

REGULARIZED MOTION ESTIMATION
TECHNIQUES AND THEIR APPLICATIONS TO
VIDEO CODING

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
Serkan Kiranyaz
September, 1996

THESIS
TK
6680.5
.K57
1996

REGULARIZED MOTION ESTIMATION
TECHNIQUES AND THEIR APPLICATIONS TO
VIDEO CODING

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


Serkan Kiranyaz

September 1996

TK
6680.5
.K57
1995


E C 35376

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.



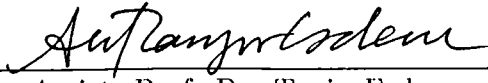
Prof. Dr. Levent Onural(Supervisor)

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.




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.



Assist. Prof. Dr. Tanju Erdem

Approved for the Institute of Engineering and Sciences:



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

ABSTRACT

REGULARIZED MOTION ESTIMATION TECHNIQUES AND THEIR APPLICATIONS TO VIDEO CODING

Serkan Kiranyaz

M.S. in Electrical and Electronics Engineering

Supervisor: Prof. Dr. Levent Onural

September 1996

Novel regularized motion estimation techniques and their possible applications to video coding are presented. A block matching motion estimation algorithm which extracts better block motion field by forming and minimizing a suitable energy function is introduced. Based on an adaptive structure onto block sizes, an advanced block matching algorithm is presented. The block sizes are adaptively adjusted according to the motion. Blockwise coarse to fine segmentation based motion estimation algorithm is introduced for further reduction on the number of bits that are spent for the coding of the block motion vectors. Motion estimation algorithms which can be used for average motion determination and artificial frame generation by fractional motion compensation are also developed. Finally, an alternative motion estimation and compensation technique which defines feature based motion vectors on the object boundaries and reconstructs the decoded frame from the interpolation of the compensated object boundaries is presented. All the algorithms developed in this thesis are simulated on real or synthetic images and their performance is demonstrated.

Keywords : Video Coding, Regularization, Motion Estimation, Motion Compensation, Motion Detection, Line Field.

ÖZET

DÜZGÜNLEŞTİRİLMİŞ HAREKET KESTİRİMİ TEKNİKLERİ VE VİDEO KODLAMADAKİ UYGULAMALARI

Serkan Kiranyaz

Elektrik ve Elektronik Mühendisliği Bölümü Yüksek Lisans

Tez Yöneticisi: Prof. Dr. Levent Onural

Eylül 1996

Yeni düzgünleştirilmiş hareket kestirimi teknikleri ve olası video kodlama uygulamaları sunulmuştur. İlk olarak, uygun bir enerji fonksiyonunu minimize ederek daha iyi bir hareket vektör alanı oluşturan hareket kestirimi algoritması tanıtılmıştır. Daha sonra blok boylarına adaptif bir yapı konulmasıyla oluşturulan bloklara dayalı hareket kestirimi algoritması sunulmuştur. Blok boyları harekete bağlı olarak bulunmaktadır. Blok hareket vektörlerine harcanan bit sayısında daha fazla indirim yapabilecek hiyerarşik yapıda bölütlemeyle bağlı bir blok hareket kestirimi algoritması tanıtılmıştır. Ayrıca ortalama hareket belirlenmesi ve parçalı harekete göre kaydırmayla sanal görüntü üretiminde kullanılabilecek hareket kestirimi algoritmaları geliştirilmiştir. Son olarak özelliklere dayalı hareket vektörlerini nesne sınırları üzerinde tanımlayan ve kaydırılmış nesne sınırlarının interpolasyonundan çözülmüş görüntüyü oluşturan hareket kestirimi ve ona göre kaydırma teknikleri sunulmuştur. Tezde geliştirilen algoritmalar gerçek ve sentetik görüntüler üzerinde denenmiş ve performansları gözlenmiştir.

Anahtar Kelimeler : Düzgünleştirme, Hareket Kestirimi, Harekete Göre Kaydırma, Çizgisel İşlevler.

ACKNOWLEDGEMENT

I would like to express my deep gratitude to my supervisor Prof. Dr. Levent Onural for his guidance, suggestions and invaluable encouragement throughout the development of this thesis.

I would also like to thank to Assist. Prof. Dr. Tanju Erdem and Assist. Prof. Dr. Orhan Arıkan for reading and commenting on the thesis.

And special thanks to all my friends for their valuable discussions and helps.

Finally, it is my pleasure to express my thanks to my family for their eternal encouragement, support and love they have given me even from so far away.

TABLE OF CONTENTS

1	Introduction	2
1.1	Basic Problems in Motion Estimation	3
1.2	Regularization of Ill Posed Problems	4
1.3	Constrained and Stochastic Motion Models	6
1.4	Motion Compensation and Video Coding	9
1.5	Block Motion Estimation Algorithms and Video Coding Implementations	10
1.6	Scope and Outline of the Thesis	13
2	Regularized Motion Estimation	15
2.1	Block Motion Estimation by Energy Minimization	16
2.1.1	Energy Based BMME Algorithm	17
2.1.2	Results	18
2.2	Adaptive Block Matching Algorithm	20
2.2.1	Adaptive Block Matching Algorithm (ABMA)	22

2.2.2	Results	25
2.3	Blockwise Coarse to Fine Segmentation of Motion Fields	26
2.3.1	Hierarchical Block Matching Algorithm	31
2.3.2	Results	33
3	Frame Rate Regulation and Frame Interpolation	38
3.1	Average Motion Determination and Frame Rate Regulation	39
3.1.1	Motion Estimation Algorithm for AMD	39
3.2	Motion Compensated Interpolation	42
3.2.1	Single MC Interpolation	47
3.2.2	Double MC Interpolation	48
3.2.3	Binary Tree Structured MC Interpolation	50
3.2.4	Results	50
4	Motion Estimation and Compensation on Line Field	57
4.1	Line Field Definition and Extraction	58
4.1.1	Line Field Extraction	59
4.1.2	Results	60
4.2	Motion Vectors on Line Fields	61
4.2.1	Extraction of Line Motion Vectors	62
4.2.2	Results	64
4.3	Reconstruction on Motion Compensated Image from Motion Compensated Line Fields	65

4.3.1 Results	66
5 Conclusions and Future Work	68
A Motion Estimation Algorithms in H.261 and H.263	71

LIST OF FIGURES

1.1	Given two frames, block motion estimation is performed within a search area in the previous frame	11
2.1	PSNR (<i>top</i>) and Bit-Rate (<i>bottom</i>) graphics of the classical (o) and regularized (*) BMME algorithms for the <i>Mother & Daughter</i> sequence.	19
2.2	PSNR (<i>top</i>) and Bit-Rate (<i>bottom</i>) graphics of the classical (o) and regularized (*) BMME algorithms for the <i>Foreman</i> sequence.	20
2.3	BMVs extracted from classical (<i>left</i>) and regularized (<i>right</i>) BMME algorithms for the frames (10-11) that are taken from the <i>Mother & Daughter</i> sequence.	21
2.4	BMVs extracted from classical (<i>left</i>) and regularized (<i>right</i>) BMME algorithms for the frames (29-30) that are taken from the <i>Foreman</i> sequence.	21
2.5	(<i>top</i>) Previous and current frames, (<i>bottom</i>) BMVs by (<i>left</i>) classical block matching algorithm, (<i>right</i>) proposed algorithm	25
2.6	(<i>top</i>) Previous and current frames (<i>Mother & Daughter</i> frames 78 & 81), (<i>middle</i>) compensated frames by (left) classical block matching algorithm, (right) proposed algorithm, (<i>bottom</i>) BMVs extracted by (left) classical block matching algorithm, (right) proposed algorithm (depth=5).	27

2.7	<i>(top)</i> Previous and current frames (<i>Foreman frames 66 & 69</i>), <i>(middle)</i> compensated frames by (left) classical block matching algorithm, (right) proposed algorithm, <i>(bottom)</i> BMVs extracted by (left) classical block matching algorithm, (right) proposed algorithm (depth=4).	28
2.8	<i>(top)</i> <i>Mother & Daughter</i> , <i>(bottom)</i> <i>Foreman</i> , zoomed parts of the compensated frames by (left) classical block matching algorithm, (right) proposed algorithm.	29
2.9	PSNR <i>(top)</i> and Bit-Rate <i>(bottom)</i> graphics of the classical BMME (o) and ABMA (*) algorithms for the <i>Mother & Daughter</i> sequence.	30
2.10	PSNR <i>(top)</i> and Bit-Rate <i>(bottom)</i> graphics of the classical BMME (o) and ABMA (*) algorithms for the <i>Foreman</i> sequence.	30
2.11	Sub-division process in Quad-Tree structure for the depths = 1,2,3,...	32
2.12	<i>(top)</i> Previous and current frames (<i>Foreman, frames: 0 & 1</i>), <i>(bottom)</i> (left) block motion vectors and segments (each gray-level shows different segmentation), (right) motion compensated image (PSNR=29.0212 dB, depth=3).	34
2.13	<i>(top)</i> Previous and current frames (<i>Foreman, frames: 77 & 78</i>), <i>(bottom)</i> (left) block motion vectors and segments (each gray-level shows different segmentation), (right) motion compensated image (PSNR=24.0913 dB,depth=3).	35
2.14	<i>(top)</i> Previous and current frames (<i>Container Ship, frames: 61 & 81</i>), <i>(bottom)</i> (left) block motion vectors and segments (each gray-level shows different segmentation), (right) motion compensated image (PSNR=29.0598 dB, depth=4).	36

2.15	(<i>top</i>) Previous and current frames (<i>Mother & Daughter</i> , frames: 47 & 52), (<i>bottom</i>) (left) block motion vectors and segments (each gray-level shows different segmentation), (right) motion compensated image (PSNR=31.3692 dB,depth=4).	37
3.1	Line fields (horizontal and vertical) representation in dual-lattice.	43
3.2	Uniformity field extracted from line field.	44
3.3	Particular positional penalization values of line field elements. .	45
3.4	<i>Single MC interpolation</i> of the interval frames M1,M2,M3 and M4 from the given previous and current frames. (IN=5).	49
3.5	<i>Double MC interpolation</i> of the interval frames M1,M2,M3 and M4 from the given frame 1 & 2. (IN=5).	49
3.6	<i>Binary tree structured MC interpolation</i> of the interval frames M1...M7. M1,M2,M4 is generated from frame 1, M7,M6 is generated from frame 2 and M3,M5 is generated from M4 by MC interpolation. Note that except M3,M5, other interval frames are compensated from the original frames. (IN=8).	51
3.7	Given previous and current frames are 120 th and 150 th frames of the <i>Akiyo</i> sequence (<i>top</i>) and 60 th and 90 th frames of the <i>Container Ship</i> sequence (<i>bottom</i>).	54
3.8	Linear interpolation (<i>top</i>), single MC interpolation, double MC interpolation, binary tree structured MC interpolation (<i>bottom</i>). Given previous and current frames are 120 th and 150 th frames of the <i>Akiyo</i> sequence.	55
3.9	Linear interpolation (<i>top</i>), single MC interpolation, double MC interpolation, binary tree structured MC interpolation (<i>bottom</i>). Given previous and current frames are 60 th and 90 th frames of the <i>Container Ship</i> sequence.	56

4.1	The discontinuities (obtained by ∇ operation) (<i>left</i>) and line field (<i>right</i>) of the 40 th frame in <i>Mother & Daughter</i> sequence.	59
4.2	Original frames: 20 th frame in <i>Container Ship</i> sequence and 40 th frame in <i>Mother & Daughter</i> sequence.	60
4.3	Line fields (horizontal, vertical and both) of the original frames.	61
4.4	Line fields (horizontal, vertical and both) of the previous (<i>top</i>), current (<i>middle</i>) and MC (<i>bottom</i>) frames.	64
4.5	Previous, current and MC frames.	66
A.1	Weights for current and two neighbor blocks	73
A.2	Candidate neighbors for predictors for each of the luminance block.	73
A.3	Candidate neighbors for predictors for a macroblock BMV	74

LIST OF TABLES

2.1	Simulation parameters for <i>ABMA</i>	26
2.2	Simulation parameters for <i>Hierarchical Block Matching Algorithm</i>	33
A.1	Simulation parameters for <i>ABMA</i>	76

Chapter 1

Introduction

Starting from the late sixties, much effort has been spent on the development of videophone or such apparatus which are operating at low transmission bit rates [1, 2, 3]. In this area, the main objective is the transmission of video frames as efficiently as possible within an acceptable loss of visual image quality. This can only be achieved by taking advantage of the interframe correlation. The key tool for that is motion estimation and compensation. Motion estimation is a highly ill-posed problem and therefore, should be solved by regularization. Regularization should be performed in such a way that motion estimation algorithms can extract a motion field which is suitable for the application aspects. Various regularization techniques [4, 5, 6] have been proposed to provide reliable estimates from ill-posed measurements [7].

Motion estimation, as its name implies, is concerned with the extraction of motion information from a sequence of video frames. It is used in a wide range of applications including video coding, computer and robot vision, traffic monitoring, military defense systems, autonomous navigation of mobile vehicles, biomedical research.

In various motion estimation applications, the motion is represented by a 2-D field which is the projection of a 3-D object motion onto the image plane. 2-D motion estimation is concerned with displacements of 2-D projections of

object points in consecutive frames for various applications of the digital video frames (i.e. video coding).

In this thesis, we are concerned with 2-D motion estimation algorithms and their applications to very low bit rate (VLBR) video coding. Since 2-D motion estimation is an ill-posed problem, we are looking for suitable regularization techniques. Furthermore, some effective improvements are also proposed for some classical methods. Especially in very low bit rate video coding, improvements for the classical block matching motion estimation algorithm can realize better block motion estimation in terms of bit rate. Also segmentation based motion estimation algorithms are shown to further improve the performance in such a way that more reduction in the bit rate can be achieved.

1.1 Basic Problems in Motion Estimation

The aim in 2-D motion estimation is the computation or extraction of the movement of the objects which are in the image plane. There are various algorithms which have been developed to estimate 2-D or 3-D motion from the video frames [8, 9, 10, 11, 12, 13, 14, 15, 16, 17]. However, there are still open problems and some difficulties in that area.

One of the main problems is the overlapping of moving objects. This situation is called as “occlusion” [18]. Occlusion effect makes the detection and estimation of the motion difficult. Also there can exist some self-occlusion effects of a single moving object. For example, a 3-D rotation of an object causes some parts of the object to be unseen and some unseen parts of the object to become visible. This problem is out of the scope of this thesis.

Another important problem in motion estimation is the relative motion between the objects and the camera. Such relative motion causes difficulties in the detection of the moving objects as well as estimation of their actual movements. This implies that the term of motion estimation often indicates a combined detection-estimation process, such as segmentation of the individual

moving objects and then estimating their motion. Since the detection of the objects requires that the motion has to be estimated beforehand while the motion estimation requires the detection of moving objects, detection-estimation processes are not independent of each other, and therefore, any motion estimation algorithm should be developed, accordingly.

In order to improve the robustness of the motion estimation algorithms, the presence of camera noise in the observed images is explicitly taken into account [19]. This is well done by a preprocessing stage for the video frames. The main idea is to obtain such a motion field which can represent the actual movement as much as possible and also which can be coded effectively. Such noise reduction techniques are not addressed in this thesis.

In the coding point of view, there is another problem or even a dilemma. It is the difference between **well matched** and **well codable** motion field. That is to say that well matched motion field is obtained by taking “good matching” into account but on the other hand a well codable motion field is extracted by regularizing the motion field and therefore, it tends to have less matching but better coding of the motion field. As a result, the motion estimation algorithms that are used in video coding should be designed in such a way that the amount of regularization should be arranged according to the application aspects (such as channel bandwidth (bit-rate) and minimum signal to noise ratio (SNR) requirement).

1.2 Regularization of Ill Posed Problems

As stated previously, motion estimation is an ill-posed problem [6, 5, 7]. The reason behind the ill-posedness is that the number of constraints is insufficient to find a unique and robust solution. It is because of the fact that there is only one constraint for each motion vector which consist of two components. That constraint depends on an assumption which may not be always true: that assumption is brightness constancy of an object point and yields the following optical flow equality:

$$I_t(x) = I_{t-1}(x - d(\vec{x})) \quad (1.1)$$

where x is any pixel location vector, I_t is the intensity (or color) value for the pixel x at time t and $d(\vec{x})$ is the candidate motion vector. For real world images, this equality usually does not hold because of noise. Therefore, it can be converted to a well-known constraint,

$$d(\vec{x}) = \arg \min [L(I_t(x), I_{t-1}(x - d(\vec{x})))] \quad (1.2)$$

where $L(\cdot)$ is the absolute difference operator. Equation 1.2 is still insufficient to obtain a unique solution for $d(\vec{x})$. The main reason of the ill-posedness is because $d(\vec{x})$ consists of two unknown components but there is only one constraint present.

So in order to obtain a unique and robust solution, this problem is regularized by adding several constraints to the problem. The choice of constraints determines the type of the regularization and it varies due to application requirements. In other words, requirements of any specific application determine the regularization technique. Especially, bit rate and signal to noise ratio (SNR) are the most significant requirements that can determine the choice of the regularization technique.

Usual regularization techniques result in a smooth motion field. In other words, those techniques put a smoothness constraint in addition to the optical flow constraint. Hence one can state the total constraint employed on the motion field as follows:

$$d(\vec{\mathbf{x}}) = \arg \min [L(I_t(\mathbf{x}), I_{t-1}(\mathbf{x} - d(\vec{\mathbf{x}}))) + \lambda R(d(\vec{\mathbf{x}}))] \quad (1.3)$$

where λ is the regularization parameter and $R(\cdot)$ is the regularization operator which imposes smoothness on to the motion field. \mathbf{x} can be either a single pixel or a group of pixels depending on the constraint aspects.

Different algorithms are formulated by different choices of $R(\cdot)$, $L(\cdot)$ and λ . In the BMME algorithms for example, $R(\cdot)$ operator is the assignment of just one (block) motion vector for the whole block of pixels. This is a smoothness

constraint for regularization. Furthermore, there are various algorithms each of which is based on stochastic models for regularization. The one which is based on a stochastic formulation is the “Gibbs Formulation” or equivalently “Markov Random Field” (MRF) modeling of the motion field.

1.3 Constrained and Stochastic Motion Models

One common approach to the motion estimation problem is by optic flow concept [9]. Optic flow refers to the distribution of instantaneous velocities of moving brightness elements in an image or video frame. Those elements can be the objects which are in the field of view of the observer and optic flow actually arises because of the relative motion of these objects and the observer. Actually optic flow is the main information source of moving objects, their spatial arrangements and structural features.

Optic flow estimation techniques are based on the assumption that the intensity of a pixel located at (x, y) on the image plane is constant over time. Let $I(x, y, t)$ represent the intensity at points on a path that is defined by $(x=x(t), y=y(t), t)$ in the 2-D image plane. Hence the following equation relates with optic flow:

$$\frac{\partial I(x, y, t)}{\partial t} = I_x \frac{\partial x}{\partial t} + I_y \frac{\partial y}{\partial t} + I_t = 0 \quad (1.4)$$

where $I_x = \frac{\partial I(x, y, t)}{\partial x}$, $I_y = \frac{\partial I(x, y, t)}{\partial y}$ and $v_x = \frac{\partial x}{\partial t}$, $v_y = \frac{\partial y}{\partial t}$. Thus the equation 1.4 becomes:

$$I_x v_x + I_y v_y + I_t = 0 \quad (1.5)$$

Since $V = [v_x v_y]'$ is the motion vector that we are looking for, equation 1.5 relates the 2-D motion with the gradient of the image and is true if the constancy of image pixel intensity assumption holds.

For regularization we need at least one more constraint to reverse the problem to a well-posed problem. Horn and Schunk [9] introduces two types of smoothness constraints. One is the sum of square of the motion field gradient:

$$\frac{\partial v_x^2}{\partial x} + \frac{\partial v_x^2}{\partial y} + \frac{\partial v_y^2}{\partial x} + \frac{\partial v_y^2}{\partial y} \quad (1.6)$$

and the other one is the square of the Laplacian operation:

$$\frac{\partial^2 v_x^2}{\partial x^2} + \frac{\partial^2 v_x^2}{\partial y^2} + \frac{\partial^2 v_y^2}{\partial x^2} + \frac{\partial^2 v_y^2}{\partial y^2} \quad (1.7)$$

Both constraints depend on the assumption that pixels close to each other tend to have the same velocity. As a result the weighted sum of optic flow terms in equation 1.5 and one of the constraint terms given in equation 1.6 or 1.7, is minimized. Thus we obtain regularized optic flow (motion) field. If the discrete estimates of the derivatives are well-behaved, resultant optic flow achieves a good estimate of the actual motion field.

