DEVELOPMENT OF FAST MOTION ESTIMATION ALGORITHMS FOR VIDEO COMPRESSION

A THESIS SUBMITTED IN PARTIAL FULFILLMENT

OF THE REQUIREMENTS FOR THE DEGREE OF

Master of Technology

in

Telematics and Signal Processing

By

PAVANKUMAR GORPUNI

Roll No: 207EC110

Department of Electronics and Communication

National Institute of Technology

Rourkela, India

2009

DEVELOPMENT OF FAST MOTION ESTIMATION ALGORITHMS FOR VIDEO COMPRESSION

A THESIS SUBMITTED IN PARTIAL FULFILLMENT

OF THE REQUIREMENTS FOR THE DEGREE OF

Master of Technology

in

Telematics and Signal Processing

By

PAVANKUMAR GORPUNI

Roll No: 207EC110

Under the Supervision of

Prof. Ganapati Panda

Department of Electronics and Communication

National Institute of Technology

Rourkela, India

2009

National Institute of Technology

Rourkela

CERTIFICATE

This is to certify that the thesis entitled, **" Development of Fast Motion Estimation Algorithms for Video Compression "** submitted by **Pavankumar Gorpuni** in partial fulfillment of the requirements for the award of Master of Technology Degree in **Electronics & Communication Engineering** with specialization in **Telematics and Signal Processing** during 2008-2009 at the National Institute of Technology, Rourkela (Deemed University) is an authentic work carried out by him under my supervision and guidance.

To the best of my knowledge, the matter embodied in the thesis has not been submitted to any other University / Institute for the award of any Degree or Diploma.

Date

Prof. G. Panda (FNAE, FNASc)

Dept. of Electronics & Communication Engg.

National Institute of Technology

Rourkela-769008

Orissa, India

Acknowledgment

First of all, I would like to express my deep sense of respect and gratitude towards my advisor and guide **Prof. Ganapati Panda**, who has been the guiding force behind this work. I want to thank him for introducing me to the field of Signal Processing and giving me the opportunity to work under him. I am greatly indebted to him for his constant encouragement and invaluable advice in every aspect of my academic life. I consider it my good fortune to have got an opportunity to work with such a wonderful person.

I express my respects to **Prof. S.K. Patra**, **Prof. K. K. Mahapatra**, **Prof. G. S. Rath**, **Prof. S. Meher** , **Prof. S.K.Behera**, and **Prof. Punam Singh** for teaching me and also helping me how to learn. They have been great sources of inspiration to me and I thank them from the bottom of my heart.

I would like to thank all faculty members and staff of the Department of Electronics and Communication Engineering, N.I.T. Rourkela for their generous help in various ways for the completion of this thesis.

I would like to thank my friends, especially **Pyari Mohan Pradhan, Nithin V george, Vikas Baghel, Shilpakesav, Satyasai Jagannath Nanda, Sasmita** for their help during the course of this work and special thanks to Ranganadham Dharmana for helping me to publish paper in conference proceedings. I am also thankful to my classmates for all the thoughtful and mind stimulating discussions we had, which prompted us to think beyond the obvious.

I am especially indebted to my parents (**Mr. Sivaprasad and Mrs. Sivalakshmi**) for their love, sacrifice, and support. They are my first teachers after I came to this world and have set great examples for me about how to live, study, and work.

Pavankumar Gorpuni

CONTENTS

Acknowledgment **i**

Contents **ii**

Abstract **iv**

List of Figures **v**

List of Tables **vii**

1 Introduction **1**

1.1 Introduction 1

1.2 Video Standards 2

1.3 Motivation 4

1.4 Thesis Organization 6

2 Fundamental Concepts of Motion Estimation **7**

2.1 Motion Estimation 7

2.2 Block Matching Algorithm 8

2.2.1 Block Matching Methods 11

2.2.2 Matching Criteria for Motion Estimation 11

2.3 Search Algorithms for Motion Estimation 16

2.3.1 Full Search Motion Estimation 17

2.3.2 Three Step Search (TSS) 18

2.3.3 Diamond Search(DS) Algorithm 19

3 Fast Motion Estimation Algorithm based on Genetic algorithm (GA) 21
3.1 Introduction to Genetic Algorithm . . . 21
3.2 Genetic Algorithm . . . 22
3.3 Block Matching Algorithm based on GA . . . 27
3.3.1 Genetic BMA procedure . . . 28
3.3.2 Experiments and Simulation Results . . . 31
3.4 Conclusion . . . 34

4 Fast Motion Estimation Algorithm based on Clonal Particle Swarm Optimization (CPSO) 35
4.1 Particle Swarm Optimization (PSO) . . . 35
4.1.1 Conventional Particle Swarm Optimization . . . 37
4.1.2 Variants of PSO . . . 38
4.1.3 Clonal Particle Swarm optimization (CPSO) . . . 39
4.2 Block Matching Algorithm Based on CPSO . . . 40
4.2.1 Flowchart of CPSO BMA . . . 44
4.2.2 Experiments and Simulation Results . . . 45
4.3 Conclusion . . . 49

5 Bidirectional Motion Estimation 50
5.1 MPEG Frame Encoding . . . 50
5.2 Bidirectional Frame Prediction . . . 52
5.3 Bidirectional Motion Estimation Based on PSO . . . 54
5.3.1 Algorithm Steps . . . 55
5.3.2 Experiments and Simulation Results . . . 58
5.4 Conclusion . . . 62

6 Conclusion and Future work 63
6.1 Conclusion . . . 63
6.2 Scope For Future Work . . . 64

References 65

Abstract

With the increasing popularity of technologies such as Internet streaming video and video conferencing, video compression has became an essential component of broadcast and entertainment media. Motion Estimation (ME) and compensation techniques, which can eliminate temporal redundancy between adjacent frames effectively, have been widely applied to popular video compression coding standards such as MPEG-2, MPEG-4. Traditional fast block matching algorithms are easily traped into the local minima resulting in degradation on video quality to some extent after decoding. Since Evolutionary Computing Techniques are suitable for achieving global optimal solution, these techniques are introduced to do Motion Estimation procedure in this thesis. Zero Motion prejudgement is also included which aims at finding static macroblocks (MB) which do not need to perform remaining search thus reduces the computational cost. Simulation results obtained show that the proposed Clonal Particle Swarm Optimization algorithm given a very good improvement in reducing the computations overhead and achieves very goood Peak Signal to Noise Ratio(PSNR) values, which makes the techniques more efficient than the conventional searching algorithms. To reduce the Motion vector overhead in Bidirectional frame prediction, in this thesis novel Bidirectional Motion Estimation algorithm based on PSO is also proposed and results shows that the proposed method can significantly reduces the computational complexity involved in the Bidirectional frame prediction and also least prediction error in all video sequences.

LIST OF FIGURES

1.1 Wireless video conferencing application. 3

2.1 Motion Compensated Video Coding 8
2.2 Block-matching Motion estimation 9
2.3 Backward Motion estimation with current frame as k and frame (k-1) as the reference frame 10
2.4 Forward Motion estimation with current frame as k and frame (k+1) as the reference frame 11
2.5 macroblock (a) partition and (b) sub partition 14
2.6 Full search motion estimation 17
2.7 (a) Large Diamond Search Pattern (b) Small Diamond Search Pattern . . . 19

3.1 One-point crossover of binary strings. 25
3.2 Genetic algorithm procedure 27
3.3 selection of intial Population 29
3.4 search window 32
3.5 PSNR(dB) comparision of GA and DS 33
3.6 Computational comparison of GA and DS 34

4.1 Particles intial positions 43
4.2 Flowchart of CPSO-BMA 44
4.3 PSNR comparison of DS, PSO, CPSO 47
4.4 computational comparison of DS, PSO, CPSO 48
4.5 original and estimatted frames of silent sequence 48
4.6 original and estimated frames of NEWS sequence 49

5.1 An input video stream 52
5.2 Encoding order 52
5.3 Bidirectional prediction scheme 53
5.4 Bidirectional search for best motion vector 55
5.5 Original and estimated frames using different methods 60
5.6 NEWS video PSNR(dB) comparasion of B-frame 61
5.7 SILENT video PSNR(dB) comparasion of B-frame 61

LIST OF TABLES

2.1 Computational Complexity of FSBM 18

3.1 Assumed Threshold values 31
3.2 Computational gain to DS 33
3.3 Average PSNR(dB) performances of DS, GA 34

4.1 Assumed Threshold values 46
4.2 Computational gain to DS ,PSO 47
4.3 Average PSNR(dB) performances of DS, GA, PSO, CPSO 47

5.1 Average mean square prediction error (AMSPE) 59
5.2 Average Search points/frame 60

CHAPTER 1

INTRODUCTION

1.1 Introduction

Digital video coding has gradually increased in importance since the 90s when MPEG-1 first emerged. It has had large impact on video delivery, storage and presentation. Compared to analog video, video coding achieves higher data compression rates without significant loss of subjective picture quality [1]. This eliminates the need of high bandwidth as required in analog video delivery. With this important characteristic, many application areas have emerged. For example, set-top box video playback using compact disk, video conferencing over IP networks, P2P video delivery, mobile TV broadcasting, etc. The specialized nature of video applications has led to the development of video processing systems having different size, quality, performance, power consumption and cost.

Digitization of video scenes was an inevitable step since it has many advantages over analog video. Digital video is virtually immune to noise, easier to transmit and is able to provide a more interactive interface to users. Furthermore, the amount of video content, e.g. TV content, can be made larger through improved video compression because the bandwidth required for analog delivery can be used for more channels in a digital video delivery system. With todays sophisticated video compression systems, end users can also stream video, edit video and share video with friends via the internet or IP networks. In contrast, analog signals are difficult to manipulate and transmit. Generally speaking, video compression is a technology for transforming video signals that aims to retain original quality under a number of constraints, e.g. storage constraint, time delay constraint or computation power constraint. It takes advantage of data redundancy between successive frames to reduce the storage requirement by applying computational resources [2]. The

design of data compression systems normally involves a tradeoff between quality, speed, resource utilization and power consumption.

In a video scene, data redundancy arises from spatial, temporal and statistical correlation between frames. These correlations are processed separately because of differences in their characteristics. Hybrid video coding architectures have been employed since the first generation of video coding standards, i.e. MPEG. MPEG consists of three main parts to reduce data redundancy from the three sources described above. Motion estimation and compensation are used to reduce temporal redundancy between successive frames in the time domain. Transform coding, also commonly used in image compression, is employed to reduce spatial dependency within a frame in the spatial domain. Entropy coding is used to reduce statistical redun- dancy over the residue and compression data. This is a lossless compression technique commonly used in file compression.

The demand for communications with moving video picture is rapidly increasing. Video is required in many remote video conferencing systems, and it is expected that in near future cellular telephone systems will send and receive real-time video. A typical system, which relays video over a low bandwidth transmission channel, is shown in Figure 1.1. The multimedia terminals could be, for example, cellular phones or handheld computers. Both terminals contain compatible codecs: a video encoder and decoder pair, whose purpose is to compress the video stream to be transmitted over a slow link, such as radio waves or Internet. Often a bidirectional connection is desired, where both terminals transmit and receive video, and thus they both need an encoder and a decoder running in real-time.

A major problem in a video is the high requirement for bandwidth. A typical system needs to send dozens of individual frames per second to create an illusion of a moving picture. For this reason, several standards for compression of the video have been developed. Each individual frame is coded so that redundancy is removed. Furthermore, between consecutive frames, a great deal of redundancy is removed with a motion compensation system.

1.2 Video Standards

Both terminals in the Figure 1.1 need to use a video decoder that is capable of decoding the video stream produced by the other terminal. Since there are endless ways to compress and encode data, and many terminal vendors which each may have an unique idea of data compression, common standards are needed, that rigidly define how the video is

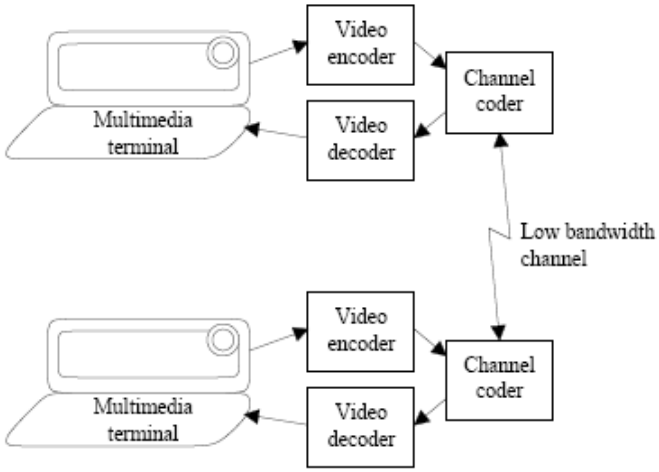

Figure 1.1: Wireless video conferencing application.

coded in the transmission channel. There are mainly two standard series in common use, both having several versions. International Telecommunications Union (ITU) started developing Recommendation H.261 in 1984, and the effort was finished in 1990 when it was approved. The standard is aimed for video conferencing and video phone services over the integrated service digital network (ISDN) with bit rate a multiple of 64 kilobits per second.

MPEG-1 is a video compression standard developed in joint operation by International Standards Organization (ISO) and International Electro-Technical Commission (IEC). The system development was started in 1988 and finished in 1990, and it was accepted as standard in 1992. MPEG-1 can be used at higher bit rates than H.261, at about 1.5 megabits per second, which is suitable for storing the compressed video stream on compact disks or for using with interactive multimedia systems [3]. The standard covers also audio associated with a video.

In 1996 a revised version of the standard, Recommendation H.263, was finalized which adopts some new techniques for compression, such as half pixel and optionally smaller block size for motion compensation. As a result it has better video quality than H.261. Recommendation H.261 divides each frame into 16×16 picture element (pixel) blocks for backward motion compensation, and H.263 can also take advantage of 8×8 pixel blocks. A new ITU standard in development is called H.26L, and it allows motion compensation

with greater variation in block sizes.

For motion estimation, MPEG-1 uses the same block size as H.261, 16×16 pixels, but in addition to backward compensation, MPEG can also apply bidirectional motion compensation. A revised standard, MPEG-2, was approved in 1994. Its target is at higher bit rates than MPEG-1, from 2 to 30 megabits per second, where applications may be digital television or video services through a fast computer network. The latest ISO/IEC video coding standard is MPEG-4, which was approved in the beginning of 1999. It is targeted at very low bit rates (832 kilobits per second) suitable for e.g. mobile video phones. MPEG-4 can be also used with higher bit rates, up to 4 megabits per second.

1.3 Motivation

Video compression is the field in electrical engineering and computer science that deals with representation of video data, for storage and/or transmission, for both analog and digital video. Video coding is often considered to be only for natural video, it can also be applied to synthetic (computer generated) video, i.e. graphics. Many representations take advantage of features of the Human Visual System to achieve an efficient representation. The biggest challenge is to reduce the size of the video data using video compression. For this reason the terms "video coding" and "video compression" are often used interchangeably by those who don't know the difference. The search for efficient video compression techniques dominated much of the research activity for video coding since the early 1980s, the first major milestone was H.261, from which JPEG adopted the idea of using the DCT; since then many other advancements have been made to algorithms such as motion estimation. Since approximately 2000 the focus has been more on Meta data and video search, resulting in MPEG-7 and MPEG-21.

Video Compression

The main problem with the uncompressed (raw) video is it contains immense amount of data and hence communication and storage capabilities are limited and are expensive. For example, if we consider a HDTV video signal with 720×1280 pixels/frame with progressive scanning at 60 frames/sec, then the transmitter must be able to send

$$\left(\frac{720 \times 1280 pixels}{frame}\right)\left(\frac{60 frames}{sec}\right)\left(\frac{3 colours}{pixel}\right)\left(\frac{8 bits}{colour}\right) = 1.3 Gb/s \qquad (1.1)$$

But the available HDTV channel bandwidth is around 20 Mb/s [2], i.e., it requires compression by a factor of 70. A Digital Versatile Disk (DVD) can only store a few seconds of raw video at television-quality resolution and frame rate and so DVD-Video storage would not be practical without video and audio compression.

Achieving Compression

Video compression can be achieved by exploiting the similarities or redundancies and irrelevancy that exists in a typical video signal. The redundancy in a video signal is based on two principles. The first is the spatial redundancy that exists in each frame. The second is the fact that most of the time, a video frame is very similar to its immediate neighbors. This is called temporal redundancy. This temoparal redundancy can be eliminared by using motion estimation and compensation procedure. Another goal of video compression is to reduce the irrelevancy in the video signal, that is to only code video features that are perceptually important and not to waste valuable bits on information that is not perceptually important or irrelevant. Identifying and reducing the redundancy in a video signal is relatively straightforward, however identifying what is perceptually relevant and what is not is very difficult and therefore irrelevancy is difficult to exploit. This can be done by using appropriate models of the Human Vision System.

