THESIS REPORT

Ph.D.

Low Complexity and High Throughput Fully DCT-Based Motion Compensated Video Coders

by U-V. Koc
Advisor: K.J.R. Liu

Ph.D. 96-9

INSTITUTE FOR SYSTEMS RESEARCH

*Sponsored by
the National Science Foundation
Engineering Research Center Program,
the University of Maryland,
Harvard University,
and Industry*

ABSTRACT

Title of Dissertation: LOW COMPLEXITY AND HIGH THROUGHPUT FULLY DCT-BASED MOTION COMPENSATED VIDEO CODERS

Ut-Va Koc, Doctor of Philosophy, 1996

Dissertation directed by: Professor K. J. Ray Liu
Department of Electrical Engineering

Many video coding standards such as H.261, MPEG1, MPEG2, HDTV and H.263, are based on the hybrid motion-compensated DCT approach. The common implementations adopt the conventional DCT-based motion-compensated video coder structure in which every raw video bit must go before being encoded through the performance-critical feedback loop consisting of a DCT, an Inverse DCT (IDCT) and a spatial-domain motion estimation/compensation. This heavily loaded feedback loop not only increases the overall complexity of the coder but also limits the throughput and becomes the bottleneck of a real-time high-end digital video system. In this dissertation, we propose a fully DCT-based motion-compensated video coder structure which eliminates IDCT and moves DCT out of the loop, resulting in a simple feedback loop with only one major component: Transform-Domain Motion Estimation/Compensation. Furthermore, different components can be jointly optimized if they operate in the same transform domain.

Based on pseudo phases and sinusoidal orthogonal principles, we develop DCT Pseudo Phase Techniques to estimate displacement directly from the DCT coefficients of shifted signals/images. We develop the DCT-based motion estimation (DXT-ME) algorithm with the computational complexity, $O(N^2)$, compared to $O(N^4)$ for the full search block matching approach (BKM-ME). Simulation shows that the DXT-ME algorithm performs as well as BKM-ME and other fast search approaches in terms of mean-square-error per pel (MSE) and bits per sample (BPS). Furthermore it has inherently highly parallel operations in computing the pseudo phases, suitable for VLSI implementation.

We develop DCT-based subpixel motion estimation algorithms without the need of image interpolation as required by conventional methods, resulting in significant reduction in computations and data flow and flexible fully DCT-based coder design because the same hardware can support different levels of required accuracy. Simulation demonstrates comparable and even better performance of the DCT-based approach than BKM-ME in terms of MSE and BPS.

We devise the DCT-based motion compensation schemes via the bilinear and cubic interpolation without any increase in computations to achieve more accurate approximation and better visual quality for motion-compensated pictures. Less computations are required in DCT domain because of sparseness and block alignment of DCT blocks and use of fast DCT to reduce computations.

LOW COMPLEXITY AND HIGH THROUGHPUT FULLY DCT-BASED MOTION COMPENSATED VIDEO CODERS

by

Ut-Va Koc

Dissertation submitted to the Faculty of the Graduate School
of The University of Maryland in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
1996

Dedication

To my parents, Kok Iong Ip and Chong I Mui, and my wife, Wen-Ling Kuo

Acknowledgements

Simple words are not enough to express my gratitude to my advisor and mentor, Professor K. J. R. Liu, for his guidance, sustained encouragement and support throughout the years of my graduate studies. My time with him has been a rewarding learning experience.

I also appreciate the efforts and time of the professors serving in my dissertation committee. In particular, I would like to thank Professor O'Leary, Professor Chellappa, Professor Fuja, and Professor Nakajima for their comments and advices on revising my dissertation.

I am grateful to the professors who have taught me in the graduate courses. I would like to thank the Institute for Systems Research (ISR) and the Electrical Engineering Department (ENEE) for providing the excellent resources and a stimulating research environment. I also wish to thank the computer staff of ISR and ENEE for their support and help in my research projects and being the lab manager of Digital Signal Processing Lab.

Thanks go to many students and friends, both past and present, in school and particularly in Digital Signal Processing Laboratory for their assistance and numerous interesting discussions. Specifically, I would like to acknowledge the contribution by Jie Chen on the unitary property of the system matrix. Without his contribution, I would not be aware of this important property of the proposed algorithm. Finally, I dedicate this thesis to my family – my parents and wife – for their constant love, patience and support.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

The demands for data, voice and multimedia services are rapidly increasing over the past few decades while the expectation of quality for these services is becoming higher and higher. To attain the highest possible quality, analog signals such as speech, audio, image, and video, are sampled and digitized as digital data for transmission/recording and reconstructed at the receiving ends in order to be free from noise and waveform distortion induced in transmission and storage. However, these digitized data are usually volumnous. Even though the technology is continuously progressing at pushing up the bandwidth limit and reducing the transmission/storage cost, still channel bandwidths and storage capacities, as tabulated in Table 1.1, are limited and relatively expensive in comparison with the volumn of these raw digital signals. To make all the digital services feasible and cost effective, data/signal compression is essential. As depicted by the Schouten diagram in Fig. 1.1, all the digital signals carry redundant and irrelevant information and are subject to be compressed by removing the redundancy and irrelevancy for efficient use of bandwidths at the lowest possible cost [30, 31].

In view of compressibility of digital signals and its importance in digital com-

CHANNEL	BANDWIDTH (BITRATE)	MEDIUM
POTS modem	$\leq$ 28.8 kbps	copper
DS0	64 kbps	copper
T1 / DS1 (24 DS0)	1.544 Mbps	copper
T3 / DS3 (28 DS1)	44.763 Mbps	copper
Cable modem	$\leq$ 30 Mbps	copper/coaxial
Ethernet	10 Mbps	copper/coaxial
Fast Ethernet	100 Mbps	copper/coaxial
ISDN	$p \times 64$ kbps	fiber
SONET / SDH	$p \times 51.84$ Mbps	fiber
FDDI (X3T9.5)	100 Mbps	fiber
CDPD	19.2 kbps	wireless
GSM1800 (DCS)	22.8 kbps	wireless
IS-54	13 kbps	wireless
IS-95	19.2, 9.6, 4.8, 2.4 kbps	wireless
PDC	11.2 kbps	wireless
TETRA	7.2 kbps	wireless
APCO Project 25	7.2 kbps	wireless

STORAGE	CAPACITY	MEDIUM
Floppy Disk	1.44 Mbytes	magnetism
CD / CD-ROM	600 Mbytes	laser
DAT	2.77 Mbits/s ($\leq$ 2 hours)	magnetism
DRAM	4–16 Mbits	seminconductor

POTS=Plain Old Telephone System	SONET=Synchronous Optical NETwork
SDH=Synchronous Digital Hierarchy	FDDI=Fiber Distributed Data Interface
CDPD=Cellular Digital Packet Data	DCS=Digital Cellular System
GSM=Global System for Mobile Communications	PDC=Personal Digital Cellular
TETRA=Trans European Trunked Radio	$p = 1, 2, \ldots$
APCO25=Associated Public Safety Communications Officers Project 25	
CD=Compact Disk	DAT=Digital Audio Tape
DRAM=Dynamic Random Access Memory	ROM=Read Only Memory

Table 1.1: List of channel bandwidths and storage capacities.

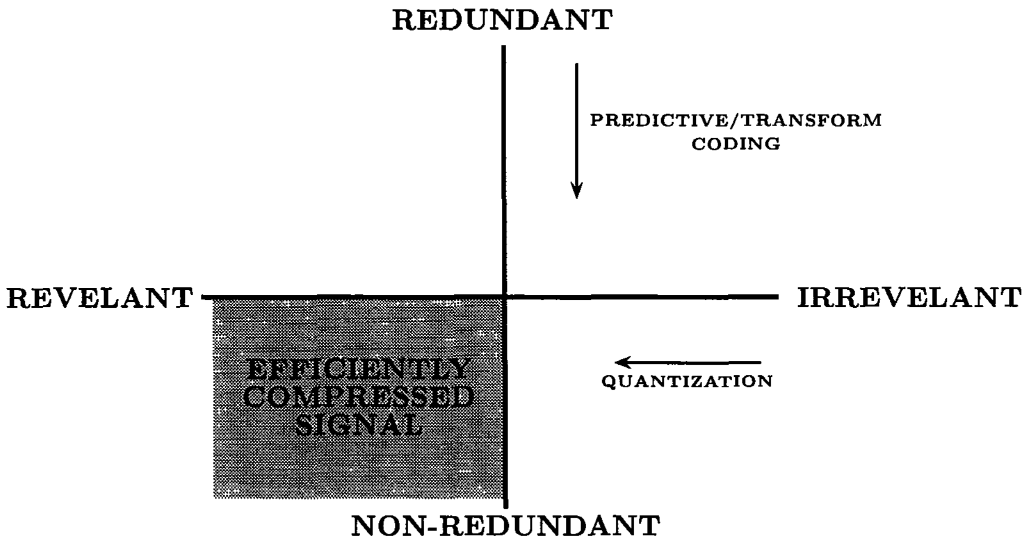


Figure 1.1: Schouten diagram shows signal compression through removal of redundancy and irrelevancy in digital signals.

munication including transmission and storage, extensive research is vigorously being pursued on data/signal compression (also called source coding in the area of digital communication) over decades [30] to reduce the data size and at the same time improve the perceived quality of the compressed signal. As illustrated in Fig. 1.2, the performance of signal compression or a source coder can be measured in four dimensions [31]:

1. Signal quality is measured in the five-point mean opinion scale (mos) associated with a set of standardized adjectival descriptions: bad, poor, fair, good, and excellent.

2. Compression efficiency indicates the number of bits per second required to transmit the compressed signal or the total number of bits for storage. An

FORMAT	RESOLUTION	RAW BITRATE	REMARK
CCIR 601	720 ppl $\times$576 lpf $\times$30 fps	149.3 Mbps	digital video
CIF	352 ppl $\times$288 lpf $\times$30 fps	36.5 Mbps	digital video
QCIF	176 ppl $\times$144 lpf $\times$30 fps	9.13 Mbps	digital video
SIF	360 ppl $\times$288 lpf $\times$30 fps	37.3 Mbps	digital video
HDTV	1280 ppl $\times$720 lpf $\times$59.94/60 fps 1920 ppl $\times$1080 lpf $\times$29.97/30 fps †	663.6 Mbps 1.3905 Gbps	digital TV
NTSC	525 lpf $\times$29.97 fps †		analog TV
PAL	625 lpf $\times$25 fps †		analog TV
SECAM	625 lpf $\times$25 fps †		analog TV
VGA	640 ppl $\times$480 lines		computer
SVGA	1024 ppl $\times$768 lines		computer

CIF=Common Intermediate Format QCIF=Quarter CIF
SIF=Source Input Format HDTV=High Definition TV
NTSC=National Television System Committee PAL=Phase Alternate Line
SECAM=Sequential Couleur avec Memoire VGA=Video Graphics Adapter
† interlaced scan: 1 frame consists of 2 fields.
ppl=pixels per line lpf=lines per frame fps=frames per sec

Table 1.2: List of source image/video formats.

alternative indicator is the compression rate defined as the ratio of the raw bit rate to the compressed bit rate.

3. The computational complexity of a compression/decompression algorithm refers to the computational requirement of the compression/decompression process, typically measured in terms of the number of arithmetic operations (ops), memory requirement, computing power requirement (millions of instructions per second or mips), power consumption, chip area required and the cost to implement.

4. Communication delay is critical to the performance of a signal compression algorithm only when two-way interactive communication is involved such as

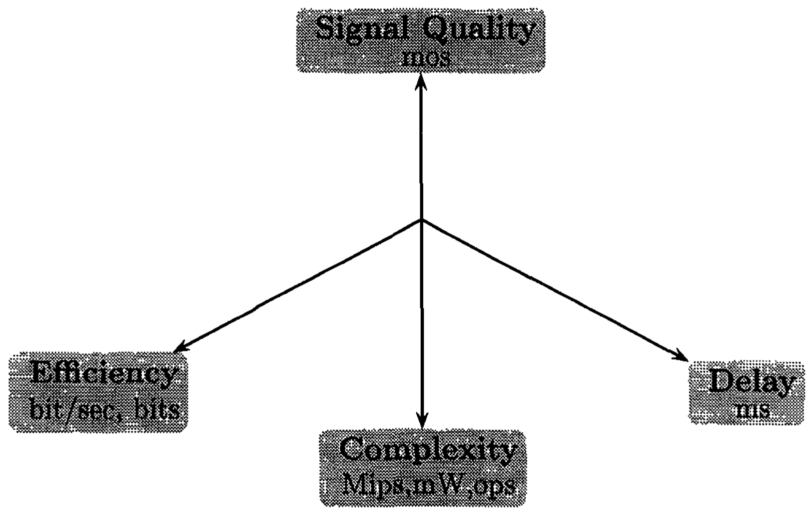


Figure 1.2: Dimensions of Performance of Signal Compression or Source Coding

in the videophone application.

Some regions in this four-dimensional space are theoretically unallowable or practically unreachable. However, there always exist tradeoffs among these four performance criteria. Depending on specific communication applications, certain tradeoffs may be more preferable than others.

In the research arena of data/signal compression, image compression [41, 40] and image sequence compression (video coding) [56, 46, 71] recently have attracted a lot of attention from the technical community due to many challenging research topics and immediate or potential applications, such as video conferencing, videophony, multimedia, high definition television (HDTV), interactive TV, telemedicine, etc. Because of emergence of various international video coding standards, advances in VLSI technology and wide-spread availability of digital computer and telecommunication networks, research efforts in video coding become directly applicable to product development and as a result increasingly important also in the industry. It can be foreseen that computing, telecommunication, networking and broadcasting will be integrated in the near future. In this merging trend, research in video coding/compression plays an increasingly important role.

1.1 Importance of Video Compression

The volumn of digital video data is notoriously huge. It is almost implausible to transmit raw video data over communication channels of limited transmission bandwidth or to save on storage devices. For the convenience of discussion, a number of commonly used source image/video formats are listed in Table 1.2.

For a high quality HDTV picture which has a spatial resolution 1920×1080 square pixels and digitized as 8-bit pixels in 3 color components at a 60 Hz interlaced scan [7], the uncompressed bit rate is about 1.3905 Gbit/sec. To compress such high-volume video data, the video processor must be of high throughput to handle such high bit rate data and low complexity to reduce the cost and increase the speed. In spite of the requirements of high throughput and low complexity for video codecs, a high compression rate is also crucial for any possible applications. For a 6MHz HDTV simulcast transmission channel bandwidth, the channel capacity is limited to 18Mbit/sec, requiring a compression rate around 77.

Consider also the Common Intermediate Format (CIF), the standard for videoconferencing recommended by CCIR [21], which contains 352 pixels per line and 288 lines per picture for the luminance signal (i.e., resolution 352×288) and 176 pels per line, 144 lines per picture for the two color difference components (chromina) [21]. At the frame rate = 30 frames per sec (fps) and 8 bits per pixel (bpp), the uncompressed bit rate for CIF is about 36.5 Mbit/sec [$= (352 \times 288 \times 30 + 176 \times 144 \times 30 \times 2) \times 8$]. Even if we use a smaller format, the Quarter CIF (QCIF) having half the number of pels and half the number of lines stated above, the bit rate of raw video data is still huge, reaching 9.1 Mbit/sec. POTS (Plain Old Telephone System), the most accessible channel by the general public, has currently only a bandwidth 28.8 kbit/sec. Even a dedicated ISDN channel has only 64 kbit/sec. Without compression, it is almost impossible or economically realistic to transmit over network or store such high-volume video data.

1.2 Organization

In order to prepare the readers for understanding the materials discussed in this dissertation, Chapter 2 will be devoted to the discussion of advances of research in video coding including the hybrid motion-compensated DCT (MC-DCT) approach and different motion estimation techniques. In Chapter 3.1, the international DCT-based video coding standards based on the MC-DCT approach will be presented. In Chapter 3.2, the disadvantages of the conventional MC-DCT video coder structure for all the DCT-based coding standards are pointed out and then the fully DCT-based coder design is proposed to overcome those disadvantages. To realize the fully DCT-based coder, DCT Pseudo-Phase Techniques are developed in Chapter 4 to estimate the motion directly from the DCT coefficients of two blocks. Application of the techniques to video coding results in the DCT-based motion estimation (DXT-ME) algorithm in Chapter 5. The interpolation-free subpixel DCT-based motion estimation algorithms are discussed in Chapter 6 to estimate the displacements of half-pel and even quarter-pel accuracy without image interpolation. In Chapter 7, the integer-pel and subpel DCT-based motion estimation algorithms are devised to complete the fully DCT-based video coder structure.

Finally, this thesis is concluded in Chapter 8.

Chapter 2

Advances in Video Coding

The research in image sequence compression or video coding is a natural extension of the research in image compression/coding active over several decades. Beyond the removal of spatial and spectral redundancy in response to our human visual system (HVS), video coding exploits further the temporal correlation between consecutive frames. In image coding, the first generation research focuses on pixel-to-pixel correlation (waveform-based) based on some statistical image models while the second generation research utilizes the knowledge of more complicated structural image models and the properties of the human visual system to achieve higher compression efficiency above the theoretical limit predicted by the classical source coding theory [41]. The second generation coding techniques can be further divided into two groups:

- Local-operator based techniques are based on the models of HVS and include pyramidal and subband coding, and anisotropic nonstationary predictive coding.

- Contour-texture oriented techniques describe an image in terms of structural primitives such as contours and textures. Two approaches were developed:

region growing based coding approach and directional decomposition coding approach.

In video coding, recent research focuses can be categorized roughly in two main groups [46, 71]: waveform-based coding and model-based (or knowledge-based) coding.

2.1 Waveform-Based Video Coding

Compression is achieved directly on a two-dimensional, discrete distribution of light intensities to attain a given acceptable level of waveform distortion with the least possible encoding rate by eliminating three types of redundancies: spatial, temporal, and spectral (chromatic). Due to the fact that high spectral correlation exists among three primary colors (red, green and blue) and the HVS is not as sensitive to the chrominance components as to the luminance component of a color image, reduction of spectral redundancy is attained by linearly transforming the color space from RGB (red-green-blue) to YUV or YCrCb (luma-chroma) and then subsampling the chrominance components (called 4:2:2 subsampling). Spatial and temporal compression can be achieved either separately (spatial/temporal) or jointly (spatio-temporal). To achieve high temporal compression, waveform-based coding usually requires motion estimation and compensation.

2.1.1 Spatial/Temporal Compression (Hybrid Approach)

This hybrid approach treats temporal compression and spatial compression separately so that the benefits of both schemes can be retained. Temporal compression is often achieved through temporal prediction or motion estimation and compensation while spatial compression is usually accomplished via transform coding, subband coding or vector quantization.

- Hybrid spatial and temporal compression — A transform coder is followed by a DPCM coder (temporal predictor).

- Hybrid temporal and spatial compression (also called vector-predictive coding) — The transform coder is put inside the feedback loop of the predictive coder.

- Motion-compensated hybrid approaches — The temporal predictor is assisted by motion estimation and compensation to further reduce temporal redundancy and result in much smaller motion compensated residuals. Depending on what spatial compression method is used, these hybrid approaches can be further categorized as follows:

  - Motion-compensated transform coding: Overlapping block-based motion estimation/compensation techniques are employed to achieve temporal compression and the resulting motion compensated prediction errors are further compressed by a transform coder. If the transform coder adopts Discrete Cosine Transform (DCT), we call this hybrid approach the motion-compensated DCT scheme (MC-DCT). When the images are generated by a first-order Markov process, DCT is equivalent to the optimum Karhunen-Loeve transform (KLT) [28] which packs most energy in as few transform coefficients as possible. Moreover, for images of real-world scenery, DCT is also a very efficient transform coder. MC-DCT is the basis of many international video coding standards and will be discussed in detail in a later chapter.

  - Motion-compensated subband coding: The motion compensated frame differences are decomposed into 2-D subbands and compression is facilitated by truncating some subbands.
  - Motion-compensated vector quantization: Either 4×4 or 8×8 vector quantization is applied to either the motion compensated residuals if motion is detected or the intra frames if no significant motion is found.

In addition to the difference in approaches, the shape of a basic encoding unit can also be a square block, an irregular block, or the full frame:

- Block-based approach — A whole frame is divided into many squared blocks, each of which is coded by different approaches such as the above motion-compensated hybrid approaches, or a fractal approach.
- Region-based approach — Unlike the block-based approach, a frame is segmented into blocks of different irregular shapes according to some criteria such as motion vector fields. Usually a patch mesh is built and the motion of the grid points is tracked instead of every pixel. Depending on how many grid points determine one segment or whether the mesh is adjusted frame by frame, it can be:
  - Either quadrangle-based (four grid points for each segment) or triangle-based (three grid points),
  - Either fixed mesh or adaptive mesh: For the adaptive mesh approach, grid points are tracked based on an energy criterion and pixels inside a segment are interpolated by a simple function found by curve fitting. This technique is also found to apply to coding mouth motion which is difficult for conventional block based approaches.

- Full-frame approach — Treat each frame as a point in a subspace and track the slow change of this subspace.

2.1.2 Spatio-temporal Compression (Joint Approach)

A video sequence is considered as a 3-dimensional signal (2 spatial dimension and 1 temporal dimension) and, therefore, spatial and temporal compressions are achieved in a uniform manner. The joint approach is considered to be an extension of the 2-D image coding and may incorporate motion compensation within the 3-D coder. Different techniques can be applied and classified as: 3-D transform coding (including 3-D DCT coding and 3-D wavelet coding), 3-D subband coding, and 3-D fractal coding. These techniques divide the whole sequence into different frequency bands. Each band will be encoded separately according to its characteristics.

2.2 Model-Based Video Coding

Images are viewed as a 2-D projection of a 3-D real scene. The concept is to construct a model with a priori knowledge of images and find the model parameters. In this way, only the model parameters need to be sent and thus a very high compression rate is achieved. Model-based video coding approach has 3 key elements: modeling, analysis and synthesis. According to different modeling steps, two major categories are:

- Object-based approach — No explicit object model is given. Every frame is composed of objects and each object is associated with 3 sets of parameters: motion A_i, shape M_i, and color S_i.

- Semantic-based approach — This approach is sometimes called compression through animation because it uses explicit object models. This approach is usually limited to coding a talking human face. A human facial model must be constructed first by means of different geometric models:
  - Surface-based parametric model: spline, harmonic surface for relatively regular geometric shapes.
  - Surface-based nonparametric model: wireframe (3D wireframe model) – planar polygonal patches of adjustable size. This is the most popular model.
  - Volume-based parametric model: generalized cylinder (GC), superquadrics. This model is capable of modeling nonrigid motion.
  - Volume-based nonparametric model: voxels.

  The problem for this approach is the time-consuming analysis step which finds the model parameters to best fit the images.

2.3 Motion Estimation and Compensation

As can be seen in the above discussion, motion estimation is effective in removing temporal redundancy for video coding. Unlike DPCM (linear prediction), it belongs to the class of nonlinear predictive coding techniques. For video compression, motion estimation techniques estimate the field associated with the spatiotemporal variation of intensity called the optical flow instead of the true motion field of objects as required in the field of computer vision. In other words, estimation of the true motion is not the ultimate goal but it is desirable to obtain the true motion information to avoid any artificial discontinuities in the predicted error. As a result, the terms "motion field" and "optical flow" are usually used interchangeably without distinction. Then motion compensation techniques are employed to predict the current frames based on the motion information and the previous frames. The purpose of video compression is to minimize the overall amount of information including motion information and prediction error information to be sent or stored for decoding. Therefore, the tradeoff of bit allocations exists between motion parameters and prediction errors. Furthermore, for limiting the coding delay, motion estimation in video coding usually utilizes either the previous frame or the next future frame as the reference, even though all the other frames in a video sequence can be referenced ideally in an accurate motion estimation procedure.

For the consideration of motion estimation in the context of video coding, three main causes give rise to the spatiotemporal intensity variation:

- Global motion or camera motion such as pan or zoom causing the apparent motion of the objects in the scene,
- Local motion of objects with respect to each other and the background,
- A change of the illumination condition which is generally not taken into account by the motion estimation techniques.

The problem of motion estimation can be approached in a deterministic framework or a stochastic (Bayesian) one. In the stochastic framework, the motion is usually modeled as a Markov random field with a joint distribution characterized as a Gibbs distribution and techniques such as maximum a posteriori (MAP) and mimimum expected cost (MEC) can be applied to motion estimation. However, for the deterministic approach, the motion is considered as an unknown quantity

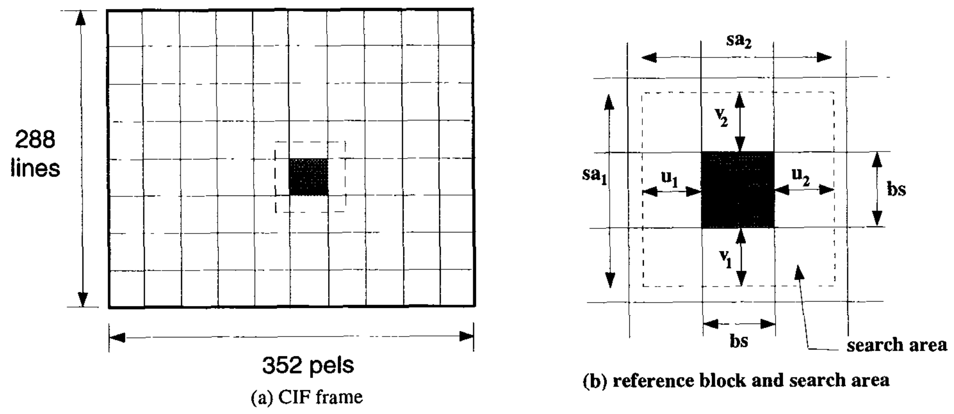


Figure 2.1: Full Search Block Matching Approach (BKM-ME)

and can be modeled as either a perspective projection or an orthographic projection from the 3D coordinate to the 2D image coordinate on the camera plane. In this framework, motion estimation techniques can be classified in four main groups [15]: block matching techniques, gradient (optical flow) techniques, pel-recursive techniques, and frequency-domain techniques.

2.3.1 Block matching techniques

By assuming only the translational motion of rigid objects on the 2D image plane, the entire image is partitioned into $N \times N$ blocks. Each block in the current block is measured against all the possible blocks in the search area of the previous frame based on some optimization criterion. Precisely, the block matching methods try to find the best motion vector satisfying

$$\hat{d} = (\hat{u}, \hat{v}) = arg \min_{(u,v) \in \mathsf{S}} \frac{\sum_{\mathsf{W}} ||x_2(m,n) - x_1(m-u, n-v)||}{N^2},$$

where $||x||$ is the metric distance defined as $||x|| = x^2$ for the Mean-Square-Error (MSE) criterion or $||x|| = |x|$ for the Mean-Absolute-Difference (MAD) criterion and S and W denote the set of allowable displacements and the measurement window respectively depending on which block matching approach is in use. For the full (exhaustive) search block matching approach, $\mathsf{W} = \{0, \ldots, N-1\}^2$ and S will include all the possible block positions in the search area.

The block matching approaches enjoy certain advantages such as simplicity in concept, direct minimization of the motion-compensated residuals in terms of MAD or MSE, and little overhead motion information. However, there are some major drawbacks: unreliable motion fields, blocking artifacts, and poor prediction along moving edges.

2.3.2 Gradient techniques

Assuming the invariant illuminance condition, it can be shown that

$$\vec{\mathbf{v}} \cdot \vec{\nabla} I(x,y,t) + \frac{\partial I(x,y,t)}{\partial t} = 0, \tag{2.1}$$

where $\vec{\mathbf{v}} = (v_x, v_y)^T = \vec{\mathbf{d}}/\Delta t$ as $\Delta t \to 0$. This equation is known as the optical flow constraint equation. Since the solution for $\vec{\mathbf{v}}$ at (x,y,t) is not unique, an additional smoothness constraint, $\min\{(\frac{\partial v_x}{\partial x})^2 + (\frac{\partial v_x}{\partial y})^2\}$ and $\min\{(\frac{\partial v_y}{\partial x})^2 + (\frac{\partial v_y}{\partial y})^2\}$, is introduced to limit the solution of $\vec{\mathbf{v}}(x,y,t)$ to vary smoothly in the spatial domain (x,y). Consequently, the optical flow is obtained by minimizing the following error term

$$\begin{aligned} \int\!\!\int \{ (\vec{\mathbf{v}} \cdot \vec{\nabla} I(x,y,t) + \frac{\partial I(x,y,t)}{\partial t})^2 + \alpha^2 [(\frac{\partial v_x}{\partial x})^2 + (\frac{\partial v_x}{\partial y})^2 \\ + (\frac{\partial v_y}{\partial x})^2 + (\frac{\partial v_y}{\partial y})^2] \} dx\, dy \end{aligned} \tag{2.2}$$

where α^2 is a weighting factor. This minimization problem is solved by an iterative Gauss-Seidel procedure.

The gradient techniques provide an accurate dense motion field but have two major drawbacks: (1) The dense motion field requires many bits to encode. (2)

The prediction error is large on moving object boundaries due to the smoothness constraint.

2.3.3 Pel-recursive techniques

Pel-recursive techniques can be considered as a subset of the gradient techniques. Given the intensity profiles in two consecutive frames, I_t and I_{t-1}, it iteratively minimizes the Displaced Frame Difference (DFD) value defined as

$$DFD(k,l,\vec{\mathbf{d}}) = \mid I_t(k,l) - I_{t-1}(k-u,l-v) \mid$$

by the steepest descent optimization algorithm where $\vec{\mathbf{d}} = (u,v)$. The resulting iterative equation to estimate the motion $\vec{\hat{\mathbf{d}}}(k,l) = (\hat{u}(k,l), \hat{v}(k,l))$ at the i^{th} iteration is given as follows:

$$\vec{\hat{\mathbf{d}}}_i(k,l) = \vec{\hat{\mathbf{d}}}_{i-1}(k,l) - \epsilon DFD(k,l,\vec{\hat{\mathbf{d}}}_{i-1}) \cdot \nabla I_{t-1}(k-\hat{u}_{i-1}, l-\hat{v}_{i-1}) \qquad (2.3)$$

where ϵ is a convergence factor, and $\nabla x_{t-1}(x,y) = \frac{\partial^2 x_{t-1}}{\partial x^2} + \frac{\partial^2 x_{t-1}}{\partial y^2}$.

The pel-recursive techniques can update the motion vectors based only on previously transmitted data and thus no overhead motion information is required because motion can be estimated at the decoder as well. However, the drawbacks include the convergence depending on the choice of ϵ, susceptibility to noise, and incapability to handle large displacements and motion discontinuities.

2.3.4 Frequency-domain techniques

Frequency-domain techniques are based on the relationship between transformed coefficients of shifted images. Several methods are available in this category: the Complex Lapped Transform (CLT) motion estimation method and the Fourier

Transform (DFT or FFT) phase correlation method, and 3-D spatio-temporal frequency-domain analysis using Wigner distributions or Gabor filters.

the CLT approach estimates the motion by finding, over all possible values of (k, l) within the search area, the minimum of the 2-dimensional cross correlation function $y(k, l)$ defined as follows:

$$\begin{aligned} y(k,l) &= \sum_{m,n=-(N-\frac{1}{2})}^{(N-\frac{1}{2})} x_1(m,n)x_2(m+k,n+l)\cos^2(\frac{m\pi}{2N})\cos^2(\frac{n\pi}{2N}) \\ &= \Re\{\sum_{v=0}^{N-1}\sum_{u=-N}^{N-1} X_1(u,v)X_2^*(u,v;k,l)\}, \end{aligned}$$

where $\{x_2(m,n) = I_{t-1}(m,n); m,n = -(N-.5),\ldots,(2N-.5)\}$ is the search area from the previous frame I_{t-1} and $\{x_1(m,n) = I_t(m,n); m,n = -(N-.5),\ldots,(N-.5)\}$ is the reference block from the current frame I_t. Here $X_1(u,v)$ and $X_2(u,v)$ are the 2-Dimensional CLT of $x_1(m,n)$ and $x_2(m,n)$ respectively. The 2-Dimensional CLT, $X(k,l)$, of $x(m,n)$ is defined as

$$X(k,l) = \frac{1}{\sqrt{2}N}\sum_{m,n=-(N-\frac{1}{2})}^{(N-\frac{1}{2})} x(m,n)e^{-j[(k+.5)\frac{m\pi}{N}+(l+.5)\frac{n\pi}{N}]}\cos(\frac{m\pi}{2N})\cos(\frac{n\pi}{2N});$$
$$\text{for } k = -N,\ldots,N-1; l = 0,\ldots,N-1.$$

The phase correlation method is based on the principle that a relative shift in the spatial domain results in a linear phase shift in the Fourier domain. It estimates the translational motion (u, v) between two $N \times N$ image matrices $\mathbf{x}_1$ and $\mathbf{x}_2$ of which the (m,n) element is $I_t(m,n)$ and $I_{t-1}(m,n)$ respectively for $m,n = 0,\ldots,N-1$. If these two image matrices differ by a translational displacement, then the displacement can be found by locating the peak of the inverse 2-D Fourier transform of the normalized cross-correlation function of the Fourier transform of these two blocks:

$$\text{IDFT}\{\frac{\mathbf{X}_1\mathbf{X}_2^*}{|\mathbf{X}_1\mathbf{X}_2^*|}\}$$

where IDFT denotes the Inverse DFT (Discrete Fourier Transform), $\mathbf{X}_1 = \text{DFT}\{\mathbf{x}_1\}$ and $\mathbf{X}_2 = \text{DFT}\{\mathbf{x}_2\}$.

Chapter 3

Motion-Compensated DCT Video Coding Approach

3.1 Motion-Compensated DCT Video Coding Standards

The motion-compensated DCT video compression scheme (MC-DCT) is the basis of several international video coding standards which are tabulated in Table 3.1, ranging from the low bit-rate and high compression-rate videophone application to the high-end high bit-rate and high quality High-Definition Television (HDTV) application requiring a modest compression rate. As mentioned in Chapter 2, the MC-DCT scheme belongs to the class of the hybrid spatial/temporal waveform-based video compression approaches. As illustrated in Fig. 3.1, the MC-DCT scheme employs motion estimation and compensation to reduce/remove temporal redundancy and then uses DCT to exploit spatial correlation among the pixels of the motion-compensated predicted frame errors (residuals). Efficient coding is accomplished by adding the quantization and variable length coding steps after

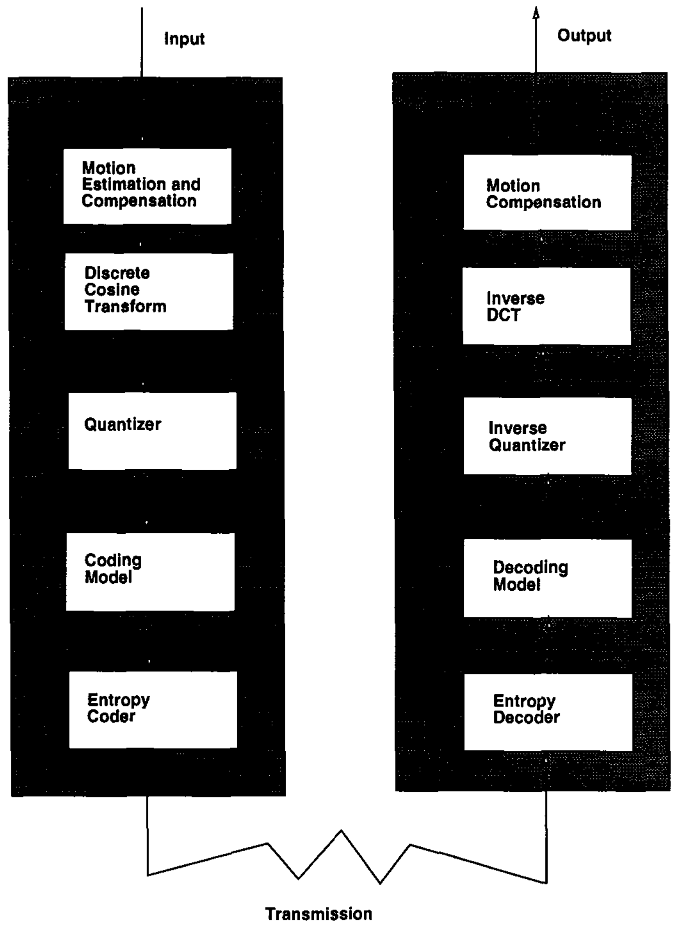


Figure 3.1: Motion-Compensated DCT (MC-DCT) Scheme.

the DCT block. Basically all the standards in Table 3.1 follows this procedure with modifications of each step to reach different targeted bitrate and application goals. In the following, the DCT-based video coding standards will be discussed to provide the readers the necessary background [71].