Another approach for the regularization of the motion estimation problem is to model the motion field as a Markov Random Field (MRF). Equivalent stochastic model namely Gibbs distribution can be used under the positivity condition (i.e. $P(\mathbf{f}) > 0$). This type of modeling results in a maximum a posteriori (MAP) estimate of the motion field. Let us first define the model:

Definition: Let $\mathbf{f} = [f_i], i \in S$ be a collection of random variables defined on a regular lattice S . \mathbf{f} is called a MRF if it satisfies the following condition:

$$P(f_i | f_j, i \neq j, \forall j \in S), \forall i = P(f_i | f_j, i \neq j, \forall j \in N_i), \forall i \quad (1.8)$$

where $P(\mathbf{f})$ is the probability density function of the random field \mathbf{f} and f_i is the value of the distinct element at i in the field, S is the entire set of sites and N_i is the neighborhood of the site i . So this condition which is the basic assumption of MRF states that given only the elements in a predefined neighborhood of the i 'th element, the probability distribution for the i 'th element is independent from the rest of the elements.

Besag [20] and also Geman and Geman [21] present the equivalence of the

MRF and Gibbs distribution. They have proven that under the positivity assumption, random field \mathbf{f} is a MRF with respect to neighborhood N_i if and only if there exist a Gibbs distribution on the same neighborhood. Gibbs distribution allows to construct a local structure through potentials and energies that describe the interactions of each element in the field. The probability density function of the Gibbs distribution is given as:

$$P(\mathbf{f}) = \frac{1}{Z} \exp\left(\frac{-H(\mathbf{f})}{T}\right) \quad (1.9)$$

where $H(\mathbf{f})$ is the energy (Hamiltonian) that describes MRF, T is the temperature of the state and Z is the partition function that can be formulated as follows:

$$Z = \sum_{f_i \in S} \exp\left(\frac{-H(f_i)}{T}\right) \quad (1.10)$$

and in order \mathbf{f} to be a random field, the equation 1.10 should be always satisfied.

In order to achieve a well-regularized motion field, the hamiltonian $H(f)$ should be properly determined. A simple choice includes just two basic energy terms: namely “matching” and “smoothness”. Those terms can be chosen as follows:

$$H_M(\vec{\mathbf{d}}) = \sum_y \sum_x (I_t(x, y) - I_{t-1}(x - d_x(x, y), y - d_y(x, y)))^2 \quad (1.11)$$

$$H_S(\vec{\mathbf{d}}) = \sum_y \sum_x \sum_{i,j \in N_{xy}} (\vec{d}(x, y) - \vec{d}(x - i, y - j))^2 \quad (1.12)$$

$$H(\vec{\mathbf{d}}) = \beta_1 H_M(\vec{\mathbf{d}}) + \beta_2 H_S(\vec{\mathbf{d}}) \quad (1.13)$$

where $\vec{\mathbf{d}}$ is motion vector field, $I_t(x, y)$ is the intensity (or color) value of a pixel located at (x, y) . $H_M(\vec{\mathbf{d}})$ is the matching term which forces motion vectors to represent true displacements. It is a well-known term from the optic flow equation and sometimes related with posteriori distribution. $H_S(\vec{\mathbf{d}})$ is the

smoothness term which imposes the basic regularization to the motion field. That term forces the motion vectors to have similar values with their neighbor motion vectors. Minimization of the total energy function $H(\vec{d})$ maximize the probability distribution (Gibbs) so that we obtain MAP estimate of the motion field [4].

1.4 Motion Compensation and Video Coding

Motion compensation in video coding is the displacement of the previous frame by an amount of estimated motion field. This action satisfies a temporal redundancy reduction and therefore, it is the basic tool which makes temporal prediction between consecutive video frames.

Motion compensated predictive coding basically depends on the following observation: a sequence of video frames in general do not change so much and therefore, have temporal correlation with each other. That is to say that except for the newly exposed scenery, each pixel in the previous frame moves along a motion trajectory and hence, if the motion field of the image is known, a reasonable prediction of the current frame can be obtained by shifting and interpolating those moving parts of the previous frame accordingly.

Many motion estimation techniques have been shown to give good bandwidth reduction and image fidelity. The one which is most common and used in most of the VLBR video coding applications is the block matching motion estimation (BMME) algorithm [17, 22, 16, 12]. In BMME algorithms, current video frame is divided into blocks. The blocks are rectangular in shape and consist of certain number of pixels each of which is assumed to undergo the same displacement, and therefore, the pixels inside a block have the same motion vector. So the algorithmic task is to find a motion vector $\vec{d}_{i,j}$ for each block such that a suitable matching criteria is maximized. Therefore, the fundamental approach towards BMME algorithm can be formulated as follows:

$$\vec{d} = \arg \min \left[\sum_{x \in S} \Phi(I_x^t, I_{x-d}^{t-1}, \vec{d}_{i,j}) \right] \quad \forall \vec{d}_{i,j} \in D \quad (1.14)$$

where $\Phi(I_x^t, I_{x-d}^{t-1}, \vec{d}_{i,j})$ is the cost function and D is the search space on the previous image. x is the position vector inside the image S . Usually, search space consists of integer translations and the minimum is found by full search or by some iterative search techniques. Many other search techniques can be applied such as three step search [23], four step search [24], log(D) algorithm [25], and so on. Those techniques have been developed in order to reduce the massive computation required by the full search.

Experimental results show that block motion field of a real world image sequence is usually smooth and varies slowly. So it makes the coding of the block motion vectors efficient in terms of bit rate.

1.5 Block Motion Estimation Algorithms and Video Coding Implementations

In this approach video frames are divided into blocks, each of which is assumed to undergo the same translation and thus block of pixels have a single block motion vector (BMV). Block motion estimation algorithms are widely used in video coding applications [16, 22, 17] and the main contribution of those algorithms is that BMV's can represent rigid body motion field with the minimum number of motion vectors.

As shown in Figure 1.1, in order to find a BMV for a block centered at (x, y) , block of image pixels is taken at frame t and an attempt is made to find the best match for it within a search area in the frame $t - 1$. If D_{max} is the maximum displacement allowed to occur either horizontally or vertically, then the area of the searched region is given by: $SA = (M + 2D_{max})(N + 2D_{max})$ pixels.

The $M \times N$ block is moved in the search area till the best match is found. The distance between the block center and the center of the best match is considered to be the BMV of that block.

There are several BMME algorithms in the literature: Robbins and

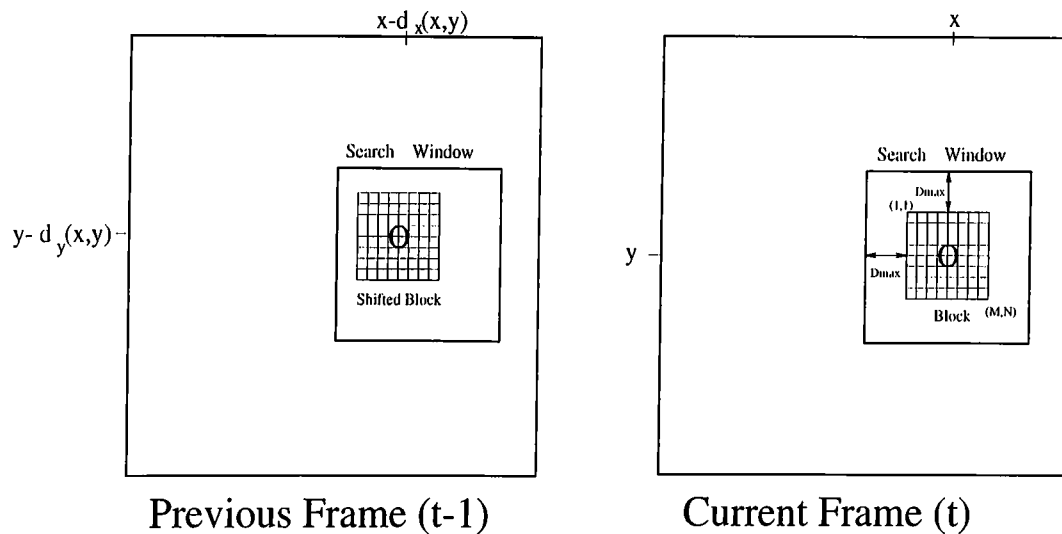


Figure 1.1: Given two frames, block motion estimation is performed within a search area in the previous frame

Netravali [26] evaluate an algorithm based on the steepest descent approach and the algorithm attempts to estimate a BMV by minimizing the square value of the displaced frame difference (DFD) which is defined as follows:

$$DFD(x, y, \vec{d}(x, y)) = I_t(x, y) - I_{t-1}(x - d_x(x, y), y - d_y(x, y)) \quad (1.15)$$

where $\vec{d}(x, y)$ is the BMV of the block centered at (x, y) , $d_x(x, y)$ and $d_y(x, y)$ are the x and y components of the BMV, respectively.

Houkes [27] presents a similar procedure using an iterative least squares linear estimation procedure. However Houkes includes a rotation and scale factor in addition to the translational motion vector. Jain and Jain [25] divide the image into fixed sized blocks whose best match is found by minimizing a distortion function between the consecutive frames.

An important issue for block matching is the block size. The visual degradation in block motion compensation is usually proportional with the block size. Smaller blocks generally reduce the visual degradation since any rotation (or nonlinear motion) can be better expressed by smaller block translations. So the performance of the BMME algorithms generally depends

on three factors: block size, matching criterion and search method. We have already discussed the first two of them. Search method is also important. Usually the search space consists of integer translations and the minimum is found by direct search techniques. The mostly used search technique is the exhaustive search. The disadvantage of the exhaustive search is that the computation time is proportional to the search area, but on the other hand, global minimum is always guaranteed to be found. Many different search techniques have been adopted such as cross search [28], three-step search [23], four step search [24] and etc. Those techniques are developed to reduce the computation time and also to find the global minimum in most of the cases. However, the main disadvantage for those techniques is that finding the global minimum is not always guaranteed.

The most famous video coding standards are MPEG phases. MPEG is an acronym for Moving Picture Experts Group which is under ISO-IEC/JTC1/SC29/WG11 and started its activity in 1988. There are two complete phases of MPEG namely MPEG-1 and MPEG-2. MPEG-1 is a standardization of coding for storage. MPEG-1 results that video and its associated audio can be stored and retrieved at about 1.5Mbits/s in a satisfactory quality.

In MPEG-1, images are in CIF format (Common Intermediate Format: 352x288) and frame rate is 30 frames/s. The draft of MPEG-1 has been finalized in June 1992. The second standard, MPEG-2, is intended for higher data rates than MPEG-1 (It is about 2-15 Mbits/s).

The last phase of MPEG, MPEG-4, mainly involves very low bit rate video coding (about several tens of Kbits/s) and has begun officially in 1993.

Another standardization organization is CCITT which formed a Specialist Group in 1984 toward a coding standard for visual telephony. In December 1990 the first picture coding standard (H.261) has been resulted [3]. Second standard is called H.263 and is primarily intended for very low bit rate video coding (about several tens of Kbits/s). The work of H.263 has been resulted in 1995.

In Appendix A, we present the motion estimation algorithms used in H.261

and H.263.

1.6 Scope and Outline of the Thesis

The scope of this thesis is the investigation of novel regularization techniques that can be used to remove the ill-posed behavior of the motion estimation problem for various applications, especially for video coding implementations. Moreover, the main contribution of this thesis is to obtain various motion field representations which can be coded efficiently.

We basically cast the motion estimation as a problem in minimization of an energy function which is formed by combining the motion constraints. Those constraints are related with the several requirements so that problem aspects can be realized. As a result, a distribution function (such as Gibbs distribution) is formed and minimized to obtain the MAP (maximum a posteriori) estimate of the motion. Therefore, the common feature of different motion estimation algorithms and related tools is the following idea: all motion estimation algorithms are modelled as GRFs such that we assign different Gibbs distributions according to the problem aspects. Therefore, all the algorithms are formalized by energy functions which differ by the constraints of the problem (and application requirements).

In Chapter 2, we focus on the regularization techniques for video coding. Since the most implemented model of motion estimation for video coding applications is the block matching algorithms, we are concerned with several advanced block matching algorithms in that chapter. Those proposed algorithms are designed to improve the performance of the block matching motion estimation (BMME) algorithms, as well as to reduce the visual perception degradation that is caused from the disadvantages of BMME algorithms. The basic disadvantages of the BMME algorithms are blocking artifacts in visual perception and redundant block motion field representation. As a result, in chapter 3 those disadvantages are shown to be reduced by using those proposed techniques.

In Chapter 3, some alternative motion estimation techniques, which can be used in a video coding implementation, are introduced. Since motion estimation is generally used for temporal prediction in a typical video coding application, we now refine the problem and present a different type of usage for motion estimation. Essentially, we present average motion determination and motion compensated interpolation concepts. Average motion determination can be used for frame rate adjustment. The frames which remain stationary can be detected by average motion determination algorithm, and those frames can be skipped. Then in the decoder side by motion compensated interpolation those stationary frames are generated artificially. So in chapter 4, we state whether or not the usage of those techniques can increase the coding performance of a particular video coding application.

In Chapter 4, we propose an alternative motion field representation which is the sparse field model. In that model, we present a motion field which is defined on the line field of the image. Since the line field consists of object boundaries which are the most important field carrying characteristic visual information of an image, we try to extract the motion compensated (MC) image from the MC line field. Further, we show that this technique can achieve high compression rate as a video coding implementation. However the visual quality is not so high, as expected.

Finally, Chapter 5 gives some interpretations about the results of the research presented in this thesis and outlines the further questions that arose in connection with the investigated algorithms that can be considered to be the subjects of future research.

Chapter 2

Regularized Motion Estimation

In this chapter regularized motion estimation algorithms which can be effectively used in very low bit rate (VLBR) video coding applications, are presented. In video coding applications, motion estimation is generally used to remove the temporal redundancy. Especially in VLBR video coding, motion estimation algorithms should be designed by taking the following three factors into account:

i) They should be accurate enough to provide an acceptable motion compensation, ii) they should have non-complex structure so that computation time would be suitable for real time execution, and iii) they should extract a motion field which can be coded effectively.

Block matching motion estimation (BMME) algorithms are the quite suitable candidates having the features described above. So it is not surprising that those are the most widely used algorithms in VLBR video coding implementations, devices and standards. Therefore, BMME algorithms are simple, fast and can be implemented in hardware very easily. But they also have serious disadvantages which can be stated as follows:

- i) BMME can cause degradations in visual quality such as blocking artifacts.
- ii) Since only the “best match” criteria is taking into account while finding

the BMVs, they can have quite arbitrary values which need high bit rate during coding.

iii) For any kind of motion (simple or complicated) between two video frames, always the same number of blocks (and BMVs) are used to represent that motion. In other words, number of blocks and block sizes are constant (predefined) and independent from the motion. That can cause redundant or insufficient usage of the blocks.

iv) BMVs are raster-scanned in the coding stage. That type of scanning breaks the vertical correlation between BMVs and therefore, the overall coding performance would be reduced as a consequence of this scanning process.

v) Global motion can not be efficiently represented by BMME algorithms. This problematic insufficiency are examined in detail in the sections 3.1, 3.2 and 3.3.

In order to overcome those disadvantages we develop some novel BMME algorithms which are presented in this chapter.

2.1 Block Motion Estimation by Energy Minimization

As discussed previously, classical BMME algorithms are usually designed by taking only matching criteria into account. Smoothness of the BMVs are only imposed by the constraint which is the assignment of only one motion vector per block. However, considering the coding efficiency this may be insufficient. Especially in VLBR video coding implementations further smoothness constraints may be necessary to achieve suitable BMVs for VLBR coding implementations.

In this section we introduce an advanced BMME algorithm which improves the BMME algorithm in order to overcome the above problem. In this approach, while searching the optimum BMV for a particular block, not only

the matching criteria but also the smoothness of the BMVs are taken into account. In order to do that, an energy function containing both matching and smoothness terms are formed and then minimized. The contribution of the smoothness term should be adaptive with respect to the block size of the algorithm. So we saw that a good way to do it is to reduce the smoothness constraint inversely proportional with the area covered by a particular block (also stated in [29]).

2.1.1 Energy Based BMME Algorithm

As stated in sections 1.3 and 2.2, we form an energy function like as in Equation 1.3. This energy function is given as:

$$Em_{xy} = \sum_i^{BS} \sum_j^{BS} (I_t(x+i, y+j) - I_{t-1}(x+i+d_x(x,y), y+j+d_y(x,y)))^2 \quad (2.1)$$

$$Es_{xy} = \sum_{i=-Nd}^{Nd} \sum_{j=-Nd}^{Nd} \|\vec{d}(x,y) - \vec{d}(x+iBS, y+jBS)\|^2 \quad (2.2)$$

$$E_{xy} = \beta_1 Em_{xy} + \beta_2 Es_{xy} \quad (2.3)$$

where E_{xy} is the total energy function for the block which is represented by the offset pair (x,y) . Those offset pairs can be the multiples of the block size BS (i.e., $(0,0)$ $(0,BS)$ (BS,BS) $(BS,0)$ $(BS,2BS)$ etc ...). Em_{xy} and Es_{xy} are the matching and smoothness constraint energy terms as before. The variables $d_x(x,y)$, $d_y(x,y)$ are the x and y components of the block motion vector $\vec{d}(x,y)$ as shown in Figure 1.1. Nd determines the size of the neighborhood.

All variables and energy terms are defined as blockwise and associated block is represented by the offset pairs (x,y) . $I_t(x+i, y+j)$ and $I_{t-1}(x+i+d_x(x,y), y+j+d_y(x,y)), \forall i, j \in [1..BS]$ are the pixel intensity (or color) values of the current and previous motion compensated special frames, respectively.

Minimization of the total energy function in Equation 2.3 for each block in the current special frame, with the block motion vectors as variables yields the

local optimum BMVs. Minimization method is the *Iterated Conditional Mode* (ICM) [30, 31].

In the minimization process, there are still some factors such as boundary problems which may cause unreasonable results. Although it was not shown in the previous energy expressions, effects of boundary problems are avoided by adding some “if” statements to those energy expressions. Those “if” statements restricts out of border situations. In Equation 2.1, the term $I_{t-1}(x + d_x(x, y), y + d_y(x, y))$ represents the shifted pixels by the candidate motion vector components. For the pixels on the image boundary, if resultant shift operation cause an “out of border” situation, an “if” statement gives infinite penalization. Therefore, such a situation is strictly avoided. Also in Equation 2.2, $\vec{d}_{x+BS_i, y+BS_j}$ are the neighbor motion vectors and the neighbor motion vectors which are out of border of the image are avoided by the same “if” statement as before by assigning infinite penalization.

Computational Complexity

In proposed algorithm, the only extra work is the computation of the “smoothness” energy term for each block. Since the calculation of “smoothness” term requires negligible computations with respect to the “matching” term of all the pixels in a block, computation time of our algorithm is slightly more than the classical BMME. Since the minimization process is carried out by ICM, the matching criteria of the pixels inside a block are once found and stored. Therefore, though the computation time is same as classical BMME algorithm, this technique requires much more memory.

2.1.2 Results

We simulate our BMME algorithm with two video sequences and test its performance with the classical BMME algorithm. The *PSNR* and *Bit Rate* graphics are given in Figures 2.1 and 2.2. Those results are obtained by taking one step compensation from the original frames. The BMVs are then entropy coded by LZV coding so that *Bit-Rate* graphic indicates the number of bits spent for coding of the BMVs. *PSNR* graphic shows the usual *Peak Signal to*

Noise Ratio which has the following formula:

$$PSNR = 10 \log_{10} \left(\frac{255^2 X_{size} Y_{size}}{\sum_i^{X_{size}} \sum_j^{Y_{size}} [I_t(i, j) - MC_t(i, j)]^2} \right) \quad (2.4)$$

where MC_t is the motion compensated image from the previous frame (I_{t-1}). In both simulations, BS is chosen as eight, and β_1 and β_2 values are 1 and 150, respectively. Frames are in QCIF format ($X_{size} = 176$, $Y_{size} = 144$) and they are gray-scale images (256 intensity levels with integer values form 0 to 255).

In Figures 2.3 and 2.4, resultant BMV fields, which are for the two typical video frames (*Mother & Daughter*, frames:10-11 and *Foreman*, frames:29-30), are shown. Those fields are obtained by using both classical and regularized BMME techniques.