Successive video frames may contain the same objects (still or moving). Motion estimation examines the movement of objects in an image sequence to try to obtain vectors representing the estimated motion. Motion compensation uses the knowledge of object motion so obtained to achieve data compression. In inter frame coding motion estimation and compensation have become powerful techniques to eliminate the temporal redundancy due to high correlation between consecutive frames. In real video scenes, motion can be a complex combination of translation and rotation. Such motion is difficult to estimate and may require large amounts of processing. However, translational motion is easily estimated and has been used successfully for motion compensated coding.

Different search algorithms are used to estimate motion between frames. When motion estimation is performed by an MPEG-2 encoder it groups pixels into 16×16 macro blocks. MPEG-4 AVC encoders can divide these macro blocks into partitions as small as 4×4, and even of variable size within the same Macro block. Partitions allow for more accuracy in motion estimation because areas with high motion can be isolated from those with less movement.

1.4 Thesis Organization

This thesis provides a fast motion estimation algorithms for video compression which are based on different evolutionary computing techniques. simulation results given a very good improvement in reducing the computations overhead and achieves very goood Peak Signal to Noise Ratio (PSNR) values which makes the techniques more efficient than the conventional searching algorithms. Thesis can be organized in the following manner, Chapter 2 focuses on the fundamental concepts of motion estimation and the existing conventional motion estimation algorithms. Chapter 3 describes the fast motion estimation algorithm based on the genetic algorithm and Chapter 4 describes the praposed Fast motion estimation algorithm based on Clonal Particle Swarm Optimization. Chpater 5 describes the praposed novel bidirectional motion estimation based on particle swarm optimization and got a good results when compared to existing techniques. chapter 6 gives the conclusions and future scope for work.

CHAPTER 2

FUNDAMENTAL CONCEPTS OF MOTION ESTIMATION

2.1 Motion Estimation

A video sequence can be considered to be a discretized three-dimensional projection of the real four-dimensional continuous space-time. The objects in the real world may move, rotate, or deform. The movements can not be observed directly, but instead the light reflected from the object surfaces and projected onto an image. The light source can be moving, and the reflected light varies depending on the angle between a surface and a light source. There may be objects occluding the light rays and casting shadows. The objects may be transparent (so that several independent motions could be observed at the same location of an image) or there might be fog, rain or snow blurring the observed image. The discretization causes noise into the video sequence, from which the video encoder makes its motion estimations. There may also be noises in the image capture device (such as a video camera) or in the electrical transmission lines. A perfect motion model would take all the factors into account and find the motion that has the maximum likelihood from the observed video sequence.

Changes between frames are mainly due to the movement of objects. Using a model of the motion of objects between frames, the encoder estimates the motion that occurred between the reference frame and the current frame. This process is called motion estimation (ME) [4]. The encoder then uses this motion model and information to move the contents of the reference frame to provide a better prediction of the current frame. This process is known as motion compensation (MC), and the prediction so produced is called the motion-compensated prediction (MCP) or the displaced-frame (DF) [5]. In this case,

the coded prediction error signal is called the displaced-frame difference (DFD).A block diagram of a motion-compensated coding system is illustrated in Figure 2.1. This is the most commonly used interframe coding method.

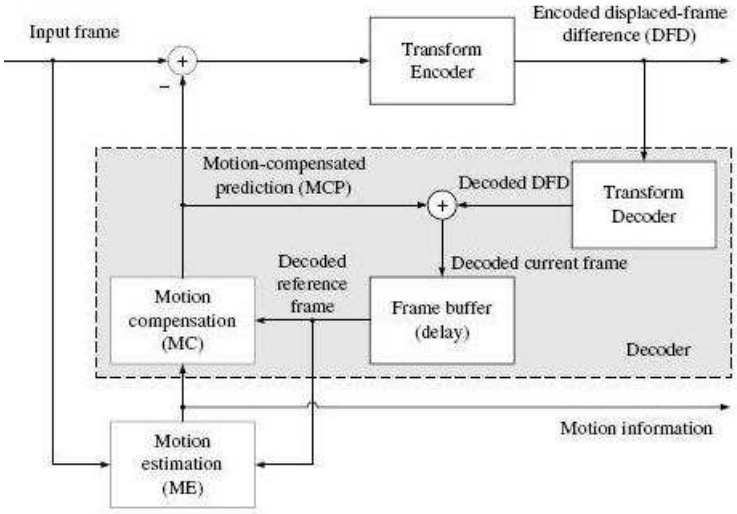

Figure 2.1: Motion Compensated Video Coding

The reference frame employed for ME can occur temporally before or after the current frame. The two cases are known as forward prediction and backward prediction, respectively. In bidirectional prediction, however, two reference frames (one each for forward and backward prediction) are employed and the two predictions are interpolated (the resulting predicted frame is called B-frame). The most commonly used ME method is the block-matching motion estimation (BMME) algorithm.

2.2 Block Matching Algorithm

Figure 2.2 illustrates a process of block-matching algorithm. In a typical Block Matching Algorithm, each frame is divided into blocks, each of which consists of luminance and chrominance blocks. Usually, for coding efficiency, motion estimation is performed only on the luminance block. Each luminance block in the present frame is matched against candidate blocks in a search area on the reference frame. These candidate blocks are just the displaced versions of original block. The best candidate block is found and its

displacement (motion vector) is recorded. In a typical interframe coder, the input frame is subtracted from the prediction of the reference frame. Consequently the motion vector and the resulting error can be transmitted instead of the original luminance block; thus interframe redundancy is removed and data compression is achieved. At receiver end, the decoder builds the frame difference signal from the received data and adds it to the reconstructed reference frames.

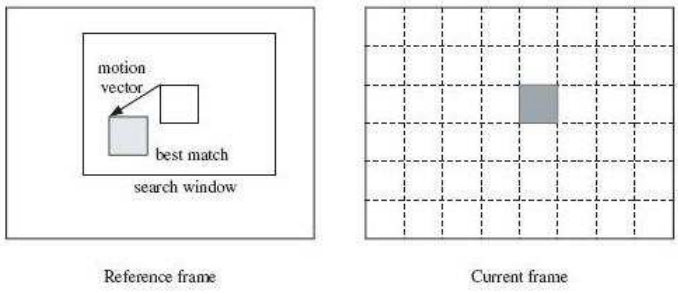

Figure 2.2: Block-matching Motion estimation

This algorithm is based on a translational model of the motion of objects between frames. It also assumes that all pels within a block undergo the same translational movement.There are many other ME methods, but BMME is normally preferred due to its simplicity and good compromise between prediction quality and motion overhead.This assumption is not strictly valid, since we capture 3-D scenes through the camera and objects do have more degrees of freedom than just the translational one. However, the assumptions are still reasonable, considering the practical movements of the objects over one frame and this makes our computations much simpler.

There are many other approaches to motion estimation, some using the frequency or wavelet domains, and designers have considered scope to invent new methods since this process does not need to be specified in coding standards. The standards need only specify how the motion vectors should be interpreted by the decoder . Block Matching (BM) is the most common method of motion estimation. Typically each macro block (16×16 pels) in the new frame is compared with shifted regions of the same size from the previous decoded frame, and the shift which results in the minimum error is selected as the best motion vector for that macro block. The motion compensated prediction frame is then formed from all the shifted regions from the previous decoded frame [5].

Backward Motion Estimation

The motion estimation generally considered as backward motion estimation, since the current frame is considered as the candidate frame and the reference frame on which the

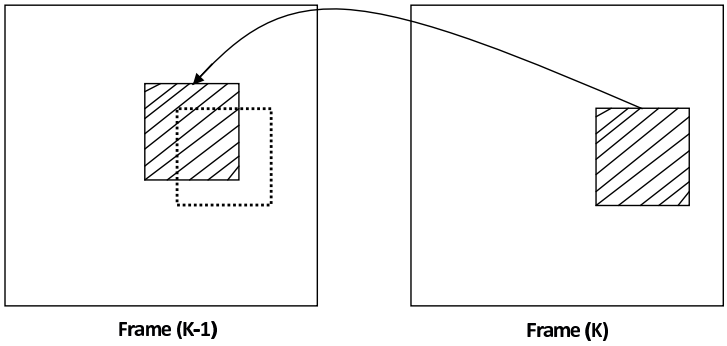

Figure 2.3: Backward Motion estimation with current frame as k and frame (k-1) as the reference frame

motion vectors are searched is a past frame, that is, the search is backward. Backward motion estimation leads to forward motion prediction.

Forward Motion Estimation

It is just the opposite of backward motion estimation. Here, the search for motion vectors is carried out on a frame that appears later than the candidates frame in temporal ordering. In other words, the search is "forward". Forward motion estimation leads to backward motion prediction. It may appear that forward motion estimation is unusual, since one requires future frames to predict the candidate frame. However, this is not unusual, since the candidate frame, for which the motion vector is being sought is not necessarily the current, that is the most recent frame. It is possible to store more than one frame and use one of the past frames as a candidate frame that uses another frame, appearing later in the temporal order as a reference.

Forward motion estimation (or backward motion compensation) is supported under the MPEG 1 & 2 standards, in addition to the conventional backward motion estimation. The standard also supports bi-directional motion compensation in which the candidate frame is predicted from a past reference as well as a future reference frame with respect to the candidates frame.

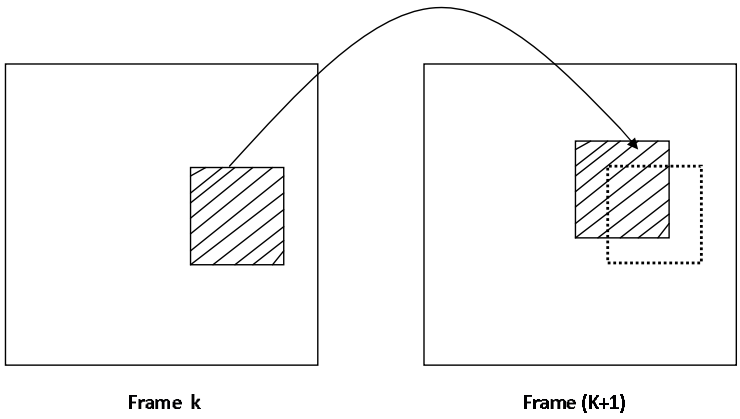

Figure 2.4: Forward Motion estimation with current frame as k and frame (k+1) as the reference frame

2.2.1 Block Matching Methods

Block-matching motion estimation (BMME) is the most widely used motion estimation method for video coding. Interest in this method was initiated by Jain and Jain and he praposed a block-matching algorithm (BMA)in 1981. The current frame, f_t , is first divided into blocks of $M \times N$ pels. The algorithm then assumes that all pels within the block undergo the same translational movement. Thus, the same motion vector, d $=[d_x,d_y]^T$, is assigned to all pels within the block. This motion vector is estimated by searching for the best match block in a larger search window of $(M + 2d_{m_x}) \times (N + 2d_{m_y})$ pels centered at the same location in a reference frame, $f_{t-\Delta t}$, where d_{m_x} and d_{m_y} are the maximum allowed motion displacements in the horizontal and vertical directions, respectively.

2.2.2 Matching Criteria for Motion Estimation

Inter frame predictive coding is used to eliminate the large amount of temporal and spatial redundancy that exists in video sequences and helps in compressing them. In conventional predictive coding the difference between the current frame and the predicted frame is coded and transmitted. The better the prediction, the smaller the error and hence the transmission bit rate when there is motion in a sequence, then a pel on the same part of the moving object is a better prediction for the current pel..There are a number of criteria to evaluate the "goodness" of a match.

Three popular matching criteria used for block-based motion estimation are

1. Mean of squarred error (MSE)

2. Sum of absolute difference (SAD)

3. Matching pel count (MPC)

To implement the block motion estimation, the candidate video frame is partitioned into a set of non overlapping blocks and the motion vector is to be determined for each such candidate block with respect to the reference. For each of these criteria, square block of size $N \times N$ pixels is considered. The intensity value of the pixel at coordinate (n_1, n_2) in the frame k is given by ,$S(n_1, n_2, k)$ where $(0 \leq n_1, n_2 \leq N-1)$. The frame k is referred to as the candidate frame and the block of pixels defined above is the candidates block.

MSE Criterion

Considering $(k-l)$ as the past references frame $l > 0$ for backward motion estimation, the mean square error of a block of pixels computed at a displacement (i, j) in the reference frame is given by

$$MSE(i,j) = \frac{1}{N^2} \sum_{n_1=0}^{N-1} \sum_{n_2=0}^{N-1} [s(n_1, n_2, k) - s(n_1 + i, n_2 + j, k - l)]^2 \tag{2.1}$$

Consider a block of pixels of size $N \times N$ in the reference frame, at a displacement of, where i and j are integers with respect to the candidate block position. The MSE is computed for each displacement position (i, j),within a specified search range in the reference image and the displacement that gives the minimum value of MSE is the displacement vector which is more commonly known as motion vector and is given by

$$[d_1, d_2] = arg \underbrace{min}_{i,j} [MSE(i,j)] \tag{2.2}$$

The MSE criterion defined in equation 2.1 requires computation of N^2 subtractions, N^2 multiplications (squaring) and $(N^2 - 1)$ additions for each candidate block at each search position. This is computationally costly and a simpler matching criterion, as defined below is often preferred over the MSE criterion.

SAD Criterion

Like the MSE criterion, the sum of absolute difference (SAD) too makes the error values as positive, but instead of summing up the squared differences, the absolute differences are summed up. The SAD measure at displacement (i, j) is defined as

$$SAD(i,j) = \frac{1}{N^2} \sum_{n_1=0}^{N-1} \sum_{n_2=0}^{N-1} [s(n_1, n_2, k) - s(n_1 + i, n_2 + j, k - l)] \tag{2.3}$$

The motion vector is determined in a manner similar to that for MSE as

$$[d_1, d_2] = arg \underbrace{min}_{i,j} [SAD(i,j)] \tag{2.4}$$

The SAD criterion shown in equation 2.3 requires N^2 computations of subtractions with absolute values and additions N^2 for each candidate block at each search position. The absence of multiplications makes this criterion computationally more attractive and facilitates easier hardware implementation.

MPC Criterion

In this criterion, the pixels of the candidates block B are compared with the corresponding pixels in the block with displacement (i, j),in the reference frame and those which are less than a specified threshold, i.e., closely matched are counted. The count for matching and the displacement (i,j),for which the count is maximum correspond to the motion vector. We define a binary valued function $count(n_1, n_2) \forall (n_1, n_2) \in B$ as

$$count(n_1, n_2) = \begin{cases} 1 & \text{if } |s(n_1, n_2, k) - s(n_1 + i, n_2 + j, k - l)| \leq \theta \\ 0 & \text{otherwise} \end{cases} \tag{2.5}$$

where, θ is a pre-determined threshold. The matching pel count (MPC) at displacement (i, j) is defined as the accumulated value of matched pixels as given by

$$MPC(i,j) = \sum_{n_1=0}^{N-1} \sum_{n_2=0}^{N-1} [count(n_1, n_2)] \tag{2.6}$$

$$[d_1, d_2] = arg \underbrace{max}_{i,j} [MPC(i,j)] \tag{2.7}$$

Block Size

Another important parameter of the BMA is the block size.If the block size is smaller, it achieves better prediction quality. This is due to a number of reasons. A smaller block size reduces the effect of the accuracy problem. In other words, with a smaller block size, there is less possibility that the block will contain different objects moving in different directions.

In addition, a smaller block size provides a better piecewise translational approximation to nontranslational motion. Since a smaller block size means that there are more blocks (and consequently more motion vectors) per frame, this improved prediction quality comes at the expense of a larger motion overhead. Most video coding standards use a block size of 16×16 as a compromise between prediction quality and motion overhead. A number of variable-block-size motion estimation methods have also been proposed in the literature. H.263 and MPEG standards allows adaptive switching between block sizes of 16×16 and 8×8 on an Macro Block (MB) basis.

Motion compensation for each 16×16 macroblock can be performed using a number of different block sizes and shapes. The original luminance component of each macroblock (16×16) may be spilt into 4 kinds of size: 16×16, 16×8, 8×16, 8×8, as shown in Figure 2.5 (a). Each of the sub-divided regions is a macroblock partition. If the 8×8 mode is chosen, each of the four 8×8 may be spilt into 4 kinds of size:8×8, 8×4, 4×8, 4×4 as shown in Figure 2.5 (b). These partitions and sub-partitions compose to a large number to a large number of possible combinations.