3.1.1 H.261

H.261 was adopted by ITU (International Telecommunication Union, formerly called CCITT) in 1990 for videoconferencing services over ISDN which has a data

STANDARD	APPLICATION	TARGETED BITRATE	REMARK
JPEG	for still image only		
H.261	teleconferencing over ISDN	$p \times 64$ kbps	
MPEG-1	video on CD-ROM	$<$ 1.5 Mbps	
MPEG-2	generic high bitrate applications	$>$ 1.5 Mbps	also called H.262
HDTV	terrestrial broadcasting	18 Mbps	based on MPEG-2
H.263	low bitrate communications over PSTN	$\sim$ 18 kbps	

Table 3.1: DCT-Based Motion-Compensated Video Coding Standards

bandwidth in a multiple of 64 kbps [21]. Because of its bidirectional communication nature, the maximum coding delay is specified to be 150 ms. The input formats used and defined in H.261 are CIF (Common Intermediate Format) and QCIF.

The video data structure in H.261 is a hierarchical structure. Each frame has a picture layer leading with a picture header specifying the picture format and frame number and followed by 12 groups-of-block layers (GOB) for CIF format and 3 GOBs for QCIF. Each GOB in turn have 33 macroblocks (MB) for both CIF and QCIF. A macroblock is the basic data unit for compression mode selection and consists of a macroblock header, the compression mode, 4 8×8 Y blocks (luma), 1 8×8 U block and 1 8×8 V block (chroma) due to subsampling the chrominance components.

There are two major modes in H.261: intra mode and inter mode. In the intra mode, only DCT is used for compression in a similar way to JPEG image compression but, in inter mode, the motion-compensated DCT approach is applied – motion estimation and compensation is used to exploit temporal correlation in addition to DCT compression. In each MB, 10 possible compression modes specify the major mode, the quantization step size (MQUANT), the motion vector data (MVD), the coded block pattern (CBP), or whether a spatial filter is applied to the motion-compensated residuals.

DCT coefficients are thresholded and then uniformly quantized with a stepsize 8 for the DC component (the DCT coefficient indexed as $(0,0)$) or the specified MQUANT value for the AC components (all DCT coefficients other than the DC component). For AC components, a central dead zone around zero is used to avoid the ringing effect. The quantized DC and AC coefficients are zigzag scanned as shown in Fig. 3.2 and then encoded with the run-length code which specifies a series of events containing a run length of zero coefficients preceding a nonzero coefficient and the value of the nonzero coefficient.

Under a constant visual quality, the encoded bit stream has a variable bit rate. For ISDN transmission, a fixed bit rate is desired. Therefore, the rate/buffer control mechanism is recommended but not specified in H.261.

3.1.2 MPEG-1

MPEG-1 was approved as an ISO (International Standards Organization) standard by late 1992 [54]. MPEG-1 is intended for video storage and playback on the storage devices such as CD-ROM, magnetic tapes and hard drives at a quality comparable to the VHS analog video. Therefore, the maximum coding delay in MPEG-1 is 1 second, enough for the purpose of unidirectional video access and much larger than the maximum delay specified in H.261. The typical bit rate is 1.5 Mbps at a CD-ROM playback speed for the combination of video, audio and system bitstreams allocated as follows:

- video – 1.150 Mbps,

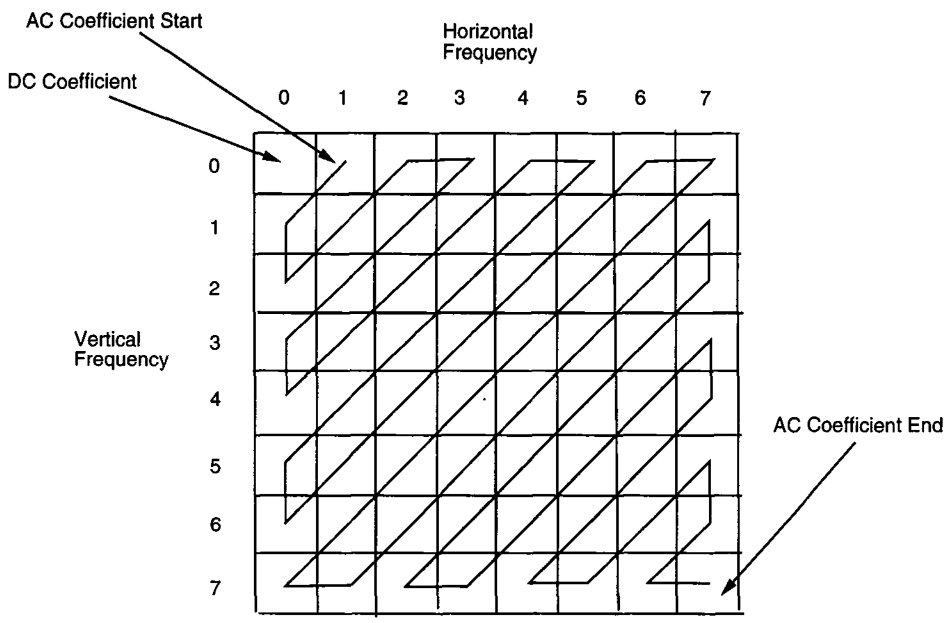


Figure 3.2: Zig-zag

- audio – 0.256 Mbps,
- system – 0.094 Mbps.

The basic input format is SIF in the progressive (noninterlaced) mode. The (Y,Cr,Cb) color space and the (4:2:0) line sampling (i.e., subsampling color frames in both the horizontal and vertical directions) are used. Unlike H.261, many video parameters, such as the picture size and the frame rate, etc, are changeable and can be specified in the MPEG-1 bitstream syntax. However, the maximum frame rate and size is limited to 720 pixels/line $\times$576 lines/frame $\times$30 frames/sec.

MPEG-1 is a generic standard defining the syntax and semantics of the encoded bitstream and implying the decoding process without limiting which algorithms or methods to use for compression. The MPEG-1 bitstream has a hierarchical data structure composed of six layers as tabulated in Table 3.2. The upper-level layer

MPEG-1 Layer	**Purpose**
Sequence Layer	Random Access Unit: Context
Group of Pictures Layer	Random Access Unit: Video Coding
Picture Layer	Primary Coding Unit
Slice Layer	Resynchronization Unit
Macroblock Layer	Motion Compensation Unit
Block Layer	DCT Unit

Table 3.2: Six layers in a MPEG-1 bitstream.

contains several lower-level layers. At the Sequence Layer, a video sequence header is inserted to specify the picture width and height, pel aspect ratio, frame rate , bit rate and buffer size. At the Group of Pictures (GOP) Layer, several frames are grouped together to form a random access unit as shown in Fig. 3.3. In other words, the decoding process must start at the first frame of the GOP Layer. At the Picture Layer, frames are classified in three types:

- Intra-frame (I): Only JPEG-like spatial compression through DCT, quantization and variable length coding is employed. The compression rate for I frames is modest. Without any reference to any other frames, an I frame is used to serve as the access point to the sequence.

- Forward predicted frame (P): Motion estimation and compensation (MC) is used to predict the current frame from the previous I or P frame. The compression rate for P frames is higher than that of I frames.

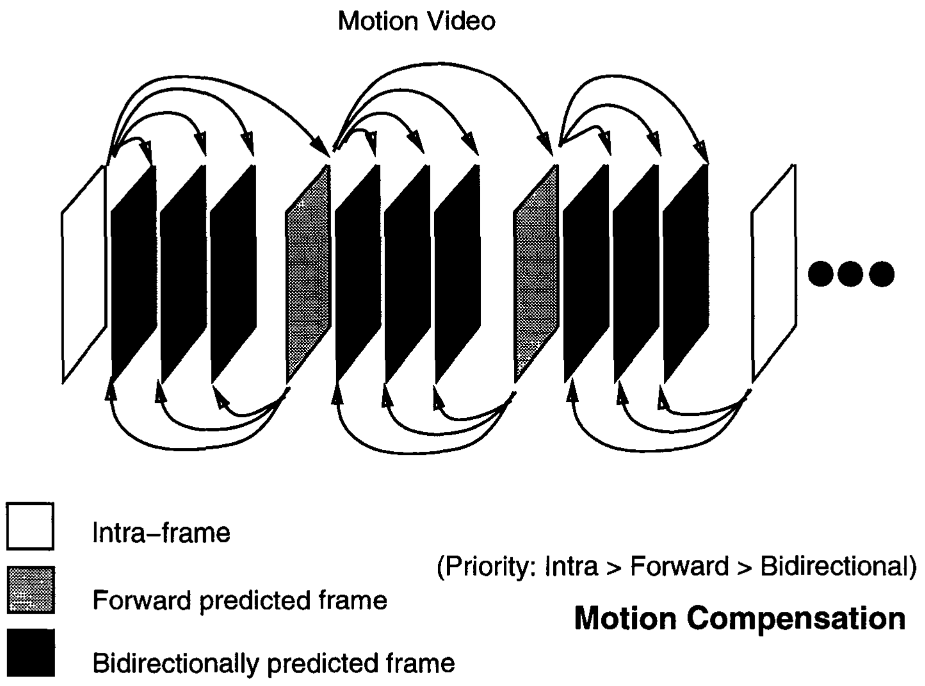

Motion Video
Intra–frame
Forward predicted frame
Bidirectionally predicted frame
(Priority: Intra > Forward > Bidirectional)
Motion Compensation

Figure 3.3: Groups of pictures.

- Bidirectionally predicted frame (B): MC is based on the previous and the next I or P frames. The prediction mode for a macroblock in B-frame can be one of the following depending on which mode produces the smallest number of bits: intra (no MC), forward predicted (MC based on the next I/P frame), backward predicted (MC based on the previous I/P frame), average (MC based on the previous and the next I/P frames). The advantages of B frames are:
  - The uncovered area can be predicted from the next frame. As a result, a very high compression rate can be achieved.
  - Better signal-to-noise reduction is resulted from MC on two frames separately.

- B frames are not used for reference by any other frames. Therefore, there is no error propagation.
- The number of B-frames in a GOP is adjustable.

However, B frames require more frame buffers and cause a larger coding delay.

Different from H.261, the DC components of all the blocks in each frame are grouped together and encoded separately due to the observation that the DC components usually have different statistical characteristics from the rest of DCT coefficients. In view of the nature of B frames, the frame reordering is required for encoding/decoding:

1. Display order:	IBBPBBPBBI...	$\leftarrow$ frame type	
	123456789A...	$\leftarrow$ frame number	At the Slice Layer,
2. Coding order:	IPBBPBBIBB...	$\leftarrow$ frame type	
	1423756A89...	$\leftarrow$ frame number	

each slice is formed from several macroblocks and used mainly for error recovery. At the Macroblock Layer, a macroblock serves as the basic compression unit similar to the case of H.261 and may be in one of 2, 8, and 12 compression modes for I, P, and B frames respectively. The basic DCT unit is a 8×8 block at the Block Layer.

In MPEG-1, the set of coding parameters is flexible. However, in order to guarantee interoperability of codecs, a special subset of the parameter space is defined as Constrained Parameter Bitstream (CPB) to represent a reasonable compromise well within the primary target of MPEG and serve as an optimal point for cost effective VLSI implementation in 1992 technology. For easy reference, the major differences between H.261 and MPEG-1 are summarized in Table 3.3 [71].

H.261	**MPEG-1**
Sequential access	Random access
Only one basic frame rate	Flexible frame rate
Only CIF/QCIF format	Flexible frame size
Only I and P frames	I, P, B frames
Full-pel motion estimation accuracy	Half-pel motion estimation accuracy
Filter of motion-compensated residuals	No filter
Variable threshold + uniform quantization	Quantization matrix
No GOP	GOP
GOB structure	Slice Layer

Table 3.3: Comparison of H.261 and MPEG-1.

3.1.3 MPEG-2 (H.262) and HDTV

MPEG-2 is designed to be the extension of MPEG-1 to accomodate different visual quality requirements at various bitrates and resolutions [55]. Because of the feasibility of MPEG-2 to realize high-quality high-resolution TV applications, the HDTV protocol adopts MPEG-2 [7].

In order to fit one standard to a variety of applications without causing unreasonable implementation difficulties, MPEG-2 uses the concept of a profile which is a subset of the full possible range of algorithmic tools (called limit syntax in the MPEG-2 term) for a particular application. There are five profiles with a hierarchical relationship. Within each profile, a number of levels are defined to limit the range of parameter values reasonable to implement and practically useful (called limit parameters). Therefore the syntax supported by a higher profile includes all the syntactic elements of lower profiles. In other words, for a given level, a Main profile decoder should be able to decode a bitstream conforming to Simple profile restrictions. For a given profile, the same syntax set is supported regardless of level.

- Simple Profile: It does not allow use of B frames and scalable coding. The maximum bitrate is 15 Mbps. This profile is intended for videotape recording.

- Main Profile: No scalability is allowed. The intended use is the Studio TV application. It is expected that 95% of MPEG-2 users will use this profile.

- SNR Scalable Profile: It is the same as the Main Profile but added with SNR Scalability is added, allowing two layers of coding (the lower layer and the enhancement layer) using different quantizer step sizes for the DCT coefficients. It can create a sharper image when combined than that obtainable from one layer alone.

- Spatially Scalable Profile: Added is spatial scalability which allows the decoder to choose different resolutions by employing a pyramidal coding approach. This profile supports only the high-1440 level intended for the consumer HDTV.

- High Profile: It is basically a scalable profile with either 4:2:0 or 4:2:2 macroblocks (i.e., chrominance subsampling in the horizontal direction but not in the vertical direction) designed for the film production SMPTE 240M standard.

There are four possible levels: Low Level, Main Level, High-1440 Level and High Level. The allowable combinations of levels and profiles and the corresponding

Level	Spatial resolution layer	Profile				
		Simple	Main	SNR	Spatial	High
High	Enhancement		1920 × 1152 × 60			1920 × 1152 × 60
	Lower					960 × 576 × 30
High-1440	Enhancement		1440 × 1152 × 60		1440 × 1152 × 60	1440 × 1152 × 60
	Lower				720 × 576 × 30	720 × 576 × 30
Main	Enhancement	720 × 576 × 30	720 × 576 × 30	720 × 576 × 30		720 × 576 × 30
	Lower					352 × 288 × 30
Low	Enhancement		352 × 288 × 30	352 × 288 × 30		
	Lower					

Table 3.4: Maximum sampling density in different combinations of levels and profiles. $x \times y \times t$ means x pixels/line, y lines/frame and t frames/sec.

sampling density are shown in Table 3.4 where a scalable MPEG-2 video stream can be broken into different layers: base or lower layers (high priority bitstream) and enhancement layers (low priority bitstream).

Different from MPEG-1 or H.261, MPEG-2 supports interlaced video input images which are scanned as even and odd fields to form frames. Therefore, there are two new picture types for interlaced video in addition to the picture types in the progressive video mode:

- Frame pictures are obtained by interleaving the lines of an odd field and its corresponding even field.
- Field pictures are formed from a field of pixels alone.

All these pictures can be either I, P, or B frames as in the case of progressive video. The differences of MPEG-2 from MPEG-1 are summarized as follows in Table 3.5.

Layer	**MPEG-2 differs from MPEG-1**
Sequence	more aspect ratios and larger allowable frame size
	4:2:2 and 4:4:4 macroblocks
	indication of source video types, color primaries, etc.
Picture	user-selectable DC precision
	concealment of motion vectors for I-pictures to increase robustness
	non-linear macroblock quantization factor
	signal source composite video characteristics
Macroblock	no more macroblock stuffing

Table 3.5: Differences of MPEG-2 from MPEG-1.

3.1.4 H.263

H.263 is the video coding standard for low bitrate communication over the ordinary telephone line (POTS) [22]. As in other standards, the video bitrate may be variable and controlled by the terminal or the network. H.263 allows five standardised picture formats with the color space (Y, C_B, C_R) sampled in the 4:2:0 format:

- sub-QCIF: 96 lines/frame with 128 pixels each line for the luminance;
- QCIF: 144 lines/frame with 176 pixels each line for the luminance;
- CIF: 288 lines/frame with 352 pixels each line for the luminance;
- 4CIF: 576 lines/frame with 704 pixels each line for the luminance;
- 16CIF: 1152 lines/frame with 1408 pixels each line for the luminance;

at the frame rate 29.97 frames/sec (exactly 30,000 frames for 1,001 seconds) in the progressive mode.

H.263 also has a hierarchical data structure with four layers:

1. Picture Layer: Each frame consists of 6 groups of blocks (GOBs) for sub-QCIF, 9 for QCIF, 18 for CIF, 4CIF and 16CIF. Data for each picture starts with byte aligned Picture Start Code (PSC) and other picture header information followed by GOBs, an end-of-sequence (EOS) code and stuffing bits for byte alignment. The type information in the header specifies the picture type (INTRA frame = I-picture and INTER frame = P-picture) and four optional modes: Unrestricted Motion Vector mode, Syntax-based Arithmetic Coding mode, Advanced Prediction mode, PB-frames mode.

2. Group of Blocks Layer: Each GOB has 1 macroblock row for sub-QCIF, QCIF and CIF, 2 macroblock rows for 4CIF and 4 for 16CIF. Data for GOB starts with stuffing bits plus byte aligned GOB Start Code (GBSC), and other header information, and ends with macroblock data.

3. Macroblock Layer: Each macroblock in turn has 4 8×8 luminance blocks and 2 spatially corresponding 8×8 color difference blocks in the default mode. The macroblock data include the macroblock header specifying the macroblock type, motion vector data, and block data.

4. Block Layer: DC components of DCT coefficients are encoded as INTRADC with a fixed-length code separately from the rest of the coefficients encoded as TCOEF with a run-length code.

Quantization information is spread throughout all the layers.

In the default mode, motion vectors are restricted to the range $[-16, 15.5]$ such that all referenced pixels are confined in the coded picture area. However, in the

Unrestricted Motion Vector mode, motion vectors are allowed to point outside the picture with a maximum range $[-31.5, 31.5]$ under certain restrictions and the pixel values outside the picture is set to the edge pixel value by limiting the motion vector to the last full pixel position inside the coded picture area.

In the Syntax-based Arithmetic Coding (SAC) mode, all the variable length coding/decoding operations are replaced with arithmetic coding/decoding. The use of the variable length codec (VLC/VLD, usually accomplished by Huffman coding) implies that each symbol must be encoded into a fixed integral number of bits but an arithmetic coder can remove this restriction and allow a variable non-integral number of bits of the code length [63]. Thus significantly fewer bits are produced.

The Advanced Prediction mode allows overlapped block motion compensation and four motion vectors per macroblock. The option of four motion vectors per macroblock implies that each 8×8 block is associated with one motion vector. For overlapped motion compensation, an 8×8 luminance block $\hat{p}(i,j)$ is a weighted sum of three prediction values created as follows:

$$\hat{p}(i,j) = [\sum_{k=0}^{2} p(i + \hat{u}_k, j + \hat{v}_k) H_k(i,j) + 4]/8$$

where $(\hat{u}_k, \hat{v}_k)$ is the motion vector for the current block ($k = 0$), the block either above or below ($k = 1$), or the block either to the left or right of the current block. $p(i,j)$ is the reference (previous) frame and $\{H_k(i,j);\ k = 0, \ldots, 2\}$ are defined as follows:

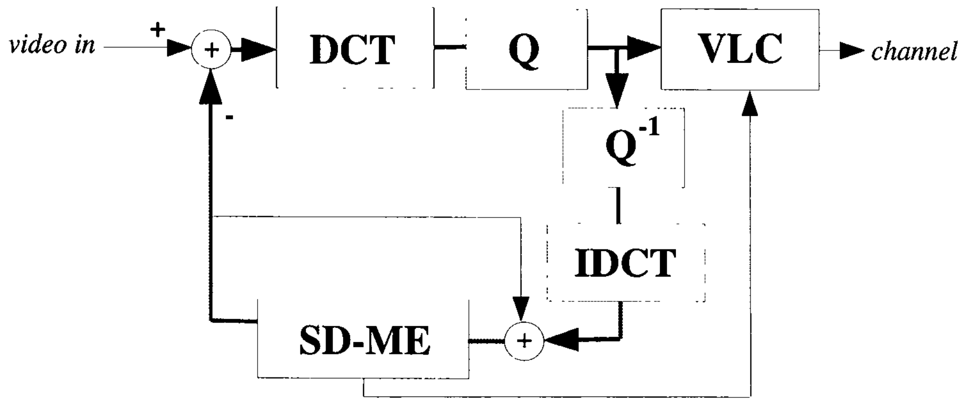
$$H_0 = \begin{bmatrix} 4 & 5 & 5 & 5 & 5 & 5 & 5 & 4 \\ 5 & 5 & 5 & 5 & 5 & 5 & 5 & 5 \\ 5 & 5 & 6 & 6 & 6 & 6 & 5 & 5 \\ 5 & 5 & 6 & 6 & 6 & 6 & 5 & 5 \\ 5 & 5 & 6 & 6 & 6 & 6 & 5 & 5 \\ 5 & 5 & 6 & 6 & 6 & 6 & 5 & 5 \\ 5 & 5 & 5 & 5 & 5 & 5 & 5 & 5 \\ 4 & 5 & 5 & 5 & 5 & 5 & 5 & 4 \end{bmatrix}, H_1 = \begin{bmatrix} 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 \\ 1 & 1 & 2 & 2 & 2 & 2 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 2 & 2 & 2 & 2 & 1 & 1 \\ 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 \end{bmatrix}, H_2 = H_1^T.$$

In the PB-frames mode, A PB-frame consists of two pictures (one P-picture predicted from the previous decoded P-picture and one B-picture predicted both from the previous decoded P-picture and the P-picture currently being decoded) coded as one unit. This is different from MPEG-2 where B-frames are encoded separately from P-frames and can be predicted from I-frames. Coding the PB-frame as one unit will reduce the decoding delay caused by the birectional prediction, critical to the bidirectional interactive videophone application. Therefore, in a PB-frame, a macroblock comprises 12 blocks instead of 6, of which 6 are for the P-picture and the other 6 blocks for the B-picture. Similar to MPEG-2, the prediction of the B-block requires forward and backward predictions. Pixels not predicted bidirectionally are predicted with forward prediction only.

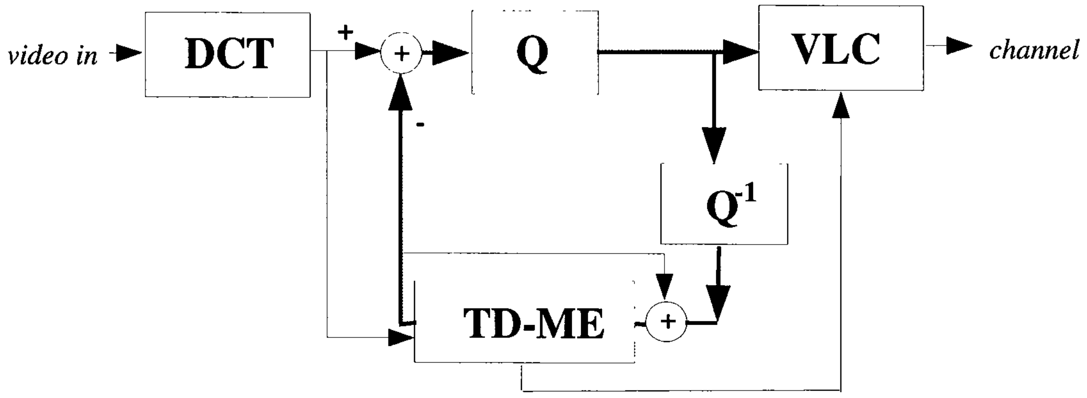
3.2 Fully DCT-Based Motion-Compensated Video Coder Structure

In most international video coding standards such as CCITT H.261 [21], MPEG1 [54], MPEG2 [55] as well as the proposed HDTV standard, Discrete Cosine Transform (DCT) and block-based motion estimation are the essential elements to achieve spatial and temporal compression, respectively. Most implementations of a standard-compliant coder adopt the conventional motion-compensated DCT video coder structure as shown in Fig. 3.4(a). The feedback loop for temporal prediction consists of a DCT, an Inverse DCT (IDCT) and a spatial-domain motion estimator (SD-ME) which is usually the full search block matching approach (BKM). This is undesirable. In addition to the additional complexity added to the overall architecture, this feedback loop limits the throughput of the coder and becomes the bottleneck of a real-time high-end video codec. A compromise is to remove the loop and perform open-loop motion estimation based upon original images instead of recontructed images in sacrifice of the performance of the coder [52, 46].

The presence of the IDCT block inside the feedback loop of the conventional video coder design comes from the fact that currently available motion estimation algorithms can only estimate motion in the spatial domain rather than directly in the DCT domain. Therefore, developing a transform-domain motion estimation algorithm will be able to eliminate this IDCT. Furthermore, the DCT block in the feedback loop is used to compute the DCT coefficients of motion compensated residuals. However, for motion compensation in the DCT domain, this DCT block can be moved out of the feedback loop. From these two observations, an alternative solution without degradation of the performance is to develop motion estimation and compensation algorithms which can work in the DCT domain. In this way, the DCT can be moved out of the loop as depicted in Fig. 3.4(b) and thus the operating speed of this DCT can be reduced to the data rate of the incoming stream. Moreover, the IDCT is removed from the feedback loop which now has only two simple components Q and Q^{-1} (the quantizers) in addition to the transform-domain motion estimator (TD-ME). This not only reduces the complexity of the coder but also resolves the bottleneck problem without any tradeoff of performance. Furthermore, as pointed out in [46], different components can be jointly optimized if they operate in the same transform domain. It should be stressed that by using DCT-based estimation and compensation methods, standard-compliant bit streams can be formed in accordance to the specification of any standard such as MPEG without any need to change the structure of any standard-compliant decoder. To realize this fully DCT-based video coder architecture to boost the system throughput and reduce the total number of components, we develop in this thesis DCT-based algorithms which perform motion estimation and compensation directly on the DCT coefficients of video frames.



(a) Conventional hybrid motion-compensated DCT video coder



(b) Fully DCT-based motion-compensated video coder

Figure 3.4: Different motion-compensated DCT video coder structures: (a) motion estimation/compensation are performed in the spatial domain; (b) motion estimation/compensation are completed in the transform (DCT) domain.

Chapter 4

DCT Pseudo Phase Techniques

In recent years, great interest has been taken in motion estimation from two 2-D signals or a sequence of images due to its various promising applications [3] in computer vision, image registration, target tracking, video coding with application to high definition television (HDTV), multimedia, video telephony, etc. Extensive research has been done over many years in developing new algorithms [3, 56, 15] and designing cost-effective and massively parallel hardware architectures [83, 37, 75, 32] suitable for current VLSI technology. Similar interests are also found in estimation of shift for the case of one-dimensional signals, a common problem in many areas of signal processing such as time delay estimation [1, 27], and optical displacement measurement [17]. As a matter of fact, shift estimation for 1-D signals and translational motion estimation for 2-D images inherently address the same problem and can use similar techniques.

For video coding applications, the most commonly used motion estimation scheme is the Full Search Block Matching Algorithm (BKM-ME) which searches for the best candidate block among all the blocks in a search area of larger size in terms of either the mean-square error (MSE) [29] or the mean of the absolute frame difference (MAD) [35]. The computational complexity of this approach is very high, i.e. $O(N^4)$ for a $N \times N$ block due to the need of sliding the reference block over the whole search area. Even so, BKM-ME has been successfully implemented on VLSI chips [83, 37, 75]. To reduce the number of computations, a number of suboptimal fast block matching algorithms have been proposed [29, 35, 70, 18, 48, 47]. However, these algorithms require three or more sequential steps to find suboptimal estimates. Recently a correlation-based approach (CLT-ME) [85] using the Complex Lapped Transform (CLT) to avoid the blocking effect was proposed but it still requires searching over a larger search area and thus results in a very high computational burden. Moreover, motion estimation using the CLT-ME is accurate on moving sharp edges but not on blur edges.

In addition to block-based approaches, pel-based estimation methods such as the Pel-Recursive Algorithm (PRA-ME) [57, 64] and Optical Flow Approach (OFA-ME) [68] are very vulnerable to noise by virtue of involving only local operations and may suffer from instability problems.

For the category of transform-domain motion estimation algorithms, the FFT phase correlation method was first proposed by Kuglin and Hines [39] and then further investigated by Thomas [72] and Girod [19]. This FFT approach utilizes correlation of FFT coefficients to estimate shifts between two images from the FFT phase shifts. However, the fast Fourier transform operates on complex numbers and is not used in most video standards. Furthermore, correlation multiplies any distortion already present in the FFT of signals. For multiframe motion detection, 3D-FFT has been successfully used to estimate motion in several consecutive frames [62, 36], based on the phenomenon that the spatial and temporal frequencies of a moving object lie on a plane of spatiotemporal space [23]. This requires processing of several frames rather than two.

In this chapter, we present new techniques called the DCT pseudo-phase techniques applicable to delay estimation for 1-D signals or motion estimation for 2-D images. Unlike other fast block search motion estimation methods (such as logarithmic, three-step search, cross, subsampled methods, etc) which simply pick several displacement candidates out of all possible displacement values in terms of minimum MAD values of a reduced number of pixels, these techniques employ the sinusoidal orthogonal principles to extract shift or displacement information from the pseudo phases hidden in the discrete cosine transform (DCT) coefficients of signals/images without doing any correlation. Under the 2-D translational motion model, the techniques result in the DCT-Based Motion Estimation (DXT-ME) algorithm, a novel algorithm for motion estimation to estimate displacements in the DCT domain. Even though this proposed algorithm is equally applicable to other motion detection/estimation scenarios such as target tracking, image registration, etc., in application to video coding, this algorithm has certain merits over conventional methods. In addition to low computational complexity (on the order of N^2 compared to N^4 for BKM-ME) and robustness of the DCT pseudo phase techniques, this algorithm takes DCT coefficients of images as input to estimate motions and therefore can be incorporated efficiently with the DCT-based coders used for most current video compression standards as the Fully DCT-Based Video Coder structure. This combines both the DCT and motion estimation into a single component to further reduce the coder complexity and at the same time increases the system throughput as explained in detail in Chapter 3.2. Finally, due to the fact that the computation of pseudo phases involves only highly local operations, a highly parallel pipelined architecture for this algorithm is possible.

In the next section, we introduce the DCT pseudo phase techniques with application to estimation of shift between 1-D signals. In Section 4.3, we consider the 2-D translation motion model and extend the DCT pseudo phase techniques to the DXT-ME algorithm for two dimensional signals and images. Properties of the algorithm are also discussed. In application to translational image registration at a very low SNR level (10 dB and even down to 0 dB), simulation shows that the DXT-ME algorithm can estimate accurately motion from entire noisy images while the full search block matching algorithm fails in certain noisy blocks. Finally, this chapter is concluded in Section 4.8.