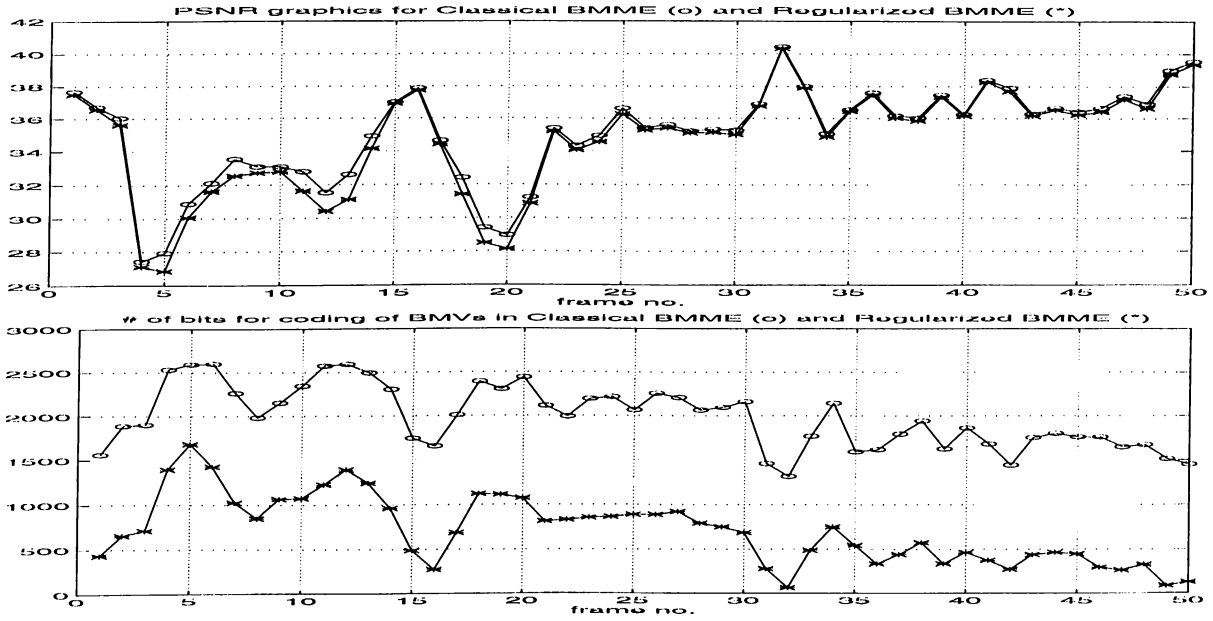


Figure 2.1: PSNR (*top*) and Bit-Rate (*bottom*) graphics of the classical (o) and regularized (*) BMME algorithms for the *Mother & Daughter* sequence.

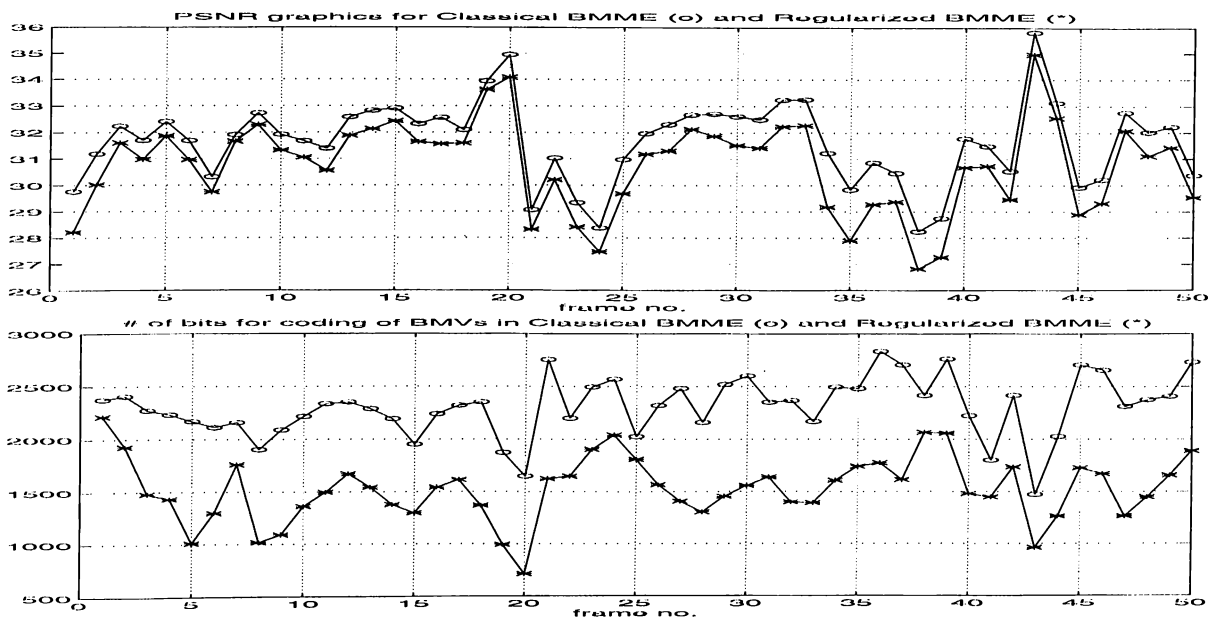


Figure 2.2: PSNR (*top*) and Bit-Rate (*bottom*) graphics of the classical (o) and regularized (*) BMME algorithms for the *Foreman* sequence.

We achieve quite good results: the number of bits spent for coding of the BMVs are reduced almost twice without any significant visual quality (or PSNR) reduction. However, we are still far away from our total objectives that are stated at the beginning of this chapter. Therefore, in the next section we focus on an adaptive BMME algorithm which can almost achieve the same visual perception quality (and also PSNR) while further decreasing the bit-rate for BMVs.

2.2 Adaptive Block Matching Algorithm

In this section, we propose an adaptive block matching algorithm which is shown to almost solve the problems explained at the beginning of this chapter. This algorithm, first of all, operates on the variable sized blocks such that the block size is determined adaptively by the matching criteria, i.e., if a good matching of a large (parent) block could not be achieved, by sub-dividing that parent block we try to improve the matching performance. So only the

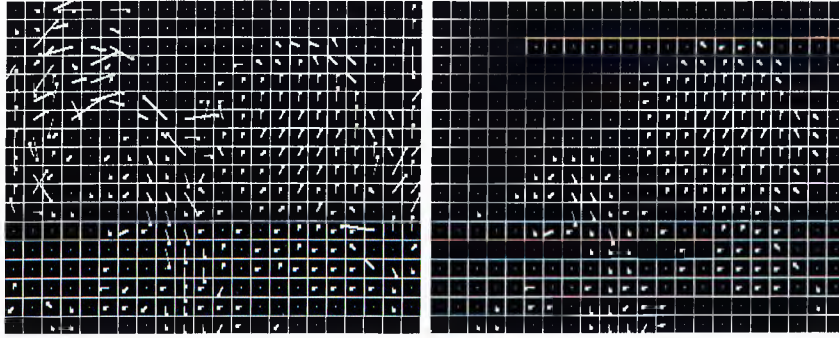


Figure 2.3: BMVs extracted from classical (*left*) and regularized (*right*) BMME algorithms for the frames (10-11) that are taken from the *Mother & Daughter* sequence.

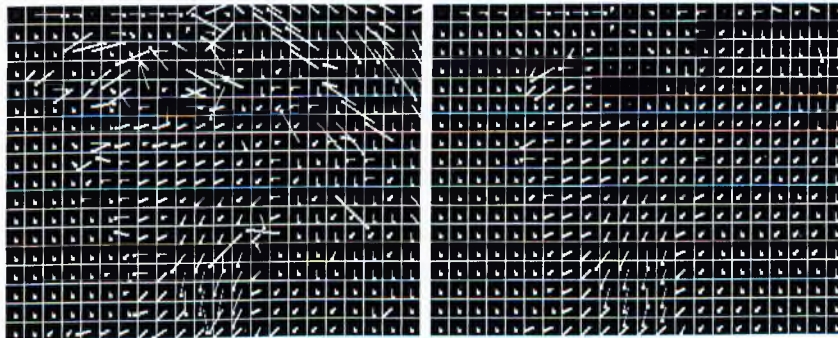


Figure 2.4: BMVs extracted from classical (*left*) and regularized (*right*) BMME algorithms for the frames (29-30) that are taken from the *Foreman* sequence.

least number of blocks, which are required to represent the motion between frames, are used. If a subdivision occurs, there are two possibilities: If the parent BMV has an acceptable matching score, it influences the child BMVs, otherwise child BMVs are determined independently. By this way, starting from the root parent block, that is the image itself, producing the child blocks if needed, we can achieve desired regularization. As a result, without significant visual degradation, we can obtain the reduced description of the (true) motion field in terms of bit consumption.

2.2.1 Adaptive Block Matching Algorithm (ABMA)

Before explaining the whole algorithm of ABMA, let us first define a few parameters as follows:

Matching Error: It is the indication of mean square error (MSE) for a block. The matching error of a block is equal to the ratio of sum of intensity difference squares to block area. So it is the average difference square (per pixel) for a block.

Depth: It is the number of the root parent block (image) sub-division.

Satisfaction Threshold: It is the maximum error for a candidate BMV to assign it as the BMV of that block. Above the satisfaction threshold, the block is sub-divided into four child blocks.

Effective Threshold: It indicates whether or not the parent block BMV affects the BMVs of the child blocks. The matching error which is above the effective threshold is assumed to give very bad matching performance so that parent BMV is now totally ignored.

Parent Multiplier: It is simply the parent block effect on the child blocks BMVs. If the matching criteria of a parent block is in between the satisfaction threshold and the effective threshold, that parent BMV is permitted to be used in the estimation process of the child BMVs.

Motion Estimation Algorithm

After the general parameters are defined, ABMA starts by taking the root parent block (image) as the current block and then finds the BMV which achieves the minimum matching error. The minimum matching error of the root parent block (image) is compared with the satisfaction threshold and effective threshold. If it is under satisfaction threshold, ABMA stops and no further sub-division is carried out afterwards. That means just one BMV which is the root parent block motion vector is sufficient to represent the motion between video frames (i.e., as in the case of global camera motions or a single object displacement as in Figure 2.5).

Otherwise, if the matching error is above the satisfaction threshold, parent block is divided into four child blocks. The child BMVs are determined by the minimization of an energy function that consists of two terms: “matching” and “parent resemblance”. Those terms and energy function are formulated as follows:

$$Em(b) = \sum_{i=Xst}^{Xend} \sum_{j=Yst}^{Yend} (I_t(i, j) - I_{t-1}(i + d_x(b), j + d_y(b)))^2 \quad (2.5)$$

$$Es(b) = \| (d_{par}^{\vec{}} - \vec{d}(b)) \|^2 \quad (2.6)$$

$$E = \sum_{\forall b \in I_t} (Em(b) + P_M Es(b)) \quad (2.7)$$

where E is the total energy function which is the sum of matching $Em(b)$ and parent resemblance $Es(b)$ terms for all blocks in the current frame. The coefficient P_M is the parent multiplier which has a nonzero value if the matching error of the parent block is under the effective threshold. The child BMVs are represented by $\vec{d}(b)$, $b = 1, 2, 3$ or 4 and their parent BMV is $d_{par}^{\vec{}}$. (Xst, Yst) and $(Xend, Yend)$ are the corner points of the block b . As before $I_t(i, j)$ and $I_{t-1}(i + d_x(b), j + d_y(b))$ are the current and previous compensated (by the BMV: $\vec{d}(b) = [d_x(b) \ d_y(b)]'$) blocks.

As a result the motion estimation is realized in a quad-tree structure such that each (parent) block inside the current frame is either sub-divided into four child blocks or finds itself a BMV by minimizing the energy term in Equation 2.7. Sub-division process is allowed to be continued up to a predefined depth value (i.e., zeroth depth represents the root-parent block, if process is over after the fifth consequent sub-division, $max. \ depth = 5$). In a certain depth, the block which is in that depth can have the size as $(\frac{Xsize}{2^{depth}} \times \frac{Ysize}{2^{depth}})$ (i.e., for a QCIF (176x144) image and $depth = 4$, a block has the size (11×9)). In practice maximum depth value can not be allowed to exceed five for the QCIF images.

Simulation results show that we can have better results if for each block the parent multiplier P_M is adaptively determined by using the depth. This is also an expected result because as depth increases (blocks become smaller) the relation between the child blocks and their parent block increases. This

is a consequence of the spatial correlation increase between smaller blocks. Since the block size reduction is proportional with the 2^{depth} , parent multiplier should be proportional with the same factor, that is:

$$P_M = P_M^0 2^{depth} \quad (2.8)$$

where P_M^0 is a constant real number which determines the amount of regularization for the BMV field. It is usually chosen according to the bit-rate for BMVs. For low bit-rate applications the value of P_M^0 is chosen larger than 8.

Back Propagation Process

This process is nothing but the grouping of the child blocks which have the same BMVs to a single parent block. If all the child blocks have the same BMV, there is no need to use four (same) BMVs for them instead of only one. Therefore, they are combined to create one parent block. Such a situation can occur in such a case: sometimes the sub-division of a parent block may not create the child blocks which achieve better matching than their parent block and thus all the child blocks can have the same BMV (which is their parent BMV). In such situations *Back Propagation* process reduces the number of blocks so that the number of bits spent for the coding of BMVs, are reduced.

Computational Complexity

We again compare the computation time with the classical BMME algorithm. In classical BMME algorithm the matching scores of every pixel in the image are calculated once in order to determine the matching score of the constant-sized blocks. Therefore, the computation time for our algorithm would be almost the same with the classical BMME algorithm because of the following reason: Since in the first depth, all the matching criteria (MSE) of the pixels are found once and then stored, for the remaining depths, only the parent resemblance term is to be calculated for each (child) block. That calculation requires only one subtraction and one multiplication for each block in the image and therefore, the computation time for it is negligible with respect to the calculation of the matching scores of the pixels. Thus the abovementioned result holds.

2.2.2 Results

We compare the performance of the ABMA with the classical block matching algorithm. First, consider the simple motion of a rectangle shown in Figure 2.5 (top). In this example, the frame size is (176x144) and blocks are (16x16) pixels for the classical block matching algorithm. So, there are 99 blocks to represent that simple motion where the BMVs are shown in Figure 2.5 (bottom-left). When the ABMA is applied to this example, with only one block which is the frame itself, the motion is represented as shown in Figure 2.5 (bottom-right).

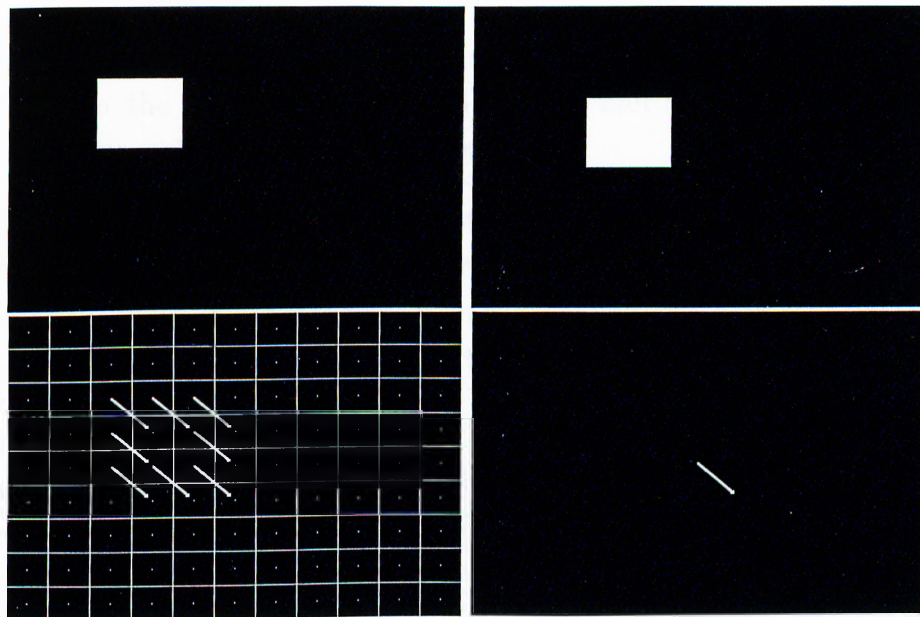


Figure 2.5: (top) Previous and current frames, (bottom) BMVs by (left) classical block matching algorithm, (right) proposed algorithm

The ABMA is also applied to “head and shoulders” type video frames as shown in Figures 2.6 and 2.7. The previous and current images are shown at the top. In the middle row, the motion compensated frames by using ABMA and classical block matching algorithm are shown. At the bottom, the BMVs and their associated blocks are illustrated. Also, in order to emphasize the effect of ABMA on the blocking artifacts compared to the classical block matching algorithm, some zoomed parts of the original video frames are shown in Figure 2.8. The parameters used in those simulations are given in Table 2.1.

in Table 2.1, P_M^0 values are chosen according to the amount of regularization

PARAMETERS	<i>WhiteRectangle</i>	<i>Mother&Daughter</i>	<i>Foreman</i>
Depth	1	5	4
Satisfaction Rate	25	25	25
Effective Rate	100	100	100
Parent Multiplier Const. (P_M^0)	0.5	3	3
Search Range	-10,+10	-7,+7	-7,+7

Table 2.1: Simulation parameters for *ABMA*

required. Normally we can choose P_M^0 value between 0 (for no regularization) and 20 (sufficiently high regularization factor even for depth 6). Therefore, according to the smoothness required, parent multiplier factor P_M^0 can be chosen as any real number in this range.

Now for the comparison between *ABMA* and the classical *BMME*, we sketch PSNR and Bit-Rate graphics of *ABMA* as shown in Figures 2.9 and 2.10. As shown in those graphs, *ABMA* is much better than the classical *BMME*, and even better than the regularized *BMME* which is discussed in the section 3.1.

Simulation results show that *ABMA* can reduce the number of bits spent for coding the *BMMs* approximately six times almost with the same PSNR values. The *BMVs* are entropy coded (same as before) so that Bit-Rate graphic indicates the number of bits spent for coding of the *BMVs*. PSNR graphic shows the usual *Peak Signal to Noise Ratio*.

2.3 Blockwise Coarse to Fine Segmentation of Motion Fields

In this section, we develop a hierarchical segmentation algorithm and a *BMME* algorithm which extracts a *BMV* field that is suitable for hierarchical segmentation. Segmentation is an efficient tool for *VLBR* video coding motion estimation algorithms. We believe that the number of bits which are spent for coding of motion vectors can be further decreased by means of segmentation.

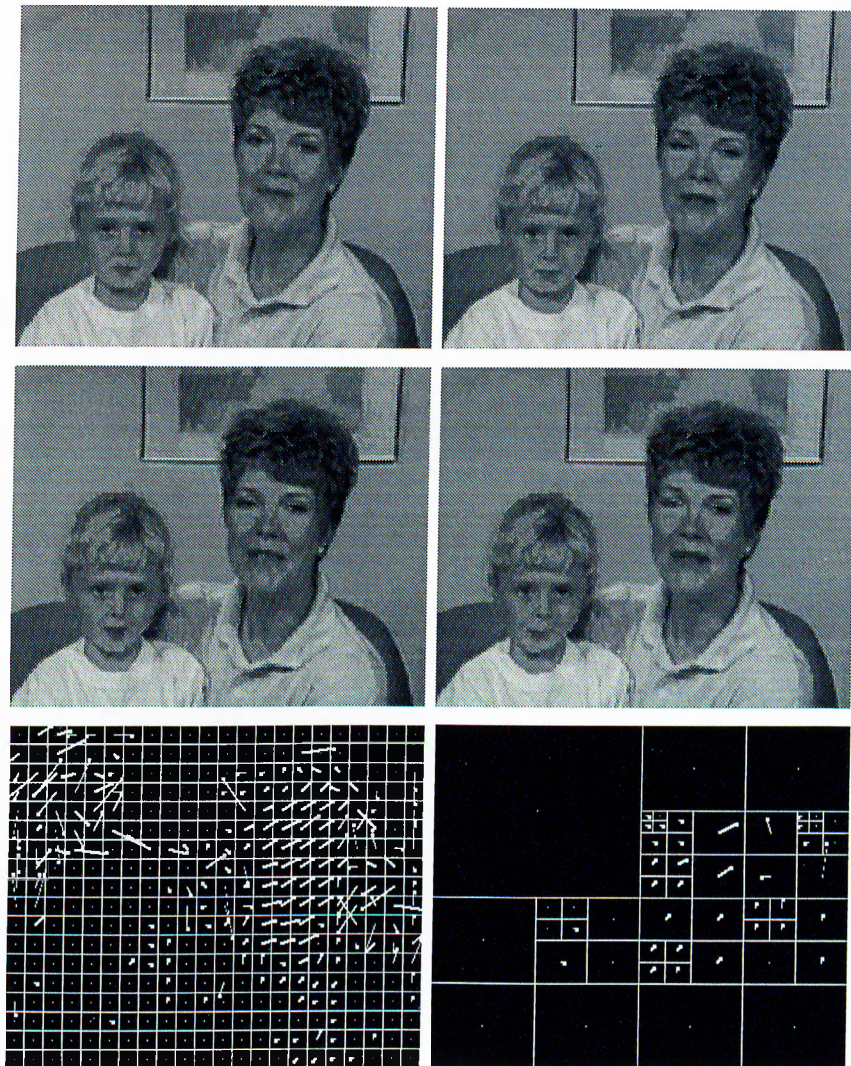


Figure 2.6: (*top*) Previous and current frames (*Mother & Daughter frames 78 & 81*), (*middle*) compensated frames by (left) classical block matching algorithm, (right) proposed algorithm, (*bottom*) BMVs extracted by (left) classical block matching algorithm, (right) proposed algorithm (depth=5).