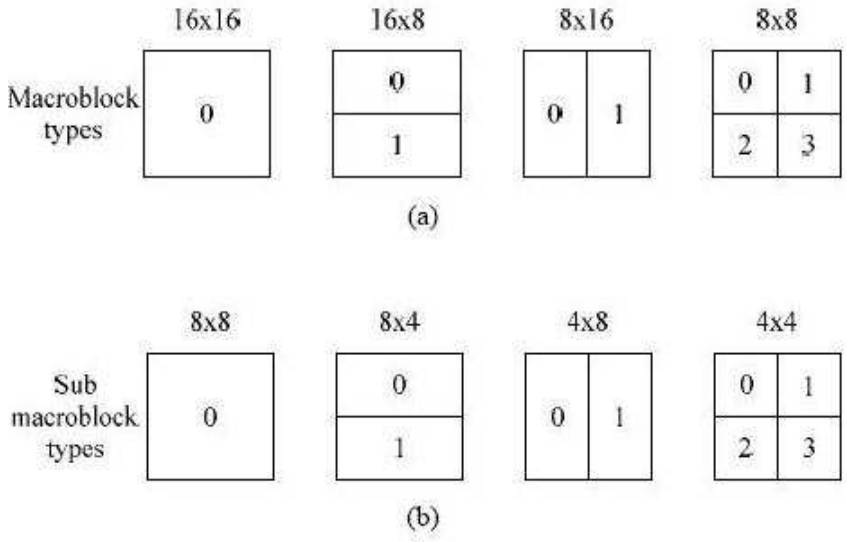

Figure 2.5: macroblock (a) partition and (b) sub partition

A separate motion vector is required for each partition or sub-partition. Each of the motion vector must be coded and transmitted, besides, the choice of partition must be

encoded in the bitstream. If we use smaller partition size (eg. 8×8, 4×4 etc), we can get smaller prediction error, but we must consume some a lot of bits to store motion vector. Otherwise, if we choice larger partition size (eg. 16×16, 16×8, 8×16), we can save a lot of bits, but the prediction error will be large. In general, a large partition size is appropriate for homogeneous areas of the frame and a small partition size may be beneficial for detailed areas. Usually, using seven different block sizes can translate into bit-rate savings of more than 15% as compared to using only a 16×16 block size.

Search Range

The maximum allowed motion displacement d_m , also known as the search range, has a direct impact on both the computational complexity and the prediction quality of the BMA. A small d_m results in poor compensation for fast-moving areas and consequently poor prediction quality. A large d_m, on the other hand, results in better prediction quality but leads to an increase in the computational complexity (since there are $(2d_m+1)^2$ possible blocks to be matched in the search window). A larger d_m can also result in longer motion vectors and consequently a slight increase in motion overhead [6]. In general, a maximum allowed displacement of $d_m = \pm 15$ pels is suffcient for low-bit-rate applications. MPEG standard uses a maximum displacement of about ± 15 pels, although this range can optionally be doubled with the unrestricted motion vector mode.

Search Accuracy

Initially, the BMA was designed to estimate motion displacements with full-pel accuracy. Clearly, this limits the performance of the algorithm, since in reality the motion of objects is completely unrelated to the sampling grid. A number of workers in the field have proposed to extend the BMA to subpel accuracy. For example, Ericsson demonstrated that a prediction gain of about 2 dB can be obtained by moving from full-pel to 1/8-pel accuracy. Girod presented an elegant theoretical analysis of motion-compensating prediction with subpel accuracy. He termed the resulting prediction gain the accuracy effect. He also showed that there is a "critical accuracy" beyond which the possibility of further improving prediction is very small. He concluded that with block sizes of 16×16, quarter-pel accuracy is desirable for broadcast TV signals, whereas half-pel accuracy appears to be suffcient for videophone signals. Today, most video coding standards adopt subpel accuracy in its halfpel form. In fact, it has been shown that most of the performance gain of H.263 over H.261 can be attributed to the move from full-pel to half-pel accuracy.

It should be pointed out, however, that the improved prediction quality of subpel accuracy comes at the expense of a significant increase in computational complexity. This increase is due to two reasons. First, the reference frame intensities have to be interpolated at subpel locations. Second, there are now more possible candidate blocks within the search window. For example, when moving from full-pel to half-pel accuracy, the number of candidate blocks in the search window increases from $(2d_m + 1)^2$ to $(4d_m + 1)^2$. To alleviate this complexity, most video codecs implement subpel accuracy as a postprocessing stage, where first a full-pel motion vector is obtained, usually using full search, and then this vector is refined to subpel accuracy using a limited search. This provides a large saving in computational complexity and at the same time maintains the improved prediction quality.

2.3 Search Algorithms for Motion Estimation

Basic Approaches to Motion Estimation

There exists two basic approaches to motion estimation

1. Pixel based motion estimation
2. Block-based motion estimation.

The pixel based motion estimation approach seeks to determine motion vectors for every pixel in the image. This is also referred to as the optical flow method, which works on the fundamental assumption of brightness constancy, that is the intensity of a pixel remains constant, when it is displaced. However, no unique match for a pixel in the reference frame is found in the direction normal to the intensity gradient. It is for this reason that an additional constraint is also introduced in terms of the smoothness of velocity (or displacement) vectors in the neighborhood. The smoothness constraint makes the algorithm interactive and requires excessively large computation time, making it unsuitable for practical and real time implementation.

An alternative and faster approach is the block based motion estimation. In this method, the candidates frame is divided into non-overlapping blocks (of size 16×16, or 8×8 or even 4×4 pixels in the recent standards) and for each such candidate block, the best motion vector is determined in the reference frame. Here, a single motion vector is computed for the entire block, whereby we make an inherent assumption that the entire block undergoes translational motion. This assumption is reasonably valid, except

for the object boundaries and smaller block size leads to better motion estimation and compression.

Block based motion estimation is accepted in all the video coding standards proposed till date. It is easy to implement in hardware and real time motion estimation and prediction is possible.

2.3.1 Full Search Motion Estimation

In order to get the best match block in the reference frame, it is necessary to compare the current block with all the candidate blocks of the reference frames. Full search motion estimation calculates the sum absolute difference (SAD) value at each possible location in the search window. Full search computed the all candidate blocks intensive for the large search window.

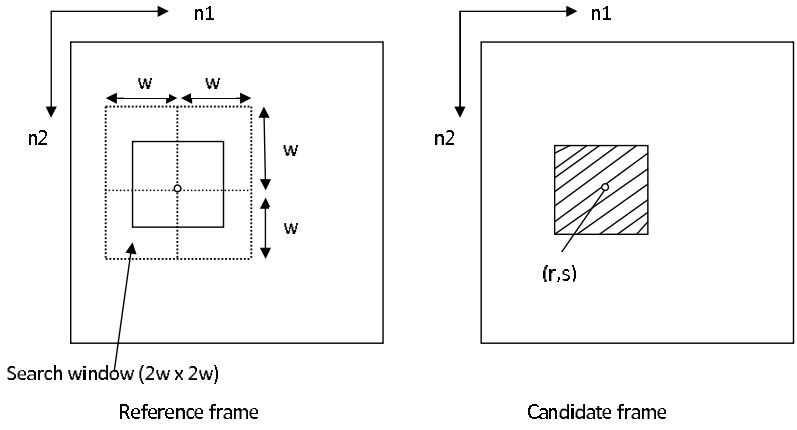

Figure 2.6: Full search motion estimation

consider a block of $N \times N$ pixels from the candidates frame at the coordinate position (r, s) as shown and then consider a search window having a range $\pm w$ in both and directions in the references frame, as shown. For each of the $(2w + 1)^2$ search position (including the current row and the current column of the reference frame), the candidate block is compared with a block of size $N \times N$ pixels, according to one of the matching criteria discussed in section 2.2.2 and the best matching block, along with the motion vector is determined only after all the $(2w+1)^2$ search position are exhaustively explored.

The FSBM is optimal in the sense that if the search range is correctly defined, it is guaranteed to determine the best matching position. However, it is highly computational intensive. For each matching position, we require $O(N^2)$ computations (additions,

Method	**Computations**		
	Add/sub	**Multiplication**	**Comparison**
MSE	$(2N^2-1)(2w+1)^2$	$N^2(2w+1)^2$	$(2w+1)^2$
SAD	$(2N^2-1)(2w+1)^2$	——	$(2w+1)^2$
MPC	$(2N^2-1)(2w+1)^2$	——	$N^2(2w+1)^2$

Table 2.1: Computational Complexity of FSBM

subtractions, multiplications etc) and since there are(search positions, the number of computations for each matching criterion is given by Table 2.1

FSBM requires large number of computations. If $w = 7$ pixels, measuring that the best matching position exists within a displacement of ± 7 pixels from the current block position, it requires $15 \times 15 = 225$ search positions. For real time implementation, quick and efficient search strategies were explored.It may be noted that such quick search techniques do not make exhaustive search within the search within the search area and can at best be sub-optimal.

2.3.2 Three Step Search (TSS)

This algorithm was introduced by Koga et al in 1981 [7]. It became very popular because of its simplicity and also robust and near optimal performance. It searches for the best motion vectors in a course to fine search pattern. The algorithm may be described as:

Step 1: An initial step size is picked. Eight blocks at a distance of step size from the centre (around the centre block) are picked for comparison.

Step 2: The step size is halved. The centre is moved to the point with the minimum distortion.

The point which gives the smallest criterion value among all tested points is selected as the final motion vector m.TSS reduces radically the number of candidate vectors to test, but the amount of computation required for evaluating the matching criterion value for each vector stays the same. TSS may not find the global minimum (or maximum) of the matching criterion; instead it may find only a local minimum and this reduces the quality of the motion compensation system. On the other hand, most criteria can be easily used with TSS.

2.3.3 Diamond Search(DS) Algorithm

By exhaustively testing on all the candidate blocks , full search (FS) algorithm gives the global minimum SAD position which corresponds to the best matching block at the expense of highly computation. To overcome this defect, many fast BMAs are developed such as diamond search (DS) [8] and the cross-search algorithm [9] and new four step search [10].

The DS algorithm employs two search patterns. The first pattern, called large diamond search pattern (LDSP) shown in figure 2.7 (a), comprises nine checking points from which eight points surround the center one to compose a diamond shape. The second pattern consisting of five checking points forms a small diamond shape, called small diamond search pattern (SDSP) shown in figure 2.7 (b). In the searching procedure of the DS algorithm, LDSP is repeatedly used until the minimum block distortion (MBD) occurs at the center point. The search pattern is then switched from LDSP to SDSP when it reaches the final search stage. Among the five checking points in SDSP, the position yielding the minimum block distortion (MBD) provides the motion vector of the best matching block. The DS algorithm can not only provides similar or in many cases better results than NTSS, but also reduces complexity by as much as 75%. But it appears that DS has significant quality degradation with sequences containing significant global motion, or when coding higher resolution sequences.

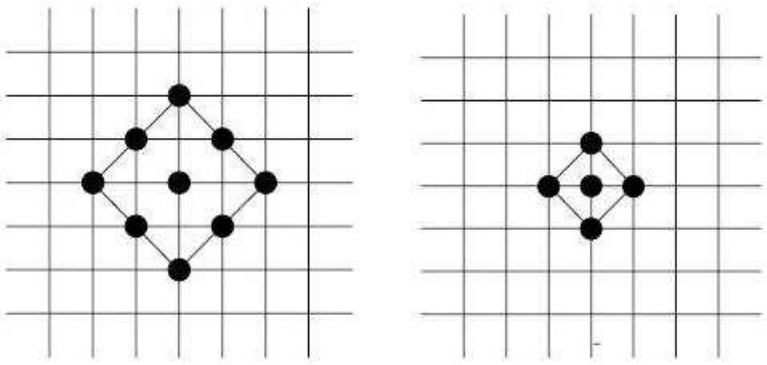

Figure 2.7: (a) Large Diamond Search Pattern (b) Small Diamond Search Pattern

The large diamond search pattern (LDSP) contains 9 search points from eight sur-

round to the center point for one and compose to a diamond shape and used at first and repeatedly until the minimum SAD point is the center point. The small diamond search pattern (SDSP) contains 5 search points.

The DS algorithm is summarized as follows.

1. The initial LDSP is centered at the origin of the search window, and the 9 checking points of LDSP are tested. If the MBD point calculated is located at the center position, go to Step 3; otherwise, go to Step 2.

2. The MBD point found in the previous search step is re-positioned as the center point to form a new LDSP. If the new MBD point obtained is located at the center position, go to Step 3; otherwise, recursively repeat this step.

3. Switch the search pattern from LDSP to SDSP. The MBD point found in this step is the final solution of the motion vector which points to the best matching block. Some insightful remarks on implementing the DS algorithm are provided as follows.

The compact shape of the search patterns used in the DS algorithm increases the possibility of finding the global minimum point located inside the search pattern. Therefore, the DS algorithm tends to produce smaller or at least similar motion estimation error compared with other fast BMAs. Unlike TSS, NTSS and 4SS, the search window size is not restricted by the searching strategy in DS algorithm. DS algorithm greatly outperforms the well-known TSS algorithm and achieves close MSE performance compared to NTSS while reducing its computation by up to 22% approximately. The DS is implemented in the MPEG-4 video-encoding environment and its efficiency is demonstrated through core experimental results. Based on these results, it is adopted and incorporated in MPEG-4 verification model.

CHAPTER 3

FAST MOTION ESTIMATION ALGORITHM BASED ON GENETIC ALGORITHM (GA)

3.1 Introduction to Genetic Algorithm

Genetic algorithms (GAs) are adaptive methods which may be used to solve search and optimisation problems. They are based on the genetic processes of biological organisms. Over many generations, natural populations evolve according to the principles of natural selection and "survival of the fittest". By mimicking this process, genetic algorithms are able to "evolve" solutions to real world problems, if they have been suitably encoded. The basic principles of GAs were first laid down rigorously by Holland.

GAs work with a population of "individuals", each representing a possible solution to a given problem. Each individual is assigned a "fitness score" according to how good a solution to the problem it is. The highly-fit individuals are given opportunities to "reproduce", by "cross breeding" with other individuals in the population. This produces new individuals as "offspring", which share some features taken from each "parent". The least fit members of the population are less likely to get selected for reproduction, and so "die out".

A whole new population of possible solutions is thus produced by selecting the best individuals from the current "generation", and mating them to produce a new set of individuals [11]. This new generation contains a higher proportion of the characteristics possessed by the good members of the previous generation. In this way, over many generations, good characteristics are spread throughout the population. By favouring

the mating of the more fit individuals, the most promising areas of the search space are explored.

3.2 Genetic Algorithm

Evaluation Function

It provides a measure of performance with respect to a particular set of parameters. The fitness function transforms that measure of performance into an allocation of reproductive opportunities. The evaluation of a string representing a set of parameters is independent of the evaluation of any other string. The fitness of that string, however, is always defined with respect to other members of the current population. In the genetic algorithm, fitness is defined by: f_i/f_A where f_i is the evaluation associated with string i and f_A is the average evaluation of all the strings in the population.

The notion of evaluation and fitness are sometimes used interchangeably, However, it is useful to distinguish between the evaluation function and the fitness function used by a genetic algorithm. The evaluation function or objective function provides a measure of performance with respect to a particular set of parameters. The fitness function transforms that measure of performance into an allocation of reproductive opportunities. The evaluation of a string representing a set of parameters is independent of the evaluation of any other string. The fitness of that string, however, is always defined with respect to other members of the current population.

Coding

Before a GA can be run, a suitable coding (or representation) for the problem must be devised. We also require a fitness function, which assigns a figure of merit to each coded solution. During the run, parents must be selected for reproduction, and recombined to generate offspring. It is assumed that a potential solution to a problem may be represented as a set of parameters (for example, the parameters that optimise a neural network). These parameters (known as genes) are joined together to form a string of values (often referred to as a chromosome. For example, if our problem is to maximise a function of three variables, F(x; y; z), we might represent each variable by a 10-bit binary number (suitably scaled), chromosome would therefore contain three genes, and consist of 30 binary digits. The set of parameters represented by a particular chromosome is referred to as a genotype. The genotype contains the information required to construct an organism

which is referred to as the phenotype. For example, in a bridge design task, the set of parameters specifying a particular design is the genotype, while the finished construction is the phenotype.

The fitness of an individual depends on the performance of the phenotype. This can be inferred from the genotype, i.e. it can be computed from the chromosome, using the fitness function. Assuming the interaction between parameters is nonlinear, the size of the search space is related to the number of bits used in the problem encoding. For a bit string encoding of length L; the size of the search space is 2^L and forms a hypercube [12]. The genetic algorithm samples the corners of this L–dimensional hypercube. Generally, most test functions are at least 30 bits in length; anything much smaller represents a space which can be enumerated. Obviously, the expression 2^L grows exponentially. As long as the number of "good solutions" to a problem are sparse with respect to the size of the search space, then random search or search by enumeration of a large search space is not a practical form of problem solving. On the other hand, any search other than random search imposes some bias in terms of how it looks for better solutions and where it looks in the search space. A genetic algorithm belongs to the class of methods known as "weak methods" because it makes relatively few assumptions about the problem that is being solved.

