "am_linux.exe" and the parameter file name is "params.dat"

A parameter file contains all input to software for proper execution. The parameters of Analysis Model, Version 4 are explained below:

PARAMETER NAME	Default Value	VALID RANGE and TYPE	EXPLANATION
RULE_PROC		0 or 1	0: Bilkent Rule Proc 1: UH Rule Proc
COLOR_REGIONS	256	pos integer	number of regions in the color seg. mask
MV_REGIONS	4	pos integer	number of regions in the motion seg. mask
MOTIONTHRESHOLD	0.5	pos floating	Threshold to label a motion region as MOVING
MOT_RES	4	pos integer	Spatial Resolution of Motion Vector Field (i.e. one vector for each MOT_RES $\times$ MOT_RES block)
MOT_LEVELS	3	pos integer	Number of hierarchy levels in HBM algorithm

PARAMETER NAME	Default Value	VALID RANGE and TYPE	EXPLANATION
BLOCKSIDE	32 16 4	pos integer	Size of measurement windows for each hierarchy level (one number should be given for each level)
RANGE	8 4 2	pos integer	Range of search window for each hierarchy level (one number should be given for each level)
STEP	2 2 1	pos integer	Step size of search window for each hierarchylevel (one number should be given for each level)
MERGESIZE	20	pos integer	Regions with size less than this number will be merged with a neighboring region
MEM	3	pos integer	Objects that are labeled as STATIONARY will be treated MEM more frames as objects. After that, they will be merged to background.
ZERO_FORCE	16.0	pos floating	Constant used in zero-forcing of motion vectors. The larger this number, more vectors will be forced to (0,0) vector.
START_FRAME	0	non-neg int	Number of frame in a source analysis shall start with.
MAX_FRAME	299	pos integer	AM will process frames until (including) this frame
FRAMESKIP	3	pos integer	To get next frame in the sequence, this many frames will be skipped. Has no effect if AFS (see below) is enabled.
WIDTH		176 or 352	Width of input frames in pixels. Rule Processor Mode 2 runs with arbitrary sizes but Mode 1 works only with QCIF or CIF images.

PARAMETER NAME	Default Value	VALID RANGE and TYPE	EXPLANATION
HEIGHT		144 or 288	Height of input frames in pixels. Rule Processor Mode 2 runs with arbitrary sizes but Mode 1 works only with QCIF or CIF images.
INPUT_Y		any string	Path to file containing Y information of source sequence frames (luminance information)
INPUT_U		any string	Path to file containing U information of source sequence frames (chrominance information)
INPUT_V		any string	Path to file containing V information of source sequence frames (chrominance information)
INPUT_A		any string	Path to file containing Alpha channel information of source sequence frames. If EVALUATION is enabled, this indicates the file that original masks should be loaded from.
MOTION_X		any string	File that information about horizontal components of motion vectors will be written to.
MOTION_Y		any string	File that information about vertical components of motion vectors will be written to.
COLORMASK		any string	File that results of color segmentation will be written to.
MVMASK		any string	File that results of motion segmentation will be written to.
MCMASK		any string	File that results of motion compensation will be written to.
PRERESULT		any string	File that preresult mask will be written to.

PARAMETER NAME	Default Value	VALID RANGE and TYPE	EXPLANATION
RESULT		any string	File that result segmentation mask will be written to.
ORIG_RESULT		any string	File that result segmentation mask –which contains different objects– will be written to.
INTERP_MASK		any string	File that interpolated result segm. mask will be written to.
CDM		any string	File that Change Detection Mask will be written to.
RESULT_FGY		any string	File that result with blended foreground will be written to (y-channel)
RESULT_FGU		any string	File that result with blended foreground will be written to (u-channel)
RESULT_FGV		any string	File that result with blended foreground will be written to (v-channel)
RESULT_BGY		any string	File that result with blended background will be written to (y-channel)
RESULT_BGU		any string	File that result with blended background will be written to (u-channel)
RESULT_BGV		any string	File that result with blended background will be written to (v-channel)
TEXT_COVER	80	pos integer	Threshold for Rule Processor of UH
GMEC_ITER	70	pos integer	Number of iterations for global motion estimation
PEL_COUNT	10000	pos integer	Number of observation points for global motion estimation