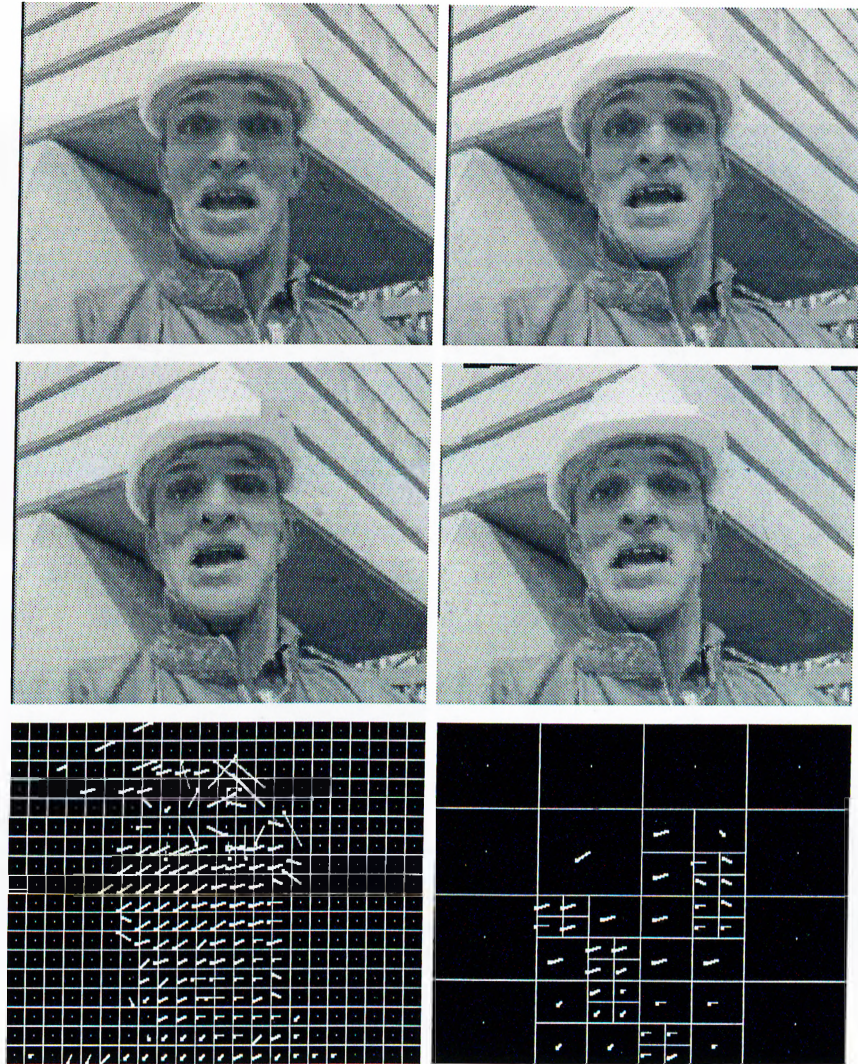


Figure 2.7: (top) Previous and current frames (*Foreman frames 66 & 69*), (middle) compensated frames by (left) classical block matching algorithm, (right) proposed algorithm, (bottom) BMVs extracted by (left) classical block matching algorithm, (right) proposed algorithm (depth=4).

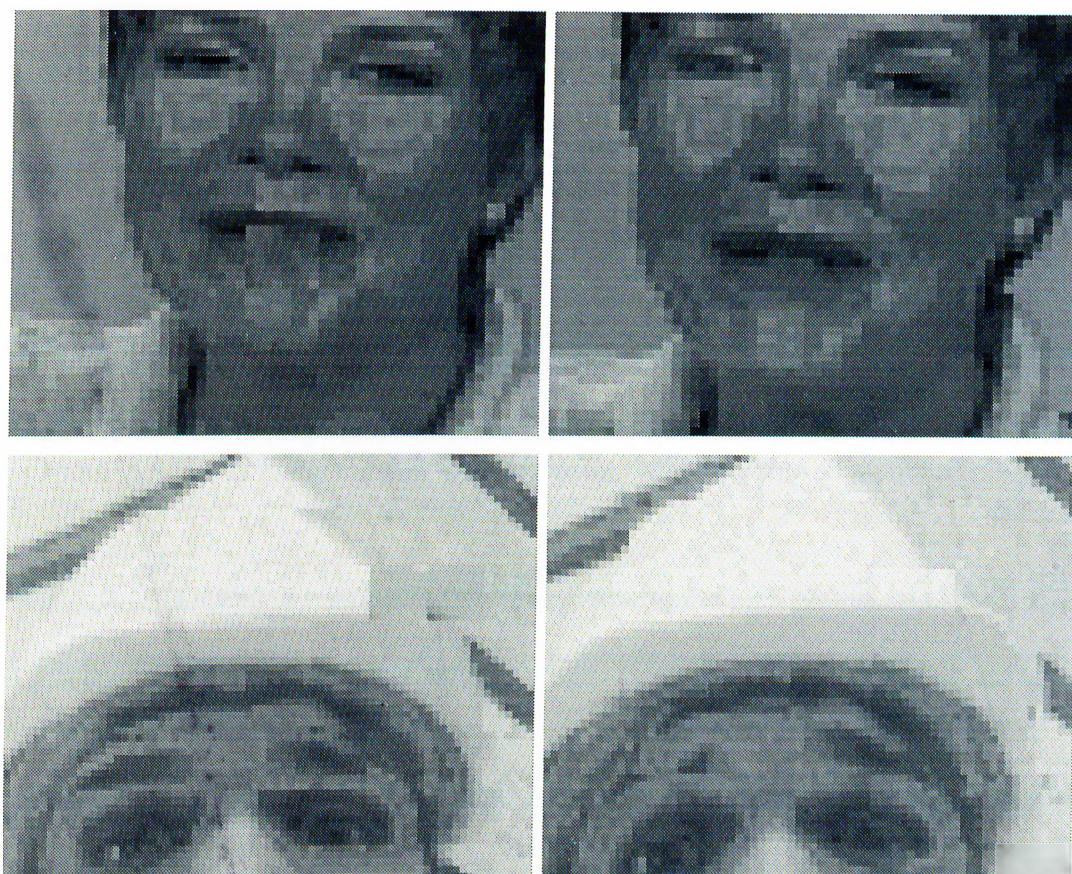


Figure 2.8: (top) *Mother & Daughter*, (bottom) *Foreman*, zoomed parts of the compensated frames by (left) classical block matching algorithm, (right) proposed algorithm.

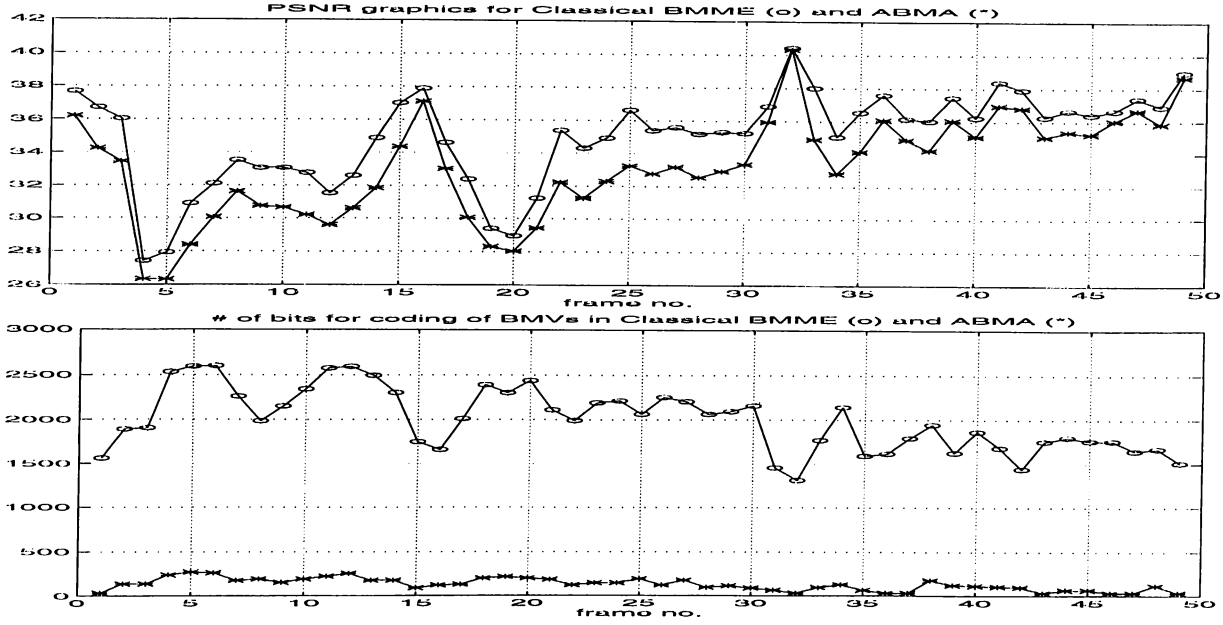


Figure 2.9: PSNR (*top*) and Bit-Rate (*bottom*) graphics of the classical BMME (o) and ABMA (*) algorithms for the *Mother & Daughter* sequence.

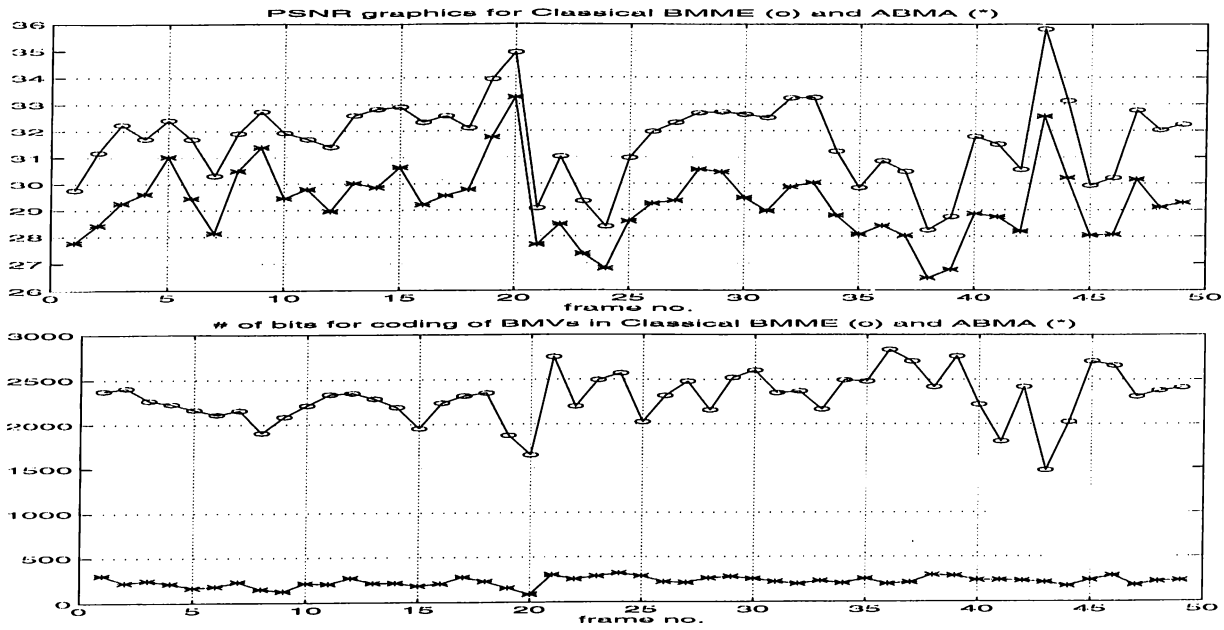


Figure 2.10: PSNR (*top*) and Bit-Rate (*bottom*) graphics of the classical BMME (o) and ABMA (*) algorithms for the *Foreman* sequence.

Segmentation of the block motion field is to combine the similar (or same) BMVs into groups so that the objects moving in a certain motion can be detected, extracted and then coded efficiently. In order to achieve this objective block motion field should have the following properties:

i) Block matching motion estimation (BMME) algorithm should extract a block motion vector field such that blocking artifacts in visual perception are minimal.

ii) Block motion field should be sufficiently smooth and correlated so that segmentation results in minimum number of segments.

iii) Block motion vector field should represent the global motion. That is to say that the motion between two video frames should be represented by the possible smallest number of BMVs so that segmentation can be performed by using minimum number of segments.

In the light of the abovementioned features, we propose a well-regularized BMME algorithm. This algorithm has a similar structure (quad-tree) as described in the previous section. However, there are certain differences at the other parts of the algorithm. For instance motion estimation criteria, parent regularization effects to the child blocks, division determination rule and the other basic structural features which are mentioned later in detail, are the basic different parts. Therefore, proposed BMME is an iterative algorithm which is repeated for every depth so that segmentation and block motion vector extraction are realised hierarchically (coarse to fine levels).

2.3.1 Hierarchical Block Matching Algorithm

This algorithm is executed in two main steps. In the first step block motion vectors are obtained and in the second step blockwise segmentation is applied. The execution is repeated for every depth (sub-division stage). Sub-division process is shown in Figure 2.11. At the first depth there is only one block which is the video frame itself, so there can be only one segment and its block motion vector. Then in the second depth parent block (frame) is subdivided into four

child blocks each of which is the quarter size of the frame. Block motion vectors are determined and segmentation is applied to those four blocks. The same algorithm is repeated for the remaining depths.

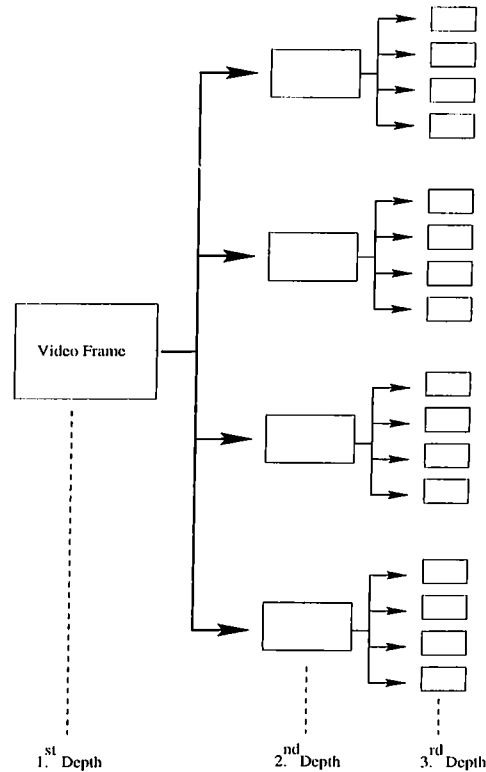


Figure 2.11: Sub-division process in Quad-Tree structure for the depths = 1,2,3,...

In any depth regularization is achieved by parent-child relation for the block motion vectors. So, in order to obtain a well-regularized block motion vector field, the parent block motion vector influences its child blocks in such a way that if the best matching rate for the child block is not greater than the parent matching rate multiplied with a coefficient, then, child block motion vector will be assigned as the block motion vector of its parent. By this way, from first depth to the last one, block motion vectors tend to have the same values as their parent block motion vector. Since all of the child blocks are generated from one root parent block, block motion vectors are forced to be similar with each other by the effects of parent blocks to their childs, and therefore, the final block motion vector becomes suitable for segmentation.

In this algorithm, the matching rate that is used in the determination of

the block motion vectors is the ratio of matched pixels to the total number of pixels in the block. Therefore, the matching rate is a real number between zero and one. In order to assign a displaced pixel as a “matched” pixel to a pixel in the current frame, the difference between the intensities (or colors) must be below a certain threshold value. Otherwise that pixel in that block is assigned to be “unmatched”.

After finding all block motion vectors for a certain depth, segmentation is achieved in the following way: first, stationary blocks ($BMV = 0$) are put in a segment which is called background segment. For the rest of the (moving) blocks, following algorithm is realized: each new block can join into a preformed segment if its BMV is in the neighborhood of that segment motion vector. Otherwise it forms a new segment and the motion vector of the segment is assigned as the BMV of that block. Thus all segments with their motion vectors are obtained for every depth by repeating this process.

2.3.2 Results

We test our algorithm by using some frames from the MPEG-4 test sequences as shown in Figure 2.12, 2.13, 2.14 and 2.15. In those Figures, top images are previous and current frames. At the bottom-left side, the resultant block motion vectors and their segmentations are illustrated. Finally, at the bottom-right side, motion compensated frame for the hierarchical block matching algorithm is shown.

The parameters used in the simulations are given in Table 2.2.

PARAMETERS	<i>Fig.2.5</i>	<i>Fig2.12.</i>	<i>Fig2.13</i>	<i>Fig2.14.</i>	<i>Fig2.15.</i>
Depth	1	3	3	4	4
Matching Threshold	20	7	7	7	7
Search Range	-10,+10	-7,+7	-7,+7	-7,+7	-7,+7
PSNR	(infinity)	29.0212	24.0913	29.0598	31.3692

Table 2.2: Simulation parameters for *Hierarchical Block Matching Algorithm*

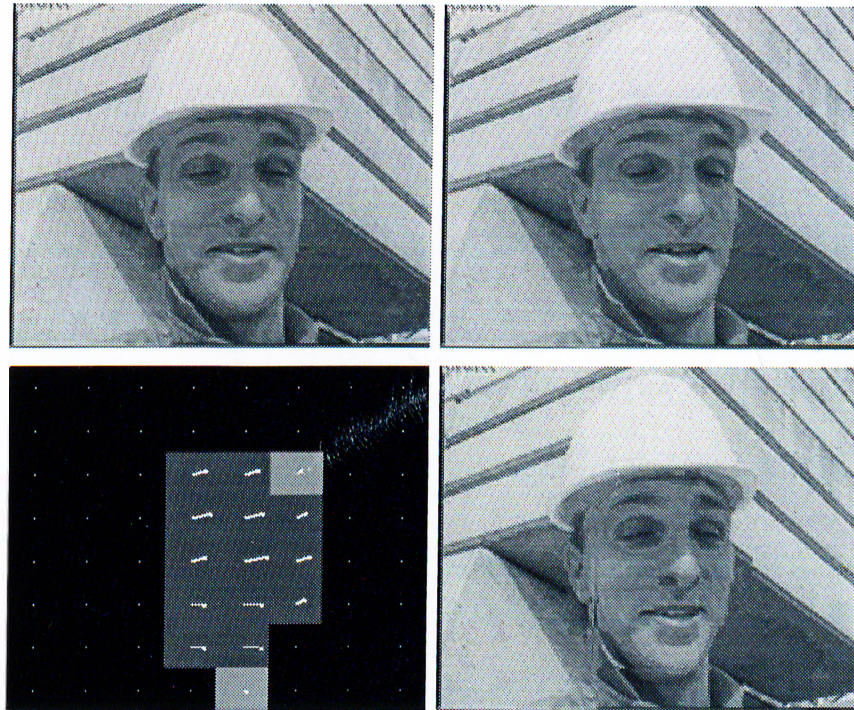


Figure 2.12: (*top*) Previous and current frames (*Foreman*, frames: 0 & 1), (*bottom*) (left) block motion vectors and segments (each gray-level shows different segmentation), (right) motion compensated image (PSNR=29.0212 dB, depth=3).

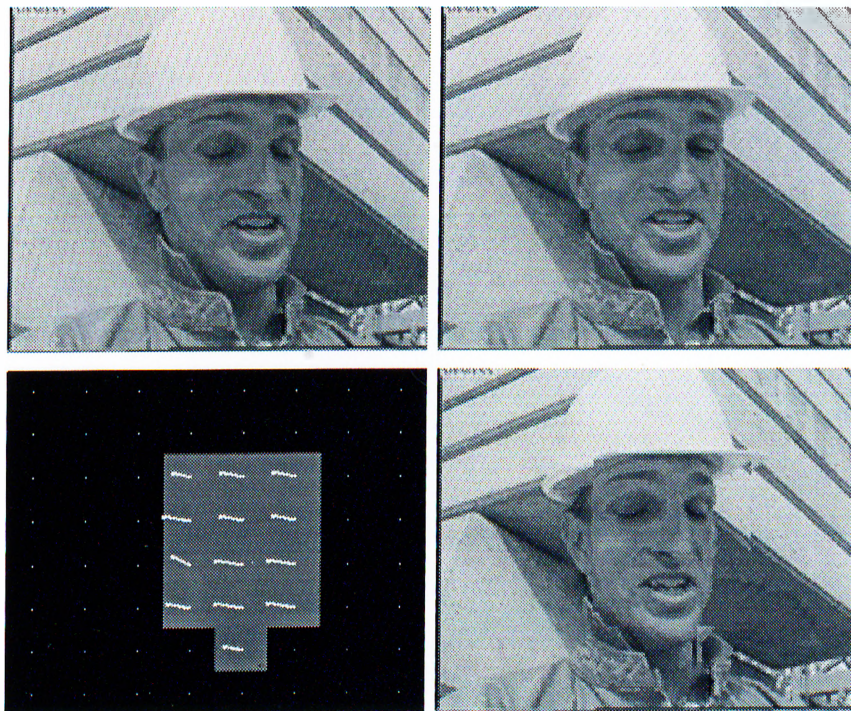


Figure 2.13: (*top*) Previous and current frames (*Foreman*, frames: 77 & 78), (*bottom*) (left) block motion vectors and segments (each gray-level shows different segmentation), (right) motion compensated image (PSNR=24.0913 dB,depth=3).