Genetic algorithms are often described as a global search method that does not use gradient information. Thus, nondifferentiable functions as well as functions with multiple local optima represent classes of problems to which genetic algorithms might be applied. Genetic algorithms, as a weak method, are robust but very general.

Chromosome Representation

Each chromosome represents the data for coordinates x and y. Each coordinate is described by 2 genes: object and strategy genes. Object gene defines the actual coordinate in the image. The value of object gene is determined from the search window size. For example, if the search window is within the range [-16, 16], then the value of object gene can take any integer number inside of this range. The search window size depends on the maximum motion vector size. Strategy gene determines whatever local or global search will be carried out. Smaller value of strategy gene more localised become the search process. The negative value defines the decrement of mutated gene and positive values respectively determine the increment of the mutated gene. The strategy parameter depends on the window size and can take any value up to its maximum. In order to implement the local search, we choose to set the strategy parameter to values 0 or 1.

Selection

Selection is the component which guides the algorithm to the solution by preferring individuals with high fitness over low-fitted ones. It can be a deterministic operation, but in most implementations it has random components.Fitness can also be assigned based on a string's rank in the population or by sampling methods, such as tournament selection. Tournament selection is one of many methods of selection in genetic algorithms which runs a "tournament" among a few individuals chosen at random from the population and selects the winner (the one with the best fitness) for crossover easily adjusted by changing the tournament size. If the tournament size is larger, weak individuals have a smaller chance to be selected.Tournament selection does not care about the spread of the scores, only the ranking. The n^{th} ranked invididual in a population of size m_u will have a $(2m_u - 2n + 1)/m_u^2$ chance of reproducing. This puts an upper and lower bound on the chances of any individual to reproduce for the next generation. Tournament selection can be generalized to include more than 2 individuals being chosen for competition, and selecting the best from this group.

After selection has been carried out the construction of the intermediate population is complete and recombination can occur. This can be viewed as creating the next population from the intermediate population. Crossover is applied to randomly paired strings with a probability denoted pc. (The population should already be sufficiently shuffled by the random selection process.) Pick a pair of strings. With probability P_c "recombine" these strings to form two new strings that are inserted into the next population.

Crossover

In sexual reproduction, as it appears in the real world, the genetic material of the two parents is mixed when the gametes of the parents merge. Usually, chromosomes are randomly split and merged, with the consequence that some genes of a child come from one parent while others come from the other parents.This mechanism is called crossover. It is a very powerful tool for introducing new genetic material and maintaining genetic diversity, but with the outstanding property that good parents also produce well-performing children or even better ones. Several investigations have come to the conclusion that crossover is the reason why sexually reproducing species have adapted faster than asexually reproducing ones.

Basically, crossover is the exchange of genes between the chromosomes of the two parents. In the simplest case, we can realize this process by cutting two strings at a

randomly chosen position and swapping the two tails. This process, which we will call one-point crossover in the following, is visualized in Figure 3.1. One-point crossover is a simple and often-used method for GAs which operate on binary strings. For other problems or different codings, other crossover methods can be useful or even necessary

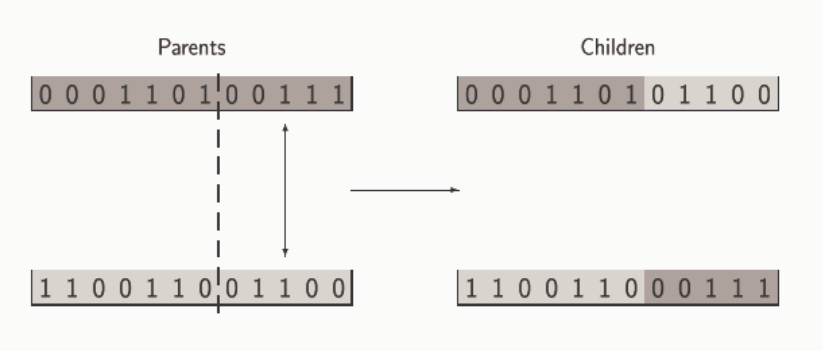

Figure 3.1: One-point crossover of binary strings.

N-point crossover: Instead of only one, N breaking points are chosen randomly. Every second section is swapped. Among this class, twopoint crossover is particularly important.

Segmented crossover: Similar to N-point crossover with the difference that the number of breaking points can vary.

Uniform crossover: For each position, it is decided randomly if the positions are swapped.

Shuffle crossover: First a randomly chosen permutation is applied to the two parents, then N-point crossover is applied to the shuffled parents, finally, the shuffled children are transformed back with the inverse permutation.

Mutation

The last ingredient of simple genetic algorithm is mutation–the random deformation of the genetic information of an individual by means of radio active radiation or other environmental influences. In real reproduction, the probability that a certain gene is mutated is almost equal for all genes. So, it is near at hand to use the following mutation technique for a given binary string s, where p_m is the probability that a single gene is modified. p_m should be rather low in order to avoid that the GA behaves chaotically like a random search.Again, similar to the case of crossover, the choice of the appropriate mutation technique depends on the coding and the problem itself. The motivation for using mutation, then, is to prevent the permanent loss of any particular bit or allele. After several

generations it is possible that selection will drive all the bits in some position to a single values either 0 or 1, If this happens without the genetic algorithm converging to a satisfactory solution, then the algorithm has prematurely converged. This may particularly be a problem if one is working with a small population.

Inversion of single bits: With probability p_m, one randomly chosen bit is negated.

Bitwise inversion: The whole string is inverted bit by bit with prob. p_m.

Random selection: With probability p_m, the string is replaced by a randomly chosen one.

Strenghts of GA

The power of GAs comes from the fact that the technique is robust and can deal successfully with a wide range of difficult problems. GAs are not guaranteed to find the global optimum solution to a problem, but they are generally good at finding "acceptably good" solutions to problems "acceptably quickly". Where specialised techniques exist for solving particular problems, they are likely to outperform GAs in both speed and accuracy of the final result.

Even where existing techniques work well, improvements have been made by hybridising them with a GA. The basic mechanism of a GA is so robust that, within fairly wide margins, parameter settings are not critical.

Weaknesses of GA

A problem with GAs is that the genes from a few comparatively highly fit individuals may rapidly come to dominate the population, causing it to converge on a local maximum. Once the population has converged, the ability of the GA to continue to search for better solutions is effectively eliminated: crossover of almost identical chromosomes produces little that is new. Only mutation remains to explore entirely new ground, and this simply performs a slow, random search.

The standard GA can be represented as shown in figure 3.2:

In the first generation the current population is also the initial population. After calculating fitness for all the strings in the current population, selection is carried out. The probability that strings in the current population are copied (i.e. duplicated) and placed in the intermediate generation is in proportion to their fitness.

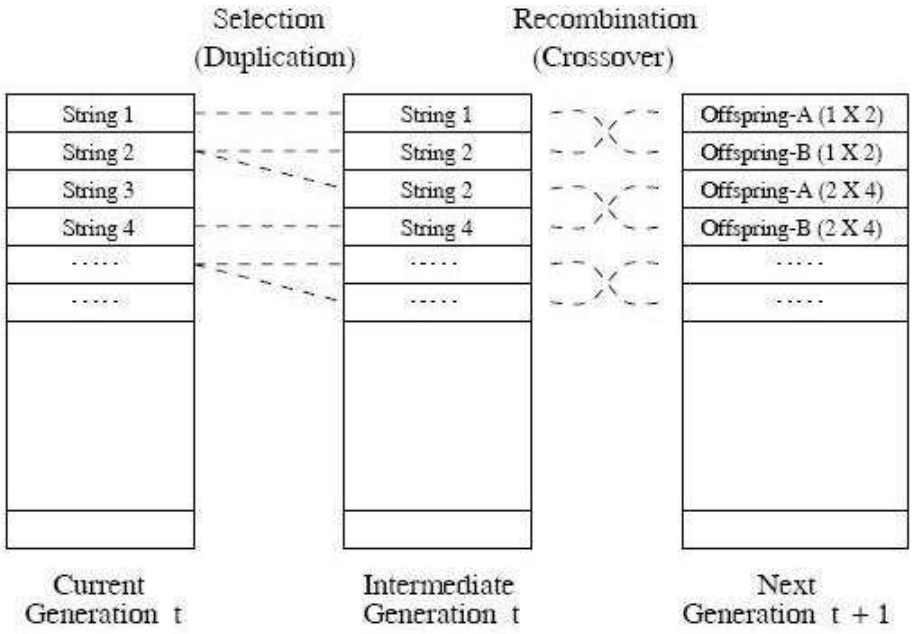

Figure 3.2: Genetic algorithm procedure

3.3 Block Matching Algorithm based on GA

Motion estimation is an essential component of many video encoding algorithms. The most popular method adopted to estimate the motion between frames is the block matching algorithm (BMA), in which a block of size $M \times N$ is compared with a corresponding block within a search area in the previous frame. Three main elements-match criterion, search area and search scheme-should be considered in the BMA. The match scheme is the most important, which plays a crucial part in the performance of BMA.

Essenticlly, the match scheme of BMA is an optimal problem .The full search BMA will always find the optimum motion vector by calculating the difference between every block in the search window from the previous frame and the current block. However, the price paid for this optimum performance is a high computational cost, which prevents it from being applied in most real-time systems. A number of fast search algorithm have been proposed to greatly reduce the computational complexity by finding a sub-optimum motion. All these algorithms rely on an assumption that the difference measure decreases monotonically as the search position moves closer to the optimum position.Because this

assumption usually does not hold, these fast algorithms may only find the local minimum and can not find the global minimum. Thus the quality of encoded video may become much worse.

The genetic algorithm is an optimum-searching process based on the laws of natural selection and genetics. It adopts the coding set of parameters instead of the parameters themselves to operate and searches the optimum based on groups of points, not a single point. Moreover, it uses a random, instead of a definite rule to work on the searching process. All of these give it high robustness as well as parallelism, and enable it to be free of the limitation in continuity and single peak requirement. It is effective in solving the problems of searching global optimum, although the computational complexity of genetic algorithm is high [13].

To solve an actual problem with genetic algorithm, the parameters of the problem are coded firstly; and then a fitness function should be chosen for determining the winner. Later the initial populations are selected and begin to evolve, that is, are processed by the selection operator, crossover operator, mutation operator and other genetic operators. The genetic fast BMA is discussed in detail as follows

3.3.1 Genetic BMA procedure

Coding scheme

The result space should be bi-directionally mapped into a space of chromosomes. Since the motion vector is represented by $MV(x, y)$, It is binary encoded into $(x_1, x_2, ..x_n, y_1, y_2, ..y_n)$, where x_i, y_i=0 or 1, $n = [\log_2 S_m] + 1$, $S_m = max(s_x, s_y)$ s_x and s_y represent the half width and the half height of the search window respectively.Here, x_1, y_1, is specially used for denoting the sign of the vector. (i.e. $x_i = 0$ denotes positive motion vector; $x_i = 1$ denotes negative motion vector.)

Fitness Function

Here we have taken the fitness function as sum of absolute difference (SAD) since SAD criterion requires N^2 computations of subtractions with absolute values and additions N^2 for each candidate block at each search position. The absence of multiplications makes this criterion computationally more attractive and facilitates easier hardware implementation.

Initial Population and Population Size

In common genetic algorithms, the initial population are selected in random. Owing to their significant effects on the performance of the search, here they are selected according to a rule that they should be as near as possible to the global optimum. This rule is based on two facts:

- Most of the motion vectors are center-biased.It has been applied in many fast BMAs. The center point of the search window and its neighboring eight points can be chosen for parts of the initial population.
- Neighboring motion vectors in one frame are similar.

  Center-biased feature denotes that match point may exist within a small zone around macro block center. selection of individuals as shown in Figure. 3.3 which distribute around center of macro block. The white dot is center of macro block and dark dots are individuals. Usually we let individuals randomly distribute in search window. However, if individual positions are around optimization position, it can speed up population convergence. Intial population size is 8 and we distribute intial population equally in all 8 directions with a view to find the matching MB in each direction with equal possibility.

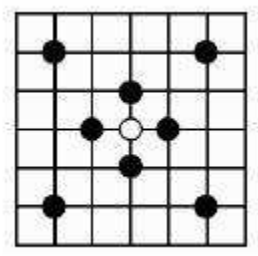

Figure 3.3: selection of intial Population

Genetic Operator

Common genetic algorithms usually choose selection operator, crossover operator and mutation operator for evolution. In this algorithm, only selection operator, mutation

operator and a special optimization operator are used according to the character of motion vectors.

Selection Operator:

In a general selection process, a chromosome is selected based on a probabilistic scheme, which costs much in computation. To reduce the complexity, a simple threshold-based selection mechanism according to the mean fitness of all chromosomes is provided. The chromosomes, whose fitness is higher than the mean fitness, will be copied to the new population. Others will be first acted on by a mutation operator, then enter the new population. Thus the chromosome with a larger fitness, namely, the search point with a smaller matching error will have enough opportunity to survive.

Mutation Operator:

After being carefully studied, the genetic mutation is found to have an actual effect in selecting the next search point. If the significant genes of the chromosomes are mutated, the next search point will be far from the center point of the search window, whereas the next will be nearer to the center if the less significant genes is mutated. Owing to its outstanding performance, the mutation operator is well used in searching the global optimum while the crossover operator is removed. Because the horizontal vector x has nothing to do with the vertical vector y, the mutation in the higher piece of the chromosome and the mutation in the lower will be performed individually, that is, at first two random numbers R_x, and R_y are generated, then the corresponding x_{R_x}, y_{R_y} will be bitwise reversed and the new chrom

Stopping Rule

Here the largest fitness of the chromosomes is available when the matching error is zero. At this time the evolution should be broken for the global optimum has been found. However, in most circumsknce the matching error can't reach zero, the evolution should be stopped under the limitation in the number of generations. Due to the center-biased characteristics of real-world motion fields, we adopt the fixed-iteration method. Generally, there are two widely adopted stopping criteria. One is fixed-iteration, that is, given a certain iteration time, saying N, the search stops after N iterations. The other is specified-threshold. For minimization problems, we specify a very small threshold, and if the change of gbest during t times of iteration is smaller than the threshold, we consider the group best value

Sequences	Format	Threshold
Akiyo	QCIF	350
Container	QCIF	300
Mother & daughter	QCIF	250
News	QCIF	250
Silent	QCIF	200

Table 3.1: Assumed Threshold values

very near to the global optimum, thus the matching procedure stops. Due to the center-biased characteristics of real-world motion fields, in this thesis fixed-iteration method in this paper for reducing the computational cost.

Genetic BMA can be summarized as follows

1. Select the intial population
2. Calculate the fitness of each chromosome and the mean fitness of this generation.
3. Copy or mutate the chromosome to the next generation according to the rule discussed above
4. Stop the evolution if the stopping rule is satisfied. Otherwise, go to step 2, 3.
5. Convert the strongest chromosome to the search point, and get the result.

Zero Motion Prejudgment (ZMP)

Zero Motion Prejudgment (ZMP) is the procedure to find the static macroblocks which contains zero motion. In real world video sequences more than 70% of the MBs are static which do not need the remaining search . So, significant reduction of computation is possible if we predict the static macroblocks by ZMP procedure before starting motion estimation procedure and the remaining search will be faster and saves memory. We first calculate the matching error (SAD) between the MB in the current frame and the MB at the same location in the reference frame and then compare it to a predetermined threshold, saying Δ. If the matching error is smaller than Δ we consider this MB static which do not need any further motion estimation, and return a [0,0] as its motion vector(MV).

3.3.2 Experiments and Simulation Results

The performance of the proposed GA based block matching algorithm is evaluated in terms of computation and average peak signal-to-noise ratio (PSNR) per frame of the

reconstructed video sequence is computed for quality measurement. Computation is defined as the average number of the error function evaluations per MV generation.For Zero motion Prejudgement threshold value for each test video sequence correspondingly based on data obtained in experiments shown in Table 3.1. This threshold value is not fixed, may vary depending on your video sequences.We divide a whole image frame into 16×16 MBs in the simulation that is $N = 16$ and the size of search window was set as 15×15.

For comparison, the performance of DS and our proposed algorithm are compared in terms of PSNR and computation and documented in Table 3.2 and Table 3.3 for 5 different video sequences. Search range P=7 means that the search will be performed within a square region of [-p, +p] around the position of the current block as shown in figure 3.4. In our simulation experiments, the block size is fixed at 16×16. To make a consistent comparison, block matching is conducted within a 15×15 search window. Summed absolute difference (SAD) is used as the matching criterion to reduce the block-matching computation in practice.