4.1 DCT Pseudo-Phase Techniques

As well known, Fourier transform (FT) of a signal, $x(t)$ is related to FT of its shifted (or delayed if t represents time) version, $x(t-\tau)$, by this equation:

$$\mathcal{F}\{x(t-\tau)\} = e^{-j\omega\tau}\mathcal{F}\{x(t)\}, \tag{4.1}$$

where $\mathcal{F}\{\cdot\}$ denotes Fourier transform. The phase of Fourier transform of the shifted signal contains the information about the amount of the shift τ, which can easily be extracted. However, the Discrete Cosine Transform (DCT) or its counterpart, the Discrete Sine Transform (DST), does not have any phase components as usually found in discrete Fourier transform (DFT), but DCT (or DST) coefficients of a shifted signal do also carry this shift information. To facilitate explanation of the DCT pseudo phase techniques, let us first consider the case of one-dimensional discrete signals. Suppose that the signal $\{x_1(n);\ n \in \{0,\dots,N-1\}\}$ is right shifted by an amount m (in our convention, a right shift means that $m > 0$) to generate another signal $\{x_2(n);\ n \in \{0,\dots,N-1\}\}$. The values of $x_1(n)$ are all zeros outside the support region $\mathcal{S}(x_1)$. Therefore,

$$x_2(n) = \begin{cases} x_1(n-m), & \text{for } n-m \in \mathcal{S}(x_1), \\ 0, & \text{elsewhere.} \end{cases}$$

The above equation implies that both signals have resemblance to each other except that the signal is shifted. It can be shown that, for $k = 1, \ldots, N-1$,

$$X_2^C(k) = Z_1^C(k)\cos[\frac{k\pi}{N}(m+\frac{1}{2})] - Z_1^S(k)\sin[\frac{k\pi}{N}(m+\frac{1}{2})], \qquad (4.2)$$

$$X_2^S(k) = Z_1^S(k)\cos[\frac{k\pi}{N}(m+\frac{1}{2})] + Z_1^C(k)\sin[\frac{k\pi}{N}(m+\frac{1}{2})]. \qquad (4.3)$$

Here X_2^S and X_2^C are DST (DST-II) and DCT (DCT-II) of the second kind of $x_2(n)$, respectively, whereas Z_1^S and Z_1^C are DST (DST-I) and DCT (DCT-I) of the first kind of $x_1(n)$, respectively, as defined as follows [84] :

$$X_2^C(k) = \frac{2}{N}C(k)\sum_{n=0}^{N-1} x_2(n)\cos[\frac{k\pi}{N}(n+0.5)];\ k \in \{0,\ldots,N-1\}, \qquad (4.4)$$

$$X_2^S(k) = \frac{2}{N}C(k)\sum_{n=0}^{N-1} x_2(n)\sin[\frac{k\pi}{N}(n+0.5)];\ k \in \{1,\ldots,N\}, \qquad (4.5)$$

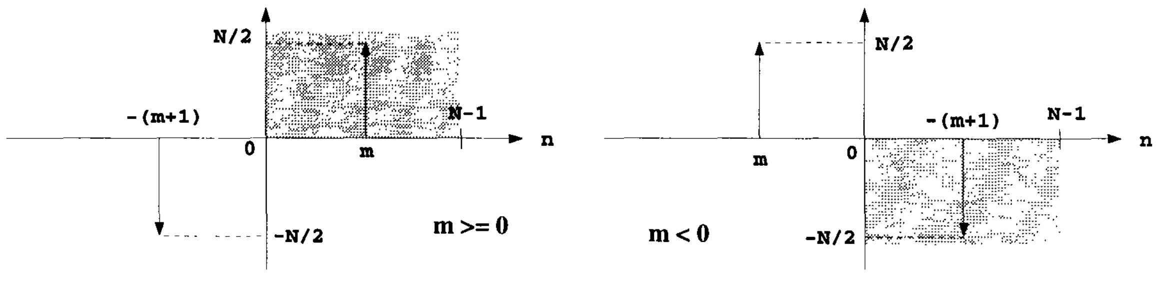
$$Z_1^C(k) = \frac{2}{N}C(k)\sum_{n=0}^{N-1} x_1(n)\cos[\frac{k\pi}{N}(n)];\ k \in \{0,\ldots,N\}, \qquad (4.6)$$

$$Z_1^S(k) = \frac{2}{N}C(k)\sum_{n=0}^{N-1} x_1(n)\sin[\frac{k\pi}{N}(n)];\ k \in \{1,\ldots,N-1\}, \qquad (4.7)$$

$$\text{where } C(k) = \begin{cases} \frac{1}{\sqrt{2}}, & \text{for } k=0 \text{ or } N, \\ 1, & \text{otherwise,} \end{cases}$$

The displacement, m, is embedded solely in the terms $g_m^s(k) = \sin[\frac{k\pi}{N}(m+\frac{1}{2})]$ and $g_m^c(k) = \cos[\frac{k\pi}{N}(m+\frac{1}{2})]$, which are called *pseudo phases* analogous to phases in Fourier transform of shifted signals. To find m, we first solve (4.2) and (4.3) for the pseudo phases and then use the sinusoidal orthogonal principles as follows:

$$\frac{2}{N}\sum_{k=1}^{N} C^2(k)\sin[\frac{k\pi}{N}(m+\frac{1}{2})]\sin[\frac{k\pi}{N}(n+\frac{1}{2})] = \delta(m-n) - \delta(m+n+1), (4.8)$$



(a) How to detect right shift (b) How to detect left shift

Figure 4.1: How the direction of shift is determined based on the sign of the peak value after application of the sinusoidal orthogonal principle for the DST-II kernel to pseudo phases

$$\frac{2}{N}\sum_{k=0}^{N-1} C^2(k)\cos[\frac{k\pi}{N}(m+\frac{1}{2})]\cos[\frac{k\pi}{N}(n+\frac{1}{2})] = \delta(m-n)+\delta(m+n+1). \tag{4.9}$$

Here $\delta(n)$ is the discrete impulse function defined as

$$\delta(n) = \begin{cases} 1, & \text{for } n = 0, \\ 0, & \text{otherwise.} \end{cases} \tag{4.10}$$

Indeed, if we replace $\sin[\frac{k\pi}{N}(m+\frac{1}{2})]$ and $\cos[\frac{k\pi}{N}(m+\frac{1}{2})]$ by the computed sine and cosine pseudo phase components, $\hat{g}^s_m(k)$ and $\hat{g}^c_m(k)$, respectively in (4.8) and (4.9), both equations simply become IDST-II and IDCT-II operations on $\hat{g}^s_m(k)$ and $\hat{g}^c_m(k)$:

$$IDSTII(\hat{g}^s_m) = \frac{2}{N}\sum_{k=1}^{N} C^2(k)\hat{g}^s_m(k)\sin[\frac{k\pi}{N}(n+\frac{1}{2})], \tag{4.11}$$

$$IDCTII(\hat{g}^c_m) = \frac{2}{N}\sum_{k=0}^{N-1} C^2(k)\hat{g}^c_m(k)\cos[\frac{k\pi}{N}(n+\frac{1}{2})]. \tag{4.12}$$

The notation $\hat{g}$ is used to distinguish the computed pseudo phase from the one in a noiseless situation (i.e. $\sin[\frac{k\pi}{N}(m+\frac{1}{2})]$ or $\cos[\frac{k\pi}{N}(m+\frac{1}{2})]$). A closer look at the right-hand side of (4.8) tells us that $\delta(m-n)$ and $\delta(m+n+1)$ have opposite signs.

This property will help us detect the shift direction. If we perform an IDST-II operation on the pseudo phases found, then the observable window of the index space in the inverse DST domain will be limited to $\{0, \ldots, N-1\}$. As illustrated in Fig. 4.1, for a right shift, one spike (generated by the positive δ function) is pointing upwards at the location $n = m$ in the gray region (i.e. the observable index space), while the other δ pointing downwards at $n = -(m+1)$ outside the gray region. In contrast, for a left shift, the negative spike at $n = -(m+1) > 0$ falls in the gray region but the positive δ function at $n = m$ stays out of the observable index space. It can easily be seen that a positive peak value in the gray region implies a right shift and a negative one means a left shift. This enables us to determine from the sign of the peak value the direction of the shift between signals.

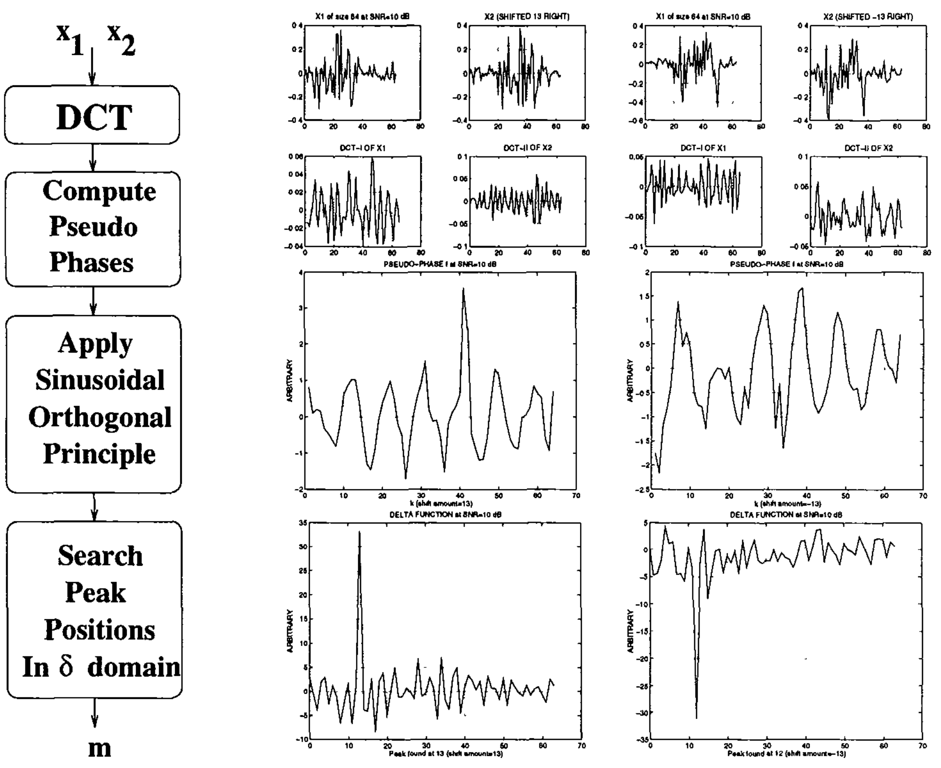
The concept of pseudo phases plus the application of sinusoidal orthogonal principles leads to the DCT pseudo phase techniques, a new approach to estimate a shift or translational motion between signals in DCT domain as depicted in Fig. 4.2 (a):

1. Compute the DCT-I and DST-I coefficients of $x_1(n)$ and the DCT-II and DST-II coefficients of $x_2(n)$.

2. Compute the pseudo phase $\hat{g}^s_m(k)$ for $k = 1, \ldots, N$ by solving this equation:

$$\hat{g}^s_m(k) = \begin{cases} \frac{Z_1^C(k)\cdot X_2^S(k) - Z_1^S(k)\cdot X_2^C(k)}{[Z_1^C(k)]^2 + [Z_1^S(k)]^2}, & \text{for } k \neq N, \\ \frac{1}{\sqrt{2}}, & \text{for } k = N. \end{cases} \tag{4.13}$$

3. Feed the computed pseudo phase, $\{\hat{g}^s_m(k);\ k = 1, \ldots, N\}$, into an IDST-II decoder to produce an output $\{d(n);\ n = 0, \ldots, N-1\}$, and search for the



(a) DCT Pseudo Phase Techniques (b) Right Shift (c) Left Shift

Figure 4.2: Illustration of one dimensional DCT pseudo phase techniques.

peak value. Then the estimated displacement $\hat{m}$ can be found by

$$\hat{m} = \begin{cases} i_p, & \text{if } d(i_p) > 0, \\ -(i_p + 1), & \text{if } d(i_p) < 0, \end{cases} \tag{4.14}$$

where $i_p = arg\max_n |d(n)|$ is the index at which the peak value is located.

In Step 1, the DCT and DST can be generated simultaneously with only $3N$ multipliers [12, 49, 50], and the computation of DCT-I can be easily obtained from DCT-II with minimal overhead as will be shown later. In Step 2, if noise is absent and there is only purely translational motion, $\hat{g}_m(k)$ will be equal to $\sin \frac{k\pi}{N}(m+0.5)$. The output $d(n)$ will then be an impulse function in the observation window. This

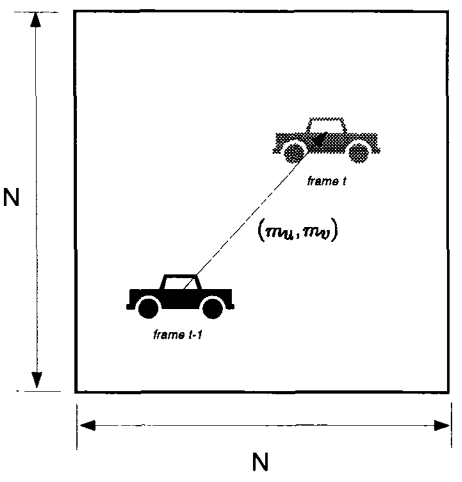


Figure 4.3: An object moves translationally by m_u in X direction and m_v in Y direction as viewed on the camera plane.

procedure is illustrated by two examples in Fig. 4.2(b) and (c) with a randomly generated signal as input at $SNR = 10$ dB. These two examples demonstrate that the DCT pseudo phase techniques are robust even in an environment of strong noise.

4.2 2-D Translational Motion Model

The DCT pseudo phase technique of extracting shift values from the pseudo phases of DCT of one dimensional signals, as explained in Section 4.1, can be extended to the two-dimensional case. Let us confine the problem of motion estimation to this 2D translational motion model in which an object moves translationally by m_u in X direction and m_v in Y direction as viewed on the camera plane and within the scope of a camera in a noiseless environment as shown in Fig. 4.3. Then by means of the DCT pseudo phase technique, we can extract the displacement vector out of the two consecutive frames of the images of that moving object by making use of the sinusoidal orthogonal principles (4.8) and (4.9). The resulted novel algorithm for this two dimensional translational motion model is called the DXT-ME algorithm which can estimate translational motion in the DCT domain.

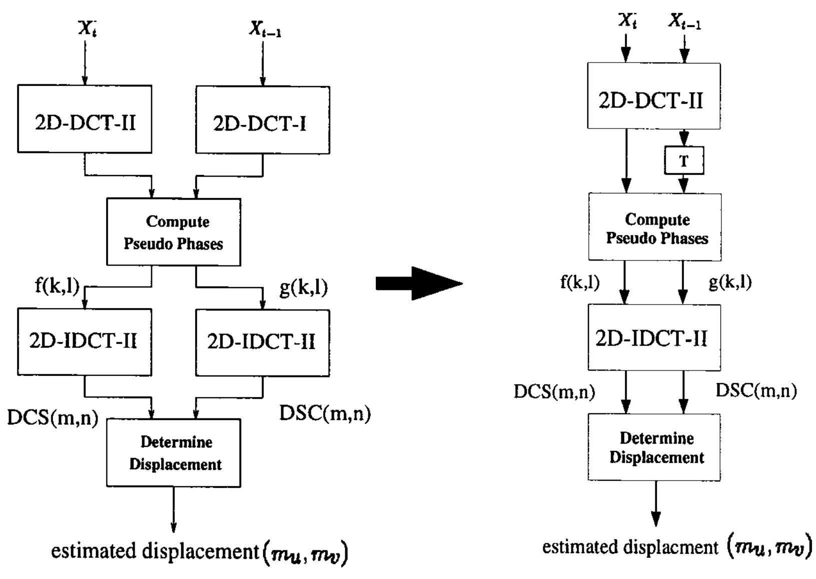
4.3 The DXT-ME Algorithm

Based on the assumption of 2D translational displacements, we can extend the DCT pseudo phase technique to the DXT-ME algorithm depicted in Fig. 4.4. The previous frame x_{t-1} and the current frame x_t are fed into 2D-DCT-II and 2D-DCT-I coders respectively. A 2D-DCT-II coder computes four coefficients, DCCTII, DCSTII, DSCTII, and DSSTII, each of which is defined as a two-dimensional separable function formed by 1D-DCT/DST-II kernels:

$$\begin{aligned}
X_t^{cc}(k,l) &= \frac{4}{N^2}C(k)C(l)\sum_{m,n=0}^{N-1} x_t(m,n)\cos[\frac{k\pi}{N}(m+0.5)]\cos[\frac{l\pi}{N}(n+0.5)], && (4.15)\\
&\text{for } k,l \in \{0,\dots,N-1\},\\
X_t^{cs}(k,l) &= \frac{4}{N^2}C(k)C(l)\sum_{m,n=0}^{N-1} x_t(m,n)\cos[\frac{k\pi}{N}(m+0.5)]\sin[\frac{l\pi}{N}(n+0.5)], && (4.16)\\
&\text{for } k \in \{0,\dots,N-1\}, l \in \{1,\dots,N\},\\
X_t^{sc}(k,l) &= \frac{4}{N^2}C(k)C(l)\sum_{m,n=0}^{N-1} x_t(m,n)\sin[\frac{k\pi}{N}(m+0.5)]\cos[\frac{l\pi}{N}(n+0.5)], && (4.17)\\
&\text{for } k \in \{1,\dots,N\}, l \in \{0,\dots,N-1\},\\
X_t^{ss}(k,l) &= \frac{4}{N^2}C(k)C(l)\sum_{m,n=0}^{N-1} x_t(m,n)\sin[\frac{k\pi}{N}(m+0.5)]\sin[\frac{l\pi}{N}(n+0.5)], && (4.18)\\
&\text{for } k,l \in \{1,\dots,N\},
\end{aligned}$$

or symbolically,

$$X_t^{cc} = DCCTII(x_t),\ X_t^{cs} = DCSTII(x_t),$$
$$X_t^{sc} = DSCTII(x_t),\ X_t^{ss} = DSSTII(x_t).$$



(a) flowchart (b) structure

Figure 4.4: Block diagram of DXT-ME

In the same fashion, the two dimensional DCT coefficients of the first kind (2D-DCT-I) are calculated based on 1D-DCT/DST-I kernels:

$$\begin{aligned}
Z_{t-1}^{cc}(k,l) &= \frac{4}{N^2}C(k)C(l)\sum_{m,n=0}^{N-1} x_{t-1}(m,n)\cos[\frac{k\pi}{N}(m)]\cos[\frac{l\pi}{N}(n)], && (4.19)\\
&\text{for } k,l \in \{0,\ldots,N\},\\
Z_{t-1}^{cs}(k,l) &= \frac{4}{N^2}C(k)C(l)\sum_{m,n=0}^{N-1} x_{t-1}(m,n)\cos[\frac{k\pi}{N}(m)]\sin[\frac{l\pi}{N}(n)], && (4.20)\\
&\text{for } k \in \{0,\ldots,N\}, l \in \{1,\ldots,N-1\},\\
Z_{t-1}^{sc}(k,l) &= \frac{4}{N^2}C(k)C(l)\sum_{m,n=0}^{N-1} x_{t-1}(m,n)\sin[\frac{k\pi}{N}(m)]\cos[\frac{l\pi}{N}(n)], && (4.21)\\
&\text{for } k \in \{1,\ldots,N-1\}, l \in \{0,\ldots,N\},\\
Z_{t-1}^{ss}(k,l) &= \frac{4}{N^2}C(k)C(l)\sum_{m,n=0}^{N-1} x_{t-1}(m,n)\sin[\frac{k\pi}{N}(m)]\sin[\frac{l\pi}{N}(n)], && (4.22)\\
&\text{for } k,l \in \{1,\ldots,N-1\},
\end{aligned}$$

or symbolically,

$$Z_{t-1}^{cc} = DCCTI(x_{t-1}),\ Z_{t-1}^{cs} = DCSTI(x_{t-1}),$$

$$Z^{sc}_{t-1} = DSCTI(x_{t-1}),\ Z^{ss}_{t-1} = DSSTI(x_{t-1}).$$

Similar to one dimensional case, assuming that only translational motion is allowed, one can derive a set of equations to relate DCT coefficients of $x_{t-1}(m,n)$ with those of $x_t(m,n)$ in the same way as in (4.2) and (4.3).

$$\mathbf{Z}_{t-1}(k,l) \cdot \vec{\theta}(k,l) = \vec{\mathbf{x}}_t(k,l), \text{ for } k,l \in \mathcal{N}, \tag{4.23}$$

where $\mathcal{N} = \{1,\ldots,N-1\}$,

$$\mathbf{Z}_{t-1}(k,l) = \begin{bmatrix} Z^{cc}_{t-1}(k,l) & -Z^{cs}_{t-1}(k,l) & -Z^{sc}_{t-1}(k,l) & Z^{ss}_{t-1}(k,l) \\ Z^{cs}_{t-1}(k,l) & Z^{cc}_{t-1}(k,l) & -Z^{ss}_{t-1}(k,l) & -Z^{sc}_{t-1}(k,l) \\ Z^{sc}_{t-1}(k,l) & -Z^{ss}_{t-1}(k,l) & Z^{cc}_{t-1}(k,l) & -Z^{cs}_{t-1}(k,l) \\ Z^{ss}_{t-1}(k,l) & Z^{sc}_{t-1}(k,l) & Z^{cs}_{t-1}(k,l) & Z^{cc}_{t-1}(k,l) \end{bmatrix}, \tag{4.24}$$

$$\vec{\theta}(k,l) = \begin{bmatrix} g^{CC}_{m_u m_v}(k,l) \\ g^{CS}_{m_u m_v}(k,l) \\ g^{SC}_{m_u m_v}(k,l) \\ g^{SS}_{m_u m_v}(k,l) \end{bmatrix} = \begin{bmatrix} \cos\frac{k\pi}{N}(m_u+0.5)\cos\frac{l\pi}{N}(m_v+0.5) \\ \cos\frac{k\pi}{N}(m_u+0.5)\sin\frac{l\pi}{N}(m_v+0.5) \\ \sin\frac{k\pi}{N}(m_u+0.5)\cos\frac{l\pi}{N}(m_v+0.5) \\ \sin\frac{k\pi}{N}(m_u+0.5)\sin\frac{l\pi}{N}(m_v+0.5) \end{bmatrix}, \tag{4.25}$$

$$\vec{\mathbf{x}}_t(k,l) = \begin{bmatrix} X^{cc}_t(k,l) & X^{cs}_t(k,l) & X^{sc}_t(k,l) & X^{ss}_t(k,l) \end{bmatrix}^T. \tag{4.26}$$

Here $\mathbf{Z}_{t-1}(k,l) \in R^{4\times 4}$ is the *system matrix* of the DXT-ME algorithm at (k,l). At the boundaries of each block in the transform domain, the DCT coefficients of $x_{t-1}(m,n)$ and $x_t(m,n)$ have one dimensional relationship as given below:

$$\begin{bmatrix} Z^{cc}_{t-1}(k,l) & -Z^{cs}_{t-1}(k,l) \\ Z^{cs}_{t-1}(k,l) & Z^{cc}_{t-1}(k,l) \end{bmatrix} \begin{bmatrix} \cos\frac{l\pi}{N}(m_v+\frac{1}{2}) \\ \sin\frac{l\pi}{N}(m_v+\frac{1}{2}) \end{bmatrix} = \begin{bmatrix} X^{cc}_t(k,l) \\ X^{cs}_t(k,l) \end{bmatrix} \tag{4.27}$$

for $k = 0,\ l \in \mathcal{N}$,

$$\begin{bmatrix} Z^{cc}_{t-1}(k,l) & -Z^{sc}_{t-1}(k,l) \\ Z^{sc}_{t-1}(k,l) & Z^{cc}_{t-1}(k,l) \end{bmatrix} \begin{bmatrix} \cos\frac{k\pi}{N}(m_u+\frac{1}{2}) \\ \sin\frac{k\pi}{N}(m_u+\frac{1}{2}) \end{bmatrix} = \begin{bmatrix} X^{cc}_t(k,l) \\ X^{sc}_t(k,l) \end{bmatrix} \tag{4.28}$$

for $l = 0,\ k \in \mathcal{N}$,

$$\begin{bmatrix} Z_{t-1}^{cc}(k,l) & -Z_{t-1}^{cs}(k,l) \\ Z_{t-1}^{cs}(k,l) & Z_{t-1}^{cc}(k,l) \end{bmatrix} \begin{bmatrix} \cos\frac{l\pi}{N}(m_v+\frac{1}{2}) \\ \sin\frac{l\pi}{N}(m_v+\frac{1}{2}) \end{bmatrix} = (-1)^{m_u} \begin{bmatrix} X_t^{sc}(k,l) \\ X_t^{ss}(k,l) \end{bmatrix} \quad (4.29)$$

for $k = N,\ l \in \mathcal{N}$,

$$\begin{bmatrix} Z_{t-1}^{cc}(k,l) & -Z_{t-1}^{sc}(k,l) \\ Z_{t-1}^{sc}(k,l) & Z_{t-1}^{cc}(k,l) \end{bmatrix} \begin{bmatrix} \cos\frac{k\pi}{N}(m_u+\frac{1}{2}) \\ \sin\frac{k\pi}{N}(m_u+\frac{1}{2}) \end{bmatrix} = (-1)^{m_v} \begin{bmatrix} X_t^{cs}(k,l) \\ X_t^{ss}(k,l) \end{bmatrix} \quad (4.30)$$

for $l = N,\ k \in \mathcal{N}$,

$$(-1)^{m_v} Z_{t-1}^{cc}(k,l) = X_t^{cs}(k,l) \quad (4.31)$$

for $k = 0,\ l = N$,

$$(-1)^{m_u} Z_{t-1}^{cc}(k,l) = X_t^{sc}(k,l) \quad (4.32)$$

for $k = N,\ l = 0$.

In a two dimensional space, an object may move in four possible directions: northeast (NE: $m_u > 0,\ m_v > 0$), northwest (NW: $m_u < 0,\ m_v > 0$), southeast (SE: $m_u > 0,\ m_v < 0$), and southwest (SW: $m_u < 0,\ m_v < 0$). As explained in Section 4.1, the orthogonal equation for the DST-II kernel in (4.8) can be applied to the pseudo phase $\hat{g}_m^s(k)$ to determine the sign of m (i.e. the direction of the shift). In order to detect the signs of both m_u and m_v (or equivalently the direction of motion), it becomes obvious from the observation in the one dimensional case that it is necessary to compute the pseudo phases $\hat{g}_{m_u m_v}^{SC}(\cdot,\cdot)$ and $\hat{g}_{m_u m_v}^{CS}(\cdot,\cdot)$ so that the signs of m_u and m_v can be determined from $\hat{g}_{m_u m_v}^{SC}(\cdot,\cdot)$ and $\hat{g}_{m_u m_v}^{CS}(\cdot,\cdot)$, respectively. By taking the block boundary equations (4.27)–(4.32) into consideration, we define two pseudo phase functions as follows:

$$f_{m_u m_v}(k,l) = \begin{cases} \hat{g}^{CS}_{m_u m_v}(k,l), & \text{for } k,l \in \mathcal{N}, \\ \frac{1}{\sqrt{2}} \frac{Z^{cc}_{t-1}(k,l) X^{cs}_t(k,l) - Z^{cs}_{t-1}(k,l) X^{cc}_t(k,l)}{(Z^{cc}_{t-1}(k,l))^2 + (Z^{cs}_{t-1}(k,l))^2}, & \text{for } k=0, l \in \mathcal{N}, \\ \frac{1}{\sqrt{2}} \frac{Z^{cc}_{t-1}(k,l) X^{cs}_t(k,l) + Z^{sc}_{t-1}(k,l) X^{ss}_t(k,l)}{(Z^{cc}_{t-1}(k,l))^2 + (Z^{sc}_{t-1}(k,l))^2}, & \text{for } l=N, k \in \mathcal{N}, \\ \frac{1}{2} \frac{X^{cs}_t(k,l)}{Z^{cc}_{t-1}(k,l)}, & \text{for } k=0, l=N, \end{cases} \tag{4.33}$$

$$g_{m_u m_v}(k,l) = \begin{cases} \hat{g}^{SC}_{m_u m_v}(k,l), & \text{for } k,l \in \mathcal{N}, \\ \frac{1}{\sqrt{2}} \frac{Z^{cc}_{t-1}(k,l) X^{sc}_t(k,l) - Z^{sc}_{t-1}(k,l) X^{cc}_t(k,l)}{(Z^{cc}_{t-1}(k,l))^2 + (Z^{sc}_{t-1}(k,l))^2}, & \text{for } l=0, k \in \mathcal{N}, \\ \frac{1}{\sqrt{2}} \frac{Z^{cc}_{t-1}(k,l) X^{sc}_t(k,l) + Z^{cs}_{t-1}(k,l) X^{ss}_t(k,l)}{(Z^{cc}_{t-1}(k,l))^2 + (Z^{cs}_{t-1}(k,l))^2}, & \text{for } k=N, l \in \mathcal{N}, \\ \frac{1}{2} \frac{X^{sc}_t(k,l)}{Z^{cc}_{t-1}(k,l)}, & \text{for } k=N, l=0, \end{cases} \tag{4.34}$$

In the computation of $f_{m_u m_v}(k,l)$ and $g_{m_u m_v}(k,l)$, if the absolute computed value is greater than 1, then this value is ill-conditioned and should be discarded. This ill-conditioned situation occurs when the denominator in (4.33)-(4.34) is close to zero in comparison to the finite machine precision or set to zero after the quantization step in the feedback loop of the encoder as shown in Fig. 3.4. Due to the fact that neighboring image pixels are highly correlated, the high-frequency DCT coefficients of an image tend to be very small and can be regarded as zero after the quantization step but the low-frequency DCT components usually have large values. Therefore, the ill-conditioned situation happens more likely when k and l are both large. It is desirable to set the value of $f_{m_u m_v}(k,l)$ (or $g_{m_u m_v}(k,l)$) as close as possible to the ideal value of $f_{m_u m_v}(k,l)$ (or $g_{m_u m_v}(k,l)$) with the infinite machine precision and no quantization. Since ideally $f_{m_u m_v}(k,l) = \cos\frac{k\pi}{N}(m_u + \frac{1}{2}) \sin\frac{l\pi}{N}(m_v + \frac{1}{2})$,

$$\begin{aligned} f_{m_u m_v}(k,l) = & \frac{1}{2}\{\sin[\frac{l\pi}{N}(m_v + \frac{1}{2}) + \frac{k\pi}{N}(m_u + \frac{1}{2})] \\ & + \sin[\frac{l\pi}{N}(m_v + \frac{1}{2}) - \frac{k\pi}{N}(m_u + \frac{1}{2})]\}. \end{aligned}$$

For small values of m_u and m_v (slow motion) and large k and l (high-frequency DCT components), it is likely that

$$\frac{l\pi}{N}(m_v + \frac{1}{2}) - \frac{k\pi}{N}(m_u + \frac{1}{2}) \approx 0,$$

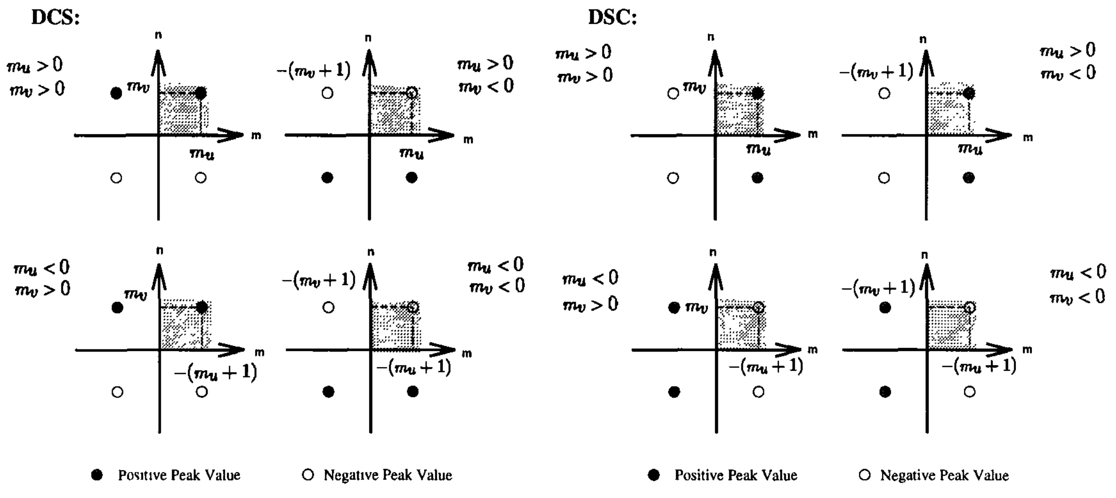
and the first term in $f_{m_u m_v}(k,l)$ is bounded by 1. Therefore, it is likely that $|f_{m_u m_v}(k,l)| \leq 0.5$. Without any other knowledge, it is reasonable to guess that $f_{m_u m_v}(k,l)$ is closer to zero than to ± 1. A similar argument follows for the case of $g_{m_u m_v}(k,l)$. Thus in our implementation, we set the corresponding variable $f_{m_u m_v}(k,l)$ or $g_{m_u m_v}(k,l)$ to be zero when the magnitudes of the computed values exceed 1. This setting for ill-conditioned computed $f_{m_u m_v}(k,l)$ and $g_{m_u m_v}(k,l)$ values is found to improve the condition of $f_{m_u m_v}(k,l)$ and $g_{m_u m_v}(k,l)$ and also the overall performance of the DXT-ME algorithm.

These two pseudo phase functions pass through 2D-IDCT-II coders (*IDCSTII* and *IDSCTII*) to generate two functions, $DCS(\cdot,\cdot)$ and $DSC(\cdot,\cdot)$ in view of the orthogonal property of DCT-II and DST-II in (4.8) and (4.9):

$$\begin{aligned} & DCS(m,n) = IDCSTII(f_{m_u m_v}) \\ &= \frac{4}{N^2}\sum_{k=0}^{N-1}\sum_{l=1}^{N} C(k)C(l) f_{m_u m_v}(k,l) \cos\frac{k\pi}{N}(m+\frac{1}{2}) \sin\frac{l\pi}{N}(n+\frac{1}{2}) \\ &= [\delta(m-m_u)+\delta(m+m_u+1)]\cdot[\delta(n-m_v)-\delta(n+m_v+1)], \end{aligned} \tag{4.35}$$

$$\begin{aligned} & DSC(m,n) = IDSCTII(g_{m_u m_v}) \\ &= \frac{4}{N^2}\sum_{k=1}^{N}\sum_{l=0}^{N-1} C(k)C(l) g_{m_u m_v}(k,l) \sin\frac{k\pi}{N}(m+\frac{1}{2}) \cos\frac{l\pi}{N}(n+\frac{1}{2}) \\ &= [\delta(m-m_u)-\delta(m+m_u+1)]\cdot[\delta(n-m_v)+\delta(n+m_v+1)]. \end{aligned} \tag{4.36}$$

By the same argument as in one dimensional case, the 2D-IDCT-II coders limit the observable index space $\{(i,j) : i,j = 0,\ldots,N-1\}$ of DCS and DSC to the



(a) from DCS (b) from DSC

Figure 4.5: How the direction of motion is determined based on the sign of the peak value

first quadrant of the entire index space shown as gray regions in Fig. 4.5 which depicts (4.35) and (4.36). Similar to one dimensional case, if m_u is positive, the observable peak value of $DSC(m, n)$ will be positive regardless of the sign of m_v since $DSC(m, n) = \delta(m - m_u) \cdot [\delta(n - m_v) + \delta(n + m_v + 1)]$ in the observable index space. Likewise, if m_u is negative, the observable peak value of $DSC(m, n)$ will be negative because $DSC(m, n) = \delta(m + m_u + 1) \cdot [\delta(n - m_v) + \delta(n + m_v + 1)]$ in the gray region. As a result, the sign of the observable peak value of DSC determines the sign of m_u. The same reasoning may apply to DCS in the determination of the sign of m_v. The estimated displacement, $\hat{d} = (\hat{m}_u, \hat{m}_v)$, can thus be found by locating the peaks of DCS and DSC over $\{0, \ldots, N - 1\}^2$ or over an index range of interest, usually, $\Phi = \{0, \ldots, N/2\}^2$ for slow motion. How the peak signs determine the direction of movement is summarized in Table 4.1. Once the

Sign of DSC Peak	**Sign of DCS Peak**	**Peak Index**	**Motion Direction**
$+$	$+$	(m_u, m_v)	northeast
$+$	$-$	$(m_u, -(m_v+1))$	southeast
$-$	$+$	$(-(m_u+1), m_v)$	northwest
$-$	$-$	$(-(m_u+1), -(m_v+1))$	southwest

Table 4.1: Determination of direction of movement (m_u, m_v) from the signs of DSC and DCS

direction is found, $\hat{d}$ can be estimated accordingly:

$$\hat{m}_u = \begin{cases} i_{DSC} = i_{DCS}, & \text{if } DSC(i_{DSC}, j_{DSC}) > 0, \\ -(i_{DSC}+1) = -(i_{DCS}+1), & \text{if } DSC(i_{DSC}, j_{DSC}) < 0, \end{cases} \tag{4.37}$$

$$\hat{m}_v = \begin{cases} j_{DCS} = j_{DSC}, & \text{if } DCS(i_{DCS}, j_{DCS}) > 0, \\ -(j_{DCS}+1) = -(j_{DSC}+1), & \text{if } DCS(i_{DCS}, j_{DCS}) < 0, \end{cases} \tag{4.38}$$

where

$$(i_{DCS}, j_{DCS}) = arg \max_{m,n\in\Phi} |DCS(m,n)|, \tag{4.39}$$

$$(i_{DSC}, j_{DSC}) = arg \max_{m,n\in\Phi} |DSC(m,n)|. \tag{4.40}$$

Normally, these two peak indices are consistent but in noisy circumstances, they may not agree. In this case, an arbitration rule must be made to pick the best index (i_D, j_D) in terms of minimum nonpeak-to-peak ratio (NPR):

$$(i_D, j_D) = \begin{cases} (i_{DSC}, j_{DSC}) & \text{if } NPR(DSC) < NPR(DCS), \\ (i_{DCS}, j_{DCS}) & \text{if } NPR(DSC) > NPR(DCS). \end{cases} \tag{4.41}$$

This index (i_D, j_D) will then be used to determine $\hat{d}$ by (4.37) and (4.38). Here NPR is defined as the ratio of the average of all absolute non-peak values to the absolute peak value. Thus $0 \leq NPR \leq 1$, and for a pure impulse function, $NPR = 0$. Such an approach to choose the best index among the two indices is found empirically to improve the noise immunity of this estimation algorithm.