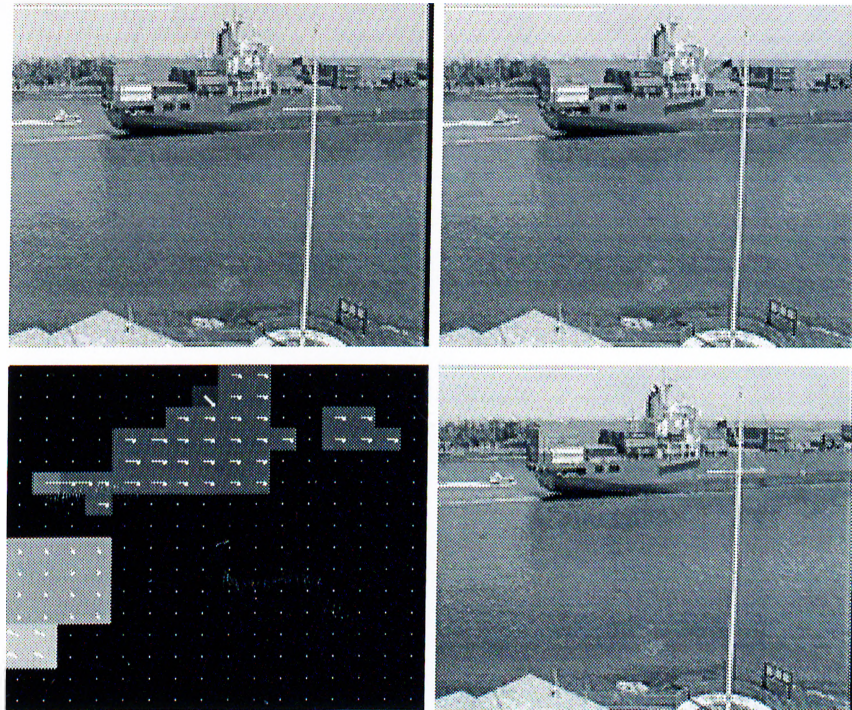


Figure 2.14: (top) Previous and current frames (*Container Ship*, frames: 61 & 81), (bottom) (left) block motion vectors and segments (each gray-level shows different segmentation), (right) motion compensated image (PSNR=29.0598 dB, depth=4).

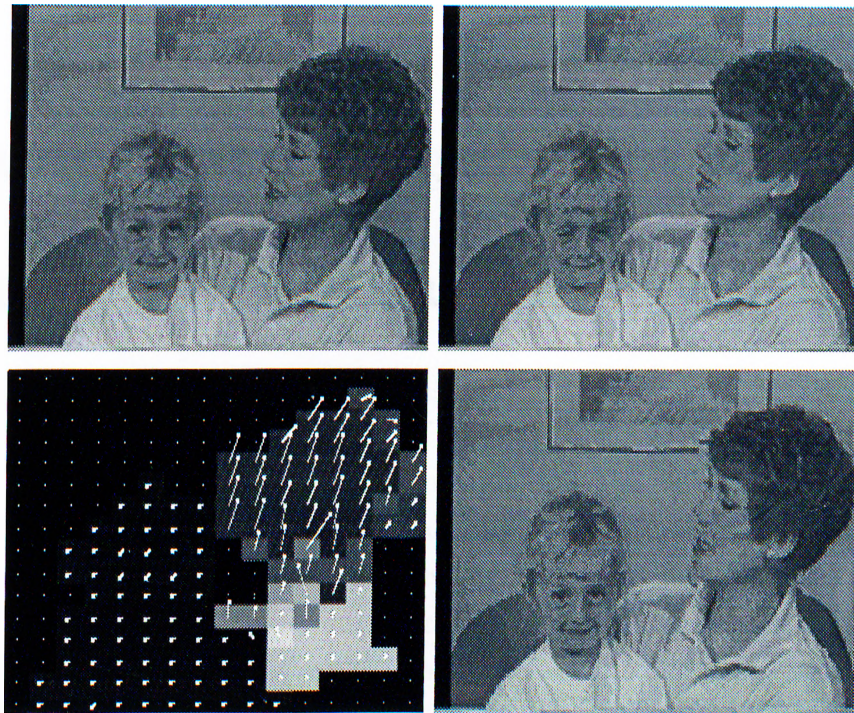


Figure 2.15: (top) Previous and current frames (*Mother & Daughter*, frames: 47 & 52), (bottom) (left) block motion vectors and segments (each gray-level shows different segmentation), (right) motion compensated image (PSNR=31.3692 dB,depth=4).

Chapter 3

Frame Rate Regulation and Frame Interpolation

Motion estimation (ME) and compensation are the basic tools which are used for temporal prediction in video coding applications. Therefore, motion compensation can realize high compressing rates and for this reason ME is usually supposed to be used only for the motion compensation (temporal estimation) of the frames. Nevertheless there are further applications of ME for video coding such as average motion determination (AMD) and motion compensated (MC) interpolation. We believe that if those tools can be used properly in a coding scheme they can improve the coding performance and also achieve higher compression rates.

In this chapter we introduce AMD and MC interpolation concepts and then we present novel ME techniques which can be used in these tools. We also discuss how they can improve the coding performance in a particular coding application.

3.1 Average Motion Determination and Frame Rate Regulation

In a video sequence there can exist a group of stationary frames. Those group of frames are very similar to each other since the amount of change is low. So especially in VLBR coding applications, those frames can be skipped if they can be estimated in the decoder (receiver) side. As a result instead of coding all the frames in the stationary group, just one of them is coded and the rest are assumed to be somehow reconstructed at the decoder side. The way of reconstruction is by “MC Interpolation” which is discussed in the next section. Then either the number of skipped frames or equivalently the “special frames” which are honored for coding are determined by AMD.

AMD, as its name implies, is an algorithm which finds the average motion between two video frames and thus finds the amount of change within two frames. In order to have accurate estimate of change between two frames, AMD algorithm should contain a well regularized ME algorithm so that it really determines the average object motion and it does not take the false transitions into account. The following section is about the ME algorithm used for AMD.

3.1.1 Motion Estimation Algorithm for AMD

In order to find the motion vectors between two video frames, an energy function (such as given in Equation 2.3) in terms of motion constraints, is formulated and then minimized [4, 6]. Energy terms are written in terms of one or more variables. That energy function is a local energy function which is defined in a predefined neighborhood in the image. Minimization is performed through a well known algorithm called ICM [30, 31].

In the applied motion estimation algorithm for AMD [8], the variables are vertical and horizontal components of the motion vector of a pixel located at (x,y) coordinates, and the binary line field [32, 5, 33]. The line field is used

to indicate the discontinuities of the motion vector field on the image. It is defined on the dual lattice as shown in Figure 3.1. The line field has two types of elements defined along vertical and horizontal directions such that vertical line field elements indicate the continuity or discontinuity of the motion vectors of the pixels which are neighbors to each other horizontally, and the horizontal line field elements indicate the continuity or discontinuity of the motion vectors of the pixels which are neighbor to each other vertically. Those line fields get the value 1 if a large amount of discontinuity between neighbor motion vectors exists. Otherwise, they get the value 0.

After defining line fields and motion vectors as such, in order to obtain local-optimum values for the motion vectors an energy function is formulated as follows:

$$Em_{xy} = [I_t(x, y) - I_{t-1}(x + d_x(x, y), y + d_y(x, y))]^2 \quad (3.1)$$

$$Es_{xy} = \sum_{i=1}^8 \| (\vec{d}_{xy}(i) - \vec{d}_{xy}(0)) \|^2 r_{xy}(i) \quad (3.2)$$

$$El_{xy} = \sum_{Nh} (L_{h,crossings} + L_{h,inclusion} + L_{h,parallel}) \quad (3.3)$$

$$+ \sum_{Nv} (L_{v,crossings} + L_{v,inclusion} + L_{v,parallel})$$

$$E_{xy} = \beta_1 Em_{xy} + \beta_2 Es_{xy} + \beta_3 El_{xy}. \quad (3.4)$$

Here, E_{xy} is the total local energy function associated with the pixel located at (x, y) coordinates. It consists of three different energy terms: matching term (Em_{xy}), smoothness term (Es_{xy}) and line field term (El_{xy}). $I_t(x, y)$ and $I_{t-1}(x + d_x(x, y), y + d_y(x, y))$ are the current and previous motion compensated image pixel color values at location (x, y) . In Equation 3.1, $d_x(x, y)$ and $d_y(x, y)$ are the vertical and horizontal components of the motion vector \vec{d}_{xy} , at the location (x, y) , respectively. In Equation 3.2, $\vec{d}_{xy}(i)$ is the motion vector and its location with the variation of i is shown in Figure 3.2. Also in Equation 3.2, r_{xy} is the uniformity field extracted from the line field as follows (see Figure 3.2):

$$r_{xy}(1) = (1 - l_{x-1, y-1}^h)(1 - l_{x-1, y-1}^v)(1 - l_{x, y-1}^h)(1 - l_{x-1, y}^v) \quad (3.5)$$

$$r_{xy}(2) = (1 - l_{x, y-1}^h) \quad (3.6)$$

$$r_{xy}(3) = (1 - l_{x+1,y+1}^h)(1 - l_{x,y-1}^v)(1 - l_{x,y-1}^h)(1 - l_{x,y}^v) \quad (3.7)$$

$$r_{xy}(4) = (1 - l_{x-1,y}^v) \quad (3.8)$$

$$r_{xy}(2) = (1 - l_{x,y}^v) \quad (3.9)$$

$$r_{xy}(6) = (1 - l_{x-1,y}^h)(1 - l_{x-1,y+1}^v)(1 - l_{x,y}^h)(1 - l_{x-1,y}^v) \quad (3.10)$$

$$r_{xy}(7) = (1 - l_{x,y}^h) \quad (3.11)$$

$$r_{xy}(8) = (1 - l_{x,y}^h)(1 - l_{x,y}^v)(1 - l_{x+1,y}^h)(1 - l_{x,y+1}^v). \quad (3.12)$$

Here, in Equations 3.5 to 3.12, $l_{x,y}^h$ and $l_{x,y}^v$ are the binary values of the horizontal and vertical line field elements which are defined on the dual lattice as shown in Figure 3.1. In Equation 3.3, $L_{h,crossings}$, $L_{h,inclusion}$, $L_{h,parallel}$, $L_{v,crossings}$, $L_{v,inclusion}$, $L_{v,parallel}$ are the real values which penalizes several line field positions shown in Figure 3.3. N_h and N_v are the neighborhood regions of the vertical and horizontal line elements ($l_{x,y}^h$ and $l_{x,y}^v$), and is taken to be one pixel back and forth from the location (x,y) as shown in Figure 3.3. Finally, total local energy expression is given by Equation 3.4.

Minimization of the total energy term in Equation 3.4 yields the local optimum solution of the correct motion between last previous frame and current candidate frame. Hence the average motion is calculated using the optimal motion vectors as follows :

$$sum_x = \frac{\sqrt{\sum_x^{X_{size}} \sum_y^{Y_{size}} d_x(x,y)^2}}{X_{size} Y_{size}} \quad (3.13)$$

$$sum_y = \frac{\sqrt{\sum_x^{X_{size}} \sum_y^{Y_{size}} d_y(x,y)^2}}{X_{size} Y_{size}} \quad (3.14)$$

$$Average\ Motion = \sqrt{sum_x^2 + sum_y^2}. \quad (3.15)$$

So, once the motion vectors of each pixel of the current image are found, by the formulas given in Equations 3.13, 3.14 and 3.15, the average motion is computed. As a result, if the average motion between those frames is over a predefined threshold value, then the current candidate frame is taken to be next special frame and it is honored for coding. Otherwise all the frames which have insufficient motion between the last special frame are assumed to be estimated by the decoder. Thus, those frames are skipped by the coder.

Computational Complexity of the Motion Estimation Algorithm

Since this is a dense motion field estimation, we have three factors which directly effects the computation time and determines the computational complexity: i) image size, ii) iteration number and iii) search range of motion vectors. Since the line elements take binary values for each pixel computation time for determination of the line elements can be ignored with respect to the computation time for the motion vectors.

The skipped frames are reconstructed at the receiver side from the special frames that are honored for coding. In the next section we show MC interpolation techniques for the reconstruction of the skipped frames.

3.2 Motion Compensated Interpolation

In a very low bit rate (VLBR) video codec (coder-decoder) temporal and spatial redundancies are removed from the video sequence in order to achieve high compression rates. In addition to that compression performance can be further improved by reducing the frame rate to be coded. By this way, the number of coded frames are decreased and this can realize significant bit savings. One way to do that is to skip certain (predefined) number of frames from the video sequence. Alternatively, a better approach for it is to skip certain number of frames which can be artificially generated at the decoder side within a acceptable loss of visual quality. The basic technique for the artificial generation is what is called “interpolation”.

Given two frames, interpolation generates certain number of skipped frames that are not honored for coding at the encoder side. The simplest interpolation is linear interpolation. In this technique the pixel color values are obtained by linear interpolation from the pixel color values of the given frames. The disadvantage of the interpolation by linear interpolation is that it causes visual artifacts such as blurring effects at the edges of the moving objects. Because this is visually unacceptable most of the time, more powerful techniques were developed such as motion compensated (MC) interpolation.

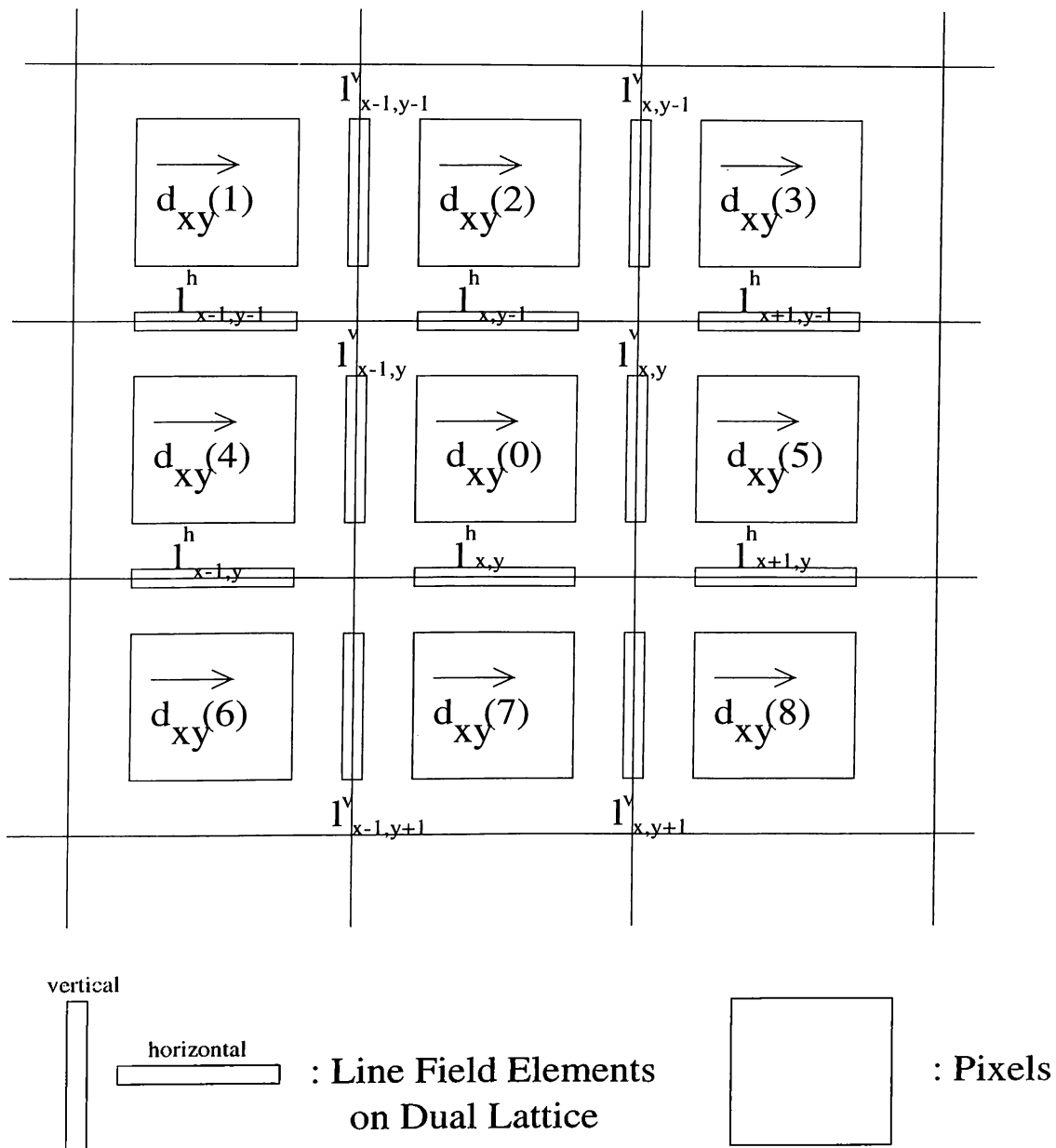


Figure 3.1: Line fields (horizontal and vertical) representation in dual-lattice.

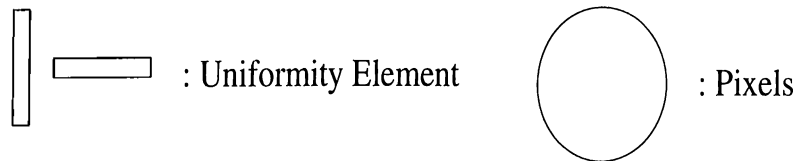
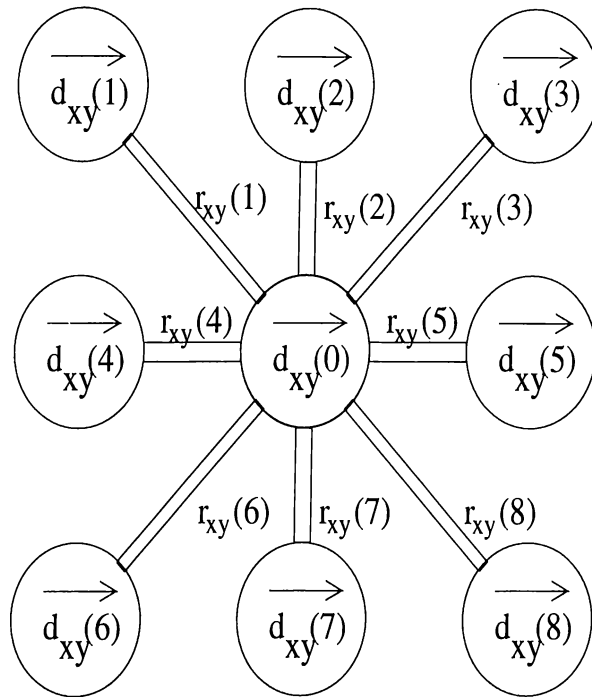


Figure 3.2: Uniformity field extracted from line field.

 <p>(0.0) (7.2)</p>	 <p>(0.0) (0.0)</p>	 <p>(0.0) (0.0)</p>	 <p>(0.0) (0.0)</p>		
 <p>(0.6) (0.6)</p>	 <p>(0.6) (0.6)</p>	 <p>(0.0) (0.0)</p>	 <p>(0.0) (0.0)</p>		
 <p>(4.8) (30)</p>	 <p>(4.8) (30)</p>	 <p>(1000.0)</p>	 <p>(3.2) (3.2)</p>		
L _{crossings}		L _{inclusion}		L _{parallel}	
: Line Process 'off' (l=0) : Line Process 'on' (l=1) : Pixels					

Figure 3.3: Particular positional penalization values of line field elements.

In MC interpolation schemes [34], motion estimation is performed between two frames and the interval frames are generated by fractional motion compensation. There are some serious problems that can cause visual degradations. First of all, there can exist some problems if the motion vectors are chosen as they indicate where all of the pixels in the previous frame tend to go in the current frame. One of them is what will happen to the unpointed pixels in the current frame. This is possible because motion vectors could not cover all of the pixels at the compensated frames. Another problem can occur if the motion vectors do not follow the object motion trajectory. In that case false transitions may occur and therefore, motion compensation in the interpolation will cause visual artifacts.