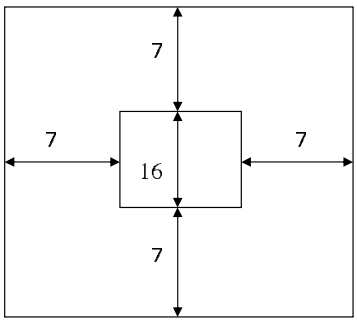

Figure 3.4: search window

To make a consistent comparison, block matching is conducted within a 15×15 search window. Summed absolute difference (SAD) is used as the matching criterion to reduce the block-matching computation in practice. In order to evaluate search performance in a wide range of motion conditions, five video sequences in QCIF were tested in this paper are Akyio, Container, mother & daughter, News, Silent from 1st frame to 100th frame. The performance of the GA based block matching algorithm is evaluated in terms of computation and average peak signal-to-noise ratio (PSNR) per frame of the reconstructed video sequence is computed for quality measurement. Computation is defined as the average number of the error function evaluations per MV generation.Here intial population size is taken as N=8, the number of the generations G=10. Figure 3.5 shows the PSNR (dB) comparison of genetic algorithm (GA) based BMA and diamond search (DS) algorithm and the figure 3.6 shows the computational comparison of GA and DS.

Sequences	GA to DS
Akiyo	3.5597
Container	4.3469
Mother & daughter	3.5288
News	4.2781
Silent	3.5234

Table 3.2: Computational gain to DS

from the simulaton results we can say that GA based block matching algorithm is several times faster than DS BMA but in PSNR comparison some acceptable drops are there. In some applications like video streaming etc.. there should be main priority to fastness of algorithm than to the quality. Anyway in future we can try variants of GA hybrid algorithms for better results.. Such approaches are suitable for achieving global optimal solution, which traditional fast BMAs are not able to obtain easily. The GA needs to set some key parameters such as population size, probability of mutation, probability of crossover, etc. If these parameters are not prefixed properly, efficiency of GA becomes lower and also it is time consuming process.

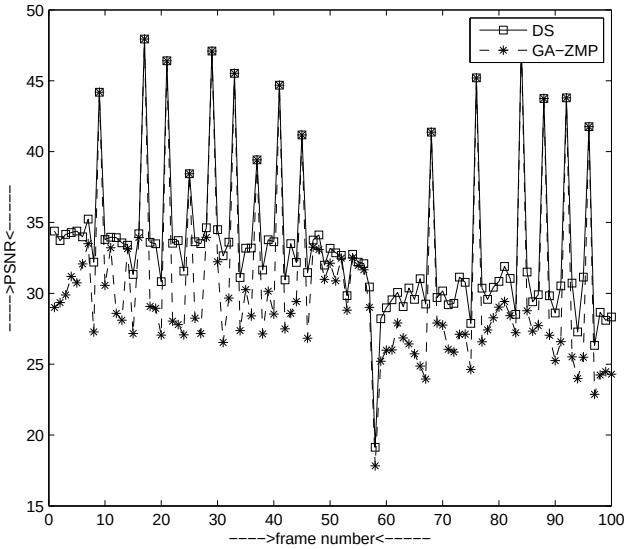

Figure 3.5: PSNR(dB) comparision of GA and DS

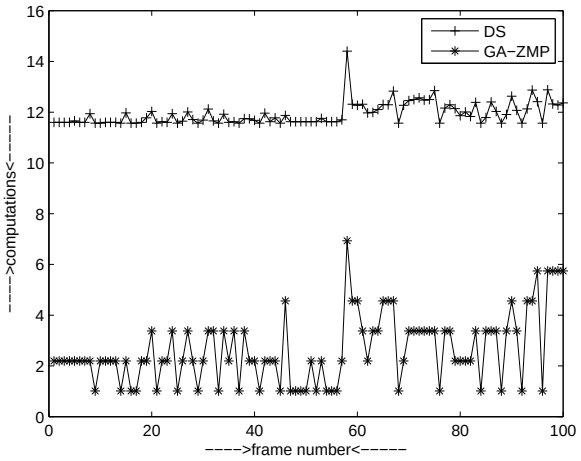

Figure 3.6: Computational comparison of GA and DS

Sequences	DS	GA
Akiyo	40.2463	38.1706
Container	37.8131	36.2360
Mother & daughter	35.7113	33.2731
News	32.2179	29.6056
Silent	32.3068	30.2544

Table 3.3: Average PSNR(dB) performances of DS, GA

3.4 Conclusion

A novel fast motion-estimation method, the genetic BMA, is praposed. It makes full use of the advantage of genetic algorithm to search the global optimum and get rid of its shortcoming of high complexity in computation. Furthermore, the fixed selection of initial population highly improves the performance. In addition, a zero-motion prejudgment (ZMP) routine is incorporated into the GA BMA to further reduce the computational cost of the algorithm. Simulation results show that the GA-ZMP BMA proposed requires less amount of computation and achieves PSNR close and acceptable PSNR performance compared to widely accepted DS BMA. The GA needs to set some key parameters such as population size, probability of mutation, probability of crossover, etc. If these parameters are not prefixed properly, efficiency of GA becomes lower and also it is time consuming process so we are going to apply Particle Swarm Optimization and other techniques in the next sections which are giving better results than GA BMA.

CHAPTER 4

FAST MOTION ESTIMATION ALGORITHM BASED ON CLONAL PARTICLE SWARM OPTIMIZATION (CPSO)

4.1 Particle Swarm Optimization (PSO)

Particle swarm optimization (PSO) is a population based stochastic optimization technique developed by Dr. Eberhart and Dr. Kennedy in 1995 [14], inspired by social behavior of bird flocking or fish schooling. It is widely accepted and focused by researchers due to its profound intelligence background and simple algorithm structure. Currently, PSO has been implemented in a wide range of research areas such as functional optimization, pattern recognition, neural network training, fuzzy system control etc. and obtained significant success. Like Genetic Algorithm(GA), PSO is also an evolutionary algorithm based on swarm intelligence. But, on the other side, unlike GA, PSO has no evolution operators such as crossover and mutation . In PSO, the potential solutions, called particles, fly through the solution space by following the current optimum particles. The original intent was to graphically simulate the graceful but unpredictable choreography of a bird flock. Through competitions and cooperations, particles follow the optimum points in the solution space to optimize the problem. Many proposals indicate that PSO is relatively more capable for global exploration and converges more quickly than many other heuristic algorithms.

PSO simulates the behaviors of bird flocking. Suppose the following scenario: a group of birds are randomly searching food in an area. There is only one piece of food in the area being searched. All the birds do not know where the food is. But they know how far the

food is in each iteration. So what's the best strategy to find the food? The effective one is to follow the bird which is nearest to the food.PSO learned from the scenario and used it to solve the optimization problems [15]. In PSO, each single solution is a "bird" in the search space. We call it "particle". All of particles have fitness values which are evaluated by the fitness function to be optimized, and have velocities which direct the flying of the particles. The particles fly through the problem space by following the current optimum particles.

Particle Swarm has two primary operators: Velocity update and Position update. During each generation each particle is accelerated toward the particles previous best position and the global best position. At each iteration a new velocity value for each particle is calculated based on its current velocity, the distance from its previous best position, and the distance from the global best position. The new velocity value is then used to calculate the next position of the particle in the search space. This process is then iterated a set number of times, or until a minimum error is achieved.

Comparisons Between Genetic Algorithm and PSO

Most of evolutionary techniques have the following procedure:

1. Random generation of an initial population
2. Reckoning of a fitness value for each subject. It will directly depend on the distance to the optimum.
3. Reproduction of the population based on fitness values.
4. If requirements are met, then stop. Otherwise go back to 2.

From the procedure, we can learn that PSO shares many common points with GA. Both algorithms start with a group of a randomly generated population, both have fitness values to evaluate the population. Both update the population and search for the optimium with random techniques. Both systems do not guarantee success.

However, PSO does not have genetic operators like crossover and mutation. Particles update themselves with the internal velocity. They also have memory, which is important to the algorithm.

Compared with genetic algorithms (GAs), the information sharing mechanism in PSO is significantly different [16]. In GAs, chromosomes share information with each other. So the whole population moves like a one group towards an optimal area. In PSO, only

gbest (or lbest) gives out the information to others. It is a one -way information sharing mechanism. The evolution only looks for the best solution. Compared with GA, all the particles tend to converge to the best solution quickly even in the local version in most cases.

4.1.1 Conventional Particle Swarm Optimization

PSO is initialized with a group of random particles (solutions) and then searches for optima by updating generations. In every iteration, each particle is updated by following two "best" values. The first one is the best solution (fitness) it has achieved so far. (The fitness value is also stored.) This value is called pbest. Another "best" value that is tracked by the particle swarm optimizer is the best value, obtained so far by any particle in the population. This best value is a global best and called gbest. When a particle takes part of the population as its topological neighbors, the best value is a local best and is called lbest. The variables pbest and gbest and their increments are both necessary. Conceptually pbest resembles autobiographical memory, as each individual remembers its own experience (though only one fact about it), and the velocity adjustment associarted with pbest has been called "simple nostalgia" in that the individual tends to return to the place thiat most satisfied it in the past. On the other hand, gbest is conceptually similar to publicized knowledge, or a group norm or standard, which individuals seek to attain. The updating formula for each particles velocity and position in conventional standard PSO is written as

$$v_{id}(t+1) = w \times v_{id}(t) + c_1 \times rand(.) \times (pbest - x_{id}(t)) + c_2 \times rand(.) \times (gbest - x_{id}(t)) \quad (4.1)$$

$$x_{id}(t+1) = x_{id}(t) + v_{id}(t+1) \quad (4.2)$$

where $i = 1, 2, \cdots, N$, N is the number of particles in the swarm, $d = 1, 2, \cdots, D$, and D is the dimension of solution space; $V_i = (V_{i1}, V_{i2}, ...V_{id})$, $Vid \in [-V_{max}, V_{max}]$ is the velocity vector of particle i which decides the particle's displacement in each iteration. Similarly, $X_i = (X_{i1}, X_{i2}, ...X_{id})$, $X_{id} \in [-X_{max}, X_{max}]$ is the position vector of particle i which is a potential solution in the solution space. the quality of the solution is measured

by a fitness function; w is the inertia weight which decreases linearly during a run; c_1, c_2 are both positive constants, called the acceleration factors which are generally set to 2; $rand(.)$ and $rand(.)$ are two independent random number distributed uniformly over the range [0, 1]; and P_g, P_i are the best solutions discovered so far by the group and itself respectively. The termination criterion for iterations is determined according to whether the presetting maximum generation or a designated value of the fitness is reached.

Particles' velocities on each dimension are clamped to a maximum velocity V_{max}. If the sum of accelerations would cause the velocity on that dimension to exceed V_{max}, which is a parameter specified by the user. Then the velocity on that dimension is limited to V_{max}.

Particle Swarm Optimization has certain parameters that require tuning to work well. This is also the case with other stochastic search algorithms e.g. for the tuning of GA mutation rates. However, changing a PSO parameter can have a proportionally large effect. Leaving out the constriction coefficient and not constricting particle speed by a maximum velocity will result in a buildup of speed, partially also because of the particle inertia. Particles with such speeds might explore the searchspace, but loose the ability to fine-tune a result. Constricting the particle speed too much might however cripple the searchspace exploration. Tuning the constriction coefficient or settings for maximum velocity is thus not a trivial task. These parameters also control the abilities of the PSO. The higher the inertia weight, the higher the particle speed. As with the constriction coefficient, the setting of the inertia must balance between having good exploration of the searchspace and good finetuning abilities. The setting for these parameters thus also determines the exploratory versus the fine-tuning abilities of the PSO. With the tradeoffs just described, the performance of the PSO is problem dependent. To improve the performance of PSO variants of PSO praposed by many sesearchers.

4.1.2 Variants of PSO

Since its invent, PSO has attracted an extensive attentions and interests of researchers from different scientific domains. Many researchers have worked on improving its performance in various ways, thereby deriving many interesting variants of PSO.

One of the variants introduces a parameter called inertia weight into the original PSO algorithms. A clever technique for creating a discrete binary version of the PSO introduced by Kennedy and Eberhart in 1997 uses the concept of velocity as a probability that a bit takes on one or zero. By analyzing the convergence behavior of the PSO, a

variant of the PSO with a constriction factor was introduced by Clerc and Kennedy , which guarantees the convergence and at the same time improves the convergence speed sharply. Parsopoulos and Vrahatis proposed a unified particle swarm optimizer (UPSO) which combined both the global version and local version together. A cooperative particle swarm optimizer was also proposed. Furthermore, El- Abd and Kamel proposed a Hierarchal Cooperative Particle Swarm Optimizer [17]. Very recently, a comprehensive learning particle swarm optimizer (CLPSO) [18] was proposed to improve the performance of the original PSO on multi-modal problems greatly by a novel learning strategy. Although there are numerous variants of the PSO, they need much time to finish evaluations of fitness function, and give similar results in the early parts of convergence [19].

In recent times another variant of PSO was praposed which is called as Clonal Particle Swarm optimization [20]. This algorithm employs clonal mechanism found in the natural immune system of creatures into the PSO. By cloning the best individual of every ten succeeding generations, the Clonal Particle Swarm optimization has better optimization solving capability and convergence performance than the conventional PSO and GA. So for our applicationm, to develop a fast motion estimation algorithm we have choosen this algorithm which will be discussed in the next section.

4.1.3 Clonal Particle Swarm optimization (CPSO)

Artificial immune system inspired by the principles of immune response is a simplified model of its natural coordinate for solving problems [21].The essence of the conventional PSO is to use particles with best known positions to guide the swarm or the population to converge to a single optimum in the search space. Clonal expansion is probably a good way to guide or direct the conventional PSO escaping from local optima whilst searching for the global optima efficiently [22]. According to the clonal expansion process in natural immune system the clonal operator is to clone one particle as N same particles in the solution space according to its fitness function at first, then generate N new particles via clonal mutation, which are related to the concentration mechanisms used for antigens and antibodies in Natural immune system. The clonal operator is used to copy one point as N same points according to its fitness function, and then generate N new particles by undergoing mutation and selection processes.

CPSO algorithm can be summarized as follows.

1. Initialization. Assume $c_1 = 2$, $c_2 = 2$, and w be from 0.9 to 0.4 linearly.

2. The state evolution of particles is iteratively updated according to Equations 4.1 and 4.2.

3. Memory the global best-fit particle P_{gB} of each generation, P_{gB}, as a mother particle of the clonal operator in Step 4.

4. Clone the global best particle. The clonal operator is used to copy one point as N same points according to its fitness function

5. Mutate some of the cloned particles depending on the probability of mutation P_m may be 0.2 or 0.3 for better results. Mutation process is implemented by using some random disturbances such as Gaussian noise.

$$P_{gB} = (1 - \mu)P_{gBC} \tag{4.3}$$

6. So for next generation mutated particles and cloned particles will be combined and updated using Equations 4.1 and 3.2, until the stop criteria met repeat above steps.

7. The algorithm can be terminated by some common stop criteria such as a given maximum number of generations or a presetting accuracy of the solution. In our experiments, we adopt the former stop criterion, i.e. a maximum number of generations.

4.2 Block Matching Algorithm Based on CPSO

With the increasing popularity of optimization algorithms digital television, Internet streaming video and video conferencing, video compression has became an essential component of broadcast and entertainment media. Motion estimation and compensation techniques, which can eliminate temporal redundancy between adjacent frames effectively, have been widely applied to popular video compression coding standards such as MPEG-2, MPEG-4, H.263 and H.264 . Block-matching algorithm (BMA) is the most popular technique adopted for motion estimation due to its simplicity for implementation. The computational complexity of motion estimation is 70 percent to 90 percent in video coding and processing systems.

Generally, the most straightforward BMA called full search (FS) simply compares the given macro block (MB) in the anchor frame with all candidate MBs in the target frame exhaustively within a predefined search region. This is not fit for real-time applications

because of its unacceptable computational cost. To speed up the search, various fast algorithms for block matching which reduce the number of search candidates have been developed. Well known examples are three-step search (TSS), Four Step Search (4SS), block-based gradient descent search (BBGDS) and diamond search (DS) have been proposed to reduce computational efforts, based on fixed search pattern and greedy search method.

From optimization theory, these traditional fast BMAs are based upon the following two assumptions: the error function has only one global optimum solution and the matching error decreases monotonically as the search point moves towards it . However, since the two assumptions can hardly be satisfied in most real-world videos, above mentioned fast BMAs are liable to get trapped in local minima resulting in degradation on video quality to some extent after decoding. This inherent shortcoming is mainly because the local search based on fix pattern cannot sufficiently explore solution space and the greedy search lacks capability of searching solutions on hypo-optima paths.

Over the last few years, promising computational intelligence methods, called evolutionary computing techniques such as genetic algorithm (GA) [13], particle swarm optimization (PSO) [23] have been successfully applied to solve motion estimation problem. Such approaches are suitable for achieving global optimal solution, which traditional fast BMAs are not able to obtain easily. The GA needs to set some key parameters such as population size, probability of mutation, probability of crossover, etc. If these parameters are not prefixed properly, efficiency of GA becomes lower and also it is time consuming process.