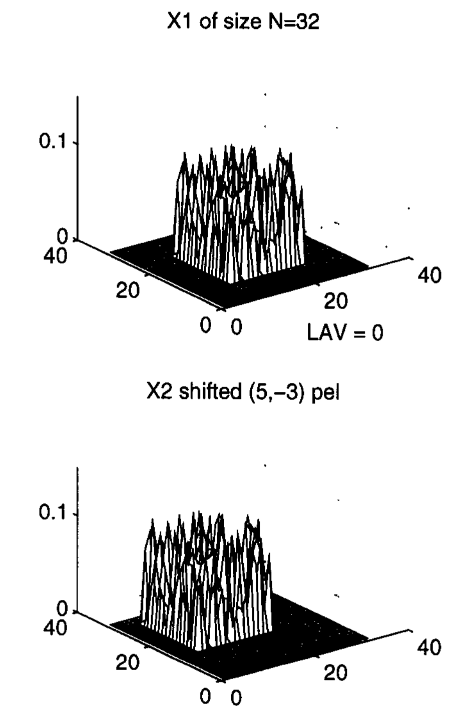
In situations where slow motion is preferred, it is better to search the peak value in a zigzag way as widely used in DCT-based hybrid video coding [21, 54]. Starting from the index $(0,0)$, zigzagly scan all the DCS (or DSC) values and mark the point as the new peak index if the value at that point (i,j) is larger than the current peak value by more than a preset threshold θ:

$$(i_{DCS}, j_{DCS}) = (i,j) \text{ if } DCS(i,j) > DCS(i_{DCS}, j_{DCS}) + \theta, \tag{4.42}$$

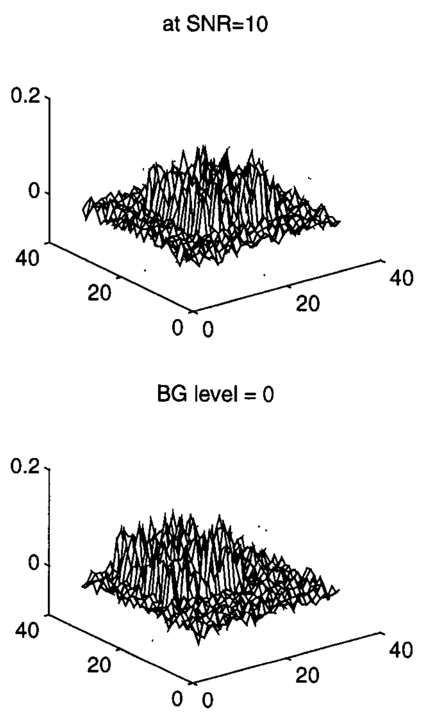
$$(i_{DSC}, j_{DSC}) = (i,j) \text{ if } DSC(i,j) > DSC(i_{DSC}, j_{DSC}) + \theta. \tag{4.43}$$

In this way, large spurious spikes at the higher index points will not affect the performance and thus improve its noise immunity further.

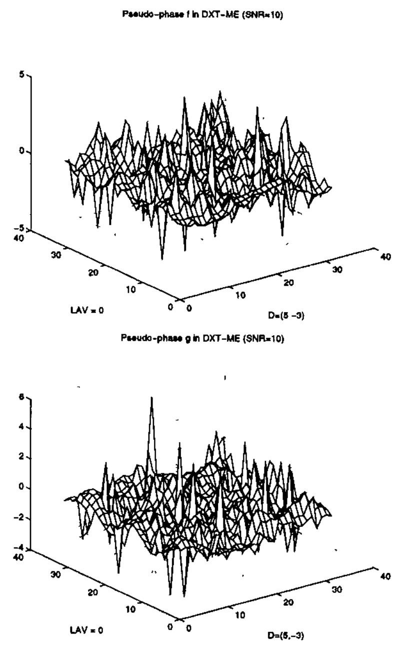
Fig. 4.6 demonstrates the DXT-ME algorithm. Images of a rectangularly-shaped moving object with arbitrary texture are generated as in Fig. 4.6(a) and corrupted by additive white Gaussian noise at SNR = 10 dB as in Fig. 4.6(b). The resulted pseudo phase functions f and g, as well as DCS and DSC, are depicted in Fig. 4.6 (c) and (d) correspondingly. Large peaks can be seen clearly in Fig. 4.6(d) on rough surfaces caused by noise in spite of noisy input images. The positions of these peaks give us an accurate motion estimate $(5, -3)$. The DXT-ME algorithm is summarized in Table 4.2.



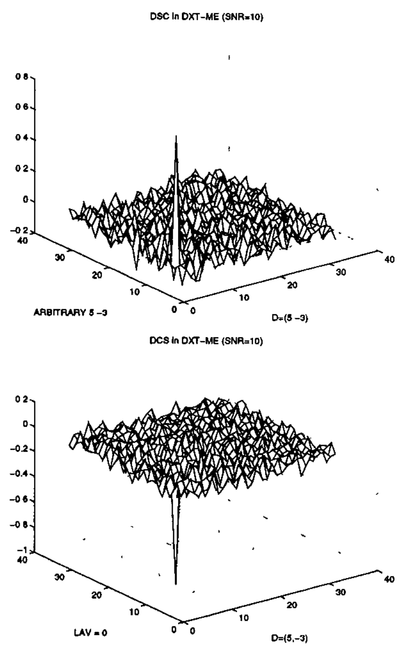
(a) original inputs x_1 and x_2



(b) noise added



(c) f and g



(d) *DSC* and *DCS*

Figure 4.6: DXT-ME performed on the images of an object moving in the direction (5, -3) with additive white Gaussian noise at SNR = 10 dB

1. Compute the 2-D DCT coefficients of second kind (2D-DCT-II) of a $N \times N$ block of pixels at the current frame t, $\{x_t(m,n); m,n \in \{0,...,N-1\}\}$.
2. Convert the stored 2D-DCT-II coefficients of the corresponding $N \times N$ block of pixels at the previous frame $t-1$, $\{x_{t-1}(m,n); m,n \in \{0,...,N-1\}\}$ to 2D DCT coefficients of first kind (2D-DCT-I) through a simple rotation unit T.
3. Find the pseudo phases $\{g^{CS}(k,l);\ k=0,1,\ldots,N-1;\ l=1,2,\ldots,N\}$ and $\{g^{SC}(k,l);\ k=1,2,\ldots,N;\ l=0,1,\ldots,N-1\}$, which are calculated from the DCT coefficients independently at each spectral location (k,l).
4. Determine the normalized pseudo phases $f(k,l)$ and $g(k,l)$ from $g^{CS}(k,l)$ and $g^{SC}(k,l)$ respectively by setting ill-formed $g^{CS}(k,l)$ and $g^{SC}(k,l)$ to zero:
$$f(k,l) = \begin{cases} C(k)C(l)g^{CS}(k,l), & \text{for } |g^{CS}(k,l)| \leq 1, \\ 0, & \text{otherwise,} \end{cases}$$
$$g(k,l) = \begin{cases} C(k)C(l)g^{SC}(k,l), & \text{for } |g^{SC}(k,l)| \leq 1, \\ 0, & \text{otherwise.} \end{cases}$$
5. Obtain the inverse DCT (2D-IDCT-II) of $f(k,l)$ and $g(k,l)$ as $DCS(m,n)$ and $DSC(m,n)$ for $m,n \in \{0,...,N-1\}$ respectively which basically are composed of impulse functions whose peak positions indicate the shift amount and peak signs reveal the direction of the movement.
6. Search for the peaks of $DCS(m,n)$ and $DSC(m,n)$ over $(m,n) \in \{0,\ldots,N-1\}^2$ (or range of interest).
7. Estimate the displacement $\hat{d} = (\hat{m}_u, \hat{m}_v)$ from the signs and positions of the peaks of $DCS(m,n)$ and $DSC(m,n)$.

Table 4.2: Summary of the DXT-ME algorithm

4.4 Unitary Property of the System Matrix

We will show that the system matrix $\mathbf{Z}_{t-1}(k,l)$ in (4.23) is a unitary matrix multiplied by a constant factor. Recall that

$$\mathbf{Z}_{t-1} = \begin{bmatrix} Z_{t-1}^{cc} & -Z_{t-1}^{cs} & -Z_{t-1}^{sc} & Z_{t-1}^{ss} \\ Z_{t-1}^{cs} & Z_{t-1}^{cc} & -Z_{t-1}^{ss} & -Z_{t-1}^{sc} \\ Z_{t-1}^{sc} & -Z_{t-1}^{ss} & Z_{t-1}^{cc} & -Z_{t-1}^{cs} \\ Z_{t-1}^{ss} & Z_{t-1}^{sc} & Z_{t-1}^{cs} & Z_{t-1}^{cc} \end{bmatrix}.$$

Define

$$K = \sqrt{(Z_{t-1}^{cc})^2 + (Z_{t-1}^{cs})^2 + (Z_{t-1}^{sc})^2 + (Z_{t-1}^{ss})^2}, \tag{4.44}$$

$$\alpha = \tan^{-1}(\frac{Z_{t-1}^{sc}}{Z_{t-1}^{cc}}), \tag{4.45}$$

$$\beta = \tan^{-1}(\frac{Z_{t-1}^{ss}}{Z_{t-1}^{cs}}). \tag{4.46}$$

Therefore,

$$Z_{t-1}^{cc} = K\cos\alpha\cos\beta, \tag{4.47}$$

$$Z_{t-1}^{cs} = K\cos\alpha\sin\beta, \tag{4.48}$$

$$Z_{t-1}^{sc} = K\sin\alpha\sin\beta, \tag{4.49}$$

$$Z_{t-1}^{ss} = K\sin\alpha\sin\beta. \tag{4.50}$$

Substituting (4.47)-(4.50) back into (4.24), we get

$$\mathbf{Z}_{t-1} = K\mathbf{A} \tag{4.51}$$

where

$$\mathbf{A} = \begin{bmatrix} \cos\alpha\cos\beta & -\cos\alpha\sin\beta & -\sin\alpha\cos\beta & +\sin\alpha\sin\beta \\ \cos\alpha\sin\beta & +\cos\alpha\cos\beta & -\sin\alpha\sin\beta & -\sin\alpha\cos\beta \\ \sin\alpha\cos\beta & -\sin\alpha\sin\beta & +\cos\alpha\cos\beta & -\cos\alpha\sin\beta \\ \sin\alpha\sin\beta & +\sin\alpha\cos\beta & +\cos\alpha\sin\beta & +\cos\alpha\cos\beta \end{bmatrix}. \tag{4.52}$$

Since

$$\mathbf{A}\mathbf{A}^T = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \mathbf{I} \text{ and } \mathbf{A}^T\mathbf{A} = \mathbf{I}, \tag{4.53}$$

the matrix $\mathbf{A}$ is unitary, i.e. $\mathbf{A} = \mathbf{A}^{-1}$. Hence,

$$\mathbf{Z}_{t-1}\mathbf{Z}_{t-1}^T = K^2\mathbf{I}. \tag{4.54}$$

As a result, the pseudo phase vector $\vec{\theta}(k,l)$ can be solved from the following equation:

$$\vec{\theta}(k,l) = (K)^{-2}\mathbf{Z}_{t-1}^T(k,l)\vec{\mathbf{x}}_t(k,l). \tag{4.55}$$

In the implementation of the above equation, it is important to make sure that $\mathbf{A}$ is unitary or alternatively the column vectors of $\mathbf{Z}_{t-1}^T$ are orthogonal to each other under the limitation of the machine precision. To check the orthogonality of the column vectors of $\mathbf{Z}_{t-1}^T$, the multiplication (inner product) of its two different column vectors must be zero or close enough to zero based on the knowledge of the machine precision.

4.5 Motion Estimation In Uniformly Bright Background

What if an object is moving in a uniformly bright background instead of a completely dark environment? It can be shown analytically and empirically that uniformly bright background introduces only very small spikes which does not affect the accuracy of the estimate. Suppose that $\{x_{t-1}(m,n)\}$ and $\{x_t(m,n)\}$ are pixel values of 2 consecutive frames of an object displaced by (m_u, m_v) on a uniformly bright background. Then let $y_t(m,n)$ and $y_{t-1}(m,n)$ be the pixel value of $x_t(m,n)$ and $x_{t-1}(m,n)$ subtracted by the background pixel value c $(c > 0)$ respectively:

$$y_t(m,n) = x_t(m,n) - c, \tag{4.56}$$

$$y_{t-1}(m,n) = x_{t-1}(m,n) - c. \tag{4.57}$$

In this way, $\{x_{t-1}(m,n)\}$ and $\{x_t(m,n)\}$ can be considered as the images of an object moving in a dark environment. Denote $\mathbf{Z}_{t-1}(k,l)$ as the system matrix of the input image x_{t-1} and $\mathbf{U}_{t-1}(k,l)$ as that of y_{t-1} for $k,l \in \mathcal{N}$. Also let $\vec{\mathbf{x}}_t(k,l)$ be the vector of the 2D-DCT-II coefficients of x_t and $\vec{\mathbf{y}}_t(k,l)$ be the vector for y_t. Applying the DXT-ME algorithm to both situations, we have, for $k,l \in \mathcal{N}$,

$$\mathbf{Z}_{t-1}(k,l) \cdot \vec{\theta}_{m_u m_v}(k,l) = \vec{\mathbf{x}}_t(k,l), \tag{4.58}$$

$$\mathbf{U}_{t-1}(k,l) \cdot \vec{\phi}_{m_u m_v}(k,l) = \vec{\mathbf{y}}_t(k,l). \tag{4.59}$$

Here $\vec{\phi}_{m_u m_v}(k,l)$ is the vector of the computed pseudo phases for the case of dark background and thus

$$\vec{\phi}_{m_u m_v}(k,l) = [g^{CC}_{m_u m_v}(k,l),\ g^{CS}_{m_u m_v}(k,l),\ g^{SC}_{m_u m_v}(k,l),\ g^{SS}_{m_u m_v}(k,l)]^T$$

but $\vec{\theta}_{m_u m_v}(k,l)$ is for uniformly bright background and

$$\vec{\theta}_{m_u m_v}(k,l) = [\hat{g}^{CC}_{m_u m_v}(k,l),\ \hat{g}^{CS}_{m_u m_v}(k,l),\ \hat{g}^{SC}_{m_u m_v}(k,l),\ \hat{g}^{SS}_{m_u m_v}(k,l)]^T \neq \vec{\phi}_{m_u m_v}(k,l).$$

Starting from the definition of each element in $\mathbf{Z}_{t-1}(k,l)$ and $\vec{\mathbf{x}}_t(k,l)$, we obtain

$$\mathbf{Z}_{t-1}(k,l) = \mathbf{U}_{t-1}(k,l) + c \cdot \mathbf{D}(k,l), \tag{4.60}$$

$$\vec{\mathbf{x}}_t(k,l) = \vec{\mathbf{y}}_t(k,l) + c \cdot \vec{\mathbf{c}}(k,l), \tag{4.61}$$

where $\mathbf{D}(k,l)$ is the system matrix with $\{d(m,n) = 1,\ \forall m,n = \{0,\ldots,N-1\}\}$ as input and $\vec{\mathbf{c}}(k,l)$ is the vector of the 2D-DCT-II coefficients of $d(m,n)$. Substituting (4.60) and (4.61) into (4.59), we get

$$\begin{aligned}\mathbf{Z}_{t-1}(k,l)\cdot\vec{\theta}_{m_u m_v}(k,l) &= \mathbf{Z}_{t-1}(k,l)\cdot\vec{\phi}_{m_u m_v}(k,l)\\ &\quad + c\cdot[\vec{\mathbf{c}}(k,l) - \mathbf{D}(k,l)\cdot\vec{\phi}_{m_u m_v}(k,l)]. \qquad (4.62)\end{aligned}$$

Since $\vec{\mathbf{c}}(k,l) = \mathbf{D}(k,l)\cdot\vec{\phi}_{00}(k,l)$, (4.62) becomes

$$\vec{\theta}_{m_u m_v}(k,l) = \vec{\phi}_{m_u m_v}(k,l) + c\mathbf{Z}_{t-1}^{-1}(k,l)\mathbf{D}(k,l)[\vec{\phi}_{00}(k,l) - \vec{\phi}_{m_u m_v}(k,l)], \quad (4.63)$$

provided that $|\mathbf{Z}_{t-1}(k,l)| \neq 0$. Similar results can also be found at block boundaries. Referring to (4.24), we know that $\mathbf{D}(k,l)$ is composed of $D^{cc}(k,l)$, $D^{cs}(k,l)$, $D^{sc}(k,l)$, and $D^{ss}(k,l)$, each of which is a separable function made up by

$$D^c(k) \equiv \frac{2}{N}C(k)\sum_{m=0}^{N-1}\cos[\frac{k\pi}{N}m] = \frac{2}{N}C(k)\{0.5[1-(-1)^k] + N\cdot\delta(k)\},$$

$$D^s(k) \equiv \frac{2}{N}C(k)\sum_{m=0}^{N-1}\sin[\frac{k\pi}{N}m] = \begin{cases}\frac{2}{N}C(k)\frac{[1-(-1)^k]}{2\tan\frac{k\pi}{2N}}, & \text{for } k\neq 0,\\ 0, & \text{for } k = 0.\end{cases}$$

From the above equations, we can see that $D^c(k) = D^s(k) = 0$ if k is even, and for odd $k > 0$, $D^c(k) = \frac{2}{N}$ while $D^s(k) = \frac{2}{N\tan\frac{k\pi}{2N}}$. Hence, $D^{cc}(k,l) = D^{cs}(k,l) = D^{sc}(k,l) = D^{ss}(k,l) = 0$ if either k or l is even. As a result, $\vec{\theta}_{m_u m_v}(k,l) = \vec{\phi}_{m_u m_v}(k,l)$ if either k or l is even. For odd indices k and l, it is possible to find a constant s and a matrix $\mathbf{N}(k,l) \in R^{4\times 4}$ such that $\mathbf{U}_{t-1}(k,l) = s[\mathbf{D}(k,l) - \mathbf{N}(k,l)]$ and $|\mathbf{N}(k,l)\mathbf{D}^{-1}(k,l)| < 1$ for $|\mathbf{D}(k,l)| \neq 0$. Thus, for $|\frac{s}{s+c}\mathbf{N}(k,l)\mathbf{D}^{-1}(k,l)| < 1$,

$$\begin{aligned}c\mathbf{Z}_{t-1}^{-1}(k,l)\mathbf{D}(k,l) &= \frac{c}{s+c}[\mathbf{I} - \frac{s}{s+c}\mathbf{N}(k,l)\mathbf{D}^{-1}(k,l)]^{-1} \qquad (4.64)\\ &= \frac{c}{s+c}\{\mathbf{I} + \frac{s}{s+c}\mathbf{N}(k,l)\mathbf{D}^{-1}(k,l)\\ &\quad + [\frac{s}{s+c}\mathbf{N}(k,l)\mathbf{D}^{-1}(k,l)]^2 + \ldots\}. \qquad (4.65)\end{aligned}$$

If we lump all the high-order terms of $\frac{s}{s+c}\mathbf{N}(k,l)\mathbf{D}^{-1}(k,l)$ in one term $\mathbf{H}(k,l)$, then

$$\vec{\theta}_{m_u m_v}(k,l) = \vec{\phi}_{m_u m_v}(k,l) + [\frac{c}{s+c} + \mathbf{H}(k,l)][\vec{\phi}_{00}(k,l) - \vec{\phi}_{m_u m_v}(k,l)]. \quad (4.66)$$

Usually, $0 \le c, s \le 255$ for the maximum gray level equal to 255. Typically $s = 1$. For moderately large c, $\mathbf{H}(k,l)$ is very small. Define the subsampled version of the pseudo-phase function $\vec{\phi}_{ab}(k,l)$ as

$$\vec{\lambda}_{ab}(k,l) \equiv \begin{cases} \vec{\phi}_{ab}(k,l), & \text{if both } k \text{ and } l \text{ are odd,} \\ 0, & \text{otherwise .} \end{cases} \quad (4.67)$$

Then

$$\vec{\theta}_{m_u m_v}(k,l) = \vec{\phi}_{m_u m_v}(k,l) + [\frac{c}{s+c} + \mathbf{H}(k,l)]\{\vec{\lambda}_{00} - \vec{\lambda}_{m_u m_v}\}. \quad (4.68)$$

Recall that a 2D-IDCT-II operation on $\vec{\phi}_{m_u m_v}(k,l)$ or $\vec{\phi}_{00}(k,l)$ produces $\vec{\delta}_{m_u m_v}$ or $\vec{\delta}_{00}$, respectively, where

$$\vec{\delta}_{ab}(m,n) = \begin{bmatrix} (\delta(m-a)+\delta(m+a+1))(\delta(n-b)+\delta(n+b+1)) \\ (\delta(m-a)+\delta(m+a+1))(\delta(n-b)-\delta(n+b+1)) \\ (\delta(m-a)-\delta(m+a+1))(\delta(n-b)+\delta(n+b+1)) \\ (\delta(m-a)-\delta(m+a+1))(\delta(n-b)-\delta(n+b+1)) \end{bmatrix}.$$

Therefore,

$$\begin{aligned} \vec{\mathbf{d}}(m,n) &\equiv \text{2D-DCT-II}\{\vec{\theta}_{m_u m_v}\} \\ &= \vec{\delta}_{m_u m_v}(m,n) + \frac{c}{s+c}\,\text{2D-DCT-II}\{\vec{\lambda}_{00} - \vec{\lambda}_{m_u m_v}\} + \vec{\mathbf{n}}(m,n), \quad (4.69) \end{aligned}$$

where $\vec{\mathbf{n}}$ is the noise term contributed from 2D-DCT-II$\{\mathbf{H}(k,l)[\vec{\lambda}_{00} - \vec{\lambda}_{m_u m_v}]\}$. Because $\vec{\lambda}_{ab}$ is equivalent to downsampling $\vec{\phi}_{ab}$ in a 2D index space and it is known that downsampling produces in the transform domain mirror images of magnitude only one-fourth of the original and of sign depending on the transform function, we obtain

$$\begin{aligned}\vec{\mathbf{E}}_{m_u m_v}(m,n) \equiv\ & \text{2D-DCT-II}\{\vec{\lambda}_{m_u m_v}\}\\ =\ & \frac{1}{4}[\vec{\delta}_{m_u m_v}(m,n) + \text{diag}(\vec{\zeta}_1)\cdot\vec{\delta}_{(N-1-m_u)m_v}(m,n)\\ & + \text{diag}(\vec{\zeta}_2)\cdot\vec{\delta}_{m_u(N-1-m_v)}(m,n)\\ & + \text{diag}(\vec{\zeta}_3)\cdot\vec{\delta}_{(N-1-m_u)(N-1-m_v)}(m,n)],\end{aligned} \quad (4.70)$$

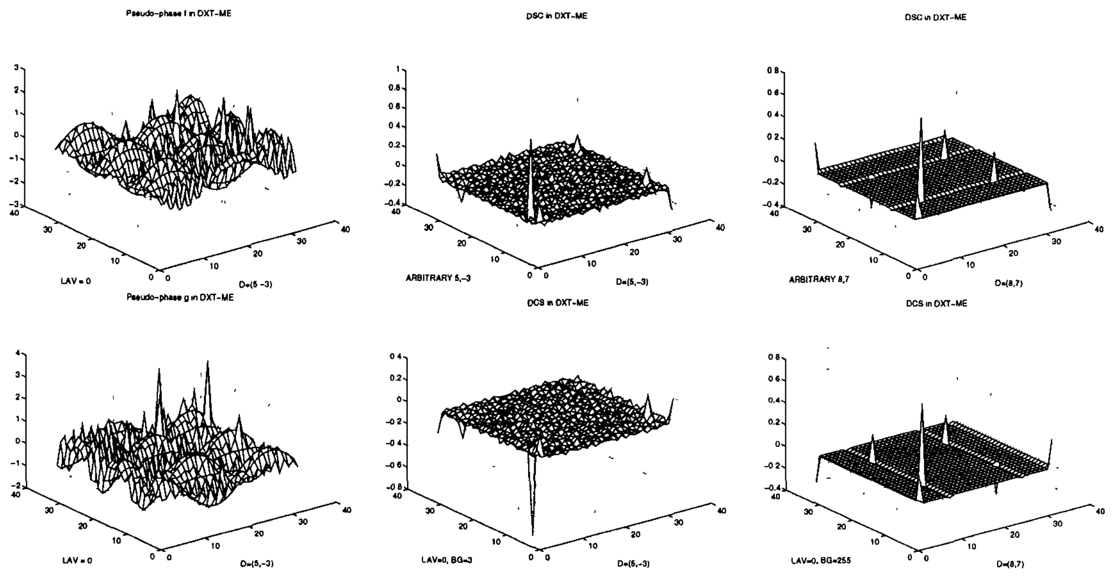
where diag$(\cdot)$ is the diagonal matrix of a vector and $\vec{\zeta}_i$ $(i = 1,2,3)$ is a vector consisting of ±1. A similar expression can also be established for 2D-DCT-II$\{\vec{\lambda}_{00}\}$. In conclusion,

$$\vec{\mathbf{d}}(m,n) = \vec{\delta}_{m_u m_v}(m,n) + \frac{c}{4(s+c)}[\vec{\mathbf{E}}_{00}(m,n) - \vec{\mathbf{E}}_{m_u m_v}(m,n)] + \vec{\mathbf{n}}(m,n). \quad (4.71)$$

The above equation predicts the presence of a very small noise term $\vec{\mathbf{n}}$ and several small spikes, $\vec{\mathbf{E}}_{00}$ and $\vec{\mathbf{E}}_{m_u m_v}$, of magnitude moderated by $\frac{c}{4(s+c)}$ which are much smaller than the displacement peak, as displayed in Fig. 4.7 (b) and (c) where $\vec{\mathbf{n}}$ for the case of $c = 3$ in (b) is observable but very small and can be regarded as noise whereas $\vec{\mathbf{n}}$ is practically absent as in (c) when $c = 255$.

4.6 Computational Issues and Complexity

The block diagram in Fig. 4.4(a) shows that a separate 2D-DCT-I is needed in addition to the standard DCT (2D-DCT-II). This is undesirable from the complexity viewpoint. However, this problem can be circumvented by considering the point-to-point relationship between 2D-DCT-I and 2D-DCT-II coefficients in the



(a) f and g (b) DSC and DCS (c) another DSC and DCS

Figure 4.7: (a)(b) An object is moving in the direction (5, -3) in a uniformly bright background ($c = 3$). (c) Another object is moving northeast (8,7) for background pixel values $= c = 255$.

frequency domain for $k, l \in \mathcal{N}$:

$$\begin{bmatrix} Z_{t-1}^{cc}(k,l) \\ Z_{t-1}^{cs}(k,l) \\ Z_{t-1}^{sc}(k,l) \\ Z_{t-1}^{ss}(k,l) \end{bmatrix} = \begin{bmatrix} +\cos\frac{k\pi}{2N}\cos\frac{l\pi}{2N} & +\cos\frac{k\pi}{2N}\sin\frac{l\pi}{2N} & +\sin\frac{k\pi}{2N}\cos\frac{l\pi}{2N} & +\sin\frac{k\pi}{2N}\sin\frac{l\pi}{2N} \\ -\cos\frac{k\pi}{2N}\sin\frac{l\pi}{2N} & +\cos\frac{k\pi}{2N}\cos\frac{l\pi}{2N} & -\sin\frac{k\pi}{2N}\sin\frac{l\pi}{2N} & +\sin\frac{k\pi}{2N}\cos\frac{l\pi}{2N} \\ -\sin\frac{k\pi}{2N}\cos\frac{l\pi}{2N} & -\sin\frac{k\pi}{2N}\sin\frac{l\pi}{2N} & +\cos\frac{k\pi}{2N}\cos\frac{l\pi}{2N} & +\cos\frac{k\pi}{2N}\sin\frac{l\pi}{2N} \\ +\sin\frac{k\pi}{2N}\sin\frac{l\pi}{2N} & -\sin\frac{k\pi}{2N}\cos\frac{l\pi}{2N} & -\cos\frac{k\pi}{2N}\sin\frac{l\pi}{2N} & +\cos\frac{k\pi}{2N}\cos\frac{l\pi}{2N} \end{bmatrix} \times \begin{bmatrix} X_{t-1}^{cc}(k,l) \\ X_{t-1}^{cs}(k,l) \\ X_{t-1}^{sc}(k,l) \\ X_{t-1}^{ss}(k,l) \end{bmatrix} \tag{4.72}$$

where X_{t-1}^{cc}, X_{t-1}^{cs}, X_{t-1}^{sc}, and X_{t-1}^{ss} are the 2D-DCT-II coefficients of the previous frame. Similar relation also exists for the coefficients at block boundaries.

Stage	Component	Computational Complexity
1	2D-DCT-II	$O_{dct} = O(N)$
	Coeff. Transformation Unit (T)	$O(N^2)$
2	Pseudo Phase Computation	$O(N^2)$
3	2D-IDCT-II	$O_{dct} = O(N)$
4	Peak Searching	$O(N^2)$
	Estimation	$O(1)$

Table 4.3: Computational complexity of each stage in DXT-ME

This observation results in the simple structure in Fig. 4.4(b), where Block T is a coefficient transformation unit realizing (4.72).

In view of the fact that the actual number of computations required by the DCT pseudo phase technique or the DXT-ME algorithm lies heavily on the specific implementation for a particular application such as motion estimation in video coding, it is more appropriate to consider the asymptotic computational complexity as generally accepted in the evaluation of algorithms in this section. Based on the straight forward implementation without further optimization, a rough count of the actual number of computations will be presented in Section 5 where the DXT-ME algorithm is used in video coding.

If the DCT has computational complexity O_{dct}, the overall complexity of DXT-ME is $O(N^2) + O_{dct}$ with the complexity of each component summarized in Table 4.3. The computational complexity of the pseudo phase computation component is only $O(N^2)$ for an $N \times N$ block and so is the unit to determine the displacement. For the computation of the pseudo phase functions $f(\cdot, \cdot)$ in (4.33) and $g(\cdot, \cdot)$ in (4.34), DSCT, DCST and DSST coefficients (regarded as DST coefficients) must be calculated in addition to DCCT coefficients (i.e. the usual 2D DCT). However all these coefficients can be generated with little overhead in the course of computing 2D DCT coefficients. As a matter of fact, a parallel and fully-pipelined 2D DCT lattice structure has been developed [12, 49, 50] to generate 2D DCT coefficients at a cost of $O(N)$ operations. This DCT coder computes DCT and DST coefficients dually due to its internal lattice architecture. These internally generated DST coefficients can be output to the DXT-ME module for pseudo phase computation. This same lattice structure can also be modified as a 2D IDCT which also has $O(N)$ complexity. To sum up, the computational complexity of this DXT-ME is only $O(N^2)$, much lower than the $O(N^4)$ complexity of BKM-ME.

A closer look at (4.33), (4.34) and (4.72) reveals that the operations of pseudo phase computation and coefficient transformation are performed independently at each point (k, l) in the transform domain and therefore are inherently highly parallel operations. Since most of the operations in the DXT-ME algorithm involve mainly pseudo phase computations and coefficient transformations in addition to DCT and Inverse DCT operations which have been studied extensively, the DXT-ME algorithm can easily be implemented on highly parallel array processors or dedicated circuits. This is very different from BKM-ME which requires shifting of pixels and summation of differences of pixel values and hence discourages parallel implementation.

4.7 Simulation for Application to Image Registration

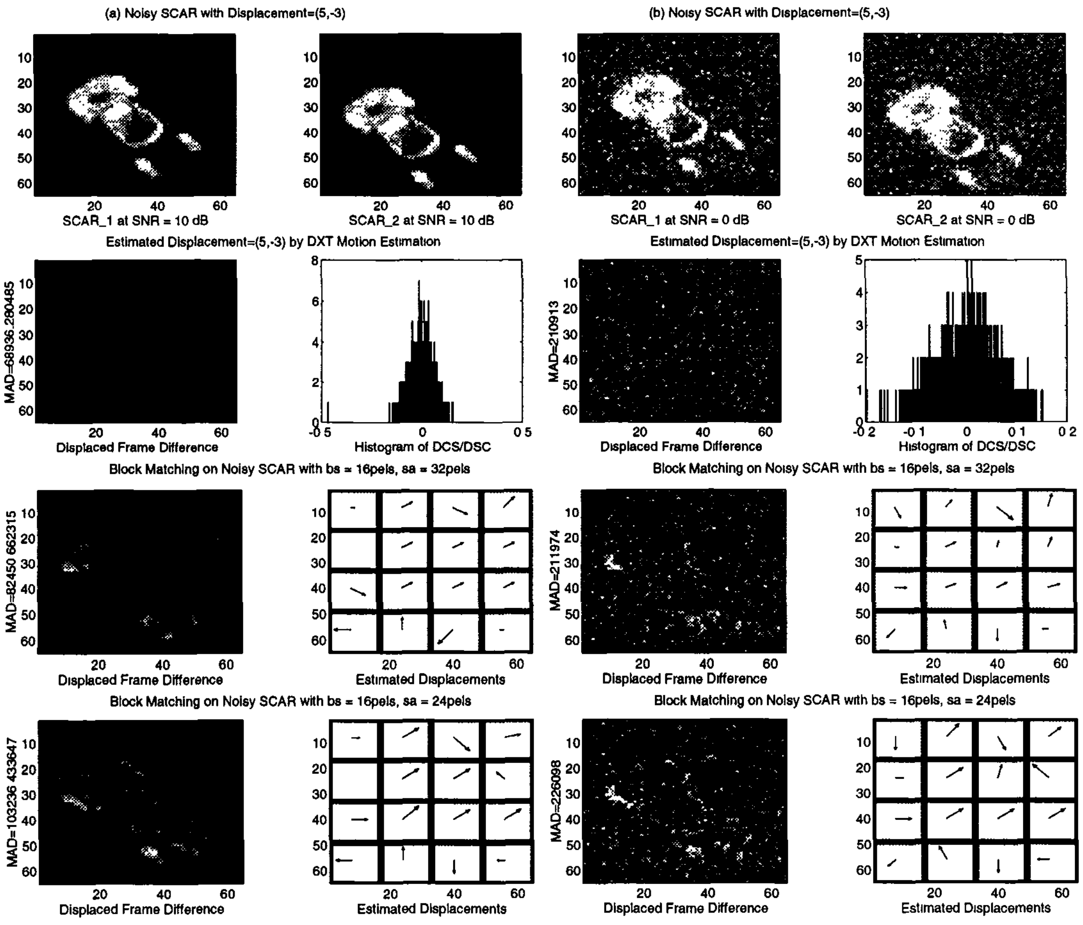


Figure 4.8: Comparison of DXT-ME of size 64 by 64 pels with Full-Search Block Matching Method (BKM-ME) of block size (bs = 16 pels) but different search areas (sa = 32 or 24 pels) on a noisy small car (SCAR) with (a) SNR = 10 dB, (b) SNR = 0 dB.