A simple interpolation method is “linear interpolation”. The basic idea behind it is to interpolate pixel intensities of the interval frames by using previous image pixels. For instance, suppose we are given two video frames such as I_t and I_{t+IN} where IN is an integer which shows the number of images which are generated artificially by interpolation. Let Im_{t+i} be the i^{th} interpolated image which is generated by the following formula:

$$Im_{t+i} = I_t + (I_{t+IN} - I_t) \frac{i}{IN}. \quad (3.16)$$

So for $i=1$ to $IN-1$, all interval frames are generated artificially by linear interpolation according to Equation 3.16. This method is extremely easy to implement but however it can have serious visual problems such as blurring artifacts that will be demonstrated later in this section .

Motion compensated interpolation techniques are developed in order to remove those blurring artifacts as well as to obtain motion tracked images. In these methods the interval frames are generated according to the simplest (and shortest) motion trajectory that can be found between those frames.

Because of the reasons explained previously, motion estimation is performed such that motion vectors indicate where all of the pixels in the current frame come from the previous frame. Therefore, every pixel of the current frame has a motion vector and by using those motion vectors, every pixel of the interpolated images can have a motion vector which is the fraction of the motion vector of the given current frame. Thus neither of the pixel remain

without motion vector. Also the motion estimation algorithm should be well regularized so that false transitions can be avoided. We use a regularized block motion estimation algorithm with sufficiently small block sizes so that blocking artifacts can be reduced. In that algorithm an energy expression containing matching and smoothness constraints for the block motion vectors, is first formed and then motion vectors are extracted by minimizing it. According to the regularized block motion estimation stated in section 3.1, we now present motion compensated (MC) interpolation techniques:

In this section, we present a motion compensated interpolation method which solves the problems mentioned previously. The basic ideas behind it are as follows: first of all, motion estimation is performed such that motion vectors indicate where all of the pixels in the current frame come from the previous frame. So, this solves the first problem mentioned before. Because it is now guaranteed that every pixel in the current frame is covered since they all have motion vectors. Secondly, the motion estimation technique is a well-regularized block matching algorithm which is a quite successful technique that can track the object motion and it was presented in detail in section 3.1.

3.2.1 Single MC Interpolation

Single MC interpolation as its name implies, is the interpolation of the interval frames only from one source which is the given previous frame. After the block motion estimation is performed between two given frames, the interval frames are obtained by fractional motion compensation from the given previous frame. Fractional block motion vectors are the uniform fractions of the block motion vectors that are obtained from the given frames and therefore, they are obtained by the following equation:

$$\vec{d}_{frac}^i = \frac{i}{IN} \vec{d} \quad (3.17)$$

where IN is the number of interpolated interval images, \vec{d} is the block motion vector between given frames and \vec{d}_{frac}^i is the block motion vector of the i^{th} (interpolated) interval image and of course it is not necessarily an integer pair.

Therefore, in single interpolation method the previous frame is first expanded by IN times in order to realize fractional motion compensation with integer block motion vectors and then i^{th} interval frame is generated by compensating the interpolated previous frame by the motion vectors $i.\vec{d}$ and then decimating the final image that is the expanded-motion compensated form of previous frame, by the factor IN . Single MC interpolation is illustrated in Figure 3.4.

One of the major disadvantage of this method is the loss of information in the generation of the interval (interpolated) frames. It is because given current frame is never used in the generation process of the interval frames. So in order to avoid from this problem we now present an advanced version of this technique which is discussed in the next section.

3.2.2 Double MC Interpolation

The main difference between double and single interpolation is that some of the interval frames are generated from the previous frame and some are generated from the current frame. Instead of calling previous and current frame, we now call frame 1 and 2 to the given frames. In double interpolation block motion estimation is performed twice such that one is between frame 1 and 2, one is between frame 2 and 1. So the nearest interval frames to frame 1 are generated by fractional motion compensation from the frame 1 with the block motion vectors on frame 1 (obtained at the first block motion estimation). Similarly, the nearest interval frames to frame 2 are generated by fractional motion compensation of the frame 2 with the block motion vectors on frame 2 (obtained at the second block motion estimation). Double MC interpolation is illustrated in Figure 3.5.

As stated before, the loss of information in the generation process of the interpolated images is avoided by this type of interpolation. However there can still be some visual artifacts in the interpolated frames because of the expansion of the frames 1 and 2 for the fractional motion compensation. Especially for the cases where IN is high, expansion by IN times of given frames will cause blurring effects especially at the object boundaries of the interpolated frames. So in order to solve this problem we now present another MC interpolation

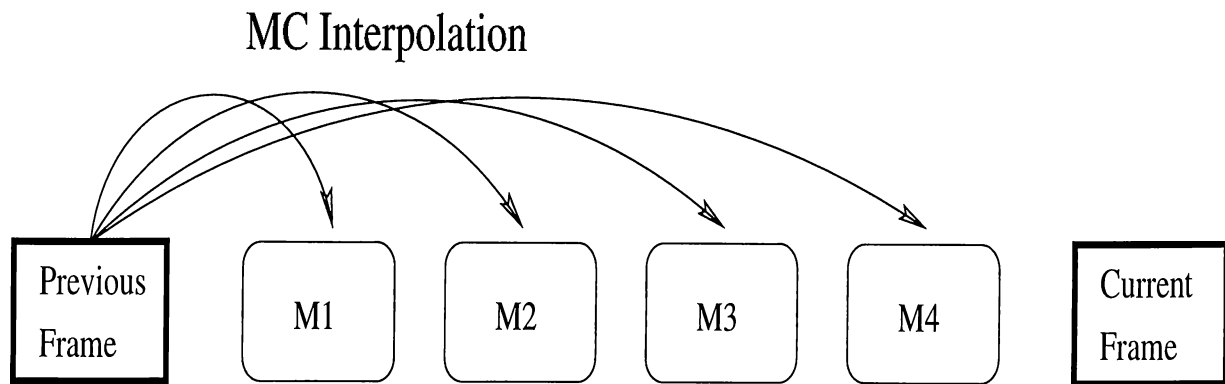


Figure 3.4: *Single MC interpolation* of the interval frames M1,M2,M3 and M4 from the given previous and current frames. (IN=5).

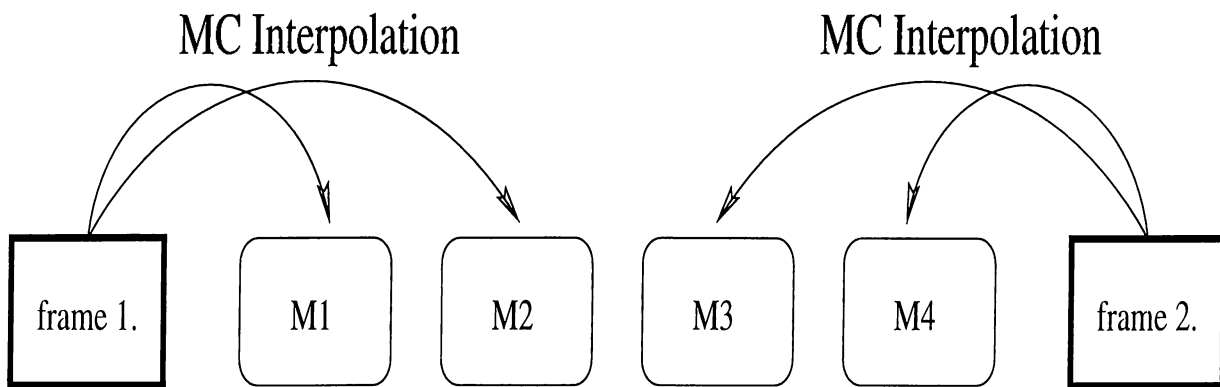


Figure 3.5: *Double MC interpolation* of the interval frames M1,M2,M3 and M4 from the given frame 1 & 2. (IN=5).

technique which is discussed in the next section.

3.2.3 Binary Tree Structured MC Interpolation

The main contribution of binary tree structured (BTS) interpolation is that the visual artifacts such as blurring can be reduced. The basic intuition behind BTS interpolation is that interval frames are generated one by one (at a time) by double MC interpolation between two frames. In other words we do not follow the general procedure of the other interpolation techniques (i.e., expansion by IN times, motion compensation and decimation by IN times). Instead, we generate every interval frame one by one by the following procedure: First block motion estimation is performed between two given frames and then one frame which is the $[\frac{IN}{2}]^{th}$ frame of the interpolated frame sequence, is generated by MC interpolation (twice expansion + motion compensation + twice decimation). Therefore, the blurring effect is caused by twice expansion (not IN times expansion). Then the same procedure is performed between frame 1 and that interpolated frame and also between frame 2 and that interpolated frame. By this way, $[\frac{IN}{4}]^{th}$ $[\frac{3IN}{4}]^{th}$ frames of the interpolated frame sequence is generated from given frames by just twice expansion. So this procedure is continued as generating always half-way frames at a time, until all the interpolated frames are generated. BTS interpolation is illustrated in Figure 3.6.

As already discussed, this interpolation method like the double MC interpolation generates the interval frames by using both frame 1 and frame 2. When the number of interpolated images is high it reduces the blurring effects as compared to other MC interpolation methods.

3.2.4 Results

We have tested the interpolation algorithms using frames from *Akiyo* and *Container Ship* sequences shown in Figure 3.7. We artificially generate eight ($IN=8$) interval frames but for illustration only 2^{nd} , 4^{th} and 6^{th} frames are

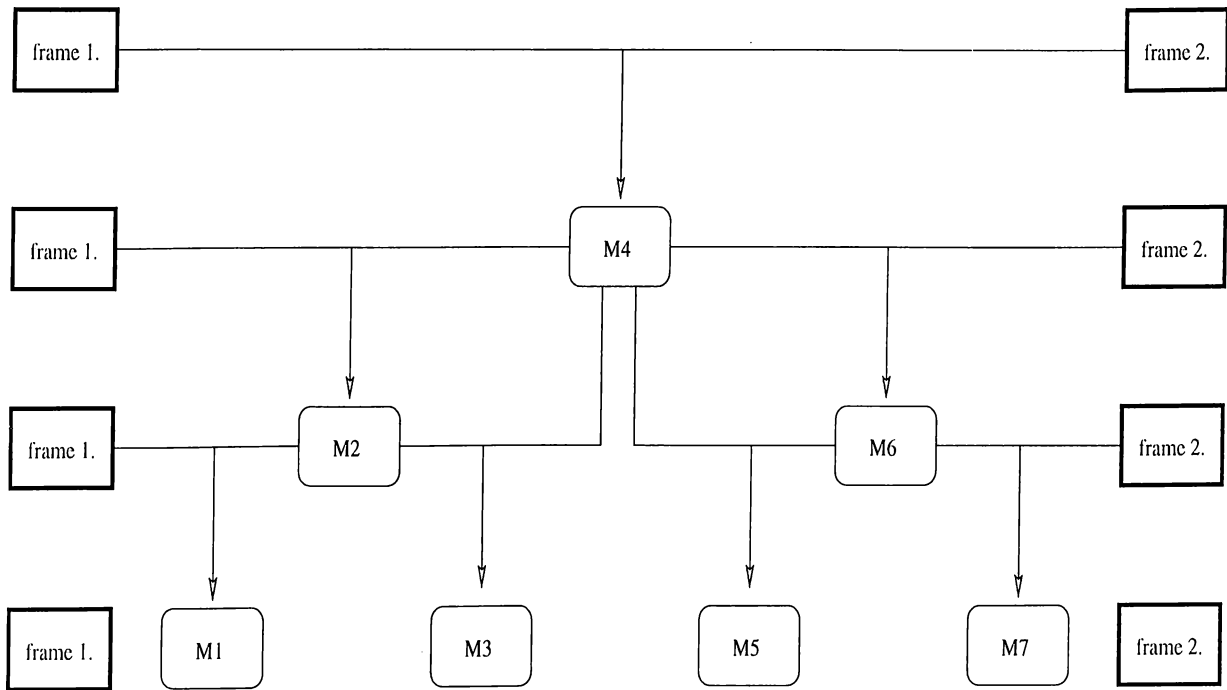


Figure 3.6: *Binary tree structured MC interpolation* of the interval frames $M1...M7$. $M1, M2, M4$ is generated from frame 1, $M7, M6$ is generated from frame 2 and $M3, M5$ is generated from $M4$ by MC interpolation. Note that except $M3, M5$, other interval frames are compensated from the original frames. ($IN=8$).

shown in Figures 3.8 and 3.9. The tested interpolation methods are (as in the order top-bottom for the Figures 3.8 and 3.9), linear interpolation, single MC interpolation, double MC interpolation and binary tree structured MC interpolation. For the *Akiyo* frames, forward block motion estimation (between frame 1 & 2) results in a PSNR value about 31.083618 dB and backward block motion estimation (between frame 2 & 1) results in a PSNR value about 31.798702. Those PSNR values for *Container Ship* frames are about 28.466227 dB and 28.560457 dB respectively. Block motion estimation is performed between (-8,+8) pixels and block size is 8.

According to simulation results we can claim that MC interpolation techniques can achieve better visual quality than the linear interpolation. Moreover, among the already presented MC interpolation techniques BTS interpolation is the best but on the other hand for small values of IN performance of the BTS interpolation will be approximately equivalent to the performance of the double interpolation.

Computational Complexity in MC Interpolation Algorithms

There are two factors which determine the computation time of the algorithms: i) motion estimation and ii) fractional motion compensation. Since motion estimation process is the regularized BMME algorithm which is presented in section 3.1, computation complexity is also given in section 3.1.

Fractional motion compensation requires expansion of the previous and current images by the number of interval frames. In single MC interpolation only the previous frame is expanded. In double MC interpolation previous and current frames are both expanded. So this requires twice (expansion) time compared to the single MC interpolation. Finally in binary tree structured MC interpolation all frames are expanded by two and this is repeated by the number of frames. So this requires significantly less amount of computation time for the expansion. On the other hand, the computation time for motion estimations (between interval frames) is increased by the number of interval frames. According to the simulation results, for small number of interval frames the computation time of the MC interpolation algorithms are almost equal to each other.

Expansion of the images requires so much memory to hold the expanded images. Instead of expansion of the whole images, one can calculate the fractional motion compensation result for each pixel by just expanding one pixel neighborhood of that pixel and then finding the fractional MC pixel color (or intensity) value. However, since this calculation must be processed for each pixel in the image, it requires eight times more calculation with respect to the expansion of the whole images. This is because there are eight neighbor pixels of a single pixel and the expansion is repeated for every pixel in the image. This means that we have eight times more calculation. So this local expansion method requires less memory but is slower than the previous method.

Simulation results show that the most important disadvantages of MC interpolation algorithms are false transitions and visual artifacts (i.e., blurring and blocking artifacts). It would be better to use linear interpolation with the MC interpolation whenever those artifacts cause significant visual degradations.

MC interpolation algorithms differ in the fractional block motion compensation process. It is obvious that double MC interpolation algorithm gives better results than the single MC interpolation algorithm. This is due to the fact that the double MC interpolation technique is realized by using much more information with respect to the single one. According to the simulation results we also observe that BTS interpolation and double MC interpolation achieve similar performances while the number of frames are less than six. Otherwise the most powerful one is the BTS interpolation because the frames are only expanded twice and thus the blurring caused from the expansion is minimized.



Figure 3.7: Given previous and current frames are 120th and 150th frames of the *Akiyo* sequence (*top*) and 60th and 90th frames of the *Container Ship* sequence (*bottom*).



Figure 3.8: Linear interpolation (*top*), single MC interpolation, double MC interpolation, binary tree structured MC interpolation (*bottom*). Given previous and current frames are 120th and 150th frames of the *Akiyo* sequence.

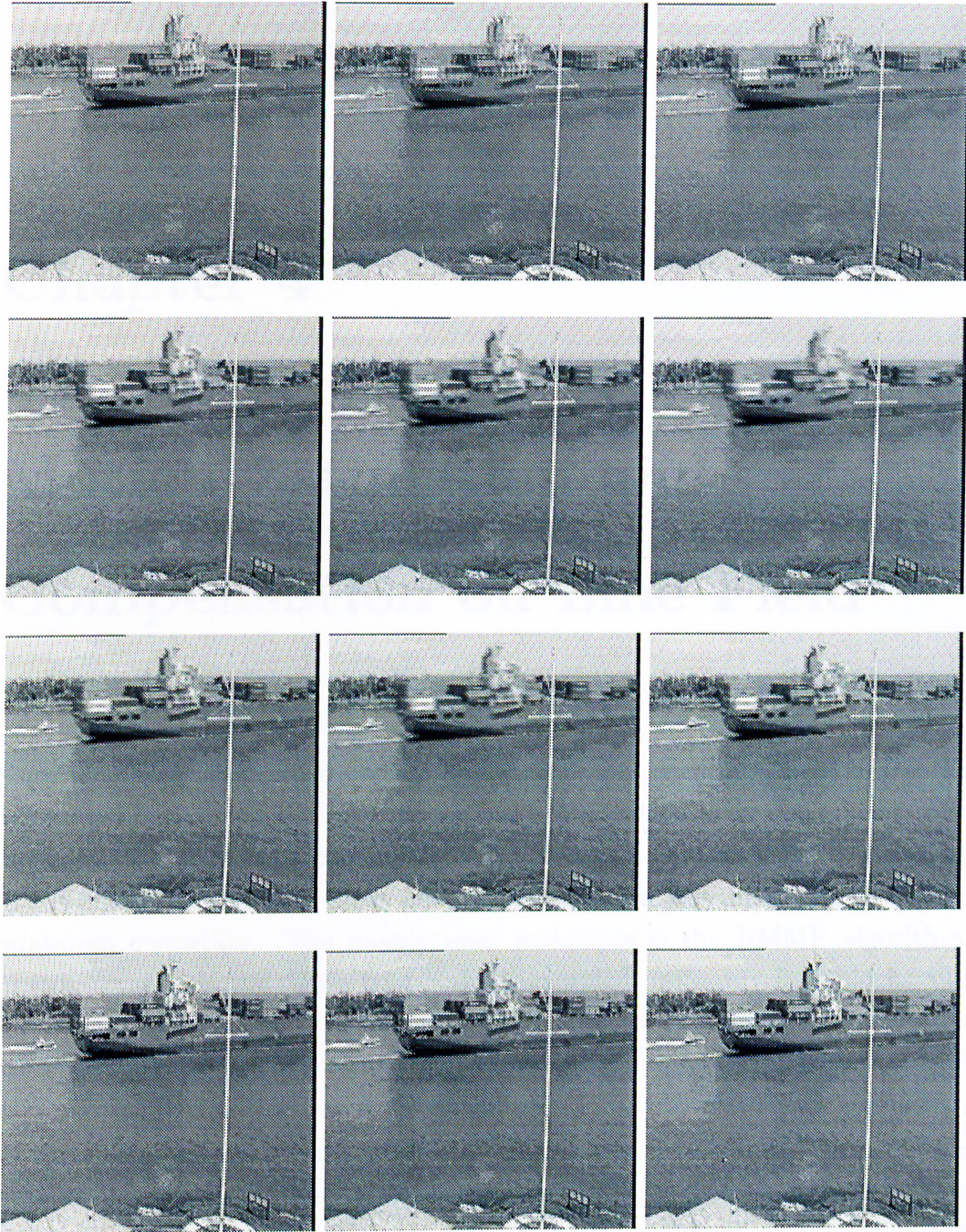


Figure 3.9: Linear interpolation (*top*), single MC interpolation, double MC interpolation, binary tree structured MC interpolation (*bottom*). Given previous and current frames are 60th and 90th frames of the *Container Ship* sequence.

Chapter 4

Motion Estimation and Compensation on Line Field

Especially in VLBR video coding applications dense motion field has to be decimated because of the bit-rate requirements. The aim in decimation is to represent the dense motion field by the decimated field as good as possible. In other words, the goal is to find the decimation scheme which gives the minimum distortion. The mostly used technique is the BMME algorithm. In this technique, the dense field is down-sampled uniformly into blocks and the motion field is represented by BMV field with the reduced information approximately by the block area. But the important question is why this kind of (uniform) decimation has to be optimum? So in this section we present an alternative sparse field representation for the motion field. Firstly we introduce the sparse field scheme which is called “line field” [32, 33, 5]. Then we show how the motion vectors can be defined on the line field and finally we describe the reconstruction of the motion compensated image from the motion compensated line field.