PARAMETER NAME	Default Value	VALID RANGE and TYPE	EXPLANATION
GRAD_X_MIN	0.0	pos floating	Minimum gradient in x-direction of observation points in global motion estimation
GRAD_X_MAX	260.0	pos floating	Maximum gradient in x-direction of observation points in global motion estimation
GRAD_Y_MIN	0.0	pos floating	Minimum gradient in y-direction of observation points in global motion estimation
GRAD_Y_MAX	260.0	pos floating	Maximum gradient in y-direction of observation points in global motion estimation
CDM_QCIF	220.4	pos floating	Threshold for change detection (QCIF, static camera)
CDM_QCIF_MB	220.4	pos floating	Threshold for change detection (QCIF, moving camera)
CDM_CIF	165.3	pos floating	Threshold for change detection (CIF, static camera)
CDM_CIF_MB	41.325	pos floating	Threshold for change detection (CIF, moving camera)
SCD_THRESH	250.0	pos floating	Threshold for scene-cut detection (MSE)
EVAL	0	0 or 1	If 1, evalutations will be performed and written to disk.
EVAL_OUT_GENERAL		any string	File that table of all evaluation results will be written to.
EVAL_OUT_ABS		any string	File that evaluation results of absolute distortion criterion will be written to.
EVAL_OUT_REL		any string	File that evaluation results of relative distortion criterion will be written to.

PARAMETER NAME	Default Value	VALID RANGE and TYPE	EXPLANATION
EVAL_OUT_TMP_ORI		any string	File that evaluation results of temporal coherency criterion will be written to (original mask)
EVAL_OUT_TMP_EST		any string	File that evaluation results of temporal coherency criterion will be written to (estimated mask)
INTERACTIVE	0	0 or 1	If 1, an initial segmentation mask will be read from disk.
INPUT_SEG_MASK		any string	Name of file containing initial segmentation mask.
MOT_SEG_ALG	0	0 or 1	0: Original Mot. Seg. algorithm 1: Mot. Seg. Alg. based on affine motion modelling
AFS	1	0 or 1	0: No AFS 1: Adaptive Frame Skip (here, FRAME_SKIP has no meaning)
AFS_THRESHOLD	600	pos integer	If AFS = 1, this sets the threshold to determine if there's "enough" motion or not.
HALF_PIX	1	0 or 1	If 1, motion vectors are calculated with sub-pixel accuracy.
MOT_EST_ALG	0	0 or 1	0: HBM 1: Gibbs-based algorithm
GIBBS_Color	25	pos integer	If gibbs is used, this sets the number of regions in the color seg. mask input to Gibbs.
GIBBS_LevMin	0	non-neg int	If gibbs is used, this sets the lowest level (finest level) for analysis 0 = pixel level
GIBBS_LevMax	3	pos integer	If gibbs is used, this sets the highest level (coarsest level) for analysis 0 = pixel level

PARAMETER NAME	Default Value	VALID RANGE and TYPE	EXPLANATION
GIBBS_IterMax	5	pos integer	If gibbs is used, this sets the maximum number of iterations for each level
GIBBS_Search1 GIBBS_Search2	3 3	pos integer pos integer	If gibbs is used, these two act as coefficients to determine the search space during iterations. See the doc for a detailed explanation.
GIBBS_LM	200.0	pos floating	In gibbs, this sets λ_m.
GIBBS_LS	100.0	pos floating	In gibbs, this sets λ_s.
GIBBS_TS1	20.0	pos floating	In gibbs, this sets T_{s_1}.
GIBBS_TS2	30.0	pos floating	In gibbs, this sets T_{s_2}.
USE_CDM_FOR_LABELLING	0	0 or 1	If 1, motion region labels are determined using CDM. If not, motion vector averages are used in labelling.