Particle swarm optimization (PSO) was originally proposed by Kennedy and Eberhart in 1995. Unlike genetic algorithm, PSO does not need genetic operators such as crossover and mutation. Thus it has advantages of easy implementation, fewer parameters adjustment, strong capability to escape from local optima and rapid convergence characteristic. In the paper, a new fast motion estimation method based on Clonal particle swarm optimization is proposed. This algorithm employs clonal mechanism found in the natural immune system of creatures into the PSO. By cloning the best individual of every ten succeeding generations, the CPSO has better optimization solving capability and convergence performance than the conventional PSO and GA.

Zero Motion Prejudgement

Zero Motion Prejudgment (ZMP) is the procedure to find the static macroblocks which contains zero motion [25]. In real world video sequences more than 70% of the MBs are

static which do not need the remaining search . So, significant reduction of computation is possible if we predict the static macroblocks by ZMP procedure before starting motion estimation procedure and the remaining search will be faster and saves memory. We first calculate the matching error (SAD) between the MB in the current frame and the MB at the same location in the reference frame and then compare it to a predetermined threshold, saying Δ. If the matching error is smaller than Δ we consider this MB static which do not need any further motion estimation, and return a [0,0] as its motion vector(MV).

Performance Evaluation Criterion

As widely adopted, we measure the amount of computation and the quality of compensated video sequence by Computation criterion and Peak Signal-to-Noise Ratio (PSNR). Computation is defined as the average number of the error function evaluations per MV generation. Due to the minimum computational cost, we choose Summed Absolute Difference (SAD) as the error function which is defined as follows:

$$SAD(i,j) = \frac{1}{N^2} \sum_{n_1=0}^{N-1} \sum_{n_2=0}^{N-1} [s(n_1, n_2, k) - s(n_1 + i, n_2 + j, k - l)] \tag{4.4}$$

Where the size of a MB is $N \times N$, I_k and I_{k-1} are current frame and reference frame.

The motion estimate quality between the original I_{ogn} and the compensated video sequences I_{cmp} is measured in PSNR which is defined as the ratio between the maximum possible power of a signal and the power of corrupting noise that affects the fidelity of its representation.

$$PSNR = 10 \log_{10} \left[\frac{I_{max}^2}{MSE} \right] \tag{4.5}$$

$$MSE(i,j) = \frac{1}{N^2} \sum_{k=1}^{K} \sum_{n_1=0}^{N-1} \sum_{n_2=0}^{N-1} [I_{ogn}(i,j,k) - I_{cmp}(i,j,k)]^2 \tag{4.6}$$

Where K is the number of frames in the video sequence, I_{max} is the maximum possible pixel value of an image. When the pixels are represented using 8 bits per sample, I_{max} is 255.

Population Size and Initial Position

Block-based matching algorithms consider each frame in the video sequence formed by many non overlapping small regions, called MB which are often square-shaped and with

fixed-size (16×16 or 8×8). Given a MB B_m in the anchor frame, the motion estimation problem is to determine a corresponding matching MB in the target frame such that the matching error between these two blocks is minimized. Then, a motion vector is computed by subtracting the coordinates of the MB in the anchor frame from that of the matching MB in the target frame. Instead of sending the entire frame pixel-by-pixel, a set of motion vectors is transmitted through the channel which greatly reduces the amount of transmission.

Center-biased feature denotes that match point may exist within a small zone around macro block center. Thus we select individuals as shown in Figure 4.1 which distribute around center of macro block. The white dot is center of macro block and dark dots are individuals. Usually we let individuals randomly distribute in search window. However, if individual positions are around optimization position, it can speed up population convergence.

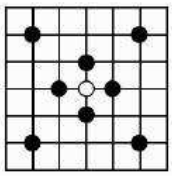

Figure 4.1: Particles intial positions

We put four particles in a cross shape with size one (size refers to the distance between any vertex point and the center-point) in the adjacent MBs and four particles in a cross shape with size two, and then rotate it by angle $\pi/2$. With two cross shapes in different sizes, we try to balance the global exploration and local refined search in order for broader searching space as well as higher matching accuracy. Moreover, we distribute particles equally in all directions (8 particles in 8 directions) with a view to find the matching MB in each direction with equal possibility.

Stopping Criterion

Generally, there are two widely adopted stopping criteria. One is fixed-iteration, that is, given a certain iteration time, saying N, the search stops after N iterations. The other is specified-threshold. For minimization problems, we specify a very small threshold, and if

the change of gbest during t times of iteration is smaller than the threshold, we consider the group best value very near to the global optimum, thus the matching procedure stops. Due to the center-biased characteristics of real-world motion fields, we adopt the fixed-iteration method in this paper for reducing the computational cost.

4.2.1 Flowchart of CPSO BMA

The block matching algorithm based on CPSO for ME is summarized as sown in figure 4.2

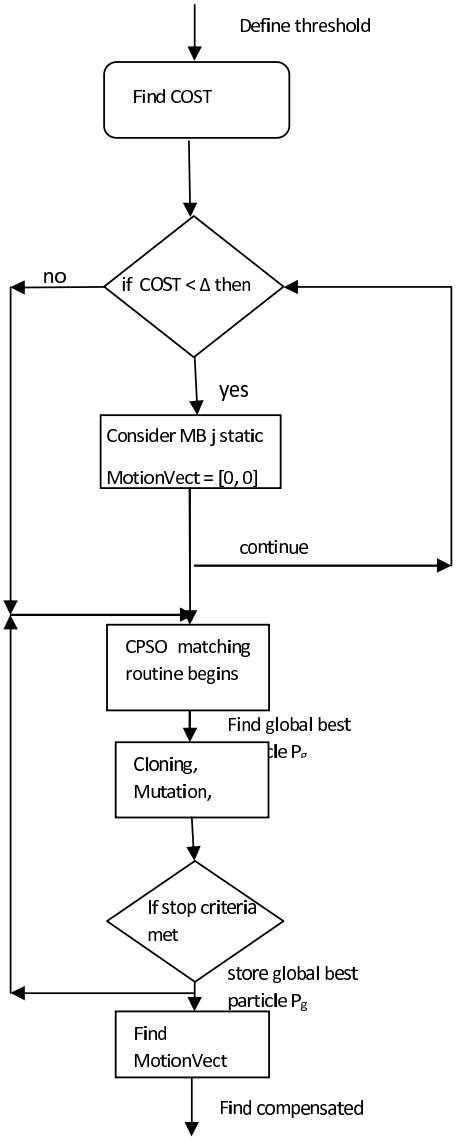

Figure 4.2: Flowchart of CPSO-BMA

4.2.2 Experiments and Simulation Results

CPSO Parameters

In this thesis, to balance between computational cost and compensated video quality, we adopt the inertia weight as in which is widely considered as the defacto PSO standard. A large inertia weight facilitates a global search while a small inertia weight facilitates a local search. By changing the inertia weight dynamically, we use the fixed-iteration stopping criterion with max 5 iterations. The max velocity is set to 5. The inertia weight w decreases linearly from 0.9 to 0.4 during a CPSO run and two Acceleration coefficients C_1 and C_2 can also control how far a particle will move in a single iteration [24]. Typically, both of them are initialized as 2.0 . μ is an Gaussian random variable with zero mean and unity variance.

Motion Estimation Parameters

We divide a whole image frame into 16×16 MBs in the simulation that is $N = 16$ and the size of search window was set as 15×15. Threshold for each test video sequence correspondingly based on data obtained in experiments shown in Table 4.1. This threshold value is not fixed, may vary depending on your video sequences. We do not restrict the range of candidate matching MBs rigidly by a search window P. Instead, through the fixed-iteration and the setting of max velocity, particles search for the matching MB in an area more flexible and adaptable.

Results and Analysis

The performance of the proposed CPSO based block matching algorithm is evaluated in terms of computation and average peak signal-to-noise ratio (PSNR) per frame of the reconstructed video sequence is computed for quality measurement. Computation is defined as the average number of the error function evaluations per MV generation.For Zero motion Prejudgement threshold value for each test video sequence correspondingly based on data obtained in experiments shown in Table 4.1.

For comparison, the performance of DS, GA, PSO, and our proposed algorithm are compared in terms of PSNR and computation and documented in Table 4.2 and Table 4.3 for 5 different video sequences. Figure 4.4 shows the computation comparison between DS, PSO and CPSO. Figure 4.3 shows the simulation results of PSNR performances of 100 frames. Figure 4.8 shows the original and decoded frames of NEWS sequence using different search algorithms. Figure 4.5 shows the shows the simulation results of PSNR

performances of DS, GA, PSO, CPSO. Search range P=7 means that the search will be performed within a square region of [-p, +p] around the position of the current block. In our simulation experiments, the block size is fixed at 16×16. To make a consistent comparison, block matching is conducted within a 15×15 search window.

In order to evaluate search performance in a wide range of motion conditions, five video sequences in QCIF were tested in this paper are Akyio, Container, mother & daughter, News, Silent from 1st frame to 100th frame. From the results obtained, CPSO shows significant computational reductions while acceptable drops in peak signal-to-noise ratio (PSNR). Notably, in sequence Akiyo, Container and Mother & Daughter, our method achieves very close PSNR performance (max difference 0.5dB in Mother & Daughter) with 6.8251, 7.5869 and 5.7950 times of computation reductions compared to DS respectively and 1.3062, 1.1518, and 1.3492 times of computation reductions compared to PSO. In sequence Silent and News our method has achieved around 1 dB less than the Diamond Search (DS) but it is more than using PSO. These silent and news contains high and moderate motion contents, where our proposed method giving slightly less PSNR but in an acceptable degree. But, in those sequences, compared to DS, CPSO consumes over 4.5 times less of computation to that of DS and 1.5 times less than that of PSO. In all of the above sequences PSNR of GA is atleast 2 dB lesser than the standard DS algorithm and also it is faster compared to DS but it is slower than the CPSO algorithm. Thus we can say that our CPSO algorithm giving better results in terms of quality and speed than other algoritms.

Sequences	Format	Threshold
Akiyo	QCIF	350
Container	QCIF	300
Mother & daughter	QCIF	250
News	QCIF	250
Silent	QCIF	200

Table 4.1: Assumed Threshold values

Referred to, a PSNR higher than 40dB typically indicates an excellent image, between 30-40dB usually means a good images (i.e., the distortion is visible but acceptable); between 20-30dB PSNR is quite poor; and finally, a PSNR lower than 20dB is unacceptable. For all five sequences tested, CPSO algorithm achieves PSNR higher than 30dB in most of the frames, thus the PSNR droppings are in an acceptable degree.

Sequences	GA to DS	PSO to DS	CPSO to DS
Akiyo	3.5597	5.2251	6.8251
Container	4.3469	6.5869	7.5869
Mother & daughter	3.5288	4.2950	5.7950
News	4.2781	5.6956	6.4956
Silent	3.5234	4.5338	6.2338

Table 4.2: Computational gain to DS ,PSO

Sequences	DS	GA	PSO	CPSO
Akiyo	40.2463	38.1706	39.0133	40.1133
Container	37.8131	36.2360	36.2071	37.6646
Mother & daughter	35.7113	33.2731	34.1635	35.2328
News	32.2179	29.6056	30.4562	31.8493
Silent	32.3068	30.2544	30.8542	31.6207

Table 4.3: Average PSNR(dB) performances of DS, GA, PSO, CPSO

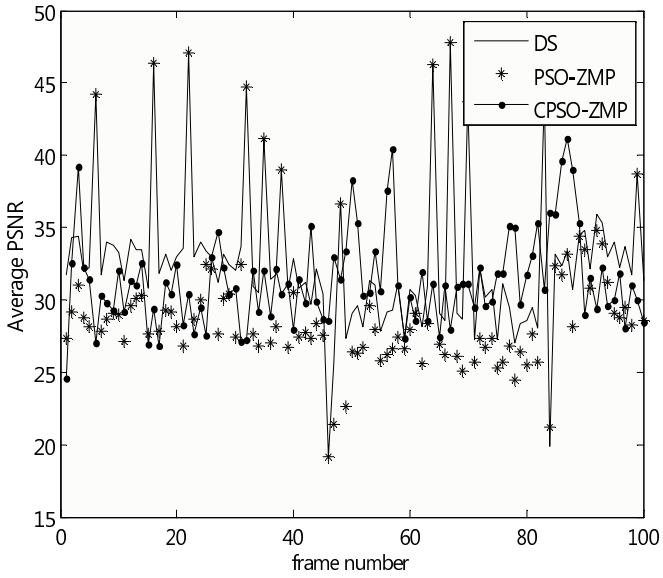

Figure 4.3: PSNR comparison of DS, PSO, CPSO

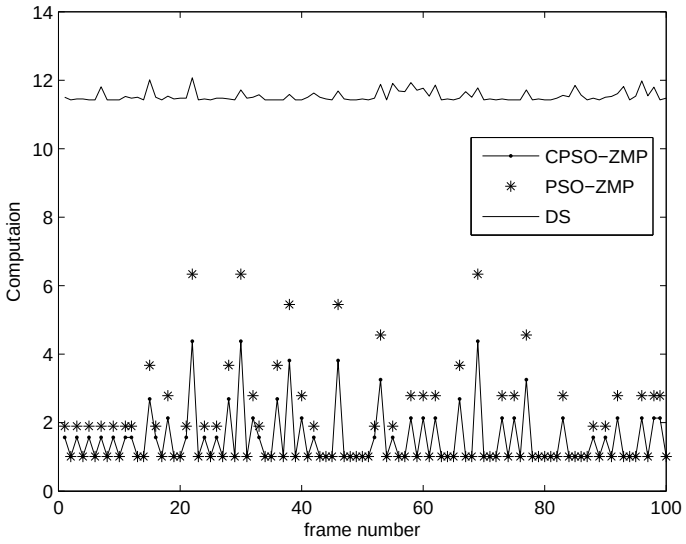

Figure 4.4: computational comparison of DS, PSO, CPSO

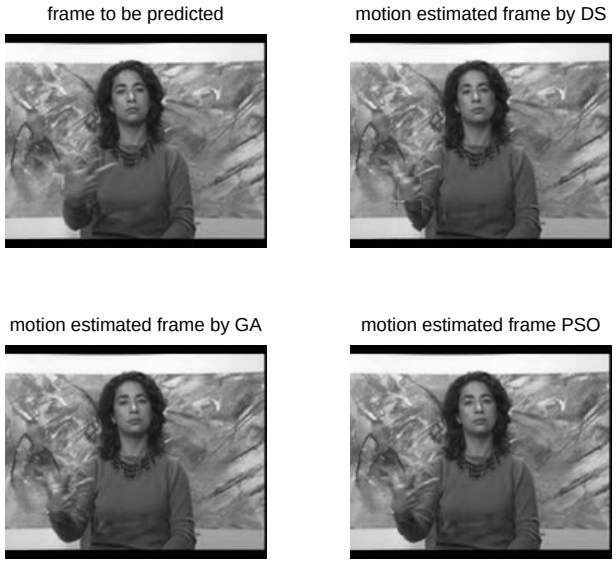

Figure 4.5: original and estimatted frames of silent sequence

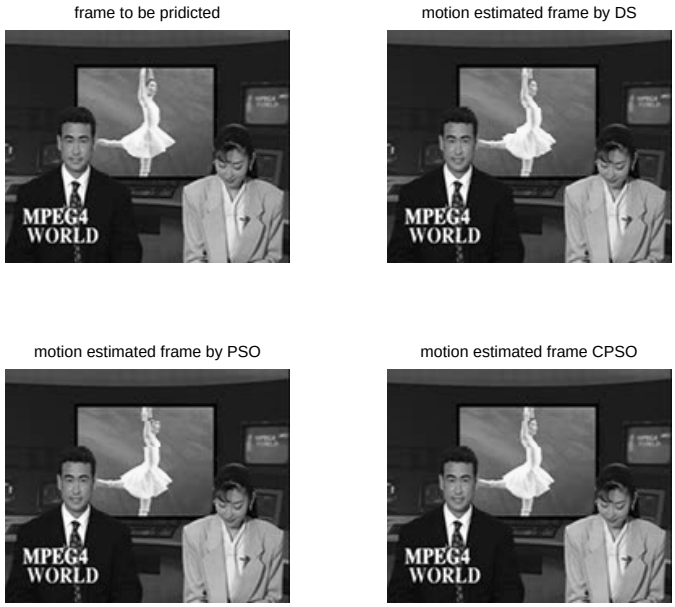

Figure 4.6: original and estimated frames of NEWS sequence

4.3 Conclusion

A fast block-matching algorithm (BMA) named fast block-matching algorithm for video motion estimation based on clonal particle swarm optimization (CPSO) is proposed. Applying immunity-clonal strategies, can effectively overcome the inherent drawback of being liable to get trapped in local optima in traditional fast BMAs such as TSS and DS. By cloning the best individual of every ten succeeding generations, the CPSO has better optimization solving capability and convergence performance than the conventional Standard PSO. In addition, skipping those static macroblocks from processing can reduce the computational cost of the algorithm. Simulation results show that the CPSO-BMA proposed requires less amount of computation and achieves PSNR in an acceptable degree of drop when compared to DS in some highly dense motion sequences. Moreover CPSO just consumes a few lines of codes due to its simplicity which makes the CPSO algorithm attractive for hardware implementation. In the future, variants of PSO might be applied to strengthen the global searching ability and to speed up the search and to avoid being trapped in local minima.