4.1 Line Field Definition and Extraction

Line field in a typical image is first proposed by Geman and Geman [21] for image restoration. As discussed in [8], forming an energy function including the line field elements leads to a non-convex optimization problem. Both stochastic and deterministic optimization methods using local iterative optimization have been applied and compared in [11, 4]. As discussed in section 4.1, line field of an image involves such a sparse region that shows basic properties such as intensity (or color) discontinuity and spatial continuity [32, 33, 5]. According to this definition one can claim that line field implies the object boundaries. Therefore, it is visually important field in an image. Recently the line field detection and reconstruction has been shown to be a good technique to image compression [35]. Additionally it is well known that human visual system for pattern recognition and motion estimation works based on the detection of the line field [36, 37].

In the work of image compression by line field, it has been shown that line field can reproduce a degraded version of its original image [38, 35]. So in lossy image coding applications, instead of coding the original image (the dense data), coding the line field and then reconstruction of the original image can be an alternative approach.

Extraction of the line field from the image is an important work and should be realized in such a way that spatially discontinuous intensity discontinuities should not be included into the line field. Those points are what we called “details” and are not object boundaries. So a line field involves the spatially continuous intensity discontinuities, and therefore, details should be excluded.

As a result line field differs from the discontinuous (intensity) points of an image. So we choose a similar formation of line field in [8] rather than any discontinuity detection algorithm. Figure 4.1 indicates the difference between line field and discontinuities that are obtained by $\nabla = \frac{\partial}{\partial x} + \frac{\partial}{\partial y}$ operation.



Figure 4.1: The discontinuities (obtained by ∇ operation) (*left*) and line field (*right*) of the 40th frame in *Mother & Daughter* sequence.

4.1.1 Line Field Extraction

Line field is extracted by forming an energy function (see section 5.1) and minimizing it. That energy function is formed in such a way that line field can represent the object boundaries. Accordingly the energy function is formulated as follows:

$$H_G(U) = \sum_x \sum_y (I(x, y) - U(x, y))^2 \quad (4.1)$$

$$H_S(U, l) = \sum_x \sum_y \sum_r \sum_t (U(x, y) - U(x - r, y - t))^2 h(i_{r,t}) \quad (4.2)$$

$$H_L(l) = \sum_{N_h} (L_{h,crossings} + L_{h,inclusion} + L_{h,parallel}) \quad (4.3)$$

$$+ \sum_{N_v} (L_{v,crossings} + L_{v,inclusion} + L_{v,parallel})$$

$$H(U, l) = \beta_1 H_G(U) + \beta_2 H_S(U, l) + \beta_3 H_L(l) \quad (4.4)$$

where U and l are the image and line field variables, $h(i_{r,t})$ is the uniformity field which is shown in figure 3.2 and $i_{r,t}$ is the integer number which takes values between 1 and 8, according to the (r,t) pair.

Line field l is defined on dual lattice and has two types: vertical l^v and horizontal l^h . Those fields are illustrated in figure 3.1. In Equation 4.4, $L_{h,crossings}$, $L_{h,inclusion}$, $L_{h,parallel}$, $L_{v,crossings}$, $L_{v,inclusion}$, $L_{v,parallel}$ are the real values which penalizes several line field positions that are shown in figure 3.3. N_h and N_v are the vertical and horizontal line elements neighborhood regions, and are taken to be one pixel back and forth from the location (x,y) as shown in the figure 3.3. Line field is again a binary field, that is, it indicates discontinuity for $(l=1)$ and otherwise continuity for $(l=0)$.

Equation 4.1 presents similarity between image field variable U and original image I , Equation 4.2 is for extracting all intensity discontinuities of an image and Equation 4.4 is the necessary term in order to discontinuities be the line field by forcing spatial continuity of the edges. Finally, Equation 4.4, as the total energy function, is minimized to get the line field of the image.

4.1.2 Results

We test the line field extraction program by using real world frames shown in figure 4.2. The extracted line fields are shown in figure 4.3. Horizontal and vertical line fields are represented in the left and middle columns. At the right, part those fields are combined by an *OR* operation in order to illustrate the total line field of each image.



Figure 4.2: Original frames: 20th frame in *Container Ship* sequence and 40th frame in *Mother & Daughter* sequence.

According to the simulation results we saw that line field of the image covers

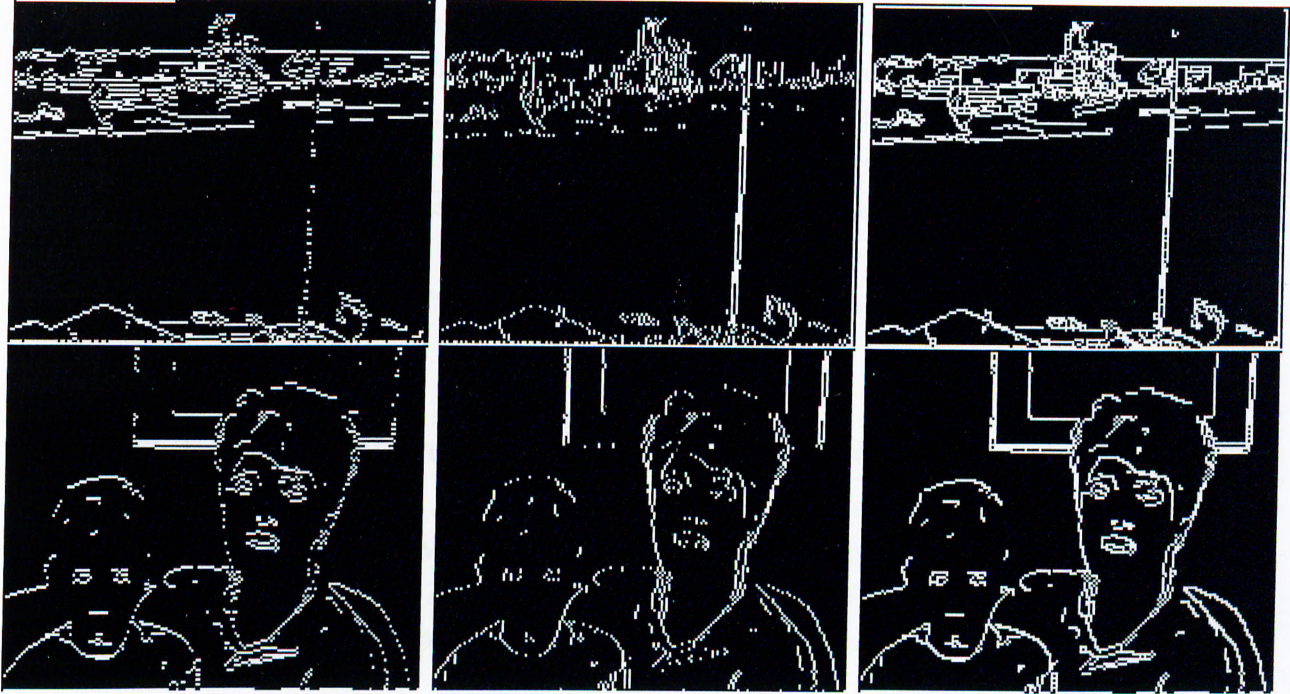


Figure 4.3: Line fields (horizontal, vertical and both) of the original frames.

the object boundaries. In addition to that this technique can be used as an image compression technique since line field is a sparse field and it involves most of the characteristic information of an image.

In the next section, we try to assign specific motion vectors on line field in order use this technique for video coding applications.

4.2 Motion Vectors on Line Fields

As mentioned in the previous section line field of an image can be used for still-image compression [38, 35]. However simulation results show that compression rate by those methods is not as high as expected. The main reason is that spatial location information has to be coded in addition to the intensity at that location.

In this section we show how line field coding scheme can be applied to video

compression in a more effective way than the still-image compression. It is well known that motion estimation and compensation are used to remove interframe temporal redundancies. By taking this into account, one can reconstruct the current frame line field from the line field of previous frame by motion estimation and compensation. Therefore, we can define two types line motion vectors: horizontal and vertical. As we define the line field on dual lattice in an image, line motion vectors have to be defined on dual lattice, too.

4.2.1 Extraction of Line Motion Vectors

Line motion vectors (LMVs) are obtained by energy minimization. The energy function is formulated in terms of motion constraints such as matching and smoothness. So we write the energy function for vertical LMVs as follows:

$$E_{m,v}^{(1)} = \sum_{x,y \in l_{t-1}^v} (I_{t-1}(x-1, y) - I_t(x-1 + d_v^x(x, y), y + d_v^y(x, y)))^2 \quad (4.5)$$

$$E_{m,v}^{(2)} = \sum_{x,y \in l_{t-1}^v} (I_{t-1}(x, y) - I_t(x + d_v^x(x, y), y + d_v^y(x, y)))^2 \quad (4.6)$$

$$E_{f,v} = \sum_{x,y \in l_{t-1}^v} (1 - l_t^v(x + d_v^x(x, y), y + d_v^y(x, y))) \quad (4.7)$$

$$E_{s,v} = \sum_{x,y \in l_{t-1}^v} \sum_{i,j \in Nd(x,y)} [\vec{d}_v(x, y) - \vec{d}_v(i, j)] \quad (4.8)$$

$$E_v = \beta_{11}^v E_{m,v}^{(1)} + \beta_{12}^v E_{m,v}^{(2)} + \beta_2^v E_{f,v} + \beta_3^v E_{s,v} \quad (4.9)$$

where I_t and I_{t-1} are the current and previous images, l_t^v and l_{t-1}^v are the current and previous vertical line fields and \vec{d}_v is the vertical LMV defined on the previous line field. In Equations 4.5 and 4.6, $E_{m,v}^{(1)}$ and $E_{m,v}^{(2)}$ are the penalizations of the intensity matching difference of the pixels which are located on the both sides of the vertical line field element. The second term $E_{f,v}$ in

Equation 4.7 penalizes the non-occurrence of the compensated line field in the current frame. The third term $E_{s,v}$ is the smoothness term of the LMV within a neighborhood $Nd(x, y)$. Finally by minimizing the total energy function in Equation 4.9 results with the smooth LMVs. As a result, those LMVs points where the vertical line field elements in previous frame tend to go in the current frame.

Similarly for the horizontal LMVs the energy function can be formulated as follows:

$$E_{m,h}^{(1)} = \sum_{x,y \in l_{t-1}^h} (I_{t-1}(x, y-1) - I_t(x + d_h^x(x, y), y-1 + d_h^y(x, y)))^2 \quad (4.10)$$

$$E_{m,h}^{(2)} = \sum_{x,y \in l_{t-1}^h} (I_{t-1}(x, y) - I_t(x + d_h^x(x, y), y + d_h^y(x, y)))^2 \quad (4.11)$$

$$E_{f,h} = \sum_{x,y \in l_{t-1}^h} (1 - l_t^h(x + d_h^x(x, y), y + d_h^y(x, y))) \quad (4.12)$$

$$E_{s,h} = \sum_{x,y \in l_{t-1}^h} \sum_{i,j \in Nd(x,y)} [\vec{d}_h(x, y) - \vec{d}_h(i, j)] \quad (4.13)$$

$$E_h = \beta_{11}^h E_{m,h}^{(1)} + \beta_{12}^h E_{m,h}^{(2)} + \beta_2^h E_{f,h} + \beta_3^h E_{s,h}. \quad (4.14)$$

As a result, given vertical and horizontal line fields, we minimize E_h and E_v to obtain vertical and horizontal LMVs. Those obtained LMVs are used to estimate the current image line field by motion compensation. So original current frame can be reconstructed from the compensated line field of the previous frame. Hence we can reconstruct the original frame by using line field of the previous frame and their LMVs. Reconstruction process will be presented in the next section.

4.2.2 Results

We test line field motion vectors by using two video frames in *Container Ship* sequence. In figure 4.4 original line fields of those frames as well as the MC line field are shown.

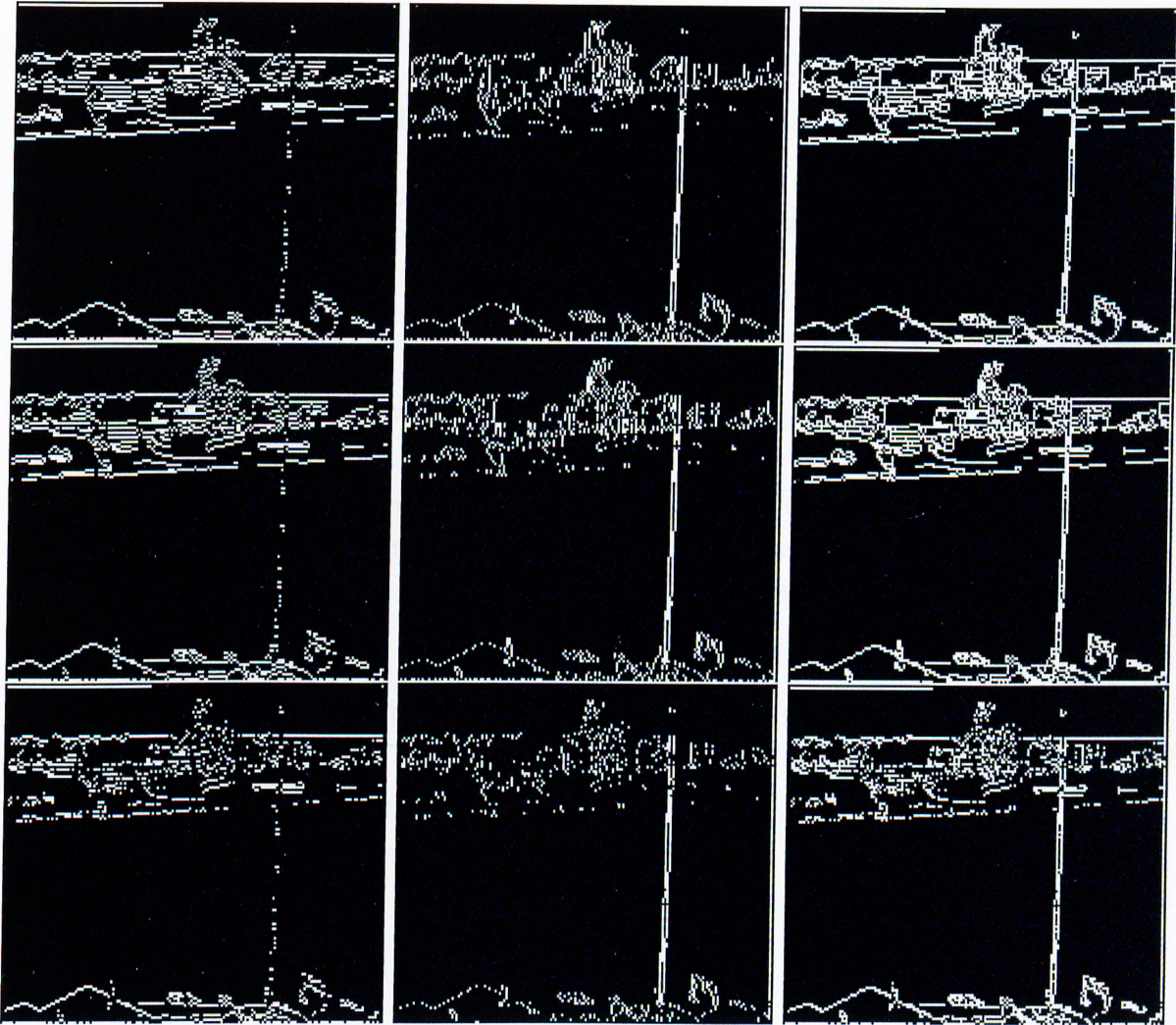


Figure 4.4: Line fields (horizontal, vertical and both) of the previous (*top*), current (*middle*) and MC (*bottom*) frames.

According to the simulation results, most of the line field of the current frame can be roughly estimated by motion compensation from the previous frame line field. But of course the compensation can not perfectly cover the original line field of the current frame. Also the pixels around the MC

line field can have different intensity levels with respect to the ones in the current frame. Since the line field is the most important field for an image, such abovementioned losses (caused from motion compensation mismatches) degrade the final MC frame visually. We show the results about it in the next section.

4.3 Reconstruction on Motion Compensated Image from Motion Compensated Line Fields

In the previous sections we demonstrated how to assign motion vectors on line field and also how to extract those by energy minimization. Those LMVs are used to reconstruct the MC line field from the previous image line field. So in this section we will demonstrate how to obtain MC image from the MC line field and also briefly discuss the effects of that type of reconstruction onto visual quality.

The reconstruction algorithm is as follows: given the line field position in dual lattice and intensity values of the neighbor pixels, we form an interpolation function composed of the following terms:

$$H_S(U, l) = \sum_x \sum_y \sum_r \sum_t (U(x, y) - U(x - r, y - t))^2 h(i_{r,t}) \quad (4.15)$$

$$H_L(l) = \sum_{N_h} (L_{h,crossings} + L_{h,inclusion} + L_{h,parallel}) \quad (4.16)$$

$$+ \sum_{N_v} (L_{v,crossings} + L_{v,inclusion} + L_{v,parallel})$$

$$H(U, l) = \alpha_1 H_S(U, l) + \alpha_2 H_L(l). \quad (4.17)$$

Those terms are the last two terms of the extraction energy function given in Equation 4.4. Therefore, if the image field is known, then by minimizing Equation 4.4, we obtain the line field of the image. Similarly if the line field is known by minimizing the similar (cross) terms in Equation 4.17 we extract the image field. Since the line field - image field pair have sufficient correspondence with each other one of them is to be almost sufficient in order to reconstruct the other.

4.3.1 Results

The reconstruction process is tested by the motion compensated line field of the 20th frame of the *Container Ship* sequence. The reconstructed MC frame is shown in figure 4.5 at the right part.



Figure 4.5: Previous, current and MC frames.

Reconstruction process is started with the initial frame which has the arbitrary values everywhere except at the neighborhood pixels of the line field. Minimization process is *Stochastic Simulated Annealing* (SSA) [39, 21]. The one important point is that at the neighborhood pixels of the line field, intensity values are not changed at the iterations so that they are allowed to take the original intensity values.

Firstly, we recognize from the simulation results is that most of the image field (especially located around the MC line field) can be reconstructed from the MC line field. However that reconstruction is degraded in visual quality and that degradation depends on the compensation error of the line field. Since the

line field is the only source for compensation and has the major characteristic information, few losses in this field can cause considerably large amount of visual degradations.

Secondly, for reconstruction we only need the line field which is the motion compensated line field of the previous frame and therefore, the next frame is reconstructed by using only the LMVs. Neither the intensity nor the spatial (position) information of the line field is necessary for this type of video coding technique. So the compression rate would be very high and the overall performance directly depends on the motion compensation success of the previous line field.

As a result we represent a dense (image) field by a sparse (line) field and use this technique in a video coding scheme. In most of the coding applications motion vectors are defined on a regular grid. Those grids are uniformly located in the image. An alternative approach is to define motion vector representatives on a more logical sparse field. In this way we can claim that line field motion vectors can be a good choice as an alternative motion field representation.

Chapter 5

Conclusions and Future Work

In this thesis, we propose new methods for regularized motion estimation and their applications to motion estimation for VLBR video coding. The main idea behind our approach is that the regularization type should be determined with respect to the application aspects.

We first introduce an improved BMME technique which casts motion estimation as a problem in energy minimization. This is achieved by modeling the motion estimation as a *Markov Random Field* (MRF) or equivalently one can write the *Gibbs Distribution* of the field in terms of Hamiltonians (energy terms) which indicate motion constraints and then find the MAP estimate of it. When we compare it with the classical BMME algorithm, our simulation results show that energy minimization based motion estimation techniques can give good performance in terms of Bit-Rate in such a way that developed algorithm can reduce the Bit-Rate twice almost with the same PSNR value. We then present hierarchically structured adaptive BMME algorithms. The main contribution of those algorithms is that they can achieve “global representation” of any motion without significant loss of image quality. In addition to that not only the estimation performance, but also the overall coding performance is improved.

We introduce alternative usage of ME in video coding applications such as

average motion determination and motion compensated (MC) interpolation. Average motion determination is the determination of the average movement or change between two frames. In order to find that average movement we use a well-regularized ME algorithm so that it detects the object motion -not the false transitions- in a dense motion field representation. So by means of average motion determination algorithms, frame rate is adaptively determined in a video coding implementation. The frames which show stationary behavior in time are not honored for coding -just skipped-. Those skipped frames are then artificially reconstructed in the decoder (receiver) side by MC interpolation techniques. Those techniques developed are based on the fractional motion compensation principle and simulated on real images that are taken from a video sequence. From the results we conclude that MC interpolation techniques achieves high performances compared with the other interpolating techniques such as linear interpolation. In addition to that the proposed MC interpolating techniques are developed in such a way that they reduce the interpolation artifacts (such as blurring and false transitions).