To test the performance of DXT-ME on noisy images, an image of a small car (SCAR_1) is manually shifted to produce the second frame (SCAR_2) with a known displacement and additive Gaussian noise is added to attain a desired signal-to-ratio (SNR) level. Since the object (small car) moves within the boundary of the frame in a completely darken background, no preprocessing is required. As can be seen in Fig. 4.8, DXT-ME is performed on the whole image of block size 64×64 and estimates the motion correctly at SNR level even down to 0 dB, whereas Full Search Block Matching Approach, BKM-ME produces some wrong motion estimates for boundary blocks and blocks of low signal energy. The MAD values also indicate better overall performance of DXT-ME over BKM-ME for these two still images. Furthermore, DXT-ME can perform on the whole frame while BKM-ME needs division of the frame into sub-blocks due to the requirement of larger search areas than reference blocks. This is one of the reasons that BKM-ME does not work so well as DXT-ME because smaller block size makes BKM-ME more susceptible to noise and operation of DXT-ME on the whole frame instead of on smaller blocks lends itself to better noise immunity. Even though the Kalman filtering approach [9] can also estimate velocity accurately for a sequence of noisy images, it requires iterative complicated computations while DXT-ME can estimate motion based upon two consecutive frames in one step, requiring low-complexity computations.

4.8 Conclusion

In this chapter, we propose novel DCT pseudo phase techniques to estimate shift/delay between two 1-D signals directly from their DCT coefficients by computing the pseudo phase shift hidden in DCT and then employing the sinusoidal orthogonal principles, applicable to signal delay estimation and remote sensing. Under the 2-D translational motion model, we further extend the pseudo phase techniques to the DCT-Based Motion Estimation (DXT-ME) algorithm for 2-D signals/images. Equally applicable to scenarios such as image registration and target tracking, the DXT-ME algorithm has certain advantages over the commonly used Full Search Block Matching approach (BKM-ME). We show that the DXT-ME algorithm exhibits accurate estimates even in a noisy situation. In addition to its robustness in a noisy environment and low computational complexity, $O(N^2)$ for an $N \times N$ block in comparison to the $O(N^4)$ complexity of BKM-ME, its ability to estimate motion completely in DCT domain makes possible the fully DCT-based motion-compensated video coder structure. Furthermore, combination of the DCT and motion estimation units can provide space for further optimization of the overall coder. In addition, the DXT-ME algorithm has only highly parallel local operations and this property makes feasible parallel implementation suitable for VLSI design. In the next chapter, we will demonstrate by simulation on a number of video sequences that in application to video coding, the DCT pseudo phase techniques perform well compared to BKM-ME and other fast block search algorithms in terms of mean square error per pel (MSE) and bits per sample (BPS) even though DXT-ME is completely different from any block search algorithms.

Chapter 5

DCT-Based Motion Estimation Approach

The immediate application of the DCT-based DXT-ME algorithm will then be motion estimation incorporated into the standard-compliant DCT-based motion-compensated video coder design. As mentioned in Chapter 3.2, most video coding standards are based on the hybrid DCT-based motion-compensated approach. The conventional video coder design requires a DCT and IDCT pair in the feedback due to the fact that motion estimation must be performed in the spatial domain with the currently available motion estimation schemes. However, if we apply the DXT-ME algorithm to estimate motion in the DCT domain, we can eliminate this pair and increase the system throughput while reduce the overall complexity. Furthermore, as pointed out in [46], different components can be jointly optimized if they operate in the same transform domain. It should be stressed that by using DCT-based estimation and compensation methods, standard-compliant bit streams can be formed in accordance to the specification of any standard such as MPEG without any need to change the structure of any standard-compliant decoder.

This chapter will be devoted to the application of the DXT-ME algorithm to video coding. We will first discuss some modifications to the DXT-ME algorithm such as preprocessing and the adaptive overlapping approach to fit to the requirement of video coding. To estimate motion in video sequences of complicated scenery, we add a preprocessing step to remove background features. For fair comparison with the full search block matching approach (BKM-ME) having a larger search area, we introduce the adaptive overlapping approach to allow a larger search area in order to alleviate the boundary effect which occurs when displacements are large compared to the block size and the contents of two blocks differ considerably. Then we will present the simulation results on a number of video sequences of different characteristics to demonstrate that the modified DXT-ME algorithm is competitive in performance to the block matching schemes commonly used in the area of video coding.

5.1 Preprocessing

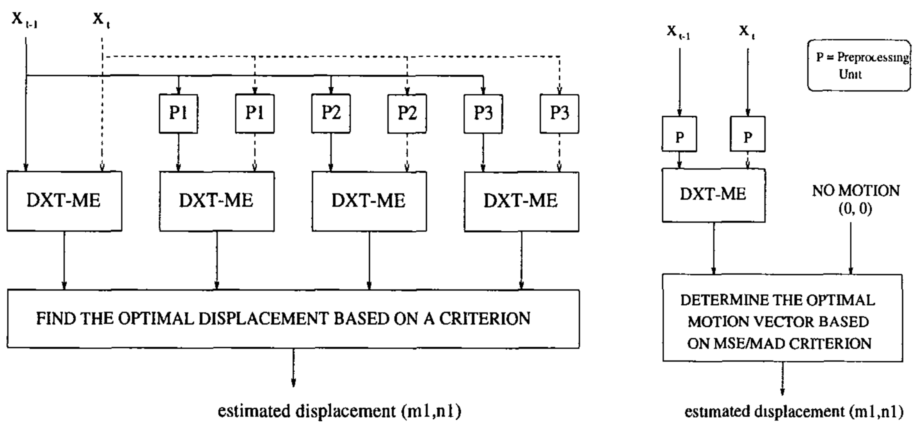
For complicated video sequences in which objects may move across the border of blocks in non-uniform background, preprocessing can be employed to enhance the features of moving objects and avoid violation of the assumption made for DXT-ME before feeding the images into the DXT-ME algorithm. Intuitively speaking, the DXT-ME algorithm tries to match the features of any object on two consecutive frames so that any translation motion can be estimated regardless of the shape and texture of the object as long as these two frames contain significant energy level of the object features. Due to this feature matching property of the DXT-ME algorithm, effective preprocessing will improve the performance of motion estimation if preprocessing can enhance the object features in the original sequence.

In order to keep the computational complexity of the overall motion estimator low, the chosen preprocessing function must be simple but effective in the sense that unwanted features will not affect the accuracy of estimation. Our study found that both edge extraction and frame differentiation are simple and efective schemes for extraction of motion information.

It is found that estimating the motion of an object from its edges is equivalent to estimating from its image projection [86]. Furthermore, since the DXT-ME algorithm assumes that an object moves within the block boundary in a completely dark environment, its edge information reduces the adverse effect of the object moving across the block boundary on the estimation accuracy. The other advantage of edge extraction is that any change in the illumination condition does not alter the edge information and in turn makes no false motion estimates by the DXT-ME algorithm. Since we only intend to extract the main features of moving objects while keeping the overall complexity low, we employ a very simple edge detection by convolving horizontal and vertical Sobel operators of size 3×3

$$H_s = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}, \; V_s = H_s^T \tag{5.1}$$

with the image to obtain horizontal and vertical gradients respectively and then combine both gradients by taking the square root of the sum of the squares of both gradients [28] . Edge detection provides us the features of moving objects but also the features of the background (stationary objects) which is undesirable. However, if the features of the background have smaller energy than those of moving objects within every block containing moving objects, then the background features will not affect the performance of DXT-ME. The computational complexity of this


X_{t-1}
X_t
P1
P1
P2
P2
P3
P3
DXT-ME
DXT-ME
DXT-ME
DXT-ME
FIND THE OPTIMAL DISPLACEMENT BASED ON A CRITERION
estimated displacement (m1,n1)
X_{t-1}
X_t
P = Preprocessing Unit
P
P
DXT-ME
NO MOTION (0, 0)
DETERMINE THE OPTIMAL MOTION VECTOR BASED ON MSE/MAD CRITERION
estimated displacement (m1,n1)

(a) E-DXT-ME (b) SE-DXT-ME

Figure 5.1: Block Diagrams of Extended DXT-ME Estimator (E-DXT-ME) and Simplified Extended DXT-ME (SE-DXT-ME)

preprocessing step is only $O(N^2)$ and thus the overall computational complexity is still $O(N^2)$.

Frame differentiation generates an image of the difference of two consecutive frames. This frame differentiated image contains no background objects but the difference of moving objects between two frames. The DXT-ME estimator operates directly on this frame differentiated sequence to predict motion in the original sequence. The estimate will be good if the moving objects are moving constantly in one direction in three consecutive frames. For 30 frames per second, the standard NTSC frame rate, objects can usually be viewed as moving at a constant speed in three consecutive frames. Obviously, this step also has only $O(N^2)$ computational complexity.

Alternatively, instead of using only one preprocessing function, we can employ several simple difference operators in the preprocessing step to extract features of images as shown in Fig. 5.1(a), in which four DXT-ME estimators generate four candidate estimates of which one can be chosen as the final estimated displacement based upon either the mean squared error per pixel (MSE) [29] or the mean of absolute differences per pixel (MAD) criteria [35].

Preferably, a simple decision rule similar to the one used in the MPEG-1 standard [54] , as depicted in Fig. 5.1(b), is used to choose among the DXT-ME estimate and no motion. This simplified extended DXT-ME algorithm works very well when combined with the adaptive overlapping approach.

5.2 Adaptive Overlapping Approach

For fair comparison with BKM-ME which has a larger search area than the block size, we adopt the adaptive overlapping approach to enlarge adaptively the block area. The enlargement of the block size diminishes the boundary effect which happens when the displacement is very large compared to the block size. As a result, the moving objects may move out partially or completely of the block, making the contents in two temporally consecutive blocks very different. However, this problem also exists for other motion estimation algorithms. That is why we need to assume that objects in the scene are moving slowly. For rapid motion, it is difficult to track motion.

In Section 4.3, we mention that we search for peaks of DSC and DCS over a fixed index range of interest $\Phi = \{0, \ldots, N/2\}^2$. However, if we follow the partitioning approach used in BKM-ME, then we may dynamically adjust Φ. At first, partition the whole current frame into $bs \times bs$ nonoverlapping reference blocks shown as the shaded area in Fig. 5.2(a). Each reference block is associated with a larger search area (of size sa) in the previous frame (the dotted region in the same

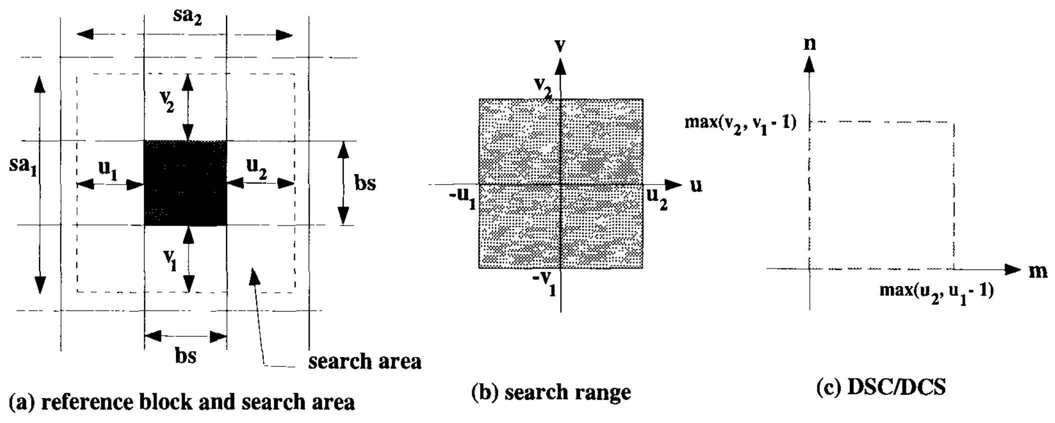


Figure 5.2: Adaptive Overlapping approach

figure) in the same way as for BKM-ME. From the position of a reference block and its associated search area, a search range $\mathcal{D} = \{(u, v) : -u_1 \leq u \leq u_2,\ -v_1 \leq v \leq v_2\}$ can then be determined as in Fig. 5.2(b). In contrast to BKM-ME, DXT-ME requires that the reference block size and the search area size must be equal. Thus, instead of using the reference block, we use the block of the same size and position in the current frame as the search area of the previous frame. The peak values of DSC and DCS are searched in a zigzag way as described in Section 4.3 over this index range $\Phi = \{0, \dots, \max(u_2, u_1 - 1)\} \times \{0, \dots, \max(v_2, v_1 - 1)\}$. In addition to the requirement that the new peak value must be larger than the current peak value by a preset threshold, it is necessary to examine if the motion estimate determined by the new peak index lies in the search region $\mathcal{D}$. Since search areas overlap on one another, the SE-DXT-ME architecture utilizing this approach is called Overlapping SE-DXT-ME. Even though the block size required by the Overlapping SE-DXT-ME algorithm is larger than the block size for one DCT block, it is still possible to estimate motion completely in the DCT domain without going back to the spatial domain by concatenating neighboring DCT blocks directly in the DCT domain

[38].

5.3 Simulation Results

A number of video sequences with different characteristics are used in our simulations to compare the performance of the DXT-ME algorithm with the Full Search Block Matching method (BKM-ME or BKM for the sake of brevity) as well as three commonly used fast search block matching approaches such as the Logarithmic Search method (LOG), the Three Step Search method (TSS), and the Subsampled Search approach (SUB) [48]. The performance of different schemes is evaluated and compared in terms of MSE (mean squared error per pel) and BPS (bits per sample) where $MSE = \frac{\sum_{m,n}[\hat{x}(m,n)-x(m,n)]^2}{N^2}$ and BPS is the ratio of the total number of bits required for each motion compensated residual frame in JPEG format (BPS) converted by the image format conversion program ALCHEMY with quality = 32 to the number of pixels. As widely used in the literature of video coding, all the block matching methods adopt the conventional MAD optimization criterion:

$$\hat{d} = (\hat{u}, \hat{v}) = arg \min_{(u,v)\in \mathsf{S}} \frac{\sum_{m,n} |x_2(m,n) - x_1(m-u, n-v)|}{N^2},$$

where S denotes the set of allowable displacements depending on which block matching approach is in use.

The first sequence is the "Flower Garden" (FG) sequence where the camera is moving before a big tree and a flower garden in front of a house as shown in Fig. 5.3(a). Each frame has 352 × 224 pixels. Simple preprocessing is applied to this sequence: edge extraction or frame differentiation as depicted in Fig. 5.3 (b) and (c) respectively. Since macroblocks, each consisting of 16 × 16 luminance

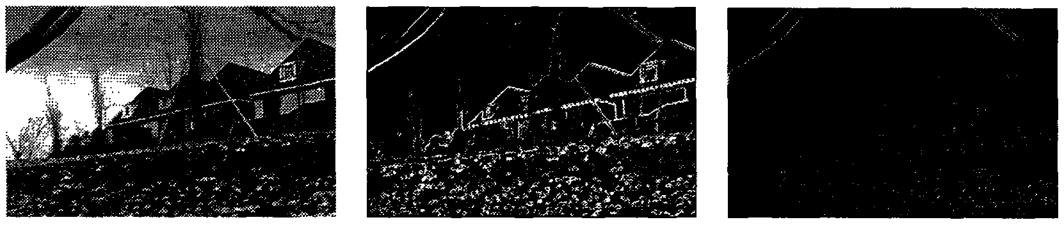
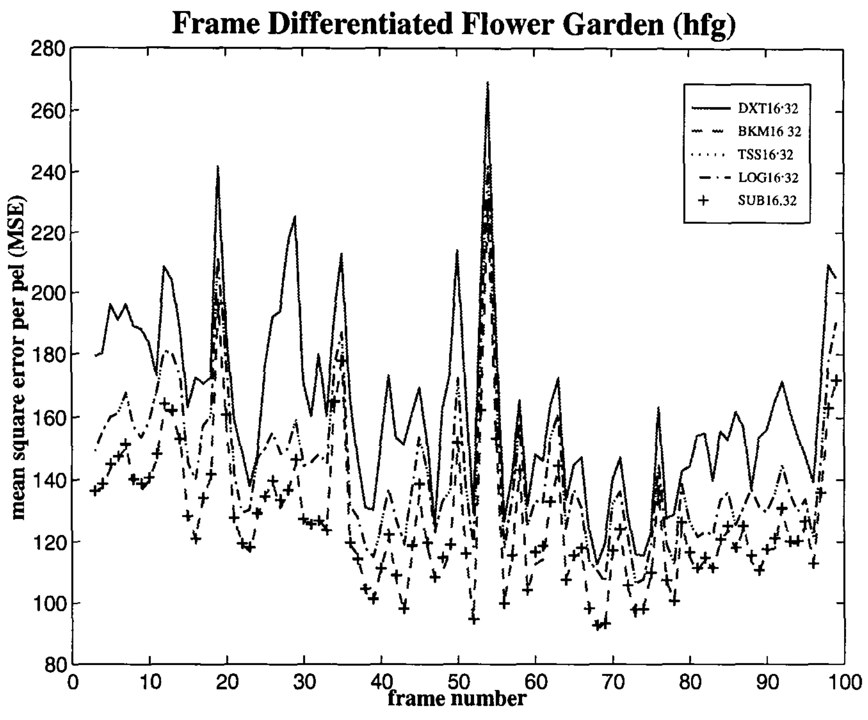

(a) Original (b) Edge extracted (c) Frame differentiated

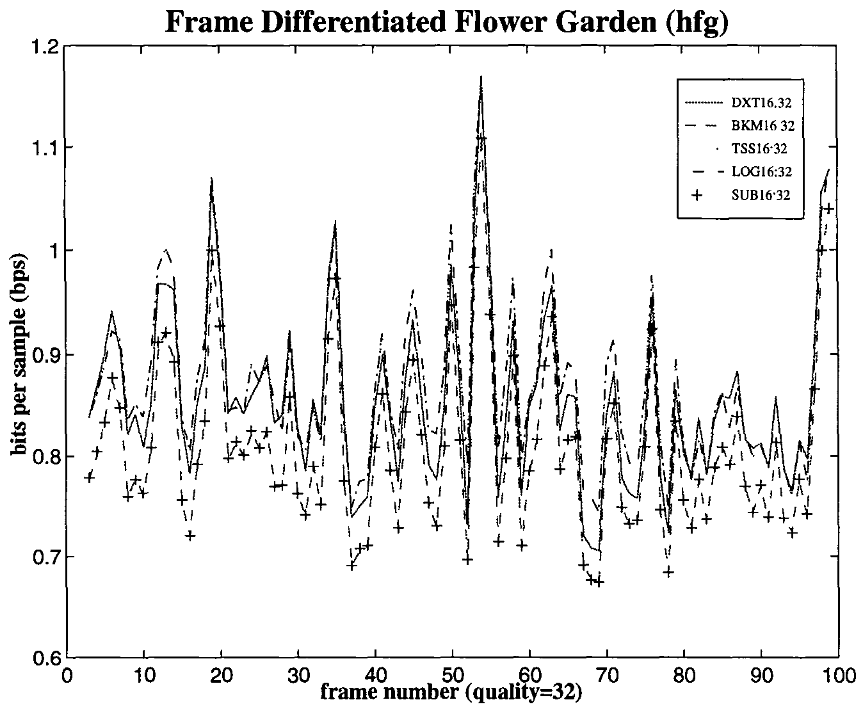
Figure 5.3: Frame 57 in the sequence "Flower Garden" (FG)

blocks and two 8×8 chrominance blocks, are considered to be the basic unit for motion estimation/compensation in MPEG standards [54], the following simulation setting is adopted for simulations on the "Flower Garden" sequence and all other subsequent sequences: 16×16 blocks on 32×32 search areas. Furthermore, the overlapping SE-DXT-ME algorithm is used for fair comparison with block matching approaches which require a larger search area.

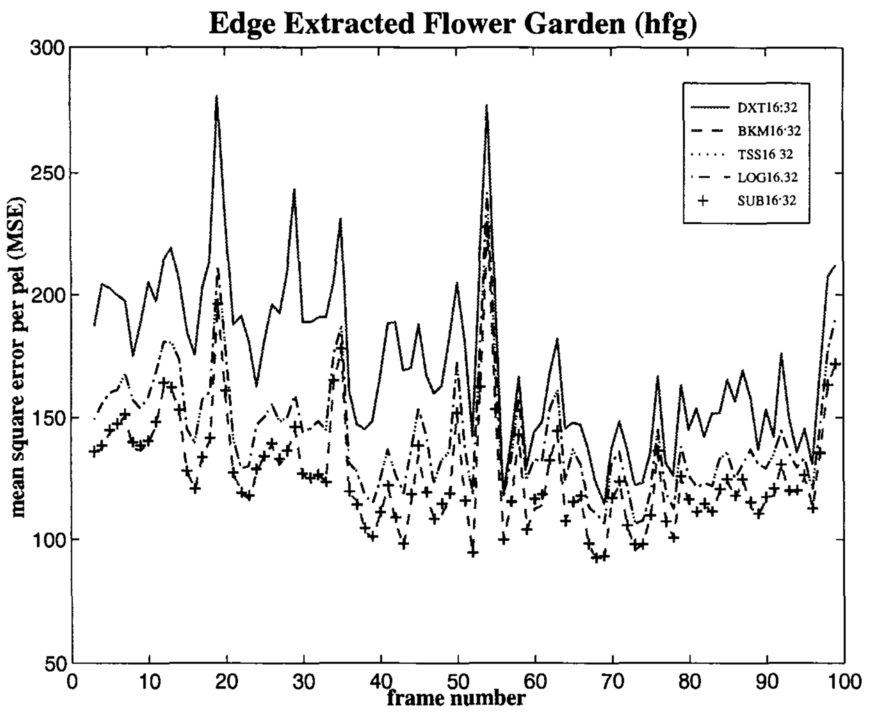
As can be seen in Fig. 5.3(b), the edge extracted frames contain significant features of moving objects in the original frames so that DXT-ME can estimate the movement of the objects based upon the information provided by the edge extracted frames. Because the camera is moving at a constant speed in one direction, the moving objects occupy almost the whole scene. Therefore, the background features do not interfere with the operation of DXT-ME much but still affect the overall performance of DXT-ME as compared to the frame differentiated preprocessing approach. The frame differentiated images of the "Flower Garden" sequence, one of which is shown in Fig. 5.3(c), have the residual energy strong enough for DXT-ME to estimate the motion directly on this frame differentiated sequence due to the constant movement of the camera.

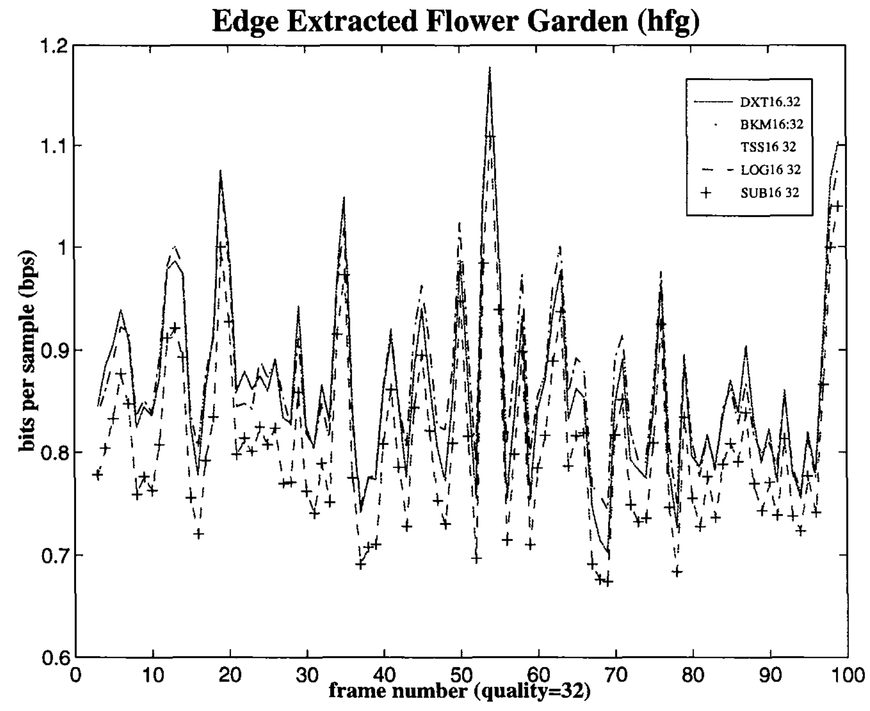
The performances for different motion estimation schemes are plotted in Fig. 5.4





(a) Preprocessed with Frame Differentiation





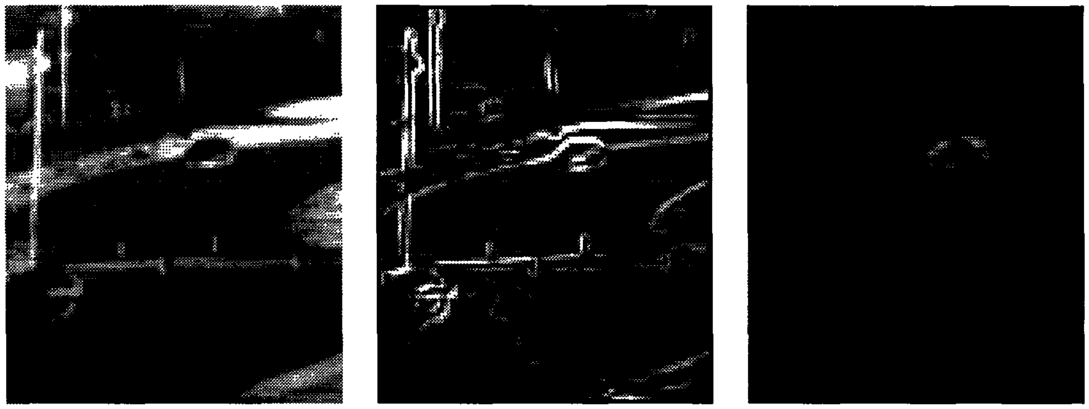
(b) Preprocessed with Edge Extraction

Figure 5.4: Comparison of Overlapping SE-DXT-ME with block matching approaches on "Flower Garden"

Approach	MSE	MSE difference	MSE ratio	BPF	BPS	BPS ratio
BKM	127.021	0.000	0%	63726	0.808	0%
Frame Differentiated DXT-ME	163.712	36.691	28.9%	67557	0.857	6.0%
Edge Extracted DXT-ME	172.686	45.665	36.0%	68091	0.864	6.8%
TSS	143.046	16.025	12.6%	68740	0.872	7.9%
LOG	143.048	16.026	12.6%	68739	0.872	7.9%
SUB	127.913	0.892	0.7%	63767	0.809	1%

Table 5.1: Performance summary of the overlapping SE-DXT-ME algorithm with either frame differentiation or edge extraction as preprocessing against full search and fast search block matching approaches (BKM, TSS, LOG, SUB) over the sequence "Flower Garden." MSE difference is the difference from the MSE value of full search block matching method (BKM) and MSE ratio is the ratio of MSE difference to the MSE of BKM.

and summarized in Table 5.1 where the MSE and BPS values of different motion estimation approaches are averaged over the whole sequence from frame 3 to frame 99 for easy comparison. It should be noted that the MSE difference in Table 5.1 is the difference of the MSE value of the corresponding motion estimation scheme from the MSE value of the full search block matching approach (BKM) and the MSE ratio is the ratio of the MSE difference to the MSE of BKM. As indicated in the performance summary table, the frame differentiated DXT-ME algorithm is 28.9% worse in terms of MSE than the full search block matching approach while the edge extracted DXT-ME algorithm is 36.0% worse. Surprisingly, even though the fast search block matching algorithms (only 12.6% worse than BKM), TSS and LOG, have smaller MSE values than the DXT-ME algorithm, TSS and

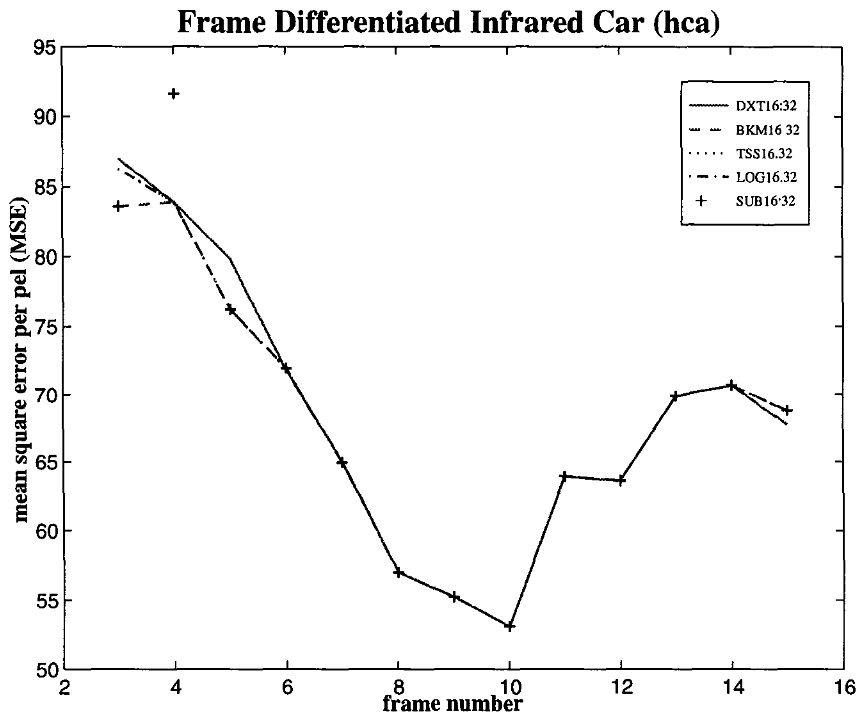


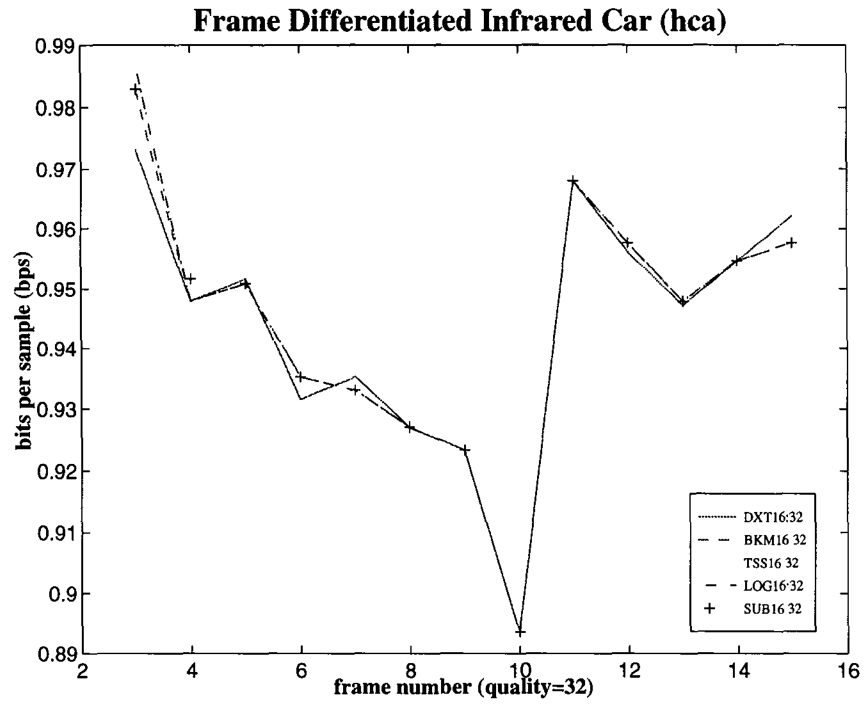
(a) Original (b) Edge extracted (c) Frame differentiated

Figure 5.5: Sequence "Infrared Car" (CAR)

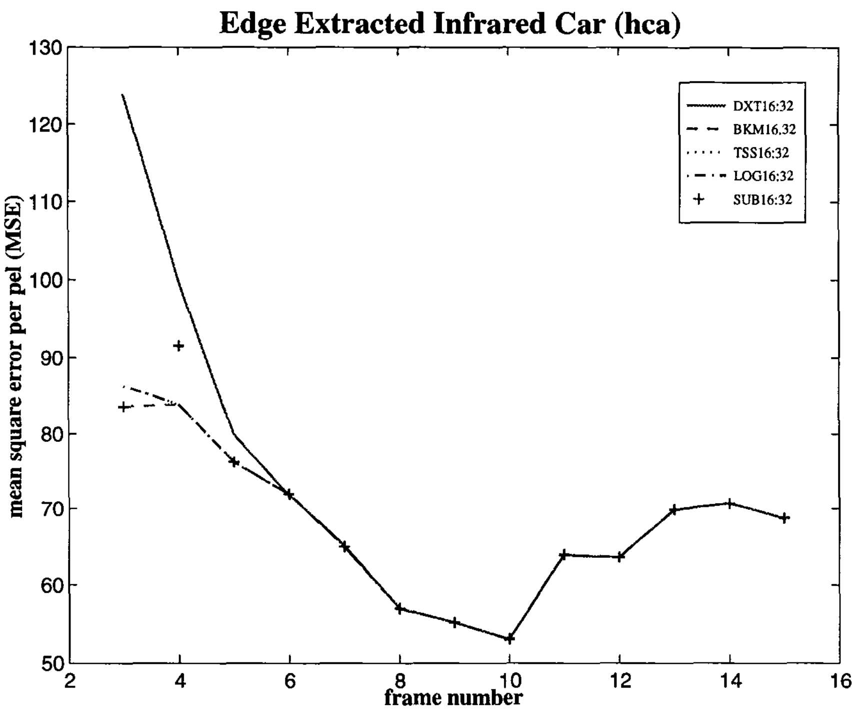
LOG have larger BPS values than the DXT-ME algorithm as can clearly be seen in Table 5.1 and Fig. 5.4. In other words, the motion-compensated residual frames generated by TSS and LOG require more bits than the DXT-ME algorithm to transmit / store after compression. This indicates that the DXT-ME algorithm is better than the logarithmic and three-step fast search block matching approaches for this "Flower Garden" sequence.