CHAPTER 5

BIDIRECTIONAL MOTION ESTIMATION

5.1 MPEG Frame Encoding

MPEG provides for up to three types of frames called the I, P and B frames. The intra-frame, or I frame, serves as a reference for predicting subsequent frames. I frames, which occur on an average of one out of every ten to fifteen frames, only contains information presented within itself. P Frames are predicted from information presented in the nearest preceding I or P frame. The bi-directional B frames are coded using prediction data from the nearest preceding I or P frame and the nearest following I or P frame. Not all MPEG 2 systems use B frames. Although a more efficient level of compression is achieved by 'B' frames, compatible receiver/decoders must have an additional memory buffer, which increases the cost of the decoder.

I Frames

An I-frame is encoded as a single image, with no reference to any past or future frames. The encoding scheme used is similar to JPEG compression. Each 8×8 block is encoded independently with one exception explained below. The block is first transformed from the spatial domain into a frequency domain using the DCT (Discrete Cosine Transform), which separates the signal into independent frequency bands. Most frequency information is in the upper left corner of the resulting 8×8 block. After this, the data is quantized. Quantization can be thought of as ignoring lower-order bits (though this process is slightly more complicated).

P Frames

Starting with an intra, or I frame, the encoder can forward predict a future frame. This is commonly referred to as a P frame, and it may also be predicted from other P frames, although only in a forward time manner. As an example, consider a group of pictures that lasts for 6 frames. In this case, the frame ordering is given as $I, P, P, P, P, P, I, P, P, P, P, \cdots$ Each P frame in this sequence is predicted from the frame immediately preceding it, whether it is an I frame or a P frame. I frames are coded spatially with no reference to any other frame in the sequence.

B Frames

The encoder also has the option of using forward/backward interpolated prediction. These frames are commonly referred to as bi-directional interpolated prediction frames, or B frames for short. As an example of the usage of I, P, and B frames, consider a group of pictures that lasts for 6 frames, and is given as $I, B, P, B, P, B, I, B, P, B, P, B, \cdots$ As in the previous I & P only example, I frames are coded spatially only and the P frames are forward predicted based on previous I and P frames. The B frames however, are coded based on a forward prediction from a previous I or P frame, as well as a backward prediction from a succeeding I or P frame. As such, the example sequence is processed by the encoder such that the first B frame is predicted from the first I frame and first P frame, the second B frame is predicted from the second and third P frames, and the third B frame is predicted from the third P frame and the first I frame of the next group of pictures. From this example, it can be seen that backward prediction requires that the future frames that are to be used for backward prediction be encoded and transmitted first, out of order. There is no defined limit to the number of consecutive B frames that may be used in a group of pictures, and of course the optimal number is application dependent.

The main advantage of the usage of B frames is coding efficiency. In most cases, B frames will result in less bits being coded overall. Quality can also be improved in the case of moving objects that reveal hidden areas within a video sequence. Backward prediction in this case allows the encoder to make more intelligent decisions on how to encode the video within these areas. Also, since B frames are not used to predict future frames, errors generated will not be propagated further within the sequence.

One disadvantage is that the frame reconstruction memory buffers within the encoder and decoder must be doubled in size to accommodate the 2 anchor frames. This is almost

never an issue for the relatively expensive encoder, and in these days of inexpensive DRAM it has become much less of an issue for the decoder as well. Another disadvantage is that there will necessarily be a delay throughout the system as the frames are delivered out of order . Most one-way systems can tolerate these delays, as they are more objectionable in applications such as video conferencing systems.

In the MPEG2, MPEG4 schemes, the order of picture coding differs to the actual order of pictures in the video stream. Namely, a group of pictures, shown in Fig. 5.1 in the MPEG2 video coder is actually encoded in the order shown in Fig. 5.2.

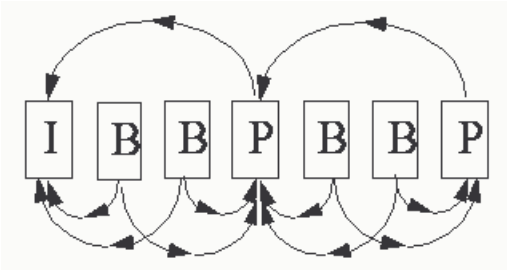

Figure 5.1: An input video stream

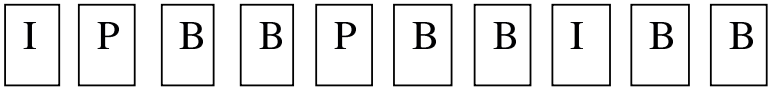

Figure 5.2: Encoding order

For instance, if a sequence is represented as $IBBPBBPBBI$, then the encoding order becomes $IPBBPBBIBB$. The bi-directional frame prediction also increases the computational complexity, since the motion estimation must be performed in both the past and future frames.

5.2 Bidirectional Frame Prediction

Forward frame prediction (P frame coding) uses a previous frame as the reference frame in order to predict a block in the current frame. Even though P frame coding does not cause any coding delay, it produces poor prediction when the frame contains hidden regions or newly entered objects. Motivated by the MPEG video standards, bi-directional frame prediction [26] (B frame coding) uses a past frame and a future frame as two reference frames for prediction. B frame coding provides a number of significant advantages, which are:

1. Hidden areas can be predicted using the future frame;

2. Less bit budget should be spent to achieve the same quality of motion compensation;

3. It has a better noise suppression quality than the forward frame prediction.

Despite the advantages, B frame coding introduces an extra delay in the encoding process, which has become a problem in applications such as teleconferencing.

Bidirectional frame prediction consists of two stages of prediction, one is forward frame prediction, backward frame prediction. In this scheme, a target macroblock of 16×16 pixels in a bidirectionally predicted frame, (or B-frame) is predicted from two reference frames which are coded frames located before and after the B-frame on the time axis. We shall refer to these reference frames as the previous reference frame and next reference frame respectively [27]. The bidirectional prediction scheme generally employed in MPEG-4 scheme is as shown in figure 5.3.

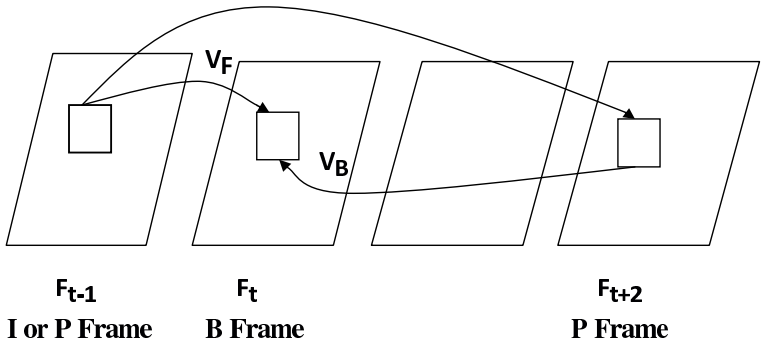

Figure 5.3: Bidirectional prediction scheme

For an input macroblock to be interpolatively predicted, one pair of forward and backward motion vectors (V_F, V_B) is generated by the encoder, coded and transmitted to the decoder. These forward and backward motion vectors try to model the motion of the objects in the scene from the previous reference frame to the current B-frame to the next reference frame. A target pixel in the macroblock is predicted by averaging the pair of pixels, or half-pel interpolated pixels, in the previous and next reference frames at coordinates corresponding to the target pixel displaced by the forward motion vector and the backward motion vector respectively. The prediction error is then coded and transmitted.

With a given number of bits for coding the prediction residual of the B-frame, the resulting distortion in reproducing this frame generally decreases with the magnitude of the prediction error. Conversely, if the B-frames are to be coded so that a certain level of

distortion is not exceeded, the number of bits required decreases with the prediction error magnitude [28]. Therefore, given the interpolative prediction scheme described above, the performance of the system relies on the effectiveness of the encoder in generating a pair of motion vectors that accurately predict the target macroblock from the reference frames. we praposed a novel effective technique for this purpose.

5.3 Bidirectional Motion Estimation Based on PSO

Generally, the most straightforward BMA called full search (FS) simply compares the given macro block (MB) in the anchor frame with all candidate MBs in the target frame exhaustively within a predefined search region. This is not fit for real-time applications because of its unacceptable computational cost. To speed up the search, various fast algorithms for block matching which reduce the number of search candidates have been developed. Well known examples are three-step search (TSS), Four Step Search (4SS), block-based gradient descent search (BBGDS) and diamond search (DS) have been proposed to reduce computational efforts, based on fixed search pattern and greedy search method.

Over the last few years, promising computational intelligence methods, called evolutionary computing techniques such as genetic algorithm (GA), particle swarm optimization (PSO) have been successfully applied to solve motion estimation problem. Such approaches are suitable for achieving global optimal solution, which traditional fast BMAs are not able to obtain easily. The GA needs to set some key parameters such as population size, probability of mutation, probability of crossover, etc. If these parameters are not prefixed properly, efficiency of GA becomes lower and also it is time consuming process. So here we are adopted PSO procedure inoder to do the bidirectional motion estimation.

Motivated by the potential improvement attainable by switching from independent search to joint search for the motion vector estimaion, and by the practical requirement of avoiding an excessively high search complexity, the proposed method is an iterative technique to jointly optimize the motion vectors by using particle swarm optimization.

Bidirectional ME forms a major computation bottleneck in video processing applications such as the detection of noise in image sequences, interpolation/ prediction of missing data in image sequences and deinterlacing of image sequences.

In general Bidirectional motion estimation is performed by following the these steps:

1. Finding forward motion vector V_F by taking past frame as reference frame

2. Finding backward motion vector V_b by taking future frame as reference

3. Find the matching error for both methods and find the average motion vector position and its matching error

4. Compare all the three errors take the motion vector which is giving the least error.

Generally, the objective of a motion estimation algorithm is to minimize a cost function that measures the interpolation error in the macroblock. Examples are the popular sum of absolute difference (SAD).The process of finding robust motion vectors using minimal computations is a heavily researched area, and various fast algorithms have been proposed.

In order to reduce the overall processing time in some video processing applications, the complexity of the bidirectional ME algorithm is looked into [29]. The proposed a novel technique to do the bidirectional motion estimation based on PSO. From the chapter 4 we can say that Block matching algorithm based on PSO giving a good results interms of quality and less number of computations. Our idea is not to find the forward and backward motion vectors individually but to find the minimum matching macroblock at each time when PSO is finding for a minimum matching block. so it will reduce the number computaions involved in finding out the minimum matching point.

5.3.1 Algorithm Steps

The praposed algorithm can be summarized in the following algorithm steps

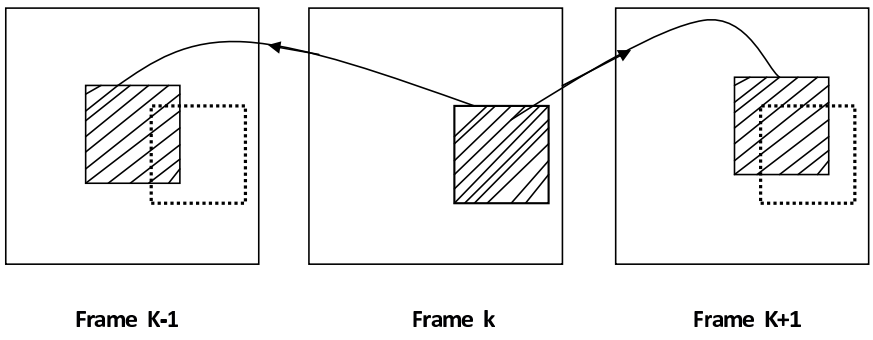

Figure 5.4: Bidirectional search for best motion vector

1. Initialization. Assume $c_1 = 2$, $c_2 = 2$, and w be from 0.9 to 0.4 linearly.

2. perform block matching algorithm based on PSO.

3. Each time find the minimum matching error (SAD) point in the past frame and the current frame as shown in fig. 5.3

4. Take the minimum out of both matching error (SAD), this one we are taking as the Cost function of our algorithm.

5. For each generation we are getting the minimum error point in the two reference frame at a time.

6. Until it reaches the stopping criteria it will continue the above steps.

7. Save the final motion vector point for motion compensation.

Since we are performing the Block matching procedure at a time in two reference frames , our objective funtion is to minimize the mean of the two matching errors between two frames.

$$costfun = min(SAD_P, SAD_F) \tag{5.1}$$

where SAD_P and $SAD_F$are the sum of absolute difference of the past frame and future frames.

Zero Motion Prejudgement

In real world video sequences more than 70% of the MBs are static which do not need the remaining search [30] . So, significant reduction of computation is possible if we predict the static macroblocks by ZMP procedure before starting motion estimation procedure and the remaining search will be faster and saves memory. We first calculate the matching error (SAD) between the MB in the current frame and the MB at the same location in the reference frame and then compare it to a predetermined threshold, saying Δ. If the matching error is smaller than Δ we consider this MB static which do not need any further motion estimation, and return a [0,0] as its motion vector(MV).

Population Size and Initial Position

Block-based matching algorithms consider each frame in the video sequence formed by many non overlapping small regions, called MB which are often square-shaped and with fixed-size (16×16 or 8×8). Given a MB B_m in the anchor frame, the motion estimation problem is to determine a corresponding matching MB in the target frame such that

the matching error between these two blocks is minimized. Then, a motion vector is computed by subtracting the coordinates of the MB in the anchor frame from that of the matching MB in the target frame. Instead of sending the entire frame pixel-by-pixel, a set of motion vectors is transmitted through the channel which greatly reduces the amount of transmission.

Performance Evaluation Criteria

As widely adopted, we measure the amount of computation in terms of Average Search points/frame and the quality of compensated video sequence by Computation criterion and Peak Signal-to-Noise Ratio (PSNR) and Average mean square prediction error (AMSPE) for each frame. Due to the minimum computational cost, we choose Summed Absolute Difference (SAD) as the error function which is given by

$$SAD(i,j) = \frac{1}{N^2} \sum_{n_1=0}^{N-1} \sum_{n_2=0}^{N-1} [s(n_1, n_2, k) - s(n_1 + i, n_2 + j, k - l)] \tag{5.2}$$

Where the size of a MB is $N \times N$, I_k and I_{k-1} are current frame and reference frame.

The motion estimate quality between the original I_{ogn} and the compensated video sequences I_{cmp} is measured in PSNR which is defined as

$$PSNR = 10 \log_{10} \left[\frac{I_{max}^2}{MSPE} \right] \tag{5.3}$$

The evaluation is based on the AMSPE and the bit rate. The results reported in the table are based on the average mean square prediction error (AMSPE), which is given by

$$AMSPE = \frac{1}{K} \sum_{k=1}^{K} MSPE(K) \tag{5.4}$$

K is the total number of the frames tested and MSPE (mean square prediction error) is the average energy per pixel in the residual image as given by

$$MSPE(i,j) = \frac{1}{N^2} \sum_{n_1=0}^{N-1} \sum_{n_2=0}^{N-1} [s(n_1, n_2, k) - s(n_1 + i, n_2 + j, k - l)]^2 \tag{5.5}$$

We consider a block of pixels of size $N \times N$ in the reference frame, at a displacement of, where i and j are integers with respect to the candidate block position. The MSE is computed for each displacement position (i, j),within a specified search range in the reference.

The praposed algorithm fastness is measured interms of Average Search points/frame.

Stopping Criterion

Generally, there are two widely adopted stopping criteria. One is fixed-iteration, that is, given a certain iteration time, saying N, the search stops after N iterations. The other is specified-threshold. For minimization problems, we specify a very small threshold, and if the change of gbest during t times of iteration is smaller than the threshold, we consider the group best value very near to the global optimum, thus the matching procedure stops. Due to the center-biased characteristics of real-world motion fields, we adopt the fixed-iteration method in this paper for reducing the computational cost.

5.3.2 Experiments and Simulation Results

The performance of the proposed Bidirectional motion estimation block matching algorithm based on Particle Swarm Optimization is evaluated in terms of Average mean square prediction error (AMSPE) , Average Search points/frame and peak signal-to-noise ratio (PSNR) per frame of the reconstructed video sequence is computed for quality measurement. Zero motion Prejudgement procedure to be done before starting of block matching procedure. significant reduction of computation is possible if we predict the static macroblocks before starting motion estimation procedure and the remaining search will be faster and saves memory. We first calculate the matching error (SAD) between the MB in the current frame and the MB at the same location in the reference frame and then compare it to a predetermined threshold, saying Δ. If the matching error is smaller than Δ, we consider this MB static which do not need any further motion estimation, and return a [0,0] as its motion vector(MV). Threshold value for each test video sequence correspondingly based on data obtained in experiments shown in Table 3.1.