References

[1] A. A. Alatan, L. Onural, M. Wollborn, R. Mech, E. Tuncel, and T. Sikora, "Image sequence analysis for emerging interactive multimedia services - the European COST 211 framework," *IEEE Transactions on Circuits, Systems, and Video Technology*, November 1998.

[2] COST211^{ter} simulation subgroup, *Description of COST211 Analysis Model, SIM(98)14*, July 1998.

[3] A. A. Alatan, E. Tuncel, and L . Onural, "Object segmentation via rule-based data fusion," in *Proceedings of Workshop on Image Analysis for Multimedia Interactive Services*, (Louvain-la-Neuve, Belgium), pp. 51–56, June 1997.

[4] A. A. Alatan, E. Tuncel, and L. Onural, "A rule-based method for object segmentation in video sequences," in *IEEE International Conference on Image Processing*, vol. 2, (Santa Barbara, California), pp. 522–525, October 1997.

[5] A. M. Tekalp, *Digital Video Processing.* Prentice Hall, 1995.

[6] International Telecommunication Union (ITU), *Recommendation H.261 (03/93) - Video codec for audiovisual services at p x 64 kbit/s*, 1993.

[7] Project COST-211bis Final Report, *Redundancy Reduction Techniques for Coding of Broadband Video Signals*, 1990.

[8] International Organization for Standardisation (ISO), *Official MPEG Home Page: http://drogo.cselt.stet.it/mpeg/standards/mpeg-1/mpeg-1.htm*, June 1996.

[9] International Organization for Standardisation (ISO), *ISO/IEC 11172-2:1993 Information technology – Coding of moving pictures and associated*

audio for digital storage media at up to about 1,5 Mbit/s – Part 2: Video, 1993.

[10] International Organization for Standardisation (ISO), *Official MPEG Home Page: http://drogo.cselt.stet.it/mpeg/standards/mpeg-2/mpeg-2.htm*, June 1996.

[11] International Organization for Standardisation (ISO), *ISO/IEC 13818-2:1996 Information technology – Generic coding of moving pictures and associated audio information: Video*, 1996.

[12] International Telecommunication Union (ITU), *ITU-T Recommendation H.263 - Video coding for low bit rate communication.*

[13] ISO/IEC JTC1/SC29/WG11 N2323, *Overview of the MPEG-4 Standard*, July 1998.

[14] ISO/IEC JTC1/SC29/WG11 N2326, *MPEG-7 Context and Objectives*, July 1998.

[15] O. J. Morris, M. J. Lee, and A. G. Constantinides, "Graph theory for image analysis: An approach based on the shortest spanning tree," *IEE Proceedings*, vol. 133, pp. 146–152, April 1986.

[16] M. J. Biggar, O. J. Morris, and A. G. Constantinides, "Segmented-image coding: Performance comparison with the discrete cosine transform," *IEE Proceedings*, vol. 135, pp. 121–132, April 1988.

[17] E. Tuncel, "Utilization of improved RSST method for video object segmentation," Master's thesis, Bilkent University, August 1997.

[18] M. Bierling, "Displacement estimation by hierarchical blockmatching," *Proceedings of SPIE Visual Communications and Image Processing*, pp. 942–951, 1988.

[19] R. Mech and M. Wollborn, "A noise robust method for 2D shape estimation of moving objects in video sequences considering a moving camera," in *Proceedings of Workshop on Image Analysis for Multimedia Interactive Services*, (Louvain-la-Neuve, Belgium), pp. 57–62, June 1997.

[20] R. Mech and P. Gerken, *Automatic Segmentation of Moving Objects (Partial Results of Core Experiment N2).* ISO/IEC JTC1/SC29/WG11 MPEG97/1949, Bristol, England, April 1997.