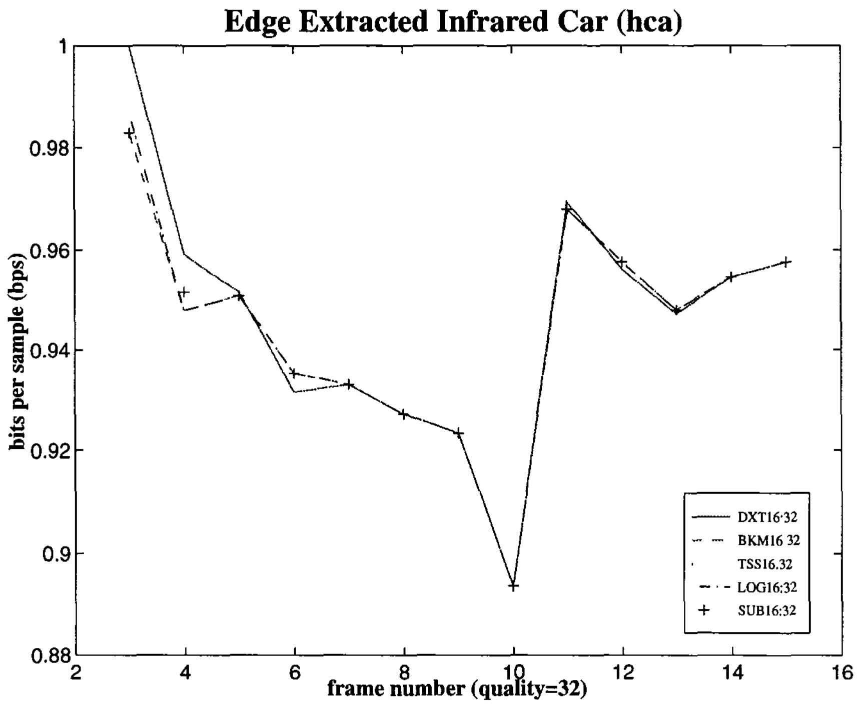
Another simulation is done on the "Infrared Car" sequence which has the frame size 96×112 and one major moving object, the car moving along the curved road towards the camera fixed on the ground. After preprocessed by edge extraction as shown in Fig. 5.5(b), the features of both the car and the background are captured in the edge extracted frames. For the first few frames, the features of the roadside behind the car mix with the features of the car moving along the roadside. This mixture is not desirable and hampers the estimation of the DXT-ME algorithm as revealed by the performance plot in Fig. 5.6 and the performance summary in Table 5.2. As to the frame differentiated images as shown in Fig. 5.5(c), the





(a) Preprocessed with Frame Differentiation





(b) Preprocessed with Edge Extraction

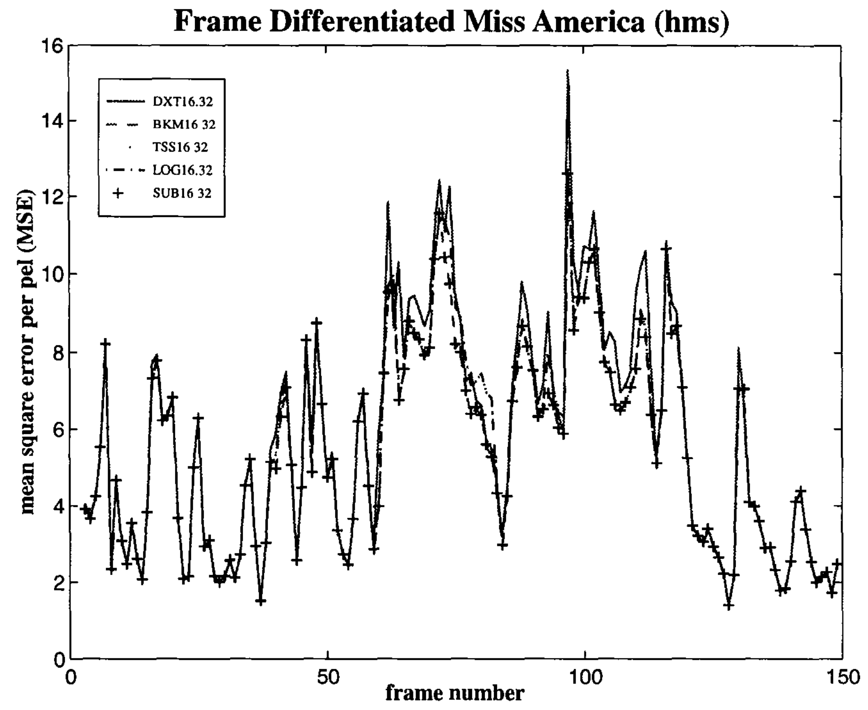
Figure 5.6: Comparison of Overlapping SE-DXT-ME with block matching approaches on "Infrared Car".

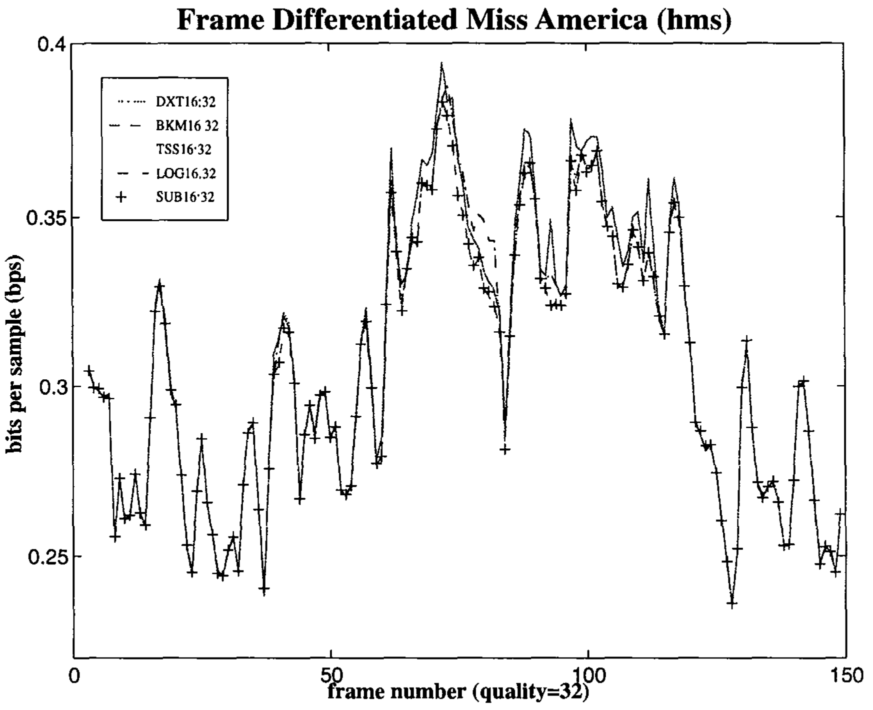
Approach	MSE	MSE difference	MSE ratio	BPF	BPS	BPS ratio
BKM	67.902	0.000	0%	10156	0.945	0%
Frame Differentiated DXT-ME	68.355	0.453	0.7%	10150	0.944	-0.1%
Edge Extracted DXT-ME	72.518	4.615	6.8%	10177	0.946	0.2%
TSS	68.108	0.206	0.3%	10159	0.945	0.0%
LOG	68.108	0.206	0.3%	10159	0.945	0.0%
SUB	68.493	0.591	0.9%	10159	0.945	0.0%

Table 5.2: Performance summary of the overlapping SE-DXT-ME algorithm with either frame differentiation or edge extraction as preprocessing against full search and fast search block matching approaches (BKM, TSS, LOG, SUB) over the sequence "Infrared Car".

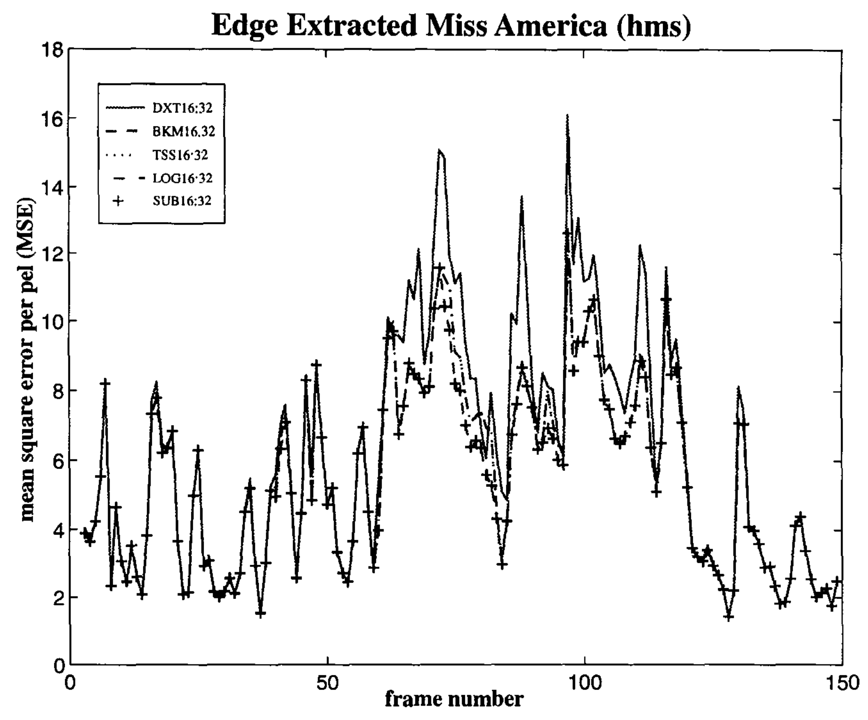
residual energy of the moving car is completely separated from the rest of the scene in most of the preprocessed frames and, therefore, lower MSE values are obtained with this preprocessing function than with edge extraction. In Table 5.2, the frame differentiated DXT-ME algorithm is only 0.7% worse than the full search block matching approach compared to 0.9% for the subsampled approach (SUB) and 0.3% for both LOG and TSS while the edge extracted DXT-ME has a MSE ratio 6.8%. However, if we compare the BPS values, we find that the frame differentiated DXT-ME requires a little less bits on average for the JPEG compressed residual frames than the full search approach (BKM).

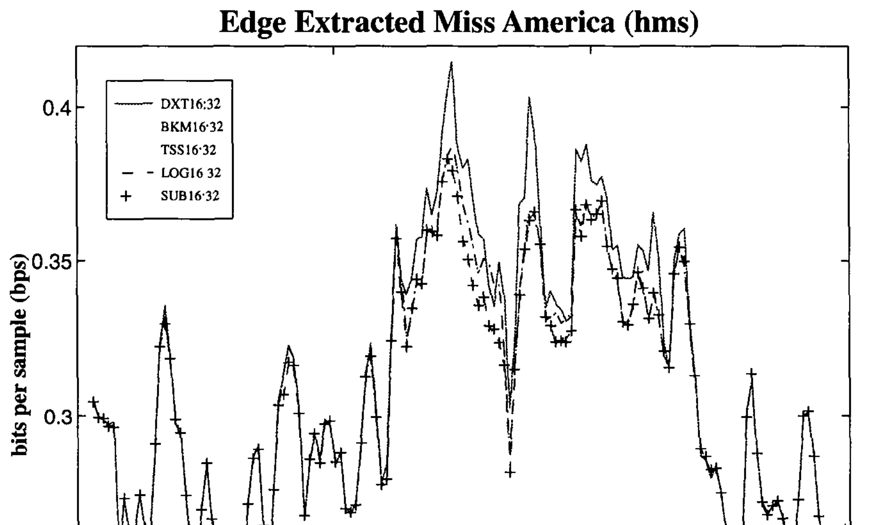
Simulation is also performed on the "Miss America" sequence in QCIF format of which each frame has 176×144 pixels. This sequence not only has translational motion of the head and shoulders but also the mouth and eyes open and close. This makes the task of motion estimation difficult for this sequence but the DXT-





(a) Preprocessed with Frame Differentiation





(b) Preprocessed with Edge Extraction

Figure 5.7: Comparison of Overlapping SE-DXT-ME with block matching approaches on "Miss America" in QCIF format

Approach	MSE	MSE difference	MSE ratio	BPF	BPS	BPS ratio
BKM	5.448	0.000	0%	7714	0.304	0%
Frame Differentiated DXT-ME	5.823	0.374	6.9%	7786	0.307	0.9%
Edge Extracted DXT-ME	6.229	0.781	14.3%	7865	0.310	2.0%
TSS	5.561	0.112	2.1%	7749	0.306	0.5%
LOG	5.561	0.113	2.1%	7749	0.306	0.5%
SUB	5.466	0.017	0.3%	7716	0.304	0.0%

Table 5.3: Performance summary of the overlapping SE-DXT-ME algorithm with either frame differentiation or edge extraction as preprocessing against full search and fast search block matching approaches (BKM, TSS, LOG, SUB) over the sequence "Miss America" in QCIF format.

ME algorithm can still perform reasonably well compared to the block matching methods, as can be found in Fig. 5.7. The performance of all the algorithms is summarized in Table 5.3 where the MSE and BPS values are averaged over the whole sequence from frame 3 to frame 149. As clearly shown in Table 5.3, the frame differentiated DXT-ME is only 6.9% worse than BKM as compared to 2.1% worse for both LOG and TSS and 0.3% worse for SUB. Furthermore, the bits per sample achieved by the frame differentiated DXT-ME is 0.307, only 0.9% larger than BKM. However, the edge extracted DXT-ME performs a little bit worse than the frame differentiated DXT-ME and achieves 2% more of MSE than BKM.

From all the above simulations, it seems that frame differentiation is a better choice for preprocessing than edge extraction due to its capability of removing background features which in some cases affect adversely the performance of the DXT-ME algorithm.

5.4 Rough Count of Computations

In the previous section, we choose the asymptotic complexity for comparison because calculation of the actual number of computations requires the knowledge of specific implementations. However, in application of the DXT-ME algorithm to video coding, we try to make a rough count of computations required by the algorithm based on the straight forward software implementation.

In DCT-based motion-compensated video coding, DCT, IDCT and peak searching are required and therefore we will count only the number of operations required in the pseudo phase computation. At each pixel position, we can solve the 4×4 linear equation by multiplying the transpose of the system matrix $\mathbf{Z}_{t-1}(k,l)$ with the DCT vector $\vec{\mathbf{x}}_t(k,l)$ because $\mathbf{Z}_{t-1}(k,l)$ is a unitary matrix. The multiplication of a matrix with a vector requires 16 multiplications and 12 additions. Therefore, the total number of operations is 7168 for a 16×16 block and 28672 for a corresponding overlapped block (32×32) while the BKM-ME approach requires 130816 additions/subtractions for block size 16×16 and search area 32×32. Still the number of operations required by the DXT-ME algorithm is smaller than BKM-ME. Further reduction of computations can be achieved by exploiting various properties in the algorithm. For example, if the denominator is found to be ill-conditioned, it is possible to skip any further computation and set the pseudo phase at that index position as zero. In this way, the required number of operations is reduced. Of course, the exact number of required operations must be counted based on the actual implementation.

Chapter 6

Interpolation-Free Subpixel Motion Estimation in DCT Domain

Accurate estimation of displacement or location of a signal or image is important in many applications of signal and image processing such as time delay estimation [42], target tracking [67], non-contact measurement [82, 4], remote sensing [6, 16], computer vision [3], image registration [13, 73], and so on. In video coding, motion estimation is proved to be very useful for reduction of temporal redundancy. Therefore, a number of motion estimation algorithms have been devised solely for video coding [56, 15] and numerous VLSI architectures have been designed for practical video applications [61]. To further improve the compression rate, motion estimation with subpixel accuracy is essential because movements in a video sequence are not necessarily multiples of the sampling grid distance in the rectangular sampling grid of a camera. It is shown that significant improvement of coding gain can be obtained with motion estimation of half pixel or finer accuracy [26]. Further investigation reveals that the temporal prediction error variance is generally decreased by subpixel motion compensation but beyond a certain "critical accuracy" the possibility of further improving prediction by more accurate motion compensation is small [20]. As suggested in [26, 19], motion compensation with 1/4-pel accuracy is sufficiently accurate for broadcast TV signals, but for videophone signals, half-pel accuracy is good enough. As a result, motion compensation with half-pel accuracy is recommended in MPEG standards [54, 55]. Implementations of half-pel motion estimation now exist [74, 5, 8].

Many subpixel motion estimation schemes have been proposed over the years [3, 56, 15]. The most commonly used spatial-domain fractional-pel motion estimation algorithms such as the block matching approach [51, 19, 14] and the pel-recursive approach [57, 58], require interpolation of images through bilinear, Lagrange, or other interpolation methods [65]. However, interpolation not only increases the complexity and data flow of a coder but also may adversely affect the accuracy of motion estimates from the interpolated images [19]. It is more desirable that subpixel accuracy of motion estimates can be obtained without interpolating the images. In the category of frequency-domain methods, the phase correlation technique [72, 87, 44] is reported to provide accurate estimates without inter-pixel interpolation but is based on the Fast Fourier Transform (FFT), which is incompatible with DCT-based video coding standards and requires a large search window at a high computational cost. Other FFT-based approaches such as in [34, 36] also have similar drawbacks.

Due to the fact that the motion compensated DCT-based hybrid approach is the backbone of several international video coding standards such as CCITT H.261 [21], MPEG1 [54], MPEG2 [55], and the emerging HDTV [7] and H.263 [22] standards, it is more desirable to estimate motion with fractional-pel accuracy *without* any inter-pixel interpolation at a low computational cost in the DCT domain so that seamless integration of the motion compensation unit with the spatial compression unit is possible.

Based upon the concept of pseudo-phases in DCT coefficients and the sinusoidal orthogonal principles, a DCT-based integer-pel motion estimation scheme (DXT-ME) of very low computational complexity ($O(N^2)$ as opposed to $O(N^4)$ for the widely used Full Search Block Matching Algorithm) was proposed in Chapter 4 to realize the fully DCT-based video coder design. In this chapter, we further explore this DCT-based concept at the subpixel level and show that if the spatial sampling of images satisfies the Nyquist criterion, the subpixel motion information is preserved in the pseudo phases of DCT coefficients of moving images. Furthermore it can be shown that with appropriate modification, the sinusoidal orthogonal principles can still be applicable except that an impulse function is replaced by a sinc function whose peak position reveals subpixel displacement. Therefore, exact subpixel motion displacement can be obtained without the use of interpolation. From these observations, we can develop a set of subpixel DCT-based motion estimation algorithms, that are fully compatible with the integer-pel motion estimator, for low-complexity and high-throughput video applications.

In this chapter, we discuss the pseudo phases carrying subpixel motion information in Section 6.1 and the subpel sinusoidal orthogonal principles in Section 6.2 for objects moving out of synchronization with the sampling grid. In Section 6.3, we propose the DCT-based half-pel (HDXT-ME) and quarter-pel (QDXT-ME and Q4DXT-ME) motion estimation algorithms whose simulation results on actual video sequences of different characteristics are presented in Section 6.4 in comparison with the popular block matching approaches.

6.1 Pseudo Phases at Subpixel Level

6.1.1 One Dimensional Signal Model

Without loss of generality, let us consider the one dimensional model in which a continuous signal $x_c(t)$ and its shifted version $x_c(t-d)$ are sampled at a sampling frequency $1/T$ to generate two sample sequences $\{x_1(n) = x_c(nT)\}$ and $\{x_2(n) = x_c(nT-d)\}$, respectively. Let us define the DCT and DST coefficients as

$$X_i^C(k) \triangleq \mathrm{DCT}\{x_i\} = \frac{2C(k)}{N}\sum_{n=0}^{N-1} x_i(n)\cos\frac{k\pi}{N}(n+\frac{1}{2}), \quad (6.1)$$

$$X_i^S(k) \triangleq \mathrm{DST}\{x_i\} = \frac{2C(k)}{N}\sum_{n=0}^{N-1} x_i(n)\sin\frac{k\pi}{N}(n+\frac{1}{2}), \quad (6.2)$$

where $$C(k) = \begin{cases} \frac{1}{\sqrt{2}}, & \text{for } k=0 \text{ or } N, \\ 1, & \text{otherwise,} \end{cases}$$

for $i = 1$ or 2. By using the sinusoidal relationship:

$$\cos\frac{k\pi}{N}(n+\frac{1}{2}) = \frac{1}{2}[e^{j\frac{k\pi}{N}(n+\frac{1}{2})} + e^{-j\frac{k\pi}{N}(n+\frac{1}{2})}], \quad (6.3)$$

$$\sin\frac{k\pi}{N}(n+\frac{1}{2}) = \frac{1}{2j}[e^{j\frac{k\pi}{N}(n+\frac{1}{2})} - e^{-j\frac{k\pi}{N}(n+\frac{1}{2})}], \quad (6.4)$$

we can show that the DCT/DST and DFT coefficients are related as follows:

$$X_i^C(k) = \frac{C(k)}{N}[\tilde{X}_i^Z(-k)e^{j\frac{k\pi}{2N}} + \tilde{X}_i^Z(k)e^{-j\frac{k\pi}{2N}}], \text{ for } k=0,\ldots,N-1, \quad (6.5)$$

$$X_i^S(k) = \frac{C(k)}{jN}[\tilde{X}_i^Z(-k)e^{j\frac{k\pi}{2N}} - \tilde{X}_i^Z(k)e^{-j\frac{k\pi}{2N}}], \text{ for } k=1,\ldots,N, \quad (6.6)$$

where $\{\tilde{X}_i^Z(k)\}$ is the DFT of the zero-padded sequence $\{x_i^Z(n)\}$ defined as

$$x_i^Z(n) = \begin{cases} x_i(n), & \text{for } n=0,\ldots,N-1, \\ 0, & \text{for } n=N,\ldots,2N-1, \end{cases} \quad (6.7)$$

so that

$$\tilde{X}_i^Z(k) \triangleq \mathrm{DFT}\{x_i^Z\} = \sum_{n=0}^{N-1} x_i(n)e^{-j\frac{2k\pi n}{2N}}, \text{ for } k=0,\ldots,2N-1. \quad (6.8)$$

From the sampling theorem, we know that the Discrete Time Fourier Transform (DTFT) of sequences $x_1(n)$ and $x_2(n)$ are related to the Fourier Transform (FT) of $x_c(t)$, $X_c(\Omega)$, in the following way:

$$X_1(\omega) \triangleq \text{DTFT}\{x_1\} = \frac{1}{T}\sum_l X_c(\frac{\omega - 2\pi l}{T}), \tag{6.9}$$

$$X_2(\omega) \triangleq \text{DTFT}\{x_2\} = \frac{1}{T}\sum_l X_c(\frac{\omega - 2\pi l}{T})\ e^{-j(\frac{\omega-2\pi l}{T})d}. \tag{6.10}$$

Furthermore, if $X_c(\Omega)$ is bandlimited in the baseband $(-\frac{\pi}{T}, \frac{\pi}{T})$, then for $\Omega = \frac{\omega}{T} \in (-\frac{\pi}{T}, \frac{\pi}{T})$,

$$X_1(\Omega T) = \frac{1}{T}X_c(\Omega), \tag{6.11}$$

$$X_2(\Omega T) = \frac{1}{T}X_c(\Omega)\ e^{-j\Omega d}. \tag{6.12}$$

Thus, the DFT of $x_1(n)$ and $x_2(n)$ are

$$\tilde{X}_1(k) \triangleq \text{DFT}\{x_1\} = \sum_{n=0}^{N-1} x_1(n)e^{-j\frac{2\pi kn}{N}} = X_1(\frac{2\pi k}{N}) = \frac{1}{T}X_c(\frac{2\pi k}{NT}), \tag{6.13}$$

$$\tilde{X}_2(k) \triangleq \text{DFT}\{x_2\} = \sum_{n=0}^{N-1} x_2(n)e^{-j\frac{2\pi kn}{N}} = X_2(\frac{2\pi k}{N}) = \frac{1}{T}X_c(\frac{2\pi k}{NT})\ e^{-j\frac{2\pi kd}{NT}}, \tag{6.14}$$

whereas the DFT of $x_1^Z(n)$ and $x_2^Z(n)$ become

$$\tilde{X}_1^Z(k) = X_1(\frac{\pi k}{N}) = \frac{1}{T}X_c(\frac{\pi k}{NT}), \tag{6.15}$$

$$\tilde{X}_2^Z(k) = X_2(\frac{\pi k}{N}) = \frac{1}{T}X_c(\frac{\pi k}{NT})\ e^{-j\frac{\pi kd}{NT}}. \tag{6.16}$$

Therefore,

$$X_2(\frac{\pi k}{N}) = X_1(\frac{\pi k}{N})e^{-j\frac{\pi kd}{NT}}. \tag{6.17}$$

Substituting (6.17) back into (6.5)-(6.6), we get

$$X_2^C(k) \;=\; \frac{C(k)}{N}[\tilde{X}_1^Z(-k)e^{j\frac{k\pi d}{NT}}e^{j\frac{k\pi}{2N}} + \tilde{X}_1^Z(k)e^{-j\frac{k\pi d}{NT}}e^{-j\frac{k\pi}{2N}}], \tag{6.18}$$

$$\text{for } k = 0, \ldots, N-1,$$

$$X_2^S(k) = \frac{C(k)}{jN}[\tilde{X}_1^Z(-k)e^{j\frac{k\pi d}{NT}}e^{j\frac{k\pi}{2N}} - \tilde{X}_1^Z(k)e^{-j\frac{k\pi d}{NT}}e^{-j\frac{k\pi}{2N}}], \tag{6.19}$$

$$\text{for } k = 1, \ldots, N.$$

Using the sinusoidal relationship in (6.4) to change natural exponents back to cosine/sine, we finally obtain the relationship between $x_1(n)$ and $x_2(n)$ in the DCT/DST domain:

$$X_2^C(k) = \frac{2C(k)}{N}\sum_{n=0}^{N-1} x_1(n)\cos\frac{k\pi}{N}(n + \frac{d}{T} + \frac{1}{2}), \text{ for } k = 0, \ldots, N-1, \tag{6.20}$$

$$X_2^S(k) = \frac{2C(k)}{N}\sum_{n=0}^{N-1} x_1(n)\sin\frac{k\pi}{N}(n + \frac{d}{T} + \frac{1}{2}), \text{ for } k = 1, \ldots, N. \tag{6.21}$$

We conclude the result in the following theorem:

THEOREM 1 *If a continuous signal $x_c(t)$ is $\frac{\pi}{T}$-bandlimited and the sampled sequences of $x_c(t)$ and $x_c(t-d)$ are $\{x_c(nT)\}$ and $\{x_c(nT-d)\}$, respectively, then their DCT and DST are related by*

$$DCT\{x_c(nT-d)\} = DCT_{\frac{d}{T}}\{x_c(nT)\}, \tag{6.22}$$

$$DST\{x_c(nT-d)\} = DST_{\frac{d}{T}}\{x_c(nT)\}, \tag{6.23}$$

where

$$DCT_\alpha\{x\} \triangleq \frac{2C(k)}{N}\sum_{n=0}^{N-1} x(n)\cos\frac{k\pi}{N}(n + \alpha + \frac{1}{2}), \tag{6.24}$$

$$DST_\beta\{x\} \triangleq \frac{2C(k)}{N}\sum_{n=0}^{N-1} x(n)\sin\frac{k\pi}{N}(n + \beta + \frac{1}{2}), \tag{6.25}$$

are the DCT and DST with α and β shifts in their kernels, respectively. Here d is the shift amount and T is the sampling interval, but d/T is not necessarily an integer.

6.1.2 Two Dimensional Image Model

Consider a moving object casting a continuous intensity profile $I_t(u,v)$ on a camera plane of the continuous coordinate (u,v) where the subscript t denotes the frame number. This intensity profile is then digitized on the fixed sampling grid of the camera with a sampling distance d to generate the current frame of pixels $x_t(m,n)$ shown in Fig. 6.1(a) where m and n are integers. Further assume that the displacement of the object between the frames $t-1$ and t is (d_u, d_v) such that $I_t(u,v) = I_{t-1}(u-d_u, v-d_v)$ where $d_u = (m_u + \nu_u)d = \lambda_u d$ and $d_v = (m_v + \nu_v)d = \lambda_v d$. Here m_u and m_v are the integer components of the displacement, and ν_u and $\nu_v \in (-\frac{1}{2}, \frac{1}{2}]$. Therefore,

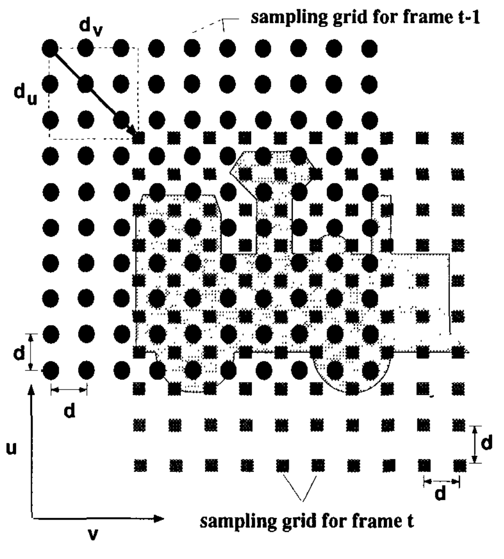
$$x_t(m,n) = I_t(md, nd) = I_{t-1}(md - d_u, nd - d_v),$$
$$x_{t-1}(m,n) = I_{t-1}(md, nd),$$

as in Fig. 6.1(b). Unlike the case of integer-pel movement, the displacement is not necessarily multiples of the sampling distance d. In other words, ν_u and ν_v do not necessarily equal zero.

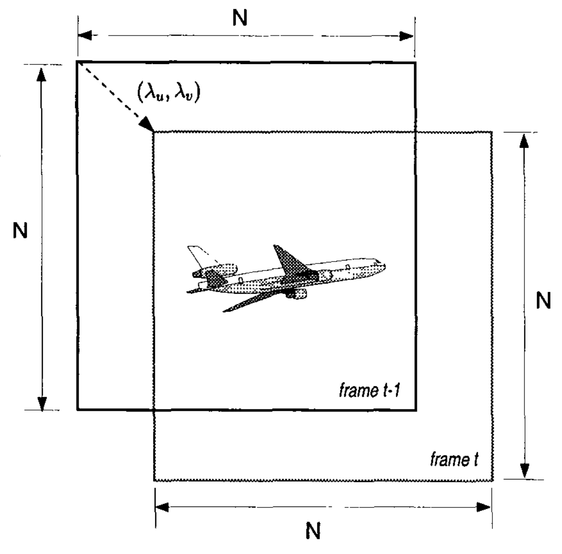
For integer-pel displacements, i. e. $\lambda_u = m_u$ and $\lambda_v = m_v$, the pseudo phases are computed by solving the *pseudo-phase motion equation* at (k,l):

$$\mathbf{Z}_{t-1}(k,l) \cdot \vec{\theta}_{m_u,m_v}(k,l) = \vec{\mathbf{x}}_t(k,l), \text{ for } k,l \in \mathcal{N} \tag{6.26}$$

where $\mathcal{N} = \{1, \ldots, N-1\}$, $\vec{\theta}_{m_u,m_v}$ is the pseudo-phase vector, and the 4×4 *system matrix* $\mathbf{Z}_{t-1}$ and the vector $\vec{\mathbf{x}}_t$ are composed from the 2D-DCT-II of $x_{t-1}(m,n)$ and



(a)



(b)

Figure 6.1: (a) The black dots and the gray squares symbolize the sampling grids for frames $I_{t-1}(u,v)$ and $I_t(u,v)$ at a sampling distance d respectively. These two frames are aligned on the common object displaced by (d_u, d_v) in the continuous coordinate (u,v). (b) Two digitized images of consecutive frames, $x_{t-1}(m,n)$ and $x_t(m,n)$, are aligned on the common object moving $(\lambda_u, \lambda_v) = (d_u/d, d_v/d)$ pixels southeast.

the 2D-DCT-I of $x_t(m,n)$ respectively:

$$\mathbf{Z}_{t-1}(k,l) = \begin{bmatrix} Z^{cc}_{t-1}(k,l) & -Z^{cs}_{t-1}(k,l) & -Z^{sc}_{t-1}(k,l) & +Z^{ss}_{t-1}(k,l) \\ Z^{cs}_{t-1}(k,l) & +Z^{cc}_{t-1}(k,l) & -Z^{ss}_{t-1}(k,l) & -Z^{sc}_{t-1}(k,l) \\ Z^{sc}_{t-1}(k,l) & -Z^{ss}_{t-1}(k,l) & +Z^{cc}_{t-1}(k,l) & -Z^{cs}_{t-1}(k,l) \\ Z^{ss}_{t-1}(k,l) & +Z^{sc}_{t-1}(k,l) & +Z^{cs}_{t-1}(k,l) & +Z^{cc}_{t-1}(k,l) \end{bmatrix},$$

$$\vec{\mathbf{x}}_t(k,l) = \begin{bmatrix} X_t^{cc}(k,l) \\ X_t^{cs}(k,l) \\ X_t^{sc}(k,l) \\ X_t^{ss}(k,l) \end{bmatrix}, \ \vec{\theta}_{m_u,m_v}(k,l) = \begin{bmatrix} g^{CC}_{m_u m_v}(k,l) \\ g^{CS}_{m_u m_v}(k,l) \\ g^{SC}_{m_u m_v}(k,l) \\ g^{SS}_{m_u m_v}(k,l) \end{bmatrix}.$$

Here the 2D-DCT-I of $x_{t-1}(m,n)$ and the 2D-DCT-II of $x_t(m,n)$ are defined in (4.15)-(4.18) and (4.19)-(4.22) respectively where $\{Z^{xx}_{t-1}; xx = cc, cs, sc, ss\}$ can be obtained by a simple rotation (4.72) in Chapter 4 from $\{X^{xx}_{t-1}; xx = cc, cs, sc, ss\}$ which are computed and stored in memory in the previous encoding cycle.

However, for non-integer pel movement, we need to use (6.22)-(6.23) in Theorem 1 to derive the system equation at the subpixel level. If the Fourier transform of the continuous intensity profile $I_t(u,v)$ is $\frac{\pi}{d}$-bandlimited and $I_t(u,v) = I_{t-1}(u-d_u, v-d_v)$, then according to Theorem 1, we can obtain the following 2-D relations:

$$\begin{aligned}
X_t^{cc}(k,l) &= \frac{4}{N^2}C(k)C(l)\sum_{m,n=0}^{N-1} x_{t-1}(m,n)\cos[\frac{k\pi}{N}(m+\lambda_u+\frac{1}{2})]\cos[\frac{l\pi}{N}(n+\lambda_v+\frac{1}{2})] \\
&\quad \text{for } k,l \in \{0,\ldots,N-1\}, && (6.27) \\
X_t^{cs}(k,l) &= \frac{4}{N^2}C(k)C(l)\sum_{m,n=0}^{N-1} x_{t-1}(m,n)\cos[\frac{k\pi}{N}(m+\lambda_u+\frac{1}{2})]\sin[\frac{l\pi}{N}(n+\lambda_v+\frac{1}{2})] \\
&\quad \text{for } k \in \{0,\ldots,N-1\}, l \in \{1,\ldots,N\}, && (6.28) \\
X_t^{sc}(k,l) &= \frac{4}{N^2}C(k)C(l)\sum_{m,n=0}^{N-1} x_{t-1}(m,n)\sin[\frac{k\pi}{N}(m+\lambda_u+\frac{1}{2})]\cos[\frac{l\pi}{N}(n+\lambda_v+\frac{1}{2})] \\
&\quad \text{for } k \in \{1,\ldots,N\}, l \in \{0,\ldots,N-1\}, && (6.29) \\
X_t^{ss}(k,l) &= \frac{4}{N^2}C(k)C(l)\sum_{m,n=0}^{N-1} x_{t-1}(m,n)\sin[\frac{k\pi}{N}(m+\lambda_u+\frac{1}{2})]\sin[\frac{l\pi}{N}(n+\lambda_v+\frac{1}{2})] \\
&\quad \text{for } k,l \in \{1,\ldots,N\}. && (6.30)
\end{aligned}$$

Thus, we can obtain the *pseudo-phase motion equation* at the subpixel level:

$$\mathbf{Z}_{t-1}(k,l) \cdot \vec{\theta}_{\lambda_u,\lambda_v}(k,l) = \vec{\mathbf{x}}_t(k,l), \text{ for } k,l \in \mathcal{N}, \qquad (6.31)$$

where $\vec{\theta}_{\lambda_u,\lambda_v}(k,l) = [g^{CC}_{\lambda_u,\lambda_v}(k,l), g^{CS}_{\lambda_u,\lambda_v}(k,l), g^{SC}_{\lambda_u,\lambda_v}(k,l), g^{SS}_{\lambda_u,\lambda_v}(k,l)]^T$. A similar relationship between the DCT coefficients of $x_t(m,n)$ and $x_{t-1}(m,n)$ at the block boundary can be obtained in the same way as in (4.27)-(4.32).