In video coding applications dense motion field should be somehow decimated because of the “very” low bit-rate requirement. One important factor in decimation is to represent the true motion field as much as possible. Another factor is to find the suitable decimation scheme which gives the minimum distortion. BMME algorithms are the mostly used techniques in VLBR video codecs, standards and devices. Those techniques are nothing but uniform decimation of the dense motion field. However such a sparse field representation may not be suitable in some certain cases. As an alternative representation we propose line field motion estimation. That technique achieve high compression ratio but the visual quality is not so high as expected. The reason for that the line field which is in fact the object boundaries of an image, can be affected by the rigid body motion. That is to say that the intensity levels at the line field of an image (object boundaries), do not remain constant over time. Even it is the most changing field in the image. For this reason the reconstructed image from the MC line field is degraded visually. In order to overcome this problem, a more advanced model which does not assume intensity constancy for line fields, should be developed. So by some refinements on the line field definition the visual quality of the MC frames can be improved.

We have quite satisfactory results from the developed BMME algorithms. According to the simulation results we can claim that those improved BMME algorithms increase the performance of an VLBR video coding algorithms. Especially the amount of bits spent for coding the BMVs is efficiently reduced without any significant visual degradation. This can allow to spend much more bits to the intra coding part of the video coding algorithms and thus visual quality is to be increased. Therefore, those BMME algorithms developed would be good candidates for the motion estimation parts of the well-known standards such as H.261 and H.263. As well as the BMME algorithms, the developed techniques in chapter four can be used to determine the frame rate at the encoder side and also to reconstruct the skipped frames artificially at the decoder side. Simulation results show that they can further increase the overall performance of any video codec (coder/decoder).

As a future research, we will deal with some advanced motion estimation techniques which can extract the object motion adaptively. So those algorithms are going to set their parameters adaptively and also object motion can be represented in a global and minimum descriptive way. In addition to that, motion estimation algorithms will be based on other characteristic features of the image such as texture information. Thus we wish to develop superior motion estimation techniques which are more suitable for human vision and motion tracking system.

A

Motion Estimation Algorithms in H.261 and H.263

In this appendix, we present the existing standard motion estimation algorithms which are currently used in practical video coding applications.

H.261:

It is the standard which is finalized by CCITT in December 1990 and is primarily intended in image coding for low and medium bit rates ($p \times 64$ kbps, $p = 1, 2, \dots, 30$) [3]. Motion estimation is performed only at the INTER mode and motion compensation is an optional tool for the decoder. In INTER mode prediction is realized by motion estimation and the error signal is coded by DCT. Since DCT imposes a block structure on the image, the motion is estimated blockwise. This is realized as follows: for each block in the current frame we try to find the closest match in the previous reconstructed frame using some suitable distortion criteria such as SAD (sum of absolute differences) which has the formula:

$$SAD(x, y, \vec{d}(x, y)) = \sum_i^{BS} \sum_j^{BS} |I_t(x+i, y+j) - I_{t-1}(x+i-d_x(x, y), y+j-d_y(x, y))| \quad (\text{A.1})$$

where BS is the block size, $\vec{d}(x, y)$ is the BMV of the block centered at (x, y) , $d_x(x, y)$ and $d_y(x, y)$ where x and y components of the BMV, respectively.

Therefore, motion estimation is the same (classical) BMME algorithm which is discussed in section 2.3.

H.263:

Starting from 1993, efforts of Specialist Group in CCITT result in a VLBR video coding standard called H.263 [40]. In order to achieve VLBR objective, motion estimation algorithm is improved compared to the one in H.261. It has some prediction modes such as *Normal*, *Restricted/Unrestricted* and *Advanced*. Those modes make the algorithm more efficient and can reduce the area of temporally unpredictable (TU) regions.

In *Normal* mode, motion estimation algorithm is the same as the one in H.261. It is the classical BMME algorithm with blocks size 16. *Restricted/Unrestricted* modes indicate the permission of a block motion vector to point towards the out of the image. In *Unrestricted* mode block motion vectors can point towards the out of the image but in *Restricted* mode they can not.

Motion Compensation

In H.263, one motion vector per macroblock (16x16) is used except in *Advanced* mode. In advanced mode, one or four block motion vectors can be used for a macroblock. If four block motion vectors are decided to be used, those vectors belong to the eight by eight sub-blocks and each of which is obtained by exhaustive search in a range $[-16, 15.5]$ within a half pixel accuracy.

In advanced mode one alternative usage is the *Overlapped Motion Compensation*. In this mode each sub-block has three block motion vectors -one for the current block and two out of four (any) neighbor blocks-. So weighted sum of those block motion vectors yields the motion vector for any

pixel in the eight by eight sub-block. The choice of two neighbor blocks are determined by the distance of that pixel to the blocks. So the nearest two blocks (one in left or right and one in up or down blocks) are chosen for the motion vector. The weights for sub-block pixels are shown in Figure A.1.

Block Motion Vector Coding

Coding of BMVs is differential vector coding. The differential coding with four vectors per macroblock is realised as follows: the vectors are obtained by adding predictors to the vector differences indicated by $BMV_{1,2,3}$ as shown in Figure A.2. The prediction value of the current BMV is the median of three BMVs ($BMV_{1,2,3}$). So the difference from prediction value is coded.

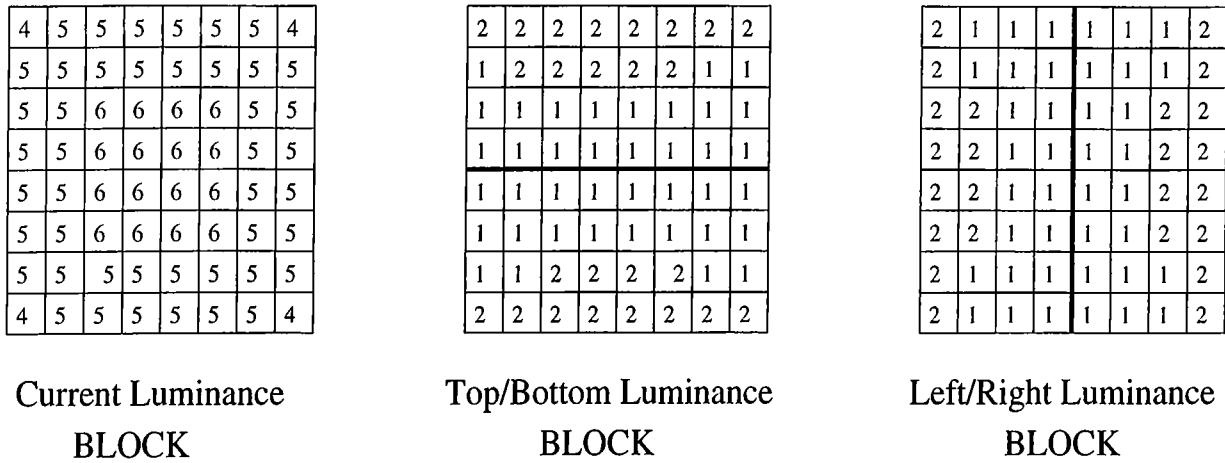


Figure A.1: Weights for current and two neighbor blocks

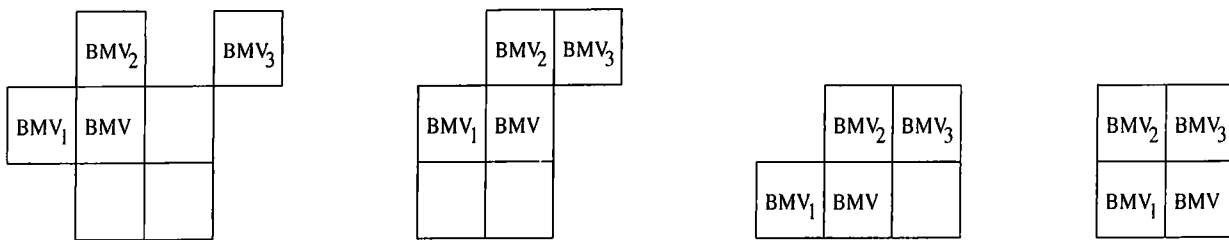


Figure A.2: Candidate neighbors for predictors for each of the luminance block.

In case of one BMV per macroblock, the candidate predictors for the differential coding are taken from three surrounding macroblocks as shown in Figure A.3. As in the case of four vectors per macroblock, the predictor is

the median value of the three candidate predictors for current BMV. In Figure A.3, GOB stands for *Group of Blocks*.

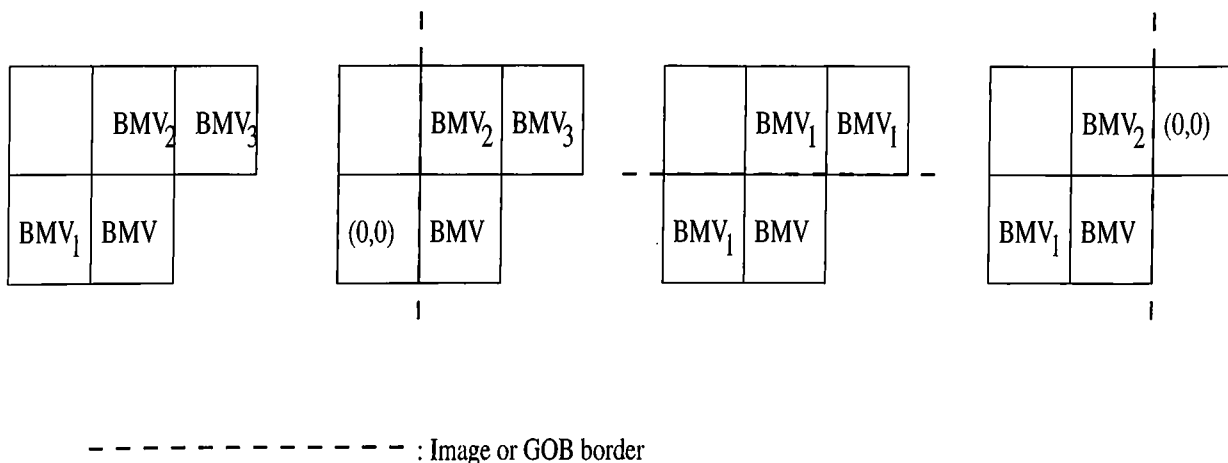


Figure A.3: Candidate neighbors for predictors for a macroblock BMV

The coding of the difference BMV values is processed by a VLC (variable length coding) table. So for each possible difference value there exist a code from the VLC table and the bit stream for the BMVs is constructed by the variable length codes of the VLC table given in Table A.1.

Motion Estimation

As it is mentioned previously, H.263 standard has mainly two modes: the intra and inter modes. The intra mode is similar to JPEG still-image compression and the details about it are out of the scope of this section. In the inter mode, first a temporal prediction is employed with or without motion compensation. That is to say that motion estimation may be processed in this stage and therefore, motion estimation is an optional process for the encoder side. The standard does not specify the motion estimation method but we can indicate some basic features about the mostly used motion estimation algorithm. SAD (as given in Equation A.1) is the usual matching criteria. Block matching based on 16×16 blocks is generally used. In the modes of the standard, there are several search techniques. For example in the advanced mode, first BMV of a macroblock is found and then the BMVs of the four luminance blocks are searched around that macroblock BMV. In some encoders the search technique for luminance blocks is also full (exhaustive) search.

However first method is rather faster and generally gives more convenient results compared with the full search. BMVs are determined in half pixel accuracy in a range $[-16,15.5]$.

Index	Vector	Differences	Bit Number	Codes
0	-16	16	13	0000 0000 0010 1
1	-15.5	16.5	13	0000 0000 0011 1
2	-15	17	12	0000 0000 0101
3	-14.5	17.5	12	0000 0000 0111
4	-14	18	12	0000 0000 1001
5	-13.5	18.5	12	0000 0000 1011
6	-13	19	12	0000 0000 1101
7	-12.5	19.5	12	0000 0000 1111
8	-12	20	11	0000 0001 001
9	-11.5	20.5	11	0000 0001 011
10	-11	21	11	0000 0001 101
11	-10.5	21.5	11	0000 0001 111
12	-10	22	11	0000 0010 001
13	-9.5	22.5	11	0000 0010 011
14	-9	23	11	0000 0010 101
15	-8.5	23.5	11	0000 0010 111
16	-8	24	11	0000 0011 001
17	-7.5	24.5	11	0000 0011 011
18	-7	25	11	0000 0011 101
19	-6.5	25.5	11	0000 0011 111
20	-6	26	11	0000 0100 001
21	-5.5	26.5	11	0000 0100 011
22	-5	27	10	0000 0100 11
23	-4.5	27.5	10	0000 0101 01
24	-4	28	10	0000 0101 11
25	-3.5	28.5	8	0000 0111
26	-3	29	8	0000 1001
27	-2.5	29.5	8	0000 1011
28	-2	30	7	0000 111
29	-1.5	30.5	5	0001 1
30	-1	31	4	0011
31	-0.5	31.5	3	011
32	0	0	1	1
33	0.5	-31.5	3	010
34	1	-31	4	0010
35	1.5	-30.5	5	0001 0
36	2	-30	7	0000 110
37	2.5	-29.5	8	0000 1010
38	3	-29	8	0000 1000
39	3.5	-28.5	8	0000 0110
40	4	-28	10	0000 0101 10
41	4.5	-27.5	10	0000 0101 00
42	5	-27	10	0000 0100 10
43	5.5	-26.5	11	0000 0100 010
44	6	-26	11	0000 0100 000
45	6.5	-25.5	11	0000 0011 110
46	7	-25	11	0000 0011 100
47	7.5	-24.5	11	0000 0011 010
48	8	-24	11	0000 0011 000
49	8.5	-23.5	11	0000 0010 110
50	9	-23	11	0000 0010 100
51	9.5	-22.5	11	0000 0010 010
52	10	-22	11	0000 0010 000
53	10.5	-21.5	11	0000 0001 110
54	11	-21	11	0000 0001 100
55	11.5	-20.5	11	0000 0001 010
56	12	-20	11	0000 0001 000
57	12.5	-19.5	12	0000 0000 1110
58	13	-19	12	0000 0000 1100
59	13.5	-18.5	12	0000 0000 1010
60	14	-18	12	0000 0000 1000
61	14.5	-17.5	12	0000 0000 0110
62	15	-17	12	0000 0000 0100
63	15.5	-16.5	13	0000 0000 0011 0

Table A.1: Simulation parameters for *ABMA*

REFERENCES

- [1] P. Gerken, "Object-based analysis-synthesis coding of image sequences at very low bit rates," *IEEE Trans. Circ. and Syst: Video Tech.*, vol. 4, pp. 228-237, September 1994.
- [2] K. Aizawa and T. S. Huang, "Model based image coding: Advanced video coding techniques for very low bit-rate applications," *IEEE Proc.*, vol. 83, pp. 259-271, February 1995.
- [3] CCITT Recommendation H.261, *Video Codec for Audiovisual Services at p.64 Kbit/s*, com xv-r 37-e edition, 1990.
- [4] I. M. Abdelqader, S. A. Rajala, W. E. Snyder, and G. L. Bilbro, "Energy minimization approach to motion estimation," *Signal Processing*, vol. 28, pp. 291-306, 1992.
- [5] D. Lee and T. Pavdlis, "One-dimensional regularization with discontinuities," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 10, pp. 822-828, November 1988.
- [6] D. Terzopoulos, "Regularization of visual problems involving discontinuities," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 8, pp. 413-424, August 1986.
- [7] T. A. Poggio, M. Bertero and V. Torre, "Ill-posed problems in early vision," *Proc. IEEE*, vol. 76, pp. 869-889, August 1988.
- [8] R. Buschmann, "Joint multigrid of 2d motion and object boundary patterns," *SPIE*, vol. 2094, pp. 106-118, 1993.

- [9] B. K. P. Horn and B. G. Schunk, "Determining optical flow," *Artif. Intell.*, vol. 23, pp. 309–354, 1984.
- [10] H.-H. Nagel and W. Enkelmann, "An investigation of smoothness constraints for the estimation of displacement vector fields from image sequences," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-8, pp. 565–593, September 1986.
- [11] E. Dubois and J. Konrad, "Comparison of stochastic and deterministic solution methods in bayesian estimation of 2d motion," *Image and Vision Computing*, vol. 8, November 1990.
- [12] M. Bierling, "Displacement estimation by hierarchical blockmatching," *3.rd SPIE Symposium on Visual Communications and Image Processing*, vol. , pp. 942–951, November 1988.
- [13] L. Boroczky, J. N. Driessen, and J. Biemond, "Adaptive algorithms for pel recursive displacement estimation," *SPIE*, vol. 1360, pp. 1210–1221, 1990.
- [14] L. Boroczky, *Pel-Recursive Motion Estimation for Image Coding*. PhD thesis, Delft University of Technology, Department of Electrical Engineering, June 1991.
- [15] T. J. Broida and R. Chellappa, "Performance bounds for estimating 3d motion parameters from a sequence of noisy images," *Journal of the Optical Society of America A*, vol. 6, pp. 879–889, June 1989.
- [16] H. Gharavi and M. Mills, "Block-matching motion estimation algorithms: New results," *IEEE Trans. Circ. and Syst. Video Tech.*, vol. 37, pp. 649–651, 1990.
- [17] V. Seferdis and M. Ghanbari, "General approach to block matching motion estimation," *Optical Engineering*, vol. 32, pp. 1464–1474, July 1993.
- [18] R. Depommier and E. Dubois, "Motion estimation with detection of occlusion areas," *ICASSP*, vol. 3, pp. 269–272, 1992.
- [19] I. M. Abdelqader and S. A. Rajala, "Motion estimation from noisy image data," *ICASSP*, vol. 5, pp. 209–212, 1993.

- [20] J. Besag, “Spatial interaction and the statistical analysis of lattice systems,” *JRSB*, vol. 34, pp. 75–83, 1974.
- [21] D. Geman and S. Geman, “Stochastic relaxation, gibbs distribution and the bayesian restoration of images,” *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-6, November 1984.
- [22] B. Liu and A. Zaccarin, “New fast algorithms for the estimation of block motion vectors,” *IEEE Trans. Circ. and Syst. Video Tech.*, vol. 3, pp. 148–157, April 1993.
- [23] T. Koga, K. Linuma, A. Hirano, and T. Ishiguro, “Motion compensated interframe coding for video conferencing,” *Proceedings of NTC*, vol. , pp. G5.3.1–G5.3.5, 1981.
- [24] L. M. Po and W. C. Ma, “A new center-biased search algorithm for block motion estimation,” *ICIP*, vol. 1, pp. 410–417, 1995.
- [25] J. R. Jain and J. K. Jain, “Displacement measurement and its application in interframe image coding,” *IEEE Trans. on Comm.*, vol. COM-29, pp. 1799–1806, December 1981.
- [26] A. N. Netravali and J. D. Robbins, “Motion-compensated coding: Some new results,” *The B. S. T. J.*, vol. BSTJ-59, pp. 1735–1745, November 1980.
- [27] M. J. Korsten and Z. Houkes, “The estimation of geometry and motion of a surface from image sequences by means of linearization of a parametric model,” *Computer Vision, Graphics, and Image Processing*, vol. 50, pp. 1–28, 1990.
- [28] R. H. J. M. Plompen, *Motion Video Coding for Visual Telephony*. PhD thesis, Delft University of Technology, Department of Electrical Engineering, 1989.
- [29] F. Heitz, P. Perez, and P. Bouthemy, “Multiscale minimization of global energy functions in some visual recovery problems,” *CVCIP Image Understanding*, vol. , pp. 125–134, January 1994.
- [30] A. M. Tekalp, *Digital Video Processing*, chapter 8, pp. 134–135. Prentice Hall, 1995.

- [31] T. N. Pappas, “An adaptive clustering algorithm for image segmentation,” *IEEE Trans. Signal Proc.*, vol. SP-40, pp. 901–914, April 1992.
- [32] T. Peli and D. Malah, “A study of edge detection algorithms,” *Comput. Graphics and Image Processing*, vol. 20, pp. 1–21, 1982.
- [33] T. A. Poggio and V. Torre, “On edge detection,” *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-8, pp. 147–163, March 1986.
- [34] R. L. Lagendijk and M. I. Sezan, “Motion compensated frame rate conversion of motion pictures,” *IEEE ICASSP*, vol. , 1992.
- [35] T. Acar, A. Gencata and M. Gokmen, “Yirtilabilir zar modeli ile goruntu sikistirma,” *SIU-95*, vol. A: Goruntu Isleme, pp. 117–122, 1995.
- [36] D. Terzopoulos, “The computation of visual surface representations,” *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 10, 1988.
- [37] T. Acar and M. Gokmen, “The computation of visual surface representations,” *Image Coding using weak membrane model images*, vol. , pp. 1221–1230, 1994.
- [38] T. Acar and M. Gokmen, “Image coding using weak membrane model of images,” *IEEE VCIP*, vol. , pp. 1221–1230, 1994.
- [39] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, “Optimization by simulated annealing,” *Science*, vol. 220, May 1983.
- [40] CCITT Recommendation H.263, *Video Codec for Audiovisual Services at low bitrates*, 1993.