[21] M. Wollborn, R. Mech, S. Colonnese, U. Mascia, G. Russo, P. Talone, J. G. Choi, M. Kim, M. H. Lee, and C. Ahn, *Description of Automatic Segmentation Techniques Developed and Tested for MPEG-4 Version 1.* ISO/IEC JTC1/SC29/WG11 MPEG97/2704, Fribourg, Switzerland, October 1997.

[22] R. Mech and M. Wollborn, "A noise robust method for segmentation of moving objects in video sequences," in *IEEE International Conference on Acoustics, Speech & Signal Processing*, (Munich, Germany), pp. 2657–2660, April 1997.

[23] T. Aach, A. Kaup, and R. Mester, "Statistical model-based change detection in moving video," *IEEE Transactions on Signal Processing*, vol. 31, pp. 165–180, March 1993.

[24] T. Aach, A. Kaup, and R. Mester, "Change detection in image sequences using gibbs random fields: a bayesian approach," in *Proceedings of International Workshop on Intelligent Signal Processing and Communication Systems*, (Sendai, Japan), pp. 56–61, October 1993.

[25] M. Hötter and R. Thoma, "Image segmentation based on object oriented mapping parameter estimation," *IEEE Transactions on Signal Processing*, vol. 15, pp. 315–334, 1988.

[26] V. Kumar and E. S. Manolakos, "Unsupervised model-based object recognition by parameter estimation of hierarchical mixtures," in *IEEE International Conference on Image Processing*, vol. 3, pp. 967–970, 1996.

[27] V. Kumar and E. S. Manolakos, "Unsupervised statistical neural networks for model-based object recognition," *IEEE Transactions on Signal Processing*, vol. 45, pp. 2709–2718, 1997.

[28] O. Yanez-Suarez and M. R. Azimi-Sadjadi, "Automated analysis of complex scenes of airborne fiberglass preparations for scanning electron microscopy imagery," in *IEEE International Conference on Image Processing*, vol. 2, pp. 438–441, 1997.

[29] X. Wan and C.-C. J. Kuo, "A multiresolution color clustering approach to image indexing and retrieval," in *IEEE International Conference on Acoustics, Speech & Signal Processing*, vol. 6, pp. 3705–3708, 1998.

[30] M. A. Wani and B. G. Batchelor, "Edge-region-based segmentation of range images," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 16, pp. 314–319, March 1994.

[31] R. Stoica, J. Zerubia, and J. M. Francos, "The two-dimansional wold decomposition for segmentation and indexing in image libraries," in *IEEE International Conference on Acoustics, Speech & Signal Processing*, vol. 5, pp. 2977–2980, 1998.

[32] S. A. Barker and P. J. W. Rayner, "Unsupervised image segmentation," in *IEEE International Conference on Acoustics, Speech & Signal Processing*, vol. 5, pp. 2757–2760, 1998.

[33] A. M. Ferman, B. Günsel, and A. M. Tekalp, "Motion and shape signatures for object-based indexing of MPEG-4 compressed video," in *IEEE International Conference on Acoustics, Speech & Signal Processing*, vol. 4, pp. 2601–2604, 1997.

[34] K. J. Han and A. H. Tewfik, "Eigen-image based video segmentation and indexing," *IEEE International Conference on Image Processing*, vol. 2, pp. 538–541, 1997.

[35] E. Sahouria and A. Zakhor, "Motion indexing of video," *IEEE International Conference on Image Processing*, vol. 2, pp. 526–529, 1997.

[36] H. Yu, G. Bozdağı, and S. Harrington, "Feature-based hierarchical video segmentation," *IEEE International Conference on Image Processing*, vol. 2, pp. 498–501, 1997.

[37] G. Iyengar and A. B. Lippman, "Videobook: An experiment in characterization of video," *IEEE International Conference on Image Processing*, vol. 3, pp. 855–858, 1996.