In (6.31), the pseudo phase vector $\vec{\theta}_{\lambda_u,\lambda_v}(k,l)$ contains the information of the subpixel movement (λ_u, λ_v). In an ideal situation where one rigid object is moving translationally within the block boundary without observable background and noise, we can find $\vec{\theta}_{\lambda_u,\lambda_v}(k,l)$ explicitly in terms of λ_u and λ_v as such:

$$\vec{\theta}_{\lambda_u,\lambda_v}(k,l) = \begin{bmatrix} g^{CC}_{\lambda_u,\lambda_v}(k,l) \\ g^{CS}_{\lambda_u,\lambda_v}(k,l) \\ g^{SC}_{\lambda_u,\lambda_v}(k,l) \\ g^{SS}_{\lambda_u,\lambda_v}(k,l) \end{bmatrix} = \begin{bmatrix} \cos\frac{k\pi}{N}(\lambda_u+\frac{1}{2})\cos\frac{l\pi}{N}(\lambda_v+\frac{1}{2}) \\ \cos\frac{k\pi}{N}(\lambda_u+\frac{1}{2})\sin\frac{l\pi}{N}(\lambda_v+\frac{1}{2}) \\ \sin\frac{k\pi}{N}(\lambda_u+\frac{1}{2})\cos\frac{l\pi}{N}(\lambda_v+\frac{1}{2}) \\ \sin\frac{k\pi}{N}(\lambda_u+\frac{1}{2})\sin\frac{l\pi}{N}(\lambda_v+\frac{1}{2}) \end{bmatrix}. \quad (6.32)$$

6.2 Subpel Sinusoidal Orthogonality Principles

In Chapter 4, estimation of integer-pel displacements in DCT domain utilizes the sinusoidal orthogonal principles:

$$\begin{aligned} \text{IDCT}\{C(k)\cos[\frac{k\pi}{N}(n+\frac{1}{2})]\} &\triangleq \frac{2}{N}\sum_{k=0}^{N-1} C^2(k)\cos[\frac{k\pi}{N}(m+\frac{1}{2})]\cos[\frac{k\pi}{N}(n+\frac{1}{2})] \\ &= \delta(m-n)+\delta(m+n+1), \\ \text{IDST}\{C(k)\sin[\frac{k\pi}{N}(n+\frac{1}{2})]\} &\triangleq \frac{2}{N}\sum_{k=1}^{N} C^2(k)\sin[\frac{k\pi}{N}(m+\frac{1}{2})]\sin[\frac{k\pi}{N}(n+\frac{1}{2})] \\ &= \delta(m-n)-\delta(m+n+1), \end{aligned}$$

where $\delta(n)$ is the discrete impulse function, and m, n are integers. This is no longer valid at the subpixel level.

In (4.9)-(4.8), we replace the integer variables m and n by the real variables u and v and define

$$\bar{L}_c(u,v) \triangleq \sum_{k=0}^{N-1} C^2(k)\cos\frac{k\pi}{N}(u+\frac{1}{2})\cos\frac{k\pi}{N}(v+\frac{1}{2}), \quad (6.33)$$

$$\bar{L}_s(u,v) \triangleq \sum_{k=0}^{N-1} C^2(k) \sin\frac{k\pi}{N}(u+\frac{1}{2}) \sin\frac{k\pi}{N}(v+\frac{1}{2}). \tag{6.34}$$

Since

$$\begin{aligned}
\bar{L}_c(u,v) &\triangleq \sum_{k=0}^{N-1} C^2(k) \cos\frac{k\pi}{N}(u+\frac{1}{2}) \cos\frac{k\pi}{N}(v+\frac{1}{2}) \\
&= -\frac{1}{2} + \sum_{k=0}^{N-1} \cos\frac{k\pi}{N}(u+\frac{1}{2}) \cos\frac{k\pi}{N}(v+\frac{1}{2}) \\
&= \frac{1}{2}[-1 + \sum_{k=0}^{N-1} \cos\frac{k\pi}{N}(u-v) + \sum_{k=0}^{N-1} \cos\frac{k\pi}{N}(u+v+1)] \\
&= -\frac{1}{2} + \frac{1}{2}[\xi(u-v) + \xi(u+v+1)],
\end{aligned}$$

and

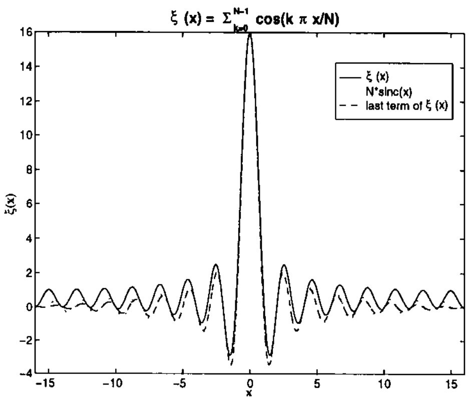
$$\begin{aligned}
\bar{L}_s(u,v) &\triangleq \sum_{k=1}^{N} C^2(k) \sin\frac{k\pi}{N}(u+\frac{1}{2}) \sin\frac{k\pi}{N}(v+\frac{1}{2}) \\
&= \frac{1}{2}\sin[\pi(u+\frac{1}{2})]\sin[\pi(v+\frac{1}{2})] + \sum_{k=0}^{N-1} \sin\frac{k\pi}{N}(u+\frac{1}{2}) \sin\frac{k\pi}{N}(v+\frac{1}{2}) \\
&= \frac{1}{2}\sin[\pi(u+\frac{1}{2})]\sin[\pi(v+\frac{1}{2})] + \frac{1}{2}[\sum_{k=0}^{N-1} \cos\frac{k\pi}{N}(u-v) - \sum_{k=0}^{N-1} \cos\frac{k\pi}{N}(u+v+1)] \\
&= \frac{1}{2}\sin[\pi(u+\frac{1}{2})]\sin[\pi(v+\frac{1}{2})] + \frac{1}{2}[\xi(u-v) - \xi(u+v+1)],
\end{aligned}$$

we show that

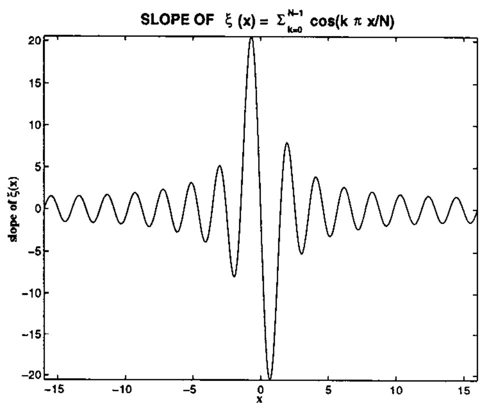
$$\bar{L}_c(u,v) = -\frac{1}{2} + \frac{1}{2}[\xi(u-v) + \xi(u+v+1)], \tag{6.35}$$
$$\bar{L}_s(u,v) = \frac{1}{2}\sin[\pi(u+\frac{1}{2})]\sin[\pi(v+\frac{1}{2})] + \frac{1}{2}[\xi(u-v) - \xi(u+v+1)], \tag{6.36}$$

where

$$\begin{aligned}
\xi(x) &\triangleq \sum_{k=0}^{N-1} \cos(\frac{k\pi}{N}x) \\
&= \frac{1}{2}[\sum_{k=0}^{N-1} e^{j\frac{k\pi}{N}x} + \sum_{k=0}^{N-1} e^{-j\frac{k\pi}{N}x}] \\
&= \frac{1}{2}[\frac{1-e^{j\pi x}}{1-e^{j\frac{\pi x}{N}}} + \frac{1-e^{-j\pi x}}{1-e^{-j\frac{\pi x}{N}}}]
\end{aligned}$$



(a) $\xi(x)$



(b) slope of $\xi(x)$

Figure 6.2: Plot of $\xi(x) = \sum_{k=0}^{N-1} \cos(\frac{k\pi}{N}x)$ and its slope for $N = 16$. Observe the similarity between the curves of N*sinc(x) and the last term of ξ.

$$
\begin{aligned}
&= \frac{1}{2}[\frac{1 - \cos\pi x - \cos\frac{\pi x}{N} + \cos\frac{\pi x}{N}(N-1)}{1 - \cos\frac{\pi x}{N}}] \\
&= \frac{1}{2}[(1 - \cos\pi x) + \frac{\sin(\pi x)\sin(\frac{\pi x}{N})}{1 - \cos(\frac{\pi x}{N})}] \\
&= \frac{1}{2}[1 - \cos\pi x + \sin\pi x \cdot \frac{\cos\frac{\pi x}{2N}}{\sin\frac{\pi x}{2N}}]. \qquad (6.37)
\end{aligned}
$$

If $\frac{(\pi x)}{(2N)}$ is so small that the second and higher order terms of $\frac{(\pi x)}{(2N)}$ can be ignored, then $\cos\frac{\pi x}{2N} \approx 1$, $\sin\frac{\pi x}{2N} \approx \frac{\pi x}{2N}$. Thus,

$$\xi(x) \approx \frac{1}{2}[1 - \cos\pi x] + N\text{sinc}(x), \qquad (6.38)$$

where $\text{sinc}(x) \triangleq \sin(\pi x)/(\pi x)$. For large N, $\xi(x)$ is approximately a sinc function whose largest peak can be identified easily at $x = 0$ as depicted in Fig. 6.2(a), where $\xi(x)$ closely resembles $N \cdot \text{sinc}(x)$, especially when x is small. The slope of $\xi(x)$ is also plotted in Fig. 6.2(b) which shows the sharpness of $\xi(x)$.

A closer look at (6.35)-(6.36) reveals that either $\bar{L}_c(u,v)$ or $\bar{L}_s(u,v)$ consists of ξ functions and one extra term which is not desirable. In order to obtain a pure form of sinc functions similar to (4.9)-(4.8), we define two modified functions $L_c(u, v)$ and $L_c(u, v)$ as follows:

$$L_c(u,v) \triangleq \sum_{k=0}^{N-1} \cos \frac{k\pi}{N}(u+\frac{1}{2}) \cos \frac{k\pi}{N}(v+\frac{1}{2}), \tag{6.39}$$

$$L_s(u,v) \triangleq \sum_{k=1}^{N-1} \sin \frac{k\pi}{N}(u+\frac{1}{2}) \sin \frac{k\pi}{N}(v+\frac{1}{2}). \tag{6.40}$$

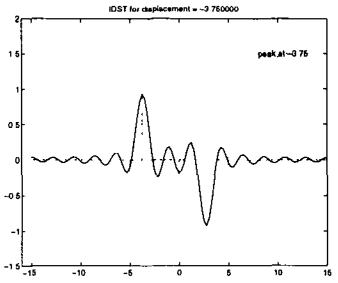
Then we can show that

$$L_c(u,v) = \frac{1}{2}[\xi(u-v) + \xi(u+v+1)], \tag{6.41}$$

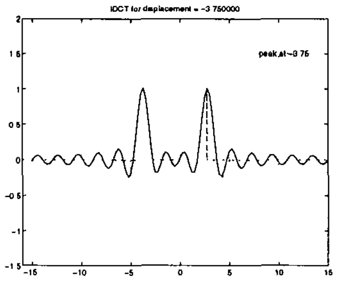
$$L_s(u,v) = \frac{1}{2}[\xi(u-v) - \xi(u+v+1)]. \tag{6.42}$$

Equations (6.39)-(6.42) are the equivalent form of the sinusoidal orthogonal principles (4.9)-(4.8) at the subpixel level. The sinc functions at the right hand side of the equations are the direct result of the rectangular window inherent in the DCT transform [60]. Fig. 6.3 (a) and (b) illustrate $L_s(x, -3.75)$ and $L_c(x, -3.75)$ respectively where two ξ functions are interacting with each other but their peak positions clearly indicate the displacement. However, when the displacement v is small (in the neighborhood of -0.5), $\xi(u-v)$ and $\xi(u+v+1)$ move close together and addition/subtraction of $\xi(u-v)$ and $\xi(u+v+1)$ changes the shape of L_s and L_c. As a result, neither L_s nor L_c looks like two ξ functions and the peak positions of L_s and L_c are different from those of $\xi(u-v)$ and $\xi(u+v+1)$, as demonstrated in Fig. 6.3 (c) and (d) respectively where the peak positions of $L_s(x, -0.75)$ and $L_c(x, -0.75)$ are -1.25 and -0.5, differing from the true displacement -0.75. In the extreme case, $\xi(u-v)$ and $\xi(u+v+1)$ cancel out each other when the displacement is -0.5 such that $L_s(x, -0.5) \equiv 0$ as shown in Fig. 6.3(e).

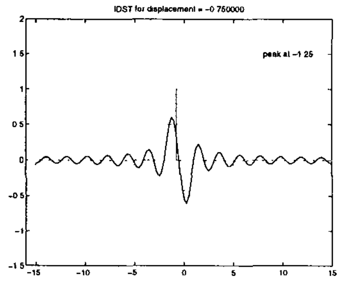
Fortunately we can eliminate the adverse interaction of the two ξ functions by simply adding L_c and L_s together since $L_c(x, v) + L_s(x, v) = \xi(x-v)$ as depicted in



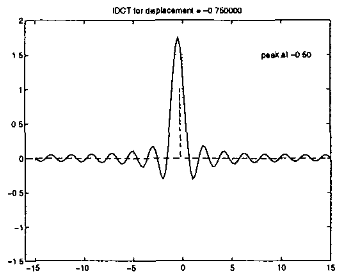
(a) $L_s(x, -3.75)$



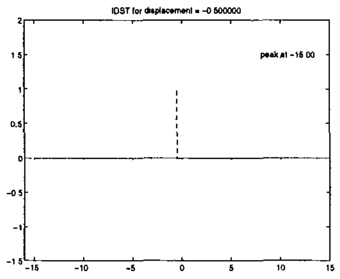
(b) $L_c(x, -3.75)$



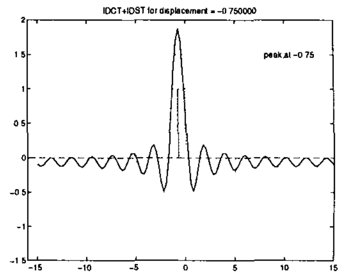
(c) $L_s(x, -0.75)$



(d) $L_c(x, -0.75)$



(e) $L_s(x, -0.5)$



(f) $L_c(x, -0.75)$ $+L_s(x, -0.75)$

Figure 6.3: Illustration of sinusoidal orthogonal principles at the subpixel level for different displacements.

Fig. 6.3(f) where the sum $L_c(x, -0.75) + L_s(x, -0.75)$ behaves like a sinc function and its peak position coincides with the displacement. Furthermore, due to the sharpness of this ξ function, we can accurately pinpoint the peak position under a noisy situation and in turn determine the motion estimate. This property enables us to devise flexible and scalable subpixel motion estimation algorithms in the subsequent sections.

6.3 DCT-Based Subpixel Motion Estimation

In this section, we apply the subpixel sinusoidal orthogonal principles to develop an exact subpixel motion displacement scheme without the use of interpolation to estimate half-pel and quarter-pel movements for high quality video applications.

6.3.1 DCT-Based Half-Pel Motion Estimation Algorithm (HDXT-ME)

From (6.31) in Section 6.1, we know that the subpixel motion information is hidden, though not obvious, in the pseudo phases. To obtain subpixel motion estimates, we can directly compute the pseudo phases in (6.31) and then locate the peaks of the sinc functions after applying the subpixel sinusoidal orthogonal principles (6.39)-(6.42) to the pseudo phases. Alternatively, we can have better flexibility and scalability by first using the DXT-ME algorithm to get an integer-pel motion estimate and then utilizing the pseudo phase functions $f(k,l)$ and $g(k,l)$ computed in the DXT-ME algorithm as in Table 4.2 to increase estimation accuracy to half-pel, due to the fact that (6.31) has exactly the same form as (6.26). Specifically, based upon the subpixel sinusoidal orthogonal principles (6.39)-(6.42), the subpixel motion information can be extracted in the form of impulse functions with peak positions closely related to the displacement.

For the sake of flexibility and modularity in design and further reduction in complexity, we adopt the second approach to devise a motion estimation scheme with arbitrary fractional pel accuracy by applying the subpixel sinusoidal orthogonal principles to the pseudo phase functions passed from the DXT-ME algorithm. The limitation of estimation accuracy will only be determined by the interaction effects of the ξ functions as explained in Section 6.2 and the slope of the ξ function at and around zero and how well the subpixel motion information is preserved in the pseudo phases after sampling.

Sign of DSC Peak	Sign of DCS Peak	Peak Index	Motion Direction
$+$	$+$	(λ_u, λ_v)	northeast
$+$	$-$	$(\lambda_u, -(\lambda_v + 1))$	southeast
$-$	$+$	$(-(\lambda_u + 1), \lambda_v)$	northwest
$-$	$-$	$(-(\lambda_u + 1), -(\lambda_v + 1))$	southwest

Table 6.1: Determination of direction of movement (λ_u, λ_v) from the signs of $\overline{DSC}$ and $\overline{DCS}$

We define $\overline{DCS}(u,v)$ and $\overline{DSC}(u,v)$ as follows:

$$\overline{DCS}(u,v) \triangleq \sum_{k=0}^{N-1} \sum_{l=1}^{N-1} [\frac{f(k,l)}{C(k)C(l)}] \cos\frac{k\pi}{N}(u+\frac{1}{2}) \sin\frac{l\pi}{N}(v+\frac{1}{2}), \tag{6.43}$$

$$\overline{DSC}(u,v) \triangleq \sum_{k=1}^{N-1} \sum_{l=0}^{N-1} [\frac{g(k,l)}{C(k)C(l)}] \sin\frac{k\pi}{N}(u+\frac{1}{2}) \cos\frac{l\pi}{N}(v+\frac{1}{2}). \tag{6.44}$$

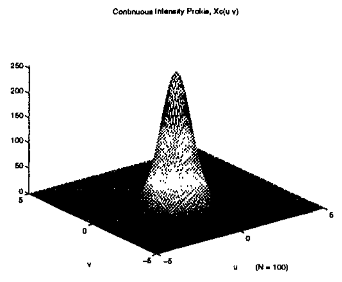
Thus, from the subpixel sinusoidal orthogonal principles (6.39)-(6.42) and the definitions of $f(k,l)$ and $g(k,l)$ in Table 4.2, we can show that

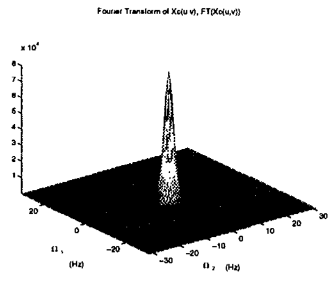
$$\overline{DCS}(u,v) = \frac{1}{4}[\xi(u-\lambda_u) + \xi(u+\lambda_u+1)] \cdot [\xi(v-\lambda_v) - \xi(v+\lambda_v+1)], \tag{6.45}$$

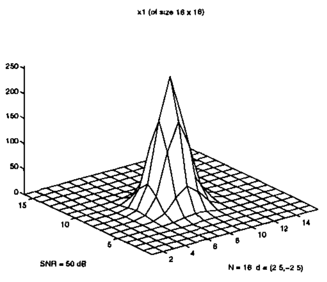
$$\overline{DSC}(u,v) = \frac{1}{4}[\xi(u-\lambda_u) - \xi(u+\lambda_u+1)] \cdot [\xi(v-\lambda_v) + \xi(v+\lambda_v+1)]. \tag{6.46}$$

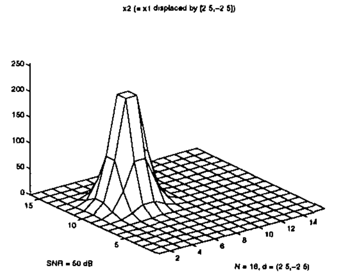
The rules to determine subpixel motion direction are summarized in Table 6.1 and similar to the rules in determination of integer-pel motion direction.

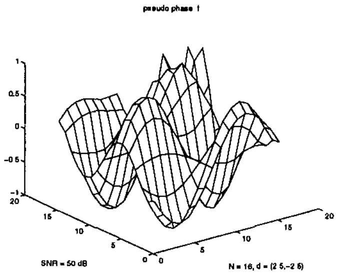
Fig. 6.4 illustrates how to estimate subpixel displacements in the DCT domain. Fig. 6.4 (c) and (d) depict the input images $x_1(m,n)$ of size 16×16 (i.e. $N = 16$) and $x_2(m,n)$ displaced from $x_1(m,n)$ by $(2.5, -2.5)$ respectively at SNR $= 50$ dB. These two images are sampled on a rectangular grid at a sampling distance $d = 0.625$ from the continuous intensity profile $x_c(u,v) = \exp(-(u^2+v^2))$ for $u, v \in [-5, 5]$

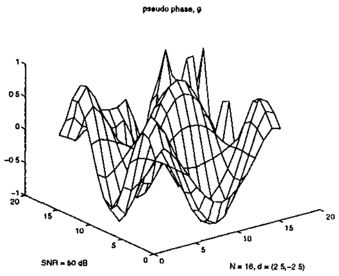

(a) continuous intensity profile $x_c(u,v)$

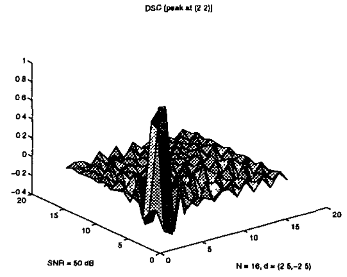

(b) FT of $x_c(u,v)$

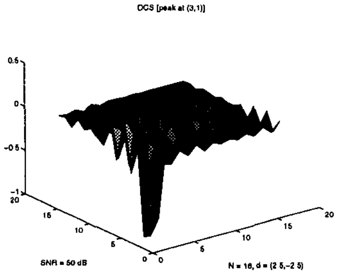

(c) 16×16 $x_1(m,n)$

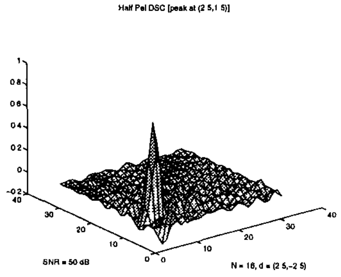

(d) 16×16 $x_2(m,n)$

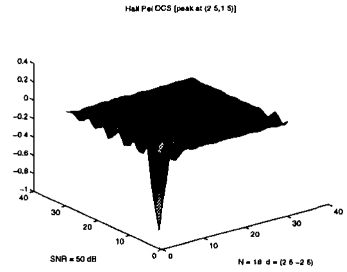

(e) pseudo phase $f(k,l)$

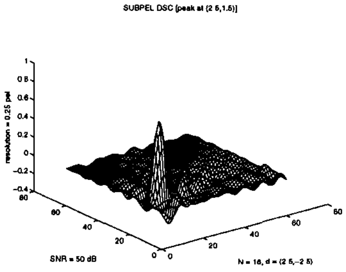

(f) pseudo phase $g(k,l)$


(g) $DSC(m,n)$


(h) $DCS(m,n)$


(i) $\overline{DSC}(u,v)$ for $u,v = 0:0.5:15$


(j) $\overline{DCS}(u,v)$ for $u,v = 0:0.5:15$


(k) $\overline{DSC}(u,v)$ for $u,v = 0:0.25:15$

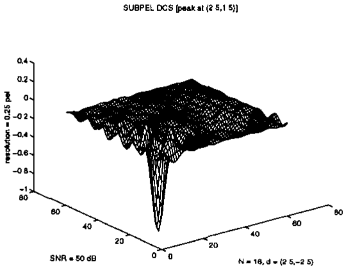

(l) $\overline{DCS}(u,v)$ for $u,v = 0:0.25:15$

Figure 6.4: Illustration of DCT-based half-pel motion estimation algorithm (HDXT-ME)

in Fig. 6.4 (a) whose Fourier transform is bandlimited as in Fig. 6.4 (b) to satisfy the condition in Theorem 1. Fig. 6.4 (e) and (f) are the 3-D plots of the pseudo phases $f(k,l)$ and $g(k,l)$ provided by the DXT-ME algorithm which also computes $DCS(m,n)$ and $DSC(m,n)$ as shown in Fig. 6.4 (g) and (h) with peaks positioned at $(3,1)$ and $(2,2)$ corresponding to the integer-pel estimated displacement vectors $(3,-2)$ and $(2,-3)$ respectively because only the first quadrant is viewed. As a matter of fact, $DCS(m,n)$ and $DSC(m,n)$ have large magnitudes at $\{(m,n); m = 2,3,\ n = 1,2\}$.

To obtain an estimate at half-pel accuracy, we calculate $\overline{DCS}(u,v)$ and $\overline{DSC}(u,v)$ in (6.43) and (6.44) respectively for $u,v = 0 : 0.5 : N-1$ as depicted in Fig. 6.4 (i) and (j) where the peaks can clearly be identified at $(2.5,1.5)$ corresponding to the motion estimate $(2.5,-2.5)$ exactly equal to the true displacement vector even though the two input images do not look alike. Note that the notation $a : r : b$ is an abbreviation of the range $\{a + i \cdot r \text{ for } i = 0, \ldots, \lfloor \frac{b-a}{r} \rfloor\} = \{a, a+r, a+2r, \ldots, b-r, b\}$. For comparison, $\overline{DCS}(u,v)$ and $\overline{DSC}(u,v)$ are also plotted in Fig. 6.4 (k) and (l) respectively for $u,v = 0 : 0.25 : N-1 = 0, 0.25, 0.5, \ldots, N-1.25, N-1$ where smooth ripples are obvious due to the ξ functions inherent in $\overline{DCS}$ and $\overline{DSC}$ of (6.45)-(6.46) and have peaks also at $(2.5,1.5)$.

Therefore, the DCT-based half-pel motion estimation algorithm (HDXT-ME) comprises three steps:

1. The DXT-ME algorithm estimates the integer components of the displacement as $(\hat{m}_u, \hat{m}_v)$.

2. The pseudo phase functions from the DXT-ME algorithm, $f(k,l)$ and $g(k,l)$, are used to compute $\overline{DCS}(u,v)$ and $\overline{DSC}(u,v)$ for $u \in \{\hat{m}_u - 0.5, \hat{m}_u, \hat{m}_u + 0.5\}$ and $v \in \{\hat{m}_v - 0.5, \hat{m}_v, \hat{m}_v + 0.5\}$ from (6.43) and (6.44) respectively.

3. Search the peak positions of $\overline{DCS}(u,v)$ and $\overline{DSC}(u,v)$ for the range of indices, $\overline{\Phi} = \{(u,v) : u \in \{\hat{m}_u - 0.5, \hat{m}_u, \hat{m}_u + 0.5\};\ v \in \{\hat{m}_v - 0.5, \hat{m}_v, \hat{m}_v + 0.5\}\}$, to find

$$(u_{\overline{DCS}}, v_{\overline{DCS}}) = arg \max_{u,v \in \overline{\Phi}} |\overline{DCS}(u,v)|, \tag{6.47}$$

$$(u_{\overline{DSC}}, v_{\overline{DSC}}) = arg \max_{u,v \in \overline{\Phi}} |\overline{DSC}(u,v)|. \tag{6.48}$$

These peak positions determine the estimated displacement vector $(\hat{\lambda}_u, \hat{\lambda}_v)$. However, if the absolute value of $\overline{DSC}(u,v)$ is less than a preset threshold $\epsilon_D > 0$, then $\hat{\lambda}_u = -0.5$. Likewise, if $|\overline{DCS}(u,v)| < \epsilon_D$, $\hat{\lambda}_v = -0.5$. Therefore

$$\hat{\lambda}_u = \begin{cases} u_{\overline{DSC}} = u_{\overline{DCS}}, & \text{if } |\overline{DSC}(u_{\overline{DSC}}, v_{\overline{DSC}})| > \epsilon_D, \\ -0.5, & \text{if } |\overline{DSC}(u_{\overline{DSC}}, v_{\overline{DSC}})| < \epsilon_D, \end{cases} \tag{6.49}$$

$$\hat{\lambda}_v = \begin{cases} v_{\overline{DCS}} = v_{\overline{DSC}}, & \text{if } |\overline{DCS}(u_{\overline{DCS}}, v_{\overline{DCS}})| > \epsilon_D, \\ -0.5, & \text{if } |\overline{DCS}(u_{\overline{DCS}}, v_{\overline{DCS}})| < \epsilon_D. \end{cases} \tag{6.50}$$

In Step 2, only those half-pel estimates around the integer-pel estimate $(\hat{m}_u, \hat{m}_v)$ are considered due to the fact that the DXT-ME algorithm finds the nearest integer-pel motion estimate $(\hat{m}_u, \hat{m}_v)$ from the subpixel displacement. This will significantly reduce the number of computations without evaluating all possible half-pel displacements.

In Step 3, the use of ϵ_D deals with the case of zero pseudo phases when the displacement is -0.5. Specifically, if $\lambda_u = -0.5$, then $g^{SC}_{\lambda_u,\lambda_v}(k,l) = 0$, $\forall k,l$ which leads to $g(k,l) = 0$ and $\overline{DSC}(u,v) = 0$. However, in a noisy situation, it is very likely that $g(k,l)$ is not exactly zero and thus neither is $\overline{DSC}(u,v)$. Therefore, ϵ_D should be set very small but large enough to accommodate the noisy case. In our experiment, ϵ_D is empirically chosen to be 0.08. Similar consideration is made on $\overline{DCS}(u,v)$ for $\lambda_v = -0.5$. It is also possible that the peak positions of $\overline{DCS}(u,v)$

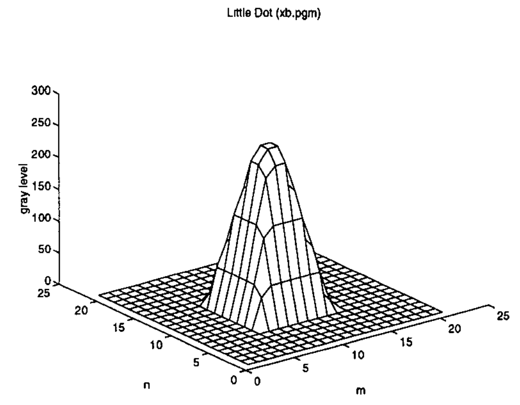
and $\overline{DSC}(u,v)$ differ in the noisy circumstances. In this case, the arbitration rule used in the DXT-ME algorithm may be applied.

To demonstrate the accuracy of this HDXT-ME algorithm, we use a 16×16 dot image x_1 in Fig. 6.5 (a) as input and displace x_1 to generate the second input image x_2 according to the true motion field $\{(\lambda_u, \lambda_v) : \lambda_u, \lambda_v = -5 : 0.5 : 4\}$ shown in Fig. 6.5 (b) through the bilinear interpolating function specified in the MPEG standard [54] which interpolates the value $x(m+u, n+v)$ from four neighboring pixel values for m, n being integers and $u, v \in [0, 1)$ in the following way:

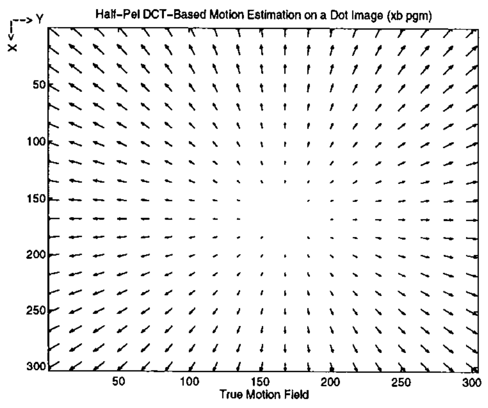
$$\begin{aligned} x(m+u,n+v) = {} & (1-u)\cdot(1-v)\cdot x(m,n) + (1-u)\cdot v\cdot x(m,n+1) \\ & + u\cdot(1-v)\cdot x(m+1,n) + u\cdot v\cdot x(m+1,n+1). \end{aligned} \tag{6.51}$$

Fig. 6.5 (c) shows the estimated motion field by the HDXT-ME algorithm which is exactly the same as the true motion field.