Experiments conducted over four video sequences demonstrate that the proposed technique is superior to the existing bi-directional motion compensation methods as are shown in tables for different video sequences schemes. The prediction error is averaged over 100 frames. The motion estimation for our bi-directional prediction coding is conducted between a B-frame and a past I or P and a future frame P. since the joint search at all the locations within the search windows in the previous and the future frame is computationally very expensive. so the praposed technique will do the motion vector search in both frame at a time. When finding out the matching error between frames each time we will find out which matching error is minimum, the one which is giving minimum error that

one is considered as the local best position for each particle position. After the limited number of generations we will get the global minimum point in both of the frames. so we reduced a lot of number of error function calulations.Due to the minimum computational cost, we choose Summed Absolute Difference (SAD) as the error function.

As widely adopted, we measure the amount of computation in terms of Average Search points/frame and the quality of compensated video sequence by Computation criterion and Peak Signal-to-Noise Ratio (PSNR) and Average mean square prediction error (AMSPE) for each frame. The pattren of the group of picture (GOP) $IBBPBBPBBI$ The evaluation is based on the AMSPE . The results reported in the table are based on the average mean square prediction error (AMSPE). The AMSPE values for all the four video sequences namely News, Mother & Daughter, Akyio, Silent are noted in a tabular form 5.1 against the Joint search algorithm based on Diamond search and Particle Swarm Optimization. To find the fastness of the algorithm we find out Average Search points/frame for all the four video sequences and are reported in the tablular form 5.2 against the Joint search algorithm based on Diamond search and Particle Swarm Optimization. As it can be seen that from the tables tha proposed bidirectional algorithm is giving less predicton error and the number of search point per each frame are less.

For comparison we find out the quality between the reconstructed video sequence is computed by PSNR of all bidirectional frames and shown in a figure 5.5 for NEWS video sequence and figure 5.6 shows the PSNR (dB) values of SILENT video sequences against the Joint search algorithm based on Diamond search and Particle Swarm Optimization. As it can be seen from these two tables and the PSNR (dB) values, the praposed method employing the bi-directional motion vectors requires less number of bits for a fixed AMSPE or produces a better prediction error for a given bit budget. These experimental results are obtained for the video sequences, It is quite clear that the proposed method can significantly reduces the computational complexity involved in the Bidirectional frame prediction and also least prediction error in all video sequences.

sequence	BI-DS	BI-PSO	NEW BI-PSO
News	125.1636	148.0847	80.3798
Mother & daughter	32.4141	47.3537	29.5195
Akiyo	11.8697	20.0268	16.3192
Silent	76.9223	98.5123	51.4913

Table 5.1: Average mean square prediction error (AMSPE)

Sequences	BI-DS	BI-PSO	NEW BI-PSO
Akiyo	21.6463	17.115	10.9515
Mother & daughter	21.3988	15.2325	11.3266
News	19.7825	11.9449	9.9448
Silent	19.905	9.7298	7.5776

Table 5.2: Average Search points/frame

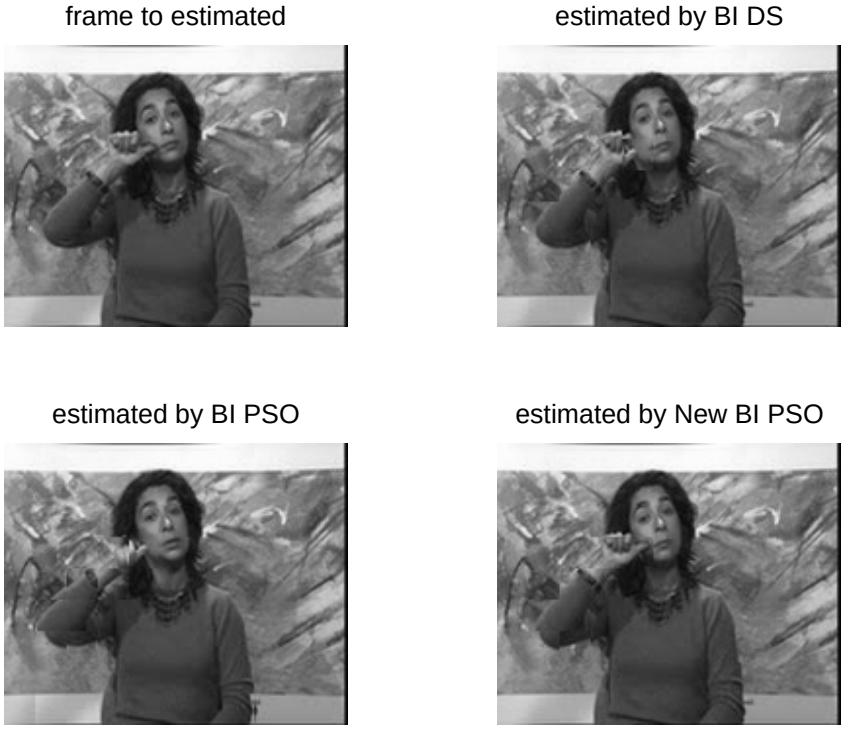

Figure 5.5: Original and estimated frames using different methods

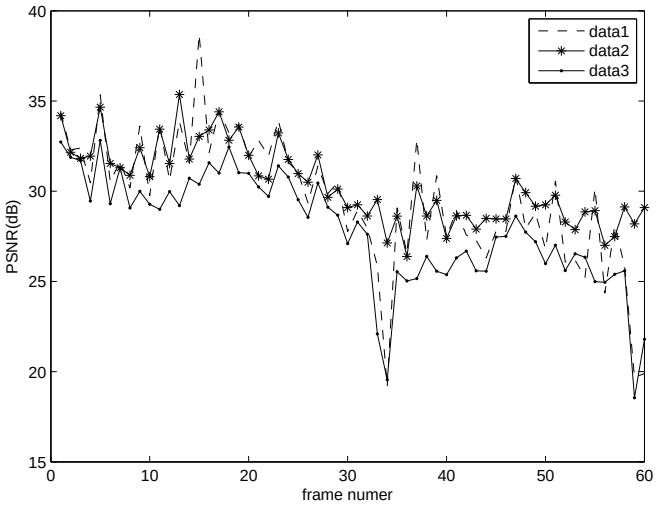

Figure 5.6: NEWS video PSNR(dB) comparasion of B-frame

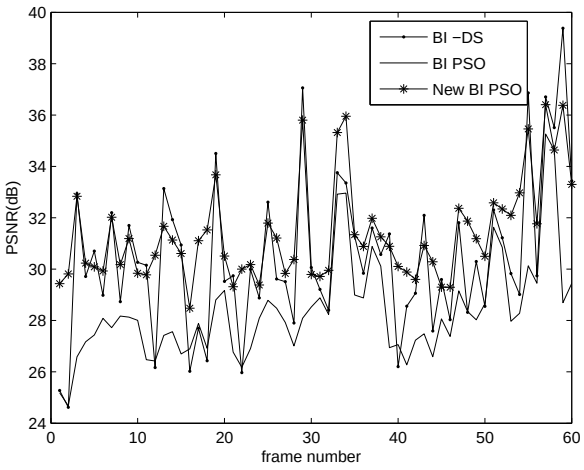

Figure 5.7: SILENT video PSNR(dB) comparasion of B-frame

5.4 Conclusion

Bidirectional ME forms a major computation bottleneck in video processing applications such as the detection of noise in image sequences, interpolation/ prediction of missing data in image sequences and deinterlacing of image sequences. The proposed Novel Bidirectional Motion Estimation algorithm that effectively reduces the number of operations in BM motion estimation without sacrificing the quality of the results. Praposed bidirectional algorithm is giving less predicton error and the number of search point per each frame are less. In addition, skipping those static macroblocks from processing can reduce the computational cost of the algorithm. Simulation results shows that the proposed algorithms gives the close match of PSNR values when compared to joint search algorithm with DS and an acceptable degree of drop when compared to joint search DS in some highly dense motion sequences. Moreover PSO just consumes a few lines of codes due to its simplicity which makes the PSO algorithm attractive for hardware implementation. In the future, variants of PSO might be applied to strengthen the global searching ability and to speed up the search and to avoid being trapped in local minima.

CHAPTER 6

CONCLUSION AND FUTURE WORK

6.1 Conclusion

Because of the Internet is more and more universal and the technology of multimedia has been progressed, the communication of the image data is a part in life. In order to employ effect in a limit transmission bandwidth, to convey the most, high quality user information .It is necessary to have more advanced compression method in image and data. Motion Estimation (ME) and compensation techniques, which can eliminate temporal redundancy between adjacent frames effectively, have been widely applied to popular video compression coding standards such as MPEG- 2, MPEG-4. Full search Motion Estimation algorithm is not fit for real-time applications because of its unacceptable computational cost. Bidirectional ME forms a major computation bottleneck in video processing applications such as the detection of noise in image sequences, interpolation/ prediction of missing data in image sequences and deinterlacing of image sequences. The computational complexity of motion estimation is 70 percent to 90 percent in video coding and processing systems, so the idea is to find a fast Motion Estimation algorithm to improve the operation.

In this thesis, To speed up the search, a novel fast block matching algorithm based different Evolutionary Computing Techniques is praposed. Traditional fast block matching algorithms are easily traped into the local minima resulting in degradation on video quality to some extent after decoding. Since these Evolutionary Computing Techniques are suitable for achieving global optimal solution, In this thesis a Novel Fast Motion Estimation algorithms based on Genetic Algorithm (GA) , Particle Swarm Optimization (PSO),

Clonal Particle Swarm Optimization (CPSO). sum of Absolute Difference (SAD) is taken as the matching Criteria due to minimum computational cost and Zero Motion prejudgement is also included which aims at finding static macroblocks(MB) which do not need to perform remaining search thus reduces the computational cost. From the simulation it is observed that CPSO algorithm giving best results among all methods in unidirectional motion estimation and the proposed Bidirectional method is a novel iterative technique which reduces the number search point in aframe and a good reconstruction of frames making these proposed methods computationally fast and efficient techniques.

6.2 Scope For Future Work

From the simulation results it is observed that CPSO just consumes a few lines of codes due to its simplicity which makes the ME algorithm based on CPSO attractive for hardware implementation. In the future, variants of PSO might be applied to strengthen the global searching ability and to speed up the search and to avoid being trapped in local minima. In future other evolutionary computing techniques also can be tried for the better results. Three important factors Block size , search area, matching criteria can be varied such as Variable block size, large search area for complex motions and small search area for low complex motions and also by taking the intial particle positiuons also make a big change. This algorithm is based on a translational model of the motion of objects between frames. It also assumes that all pels within a block undergo the same translational movement.There are many other ME methods, but Block matching ME is normally preferred due to its simplicity and good compromise between prediction quality and motion overhead. In bidirectional motion estimation also we can try to implement new techniques which will further reduce the complexity of MPEG video coding.

References

1. Tekalp Murat A. Digital video processing, Prentice Hall, PTR, 1995.

2. Y.Wang, J.Ostermann and Y.Q.Zhang. *Video Processing and Communications*. Tsinghua University Press and Prentice Hall, Beijing,China,2002.

3. I. E. G. Richardson. H.264 and MPEG-4 video compression. Wiley, Chichester, England, 2003.

4. J.R.Jain and A.K.Jain, "Displacement measurement and its application in interframe image coding," *IEEE Trans. Commun.*, vol.COM-29,PP. 1799–1808, Dec. 1981.

5. S.kappagantula and K.R.Rao, "Motion compensated interframe image Prediction", *IEEE Trans. Commun.*vol,COM 33,PP, 1011–l015,Sept, 1985.

6. B.Liu and A.Zaccarin, New fast algorithms for the estimation of block motion vectors, *IEEE Trans. Circuits Syst. Video technology*, Vol.3, pp.440–445,Dec 1995.

7. R. Li, B. Zeng, and M. L. Liou, "A new three-step search algorithm for block motion estimation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 4, pp. 438442, Aug. 1994.

8. S. Zhu and K.-K. Ma, "A new diamond search algorithm for fast blockmatchingmotion estimation," in Proc. 1997 *Int. Conf. Information Communication and Signal Processing (ICICS)*, vol. 1, Sept. 9-12, 1997,pp.292-296.

9. M. Ghanbari, "The cross-search algorithm for motion estimation," *IEEE Trans. Commun.*, vol. 38, pp. 950953, July 1990.

10. L. M. Po and W. C. Ma, "A novel four-step search algorithm for fast block motion estimation,"*IEEE Trans. Circuits Syst. Video Technol.*, vol. 6, pp. 313317, June 1996.

11. Y.Liu, L.S.Kang, Y.P.Chen, Non-Numerical Parallel Algorithms– Genetic Algorithm, Science Press, 1995.

12. D.E.Goldberg, genetic algorithm in search, Optimization & Machine learning. Reading, MA:Addison-Wesley, 1989.

13. S. Li, W.-P Xu, N.-N Zheng, H. Wang, "A novel fast motion estimation method based on genetic algorithm," *ACTA ELECTRONICA SINICA*, vol. 28, No. 6, pp. 114-117, June 2000.

14. J.Kennedy and R.C.Eberhart. Particle swarm optimization. *In Proc. IEEE International Conference on Neural Networks*, Perth, Australia, 1995.

15. Y.H.Shi and R.C.Eberhart. Empirical study of particle swarm optimization. *In Proc. IEEE Congress on Evolutionary Computation*, 1999.

16. R. C. Eberhart, Y. Shi, "Comparison between genetic algorithms and particle swarm optimization," *in Proc. IEEE Int. Conf. Evol. Comput.*, Anchorage, AK, pp. 611616, May 1998.

17. F. van den Bergh and A. P. Engelbrecht, "A cooperative approach to particle swarm optimization," *IEEE Trans. on Evolutionary Computation*, vol. 8, 2004, pp. 225-239.

18. J. J. Liang, A. K. Qin, P. N. Suganthan and S. Baskar, "Comprehensive learning particle swarm optimizer for global optimization of multimodal functions," *IEEE Trans. on Evolutionary Computation,* vol. 10, 2006, pp. 281-296.

19. Y. H. Shi and R. Eberhart, "A Modified Particle Swarm Optimizer," *Proc. IEEE World Congress on Computational Intelligence*, Alaska, ALTEC, vol. 1, 1998, pp. 69-73.

20. Y. Tan, Senior Member, IEEE, Z. M. Xiao, "Clonal Particle Swarm Optimization and Its Applications," *2007 IEEE Congress on Evolutionary Computation (CEC 2007)*, pp. 2303-2309.

21. L. N. de Castro and J. I. Timmis, "Artificial Immune System as a Novel Soft Computing Paradigm," *soft computing journal*, vol. 7, no.8,2003,pp.526-544.

22. L. N. de Castro and F. J. Von Zuben, "Learning and optimization using the clonal selection principle," *IEEE Trans. on Evol. Comp.*, vol. 6, no. 3, pp. 239-251, June 2002.

23. Xuedong Yuan, Xiaojing Shen, "Block Matching Algorithm Based on Particle Swarm Optimization for Motion Estimation," *The 2008 International Conference on Embedded Software and Systems (ICESS2008)*, pp. 191-197.

24. R.. C. Eberhart, P. Simpson, and R.. Dobbins, *Computational Intelligence PC Tools*: Academic, 1996, ch. 6, pp. 212-226.

25. Y. Nie and K.-K. Ma, "Adaptive rood pattern search for fast blockmatching motion estimation," IEEE Trans. Image Proc., vol. 11, no. 12, pp. 1442-1449, Dec 2002.

26. S.W. Wu and A.Gersho, "Joint estimation of forward/backward motion vectors for MPEG interpolative prediction," *IEEE Trans.Image Proc.*, vol.3, pp.684-687, Sept. 1994.

27. M.-K.Kim, and J.-K. "Efficient motion estimation algorithm for bidirectional prediction scheme," *Electronic Letters*, vol.30, no.8, pp.632-633, April 1994.

28. S.Katra, M-N. Chong, "Bidirectional motion estimation via vector propagation," *IEEE Trans. on CAS for Video Techn.*, Vol.8, no.8, pp.976-987, Dec.1998.

29. X.Li, Y.Lu, D.Zhao, W.Gao, S.Ma, "Enhanced direct coding for bipredictive pictures," *Proc. IEEE ISCAS*, vol.3, pp.785-788, 2004

30. Vasily G. Moshnyaga, "Static Macroblock Prediction for Low Complexity MPEG Video Codec," *TENCON 2007 - 2007 IEEE region 10 conference*, Oct. 30 2007-Nov. 2 2007,pp. 1-4.