[38] P. Csillag and L. Böröczky, "Iterative motion-based segmentation for object-based video coding," in *IEEE International Conference on Image Processing*, vol. 1, pp. 73–76, 1997.

[39] P. De Smet and D. De Vleeschauwer, "Motion-based segmentation using a thresholded merging strategy on watershed segments," in *IEEE International Conference on Image Processing*, vol. 2, pp. 490–493, 1997.

[40] F. Bartolini, V. Capellini, and L. Tucci, "Simultaneous optic flow estimation and segmentation by means of least squares techniques," in *IEEE International Conference on Image Processing*, vol. 1, pp. 97–100, 1997.

[41] Y. Yemez, B. Sankur, and E. Anarım, "An object-oriented video codec based on growing motion segmentation," in *IEEE International Conference on Image Processing*, vol. 3, pp. 444–447, 1997.

[42] J. G. Choi, S. W. Lee, and S. D. Kim, "Segmentation and motion estimation of moving objects for object-oriented analysis-synthesis coding," in *IEEE International Conference on Acoustics, Speech & Signal Processing*, vol. 4, pp. 2431–2434, 1995.

[43] M. Ebbecke, M. B. H. Ali, and A. Dengel, "Real time object detection, tracking and classification in monocular image sequences of road traffic scenes," in *IEEE International Conference on Image Processing*, vol. 2, pp. 402–405, 1997.

[44] B. K. Low and M. K. Ibrahim, "A fast and accurate algorithm for facial feature segmentation," in *IEEE International Conference on Image Processing*, vol. 2, pp. 518–521, 1997.

[45] H. Nugroho, S. Takahashi, Y. Ooi, and S. Ozawa, "Detecting human face from monocular image sequences by genetic algorithms," in *IEEE International Conference on Acoustics, Speech & Signal Processing*, vol. 4, pp. 2533–2536, 1997.

[46] N. Paragios, P. Perez, G. Tziritas, C. Labit, and P. Bouthemy, "Adaptive detection of moving objects using multiscale techniques," in *IEEE International Conference on Image Processing*, vol. 1, pp. 525–528, 1996.

[47] J. G. Choi, S. W. Lee, and S. D. Kim, "Video segmentation based on spatial and temporal information," in *IEEE International Conference on Acoustics, Speech & Signal Processing*, vol. 4, pp. 2661–2664, 1997.

[48] J. R. Ohm and P. Ma, "Feature-based cluster segmentation of image sequences," in *IEEE International Conference on Image Processing*, vol. 3, pp. 178–181, 1997.

[49] F. Moscheni and S. Bhattacharjee, "Robust region merging for spatio-temporal segmentation," in *IEEE International Conference on Image Processing*, vol. 1, pp. 501–504, 1996.

[50] F. Dufaux, F. Moscheni, and A. Lippman, "Spatio-temporal segmentation based on motion and static segmentation," *IEEE International Conference on Image Processing*, vol. 1, pp. 306–309, 1995.

[51] F. Morier, J. B. Pineau, D. Barba, and H. Sanson, "Robust segmentation of moving image sequences," in *IEEE International Conference on Image Processing*, vol. 1, pp. 719–722, 1997.

[52] L. Torres, D. Garcia, and A. Mates, "A robust motion estimation and segmentation approach to represent moving images with layers," in *IEEE International Conference on Acoustics, Speech & Signal Processing*, vol. 4, pp. 2981–2984, 1997.

[53] S. Siggelkow, R. R. Grigat, and A. Ibenthal, "Segmentation of image sequences for object oriented coding," in *IEEE International Conference on Image Processing*, vol. 2, pp. 477–480, 1996.

[54] M. M. Chang, M. I. Sezan, and A. M. Tekalp, "An algorithm for simultaneous motion estimation and scene segmentation," in *IEEE International Conference on Acoustics, Speech & Signal Processing*, vol. 5, pp. 221–224, 1994.

[55] F. Moscheni and F. Dufaux, "Region merging based on robust statistical testing," in *Visual Communications and Image Processing*, vol. 3, (Orlando, Florida), pp. 1118–1129, March 1996.

[56] F. Moscheni, F. Dufaux, and M. Kunt, "A new two-stage global/local motion estimation based on a background/foreground segmentation," in *IEEE International Conference on Acoustics, Speech & Signal Processing*, vol. 4, pp. 2261–2264, 1995.

[57] L. Kaufman and P. J. Rousseeuw, *Finding Groups in Data: an Introduction to Cluster Analysis.* Wiley, 1990.

[58] F. Moscheni, F. Dufaux, and M. Kunt, "Object tracking based on temporal and spatial information," in *IEEE International Conference on Acoustics, Speech & Signal Processing*, vol. 4, pp. 1915–1918, 1996.

[59] C. Gu and M. C. Lee, "Semantic video object segmentation and tracking using mathematical morphology and perpective motion model," in *IEEE International Conference on Image Processing*, vol. 2, pp. 514–517, 1997.

[60] E. Chalom and V. M. Bove, "Segmentation of an image sequence using multi-dimensional image attributes," in *IEEE International Conference on Image Processing*, vol. 2, pp. 525–528, 1996.

[61] Y. Altunbasak, R. Oten, and R. J. P. de Figueiredo, "Simultaneous object segmentation, multiple object tracking and alpha map generation," *IEEE International Conference on Image Processing*, vol. 1, pp. 69–72, 1997.

[62] P. E. Eren, Y. Altunbasak, and A. M. Tekalp, "Region-based affine motion segmentation using color information," in *IEEE International Conference on Acoustics, Speech & Signal Processing*, vol. 4, pp. 3005–3008, 1997.

[63] Y. Altunbasak and A. M. Tekalp, "Object-scalable content-based 2-D mesh design and tracking for object-based video coding," *IEEE Transactions on Image Processing*, September 1997.

[64] M. M. Chang, M. I. Sezan, and A. M. Tekalp, "Simultaneous motion estimation and segmentation," *IEEE Transactions on Image Processing*, vol. 6, no. 9, pp. 1326–1333, 1997.

[65] P. B. Chou and C. M. Brown, "The theory and practice of bayesian image labeling," *International Journal of Computer Vision*, vol. 4, pp. 185–210, 1990.

[66] J. Besag, "On the statistical analysis of dirty pictures," *J. R. Statist. Soc.*, vol. 48, pp. 259–302, 1986.

[67] M. Chang, A. M. Tekalp, and M. I. Sezan, "Motion-field segmentation using an adaptive MAP criterion," in *IEEE International Conference on Acoustics, Speech & Signal Processing*, vol. 5, pp. 33–36, 1993.

[68] S. Kıranyaz, "Regularized motion estimation techniques and their applications to video coding," Master's thesis, Bilkent University, September 1996.

[69] S. Kıranyaz and L. Onural, "Motion compensated frame interpolation techniques for VLBR coding," in *IEEE International Conference on Image Processing*, vol. 1, 1997.

[70] A. A. Alatan, *Object-based 3-D Motion and Structure Analysis For Video Coding Applications.* PhD thesis, Bilkent University, February 1997.

[71] A. A. Alatan and L. Onural, "Estimation and efficient encoding of depth field for video compression using 3-D structure and motion of objects," *IEEE Transactions on Image Processing*, June 1998.

[72] A. A. Alatan and L. Onural, "3-D motion estimation of rigid objects for video coding applications using an improved iterative version of the e-matrix method," *IEEE Signal Processing Letters*, February 1998.

[73] A. A. Alatan and L. Onural, "Utilization of 3-D motion models in video coding: Occlusion detection and motion compensated temporal interpolation," in *Advances In Digital Image Communication, Proceedings of 2nd Erlangen Symposium*, pp. 85–92, April 1997.