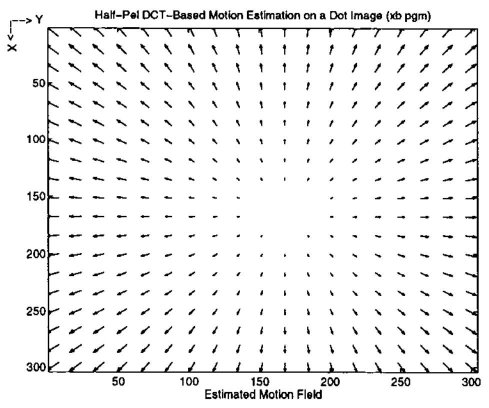
Fig. 6.6 (a)-(c) further illustrate estimation accuracy for half-pel motion estimation schemes using peak information from $L_s(u,v)$, $L_c(u,v)$, and $L_c(u,v)+L_s(u,v)$ respectively. In Fig. 6.6 (a), the "+" line indicates peak positions of $L_s(u,v)$ found in the index range $\{0 : 0.5 : 15\}$ for a block size $N = 16$ with respect to different true displacement values $\{-7 : 0.5 : 7\}$. The "o" line specifies the final estimates after determination of motion directions from the peak signs of $L_s(u,v)$ according to the rules in Table 6.1. These estimates are shown to align with the reference line $u = v$, implying their correctness. For the true displacement $= -0.5$, $L_s(-0.5, v) \equiv 0$ for all v and ϵ_D is used to decide whether the estimate should be set to -0.5. In Fig. 6.6 (b), $L_c(u,v)$ is used instead of $L_s(u,v)$ but $L_c(u,v)$ is always positive, inferring that no peak sign can be exploited to determine motion direction. In Fig. 6.6 (c), $L_c(u,v) + L_s(u,v)$ provides accurate estimates without adjustment for all true displacement values but the index range must include



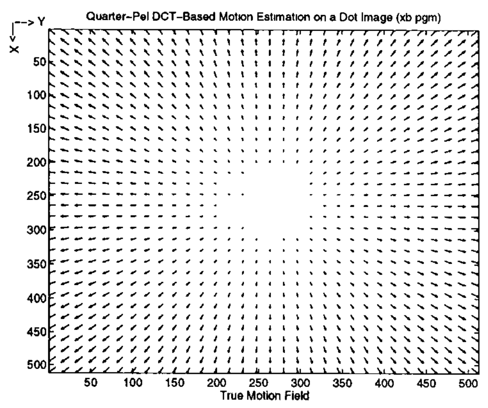
(a) Input Image



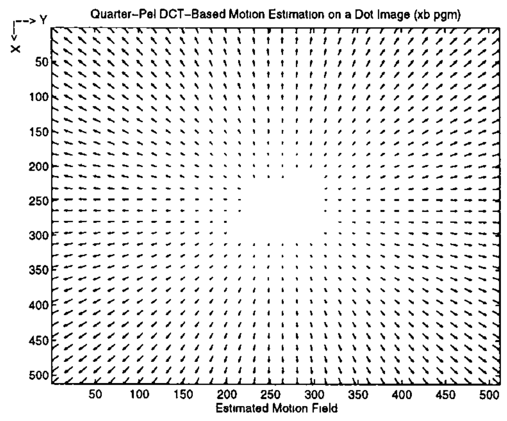
(b) True Motion Field
for Half-pel Movement



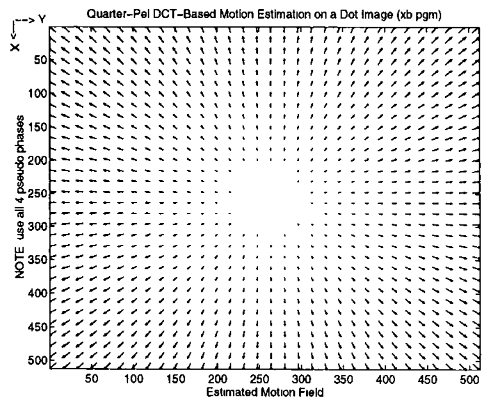
(c) Estimated Motion Field
of HDXT-ME



(d) True Motion Field
for Quarter-pel Movement

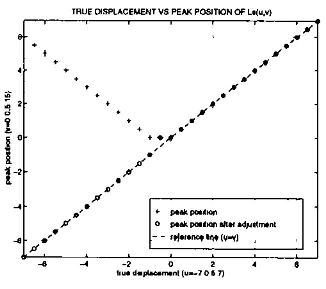


(e) Estimated Motion Field
of QDXT-ME

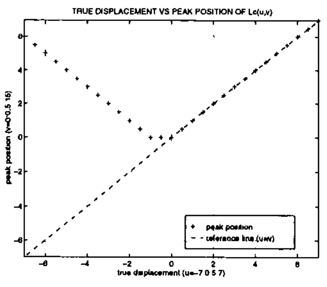


(f) Estimated Motion Field
of Q4DXT-ME

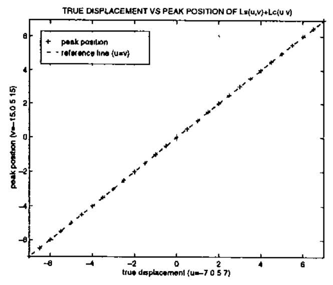
Figure 6.5: Estimated motion fields (c)(e) of HDXT-ME and QDXT-ME by moving a dot image (a) according to the true motion fields (b)(d).



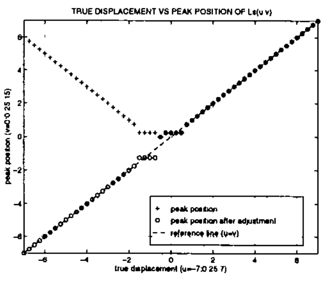
(a) $L_s(u, v)$ for half-pel estimation



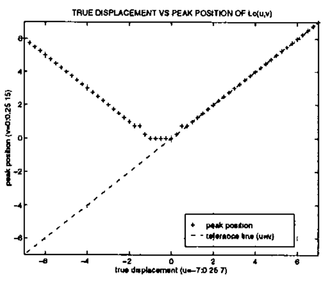
(b) $L_c(u, v)$ for half-pel estimation



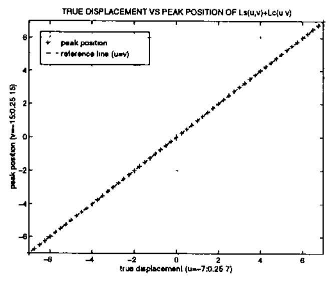
(c) $L_c(u, v) + L_s(u, v)$ for half-pel estimation



(d) $L_s(u, v)$ for quarter-pel estimation



(e) $L_c(u, v)$ for quarter-pel estimation



(f) $L_c(u, v) + L_s(u, v)$ for quarter-pel estimation

Figure 6.6: Relation between true displacements and peak positions for half-pel and quarter-pel estimation. The signs of peak values in $L_s(u, v)$ indicate the motion directions and are used to adjust the peak positions for motion estimates.

negative indices, i. e. $[-15 : 0.5 : 15]$.

In the HDXT-ME algorithm, Step 2 involves only nine $\overline{DCS}(u, v)$ and $\overline{DSC}(u, v)$ values at and around $(\hat{m}_u, \hat{m}_v)$. Since $\overline{DCS}(u, v)$ and $\overline{DSC}(u, v)$ are variants of inverse 2D-DCT-II, the parallel and fully-pipelined 2D DCT lattice structure proposed in [12, 49, 50] can be used to compute $\overline{DCS}(u, v)$ and $\overline{DSC}(u, v)$ at a cost of $O(N)$ operations in N steps. Furthermore, the searching in Step 3 requires $O(N^2)$ operations for one step. Thus, the computational complexity of the HDXT-ME algorithm is $O(N^2)$ in total.

6.3.2 DCT-Based Quarter-Pel Motion Estimation Algorithm (QDXT-ME and Q4DXT-ME)

In Section 6.2, we mention that the interaction of two ξ functions in $L_c(u,v)$ and $L_s(u,v)$ from (6.35) and (6.36) disassociates the peak locations with the displacement (λ_u, λ_v) for λ_u, $\lambda_v \in [-1.5, 0.5]$. In spite of this, in the HDXT-ME algorithm, we can still accurately estimate half-pel displacements by locating the peaks of $L_s(\lambda, v)$ for true displacements $\lambda = -N+1 : 0.5 : N-1$ and indices $v = 0 : 0.5 : N-1$ if ϵ_D is introduced to deal with the case for $\lambda = -0.5$. However, at the quarter-pel level, it does cause estimation errors around $\lambda = -0.5$ as indicated in Fig. 6.6 (d) where the peaks of $L_s(\lambda, v)$ stay at $v = 0$ for true displacements λ varying over $[-1, 0]$. As mentioned in Section 6.2, the sum of $L_c(\lambda, v)$ and $L_s(\lambda, v)$ is a pure ξ function and thus the adverse interaction is eliminated. As a result, the peak position of this sum can be used to predict precisely the displacement at either half-pel level or quarter-pel level as demonstrated in Fig. 6.6 (c) and (f) respectively. However, for two dimensional images, $\overline{DCS}$ or $\overline{DSC}$ has four ξ functions as in (6.45) or (6.46). For the DXT-ME algorithm provides two pseudo phase functions $f(k,l)$ and $g(k,l)$, only $\overline{DCS}$ and $\overline{DSC}$ are available for subpixel estimation. In this case, the sum of $\overline{DCS}$ and $\overline{DSC}$ can only annihilates two ξ functions, leaving two ξ functions as given by:

$$\overline{DCS}(u,v) + \overline{DSC}(u,v) = \frac{1}{2}[\xi(u-\lambda_u)\xi(v-\lambda_v) - \xi(u+\lambda_u+1)\xi(v+\lambda_v+1)]. \tag{6.52}$$

Even though this sum is not a single ξ function, the estimation error of using this sum is limited to 1/4 pixel for the worst case when true displacements are either -0.75 or -0.25.

The above discussion leads to the DCT-based quarter-pel motion estimation algorithm (QDXT-ME) as follows:

1. The DXT-ME algorithm computes the integer-pel estimate $(\hat{m}_u, \hat{m}_v)$.
2. $\overline{DCS}(u,v)$ and $\overline{DSC}(u,v)$ are calculated from $f(k,l)$ and $g(k,l)$ in (6.43) and (6.44) respectively for the range of indices, $\overline{\Phi} = \{(u,v) : u = \hat{m}_u - 0.75 : 0.25 : \hat{m}_u + 0.75;\ v = \hat{m}_v - 0.75 : 0.25 : \hat{m}_v + 0.75\}$.
3. Search the peak position of $D_2(u,v) \triangleq \overline{DCS}(u,v) + \overline{DSC}(u,v)$ over $\overline{\Phi}$, i. e.

$$(u_{D2}, v_{D2}) = arg \max_{u,v \in \overline{\Phi}} |D_2(u,v)|. \tag{6.53}$$

The estimated displacement vector is obtained as follows:

$$(\hat{\lambda}_u, \hat{\lambda}_v) = \begin{cases} (u_{D2}, v_{D2}), & \text{if } |D_2(u_{D2}, v_{D2})| > \epsilon_D, \\ (-0.5, -0.5), & \text{if } |D_2(u_{D2}, v_{D2})| < \epsilon_D. \end{cases} \tag{6.54}$$

Step 3 is based on the fact that $|D_2(\lambda_u, \lambda_v)| = 0$ if and only if $(\lambda_u, \lambda_v) = -0.5$. This QDXT-ME algorithm follows the same procedure as HDXT-ME except the search region and using the sum of $\overline{DCS}$ and $\overline{DSC}$. Therefore, QDXT-ME has the same computational complexity, $O(N^2)$, as HDXT-ME.

If we modify the DXT-ME algorithm to provide the other two pseudo phase functions g^{CC} and g^{SS} in addition to f and g, we can compute $\overline{DCC}$ and $\overline{DSS}$ in the following way:

$$\overline{DCC}(u,v) \triangleq \sum_{k=0}^{N-1} \sum_{l=0}^{N-1} g^{CC}(k,l) \cos\frac{k\pi}{N}(u+\frac{1}{2}) \cos\frac{l\pi}{N}(v+\frac{1}{2}), \tag{6.55}$$

$$\overline{DSS}(u,v) \triangleq \sum_{k=1}^{N-1} \sum_{l=1}^{N-1} g^{SS}(k,l) \sin\frac{k\pi}{N}(u+\frac{1}{2}) \sin\frac{l\pi}{N}(v+\frac{1}{2}). \tag{6.56}$$

Then we can show that

$$D_4(u,v) \triangleq \overline{DCC}(u,v) + \overline{DCS}(u,v) + \overline{DSC}(u,v) + \overline{DSS}(u,v) \tag{6.57}$$

$$= \xi(u-\lambda_u)\xi(v-\lambda_v). \tag{6.58}$$

This sum[1] contains only one ξ without any negative interaction effect whose peak is sharp at (λ_u, λ_v). This leads to another quarter-pel motion estimation algorithm (Q4DXT-ME), which can estimate accurately for all displacements at the quarter-pel or even finer level.

1. Find the integer-pel estimate $(\hat{m}_u, \hat{m}_v)$ by the DXT-ME algorithm.

2. Obtain four pseudo phases g^{CC}, g^{CS}, g^{SC} and g^{SS} from the modified DXT-ME algorithm. Compute $\overline{DCS}(u,v)$, $\overline{DSC}(u,v)$, $\overline{DCC}(u,v)$, and $\overline{DSS}(u,v)$ for the range of indices, $\overline{\Phi} = \{(u,v) : u = \hat{m}_u - 0.75 : 0.25 : \hat{m}_u + 0.75;\ v = \hat{m}_v - 0.75 : 0.25 : \hat{m}_v + 0.75\}$.

3. Search the peak position of $D_4(u,v)$ over $\overline{\Phi}$:

$$(u_{D4}, v_{D4}) = arg \max_{u,v \in \overline{\Phi}} |D_4(u,v)|.$$

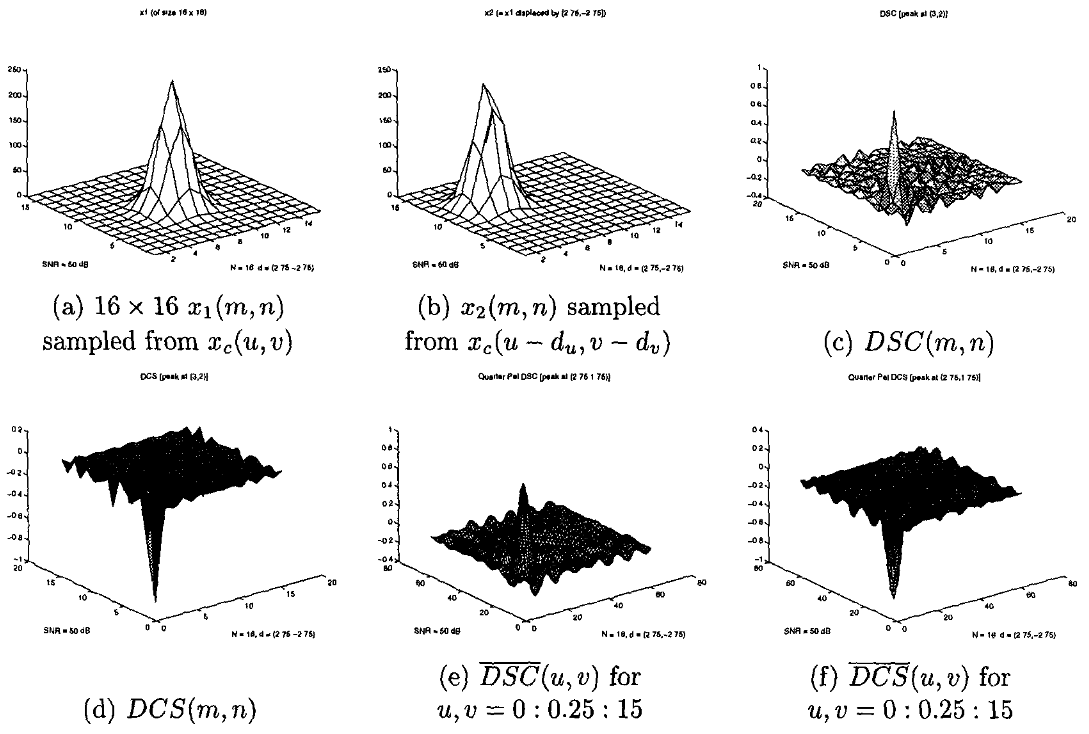
The estimated displacement vector is then the peak position:

$$(\hat{\lambda}_u, \hat{\lambda}_v) = (u_{D4}, v_{D4}).$$

Fig. 6.7 shows the procedure to estimate a quarter-pel displacement with input images $x_1(m,n)$ and $x_2(m,n)$ sampled from the continuous intensity profile $x_c(u,v)$ and its shift $x_c(u - \lambda_u d, v - \lambda_v d)$ where $(\lambda_u, \lambda_v) = (2.75, -2.75)$ and $d = 0.625$ as shown in Fig. 6.7 (a) and (b). Fig. 6.7 (c) and (d) plot $DSC(m,n)$ and $DCS(m,n)$ whose peaks are both at $(3,2)$ corresponding to the integer-pel estimate $(3,-3)$. Fig. 6.7 (e) and (f) are the graphs of $\overline{DSC}(u,v)$ and $\overline{DCS}(u,v)$ at the quarter-pel level where the estimate is found to be $(2.75, -2.75)$.

Similar to the half-pel case, Fig. 6.5 (e) and (f) demonstrate the accuracy of the estimated motion fields determined by the QDXT-ME and Q4DXT-ME algorithms

[1]These four functions can be generated naturally at the same time using the computing algorithms and architectures in [12, 49].



(a) 16×16 $x_1(m,n)$ sampled from $x_c(u,v)$

(b) $x_2(m,n)$ sampled from $x_c(u-d_u, v-d_v)$

(c) $DSC(m,n)$

(d) $DCS(m,n)$

(e) $\overline{DSC}(u,v)$ for $u,v = 0:0.25:15$

(f) $\overline{DCS}(u,v)$ for $u,v = 0:0.25:15$

Figure 6.7: Illustration of DCT-based quarter-pel motion estimation algorithm (QDXT-ME)

respectively as compared to the true motion field in Fig. 6.5 (d). The first input image $x_1(m,n)$ to both algorithms is a bandlimited dot image in Fig. 6.5 and the second input image $x_2(m,n)$ is generated by shifting $x_1(m,n)$ with respect to the true motion field in Fig. 6.5 (d) through the bilinear interpolation. Though not obvious in the graphs, the estimates of QDXT-ME around -0.5 have an estimation error up to a quarter pixel whereas Q4DXT-ME gives us perfect estimation.

6.4 Experimental Results

A set of simulations are performed on two video sequences of different characteristics: Miss America (HMS) with slow head and shoulder movement accompanying with occasional eye and mouth opening and Infrared Car (HCA) with a moving car viewed by a slightly shaking infrared camera. The performance of the DCT-based algorithms is compared with Full Search Block Matching Algorithm (BKM-ME) and its subpixel counterparts in terms of mean square error per pixel (MSE) and bits per sample (BPS). Here MSE is defined as MSE $= \{\sum_{m,n}[\hat{x}(m,n) - x(m,n)]^2\}/N^2$ where $\hat{x}(m,n)$ is the reconstructed image predicted from the original image $x(m,n)$ based upon the estimated displacement vector $\hat{\lambda} = (\hat{\lambda}_u, \hat{\lambda}_v)$. BPS is computed as the ratio of the total number of bits required for a motion-compensated residual frame compressed in JPEG format to the number of pixels for each frame. For all the MSE values computed in the experiment, the bilinear interpolation in (6.51) is used for comparison to reconstruct images displaced by a fractional pixel because the bilinear interpolation is used in MPEG standards for motion compensation [54, 55]. Furthermore, for visual comparison, all residual images, generated by subtracting the original images from the reconstructed frames predicted by various motion estimation schemes, are displayed after the saturation level is reset to 25 instead of 255 to make small pixel values of the residual images be visible. In addition, the needle maps for the estimated motion fields are superimposed on the corresponding residual images.

As usual, the integer-pel BKM-ME algorithm minimizes the MAD (Minimum Absolute Difference) function of the block $\{x_1(m,n);\ m,n = 0:1:N-1\}$ over the search area $\Phi = \{(m,n) : m,n = -\frac{N}{2} : 1 : N-1+\frac{N}{2}\}$ such that

$$\vec{\mathbf{d}}_\imath = (\hat{m}_u, \hat{m}_v) = arg \min_{u,v=-\frac{N}{2}:1:\frac{N}{2}} \frac{\sum_{m,n=0}^{N-1} |x_2(m,n) - x_1(m-u,n-v)|}{N^2}. \quad (6.59)$$

In the simulation, two levels of subpixel block-matching motion estimation algorithms are implemented for comparison:

1. Half-Pel Full Search Block Matching Algorithm (HBKM-ME) — Similar to BKM-ME, HBKM-ME searches for the displacement of minimum MAD value among the integer-pel motion estimate and 8 points of half-pel displacements around the integer-pel estimate as such:

$$\begin{aligned} \vec{\mathbf{d}}_h &= (\hat{\lambda}_u, \hat{\lambda}_v) \\ &= arg \min_{\substack{u = \hat{m}_u - \frac{1}{2}, \hat{m}_u, \hat{m}_u + \frac{1}{2}, \\ v = \hat{m}_v - \frac{1}{2}, \hat{m}_v, \hat{m}_v + \frac{1}{2}}} \frac{\sum_{m,n=0}^{N-1} |x_2(m,n) - x_1(m-u,n-v)|}{N^2}. \end{aligned} \quad (6.60)$$

Searching around the integer-pel estimate instead of all possible half-pel displacements is recommended in MPEG standards to reduce significantly the overall computational complexity.

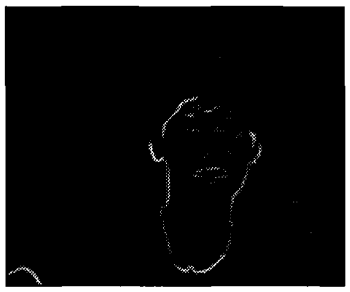
2. Quarter-Pel Full Search Block Matching Algorithm (QBKM-ME) — After the integer-pel full search block matching (BKM-ME) motion estimation, QBKM-ME considers all half-pel and quarter-pel displacements around the integer-pel motion estimate in finding the minimum MAD value. Precisely, the estimated displacement vector is

$$\begin{aligned} \vec{\mathbf{d}}_q &= (\hat{\lambda}_u, \hat{\lambda}_v) \\ &= arg \min_{\substack{u = \hat{m}_u - \frac{3}{4} \cdot \frac{1}{4} : \hat{m}_u + \frac{3}{4}, \\ v = \hat{m}_v - \frac{3}{4} : \frac{1}{4} : \hat{m}_v + \frac{3}{4}}} \frac{\sum_{m,n=0}^{N-1} |x_2(m,n) - x_1(m-u,n-v)|}{N^2}. \end{aligned} \quad (6.61)$$

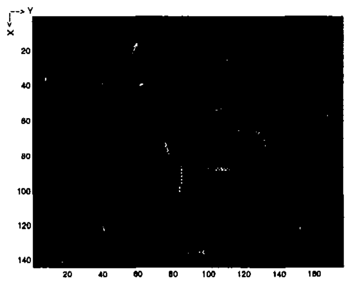
In addition to the full search block matching approaches, we also compare with three kinds of fast search block matching algorithms for integer-pel, half-pel and quarter-pel accuracy: the three step search algorithm (TSS, HTSS, QTSS), the logarithmic search algorithm (LOG, HLOG, QLOG), and the subsampled search algorithm (SUB, HSUB, QSUB) [48]. It should be noted that all half and quarter pixel values for the block matching schemes are approximated by the bilinear interpolation. However, for the DCT-based subpixel algorithms, no interpolation is needed in finding the motion estimates. Therefore, the number of operations required by HBKM-ME and QBKM-ME (even for the fast search algorithms) are twice and four times as much as BKM-ME respectively whose computational complexity is $O(N^4)$ whereas the DCT-based subpixel algorithms have only marginal increase in computations over DXT-ME of which the computational complexity is $O(N^2)$. In the following simulation, simple edge extraction and frame differentiation are adopted for preprocessing input images before the DCT-based algorithms, as described in details in Chapter 4. Either preprocessing scheme adds in only $O(N^2)$ operations as overhead, keeping the total complexity remain $O(N^2)$.

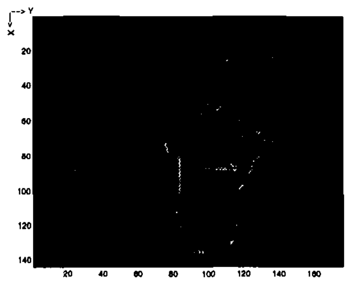
Simulation is made on the "Miss America" sequence (HMS) in QCIF format whose frame size is 176×144. The original frame 83 is shown in Fig. 6.8 (a) and the preprocessed frames in Fig. 6.8 (b)-(c) where the differentiated frame contains only very small pixel values and thus need be displayed after visualization process; otherwise, its contents will be invisible. These small DIF values indicate only slow head and shoulder motion in this sequence. The residual images for various methods in Fig. 6.8 (d)-(i) reveal that edge extraction is better than frame difference due to weak feature energy present in the frame differentiated sequence. Furthermore, there are some small patches in the clothes areas for either HDXT-ME or QDXT-ME in view of the uniform brightness in these areas removed by both preprocessing functions. This situation may be improved if a better preprocessing

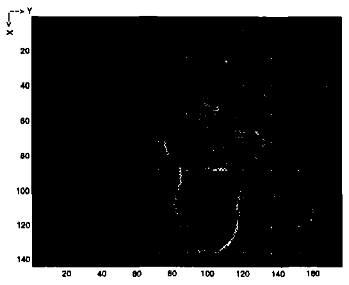
(a) Frame 83

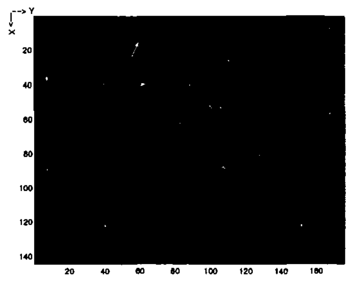

(b) Edge Extracted Frame 83

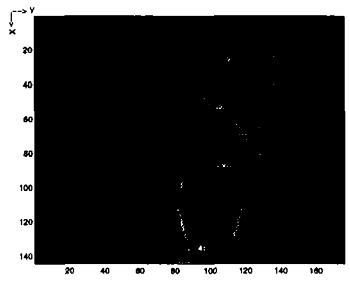
(c) Frame Differentiated Frame 83


(d) HBKM-ME


(e) HDXT-ME after edge extraction


(f) HDXT-ME after frame differentiation


(g) QBKM-ME


(h) QDXT-ME after edge extraction

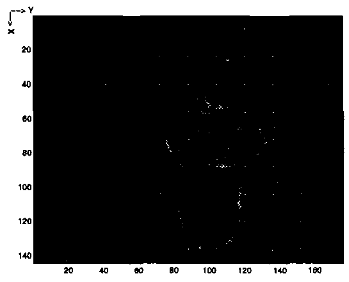
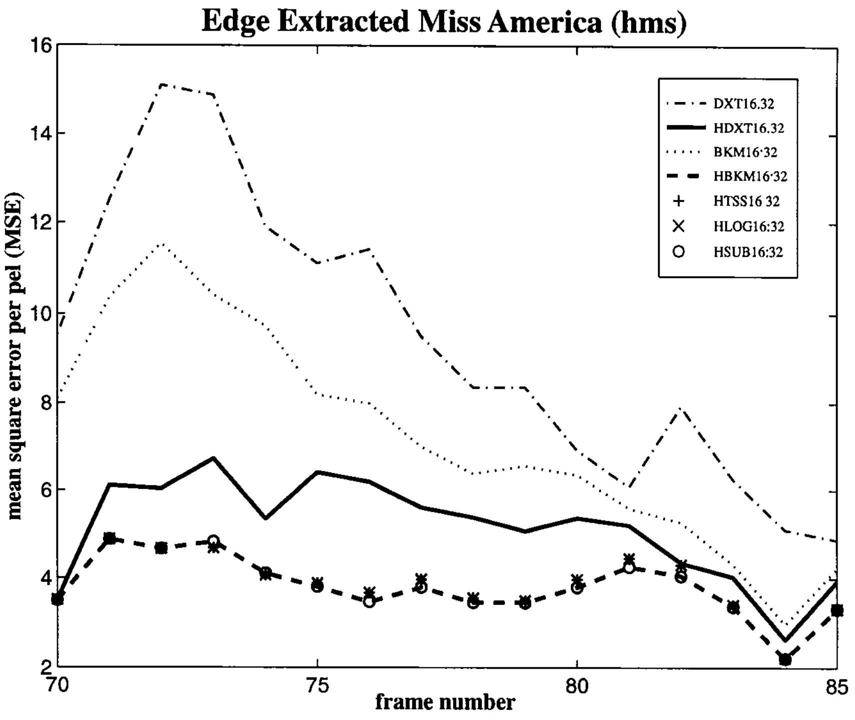
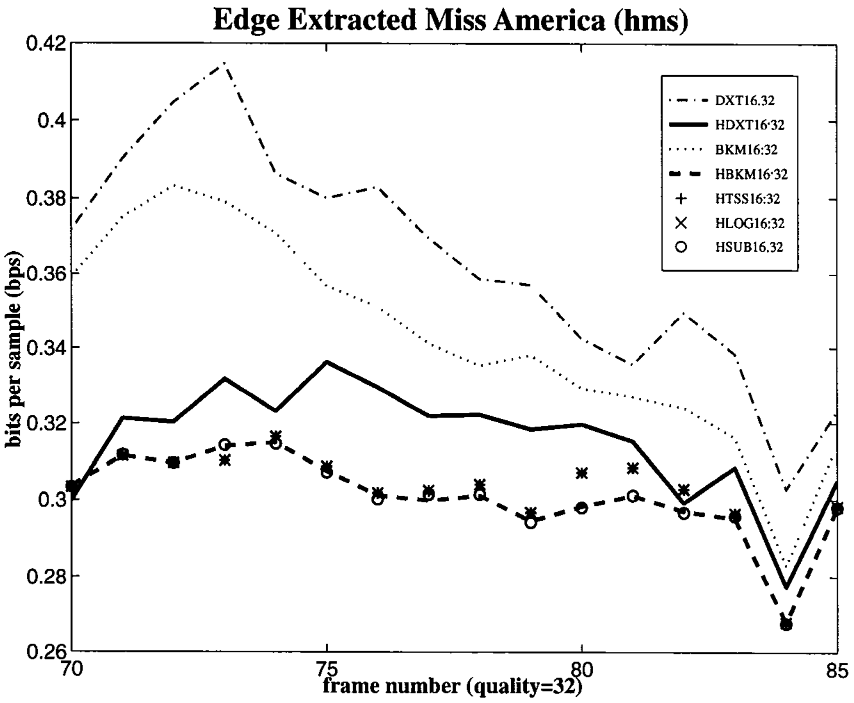
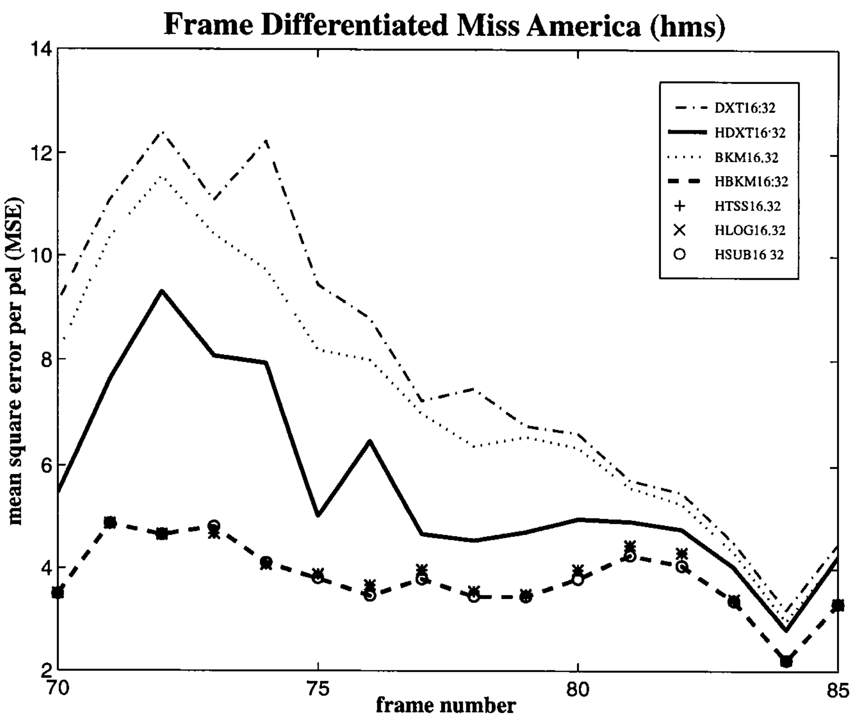

(i) QDXT-ME after frame differentiation

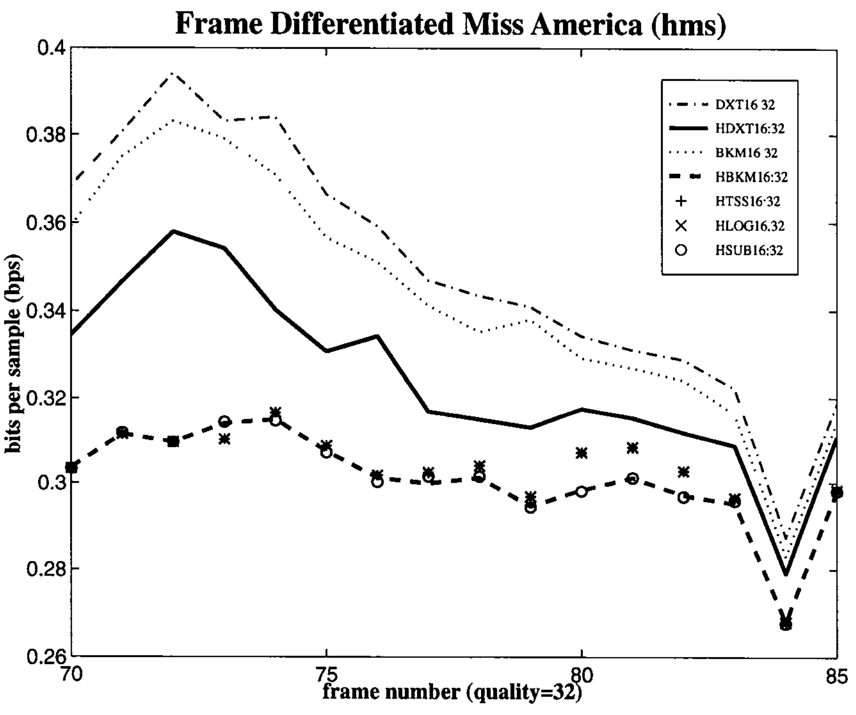
Figure 6.8: Comparison of different approaches on Frame 83 of Miss America sequence (HMS) in QCIF format for block size 16 × 16 and search size 32 × 32. Visualization is applied to (c)-(i) by setting the saturation level to 25. The needle maps for the estimated motion fields are laid over the residual images.





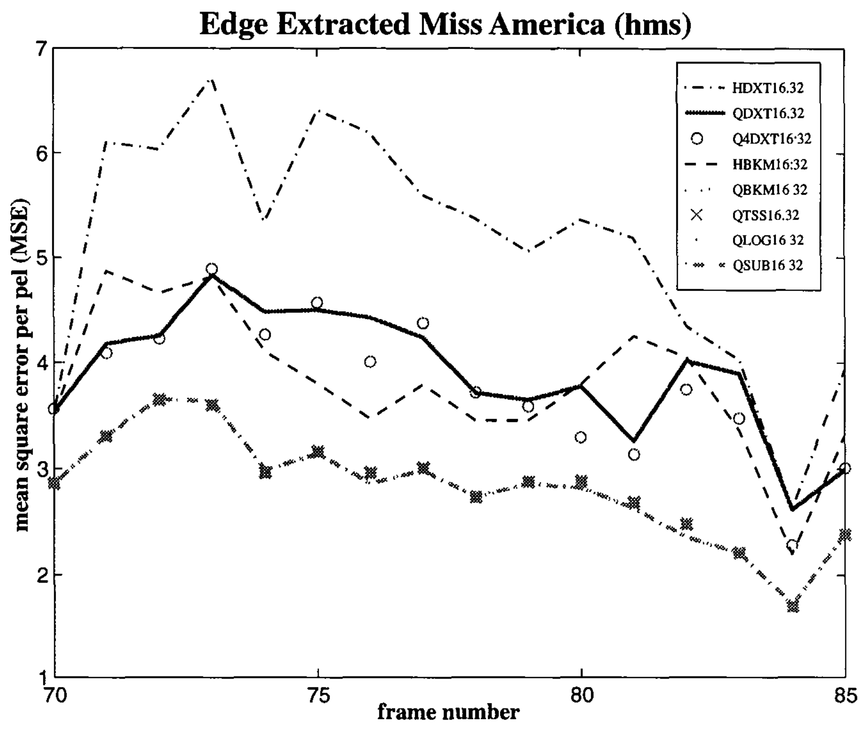
(a) Preprocessed by Edge Extraction

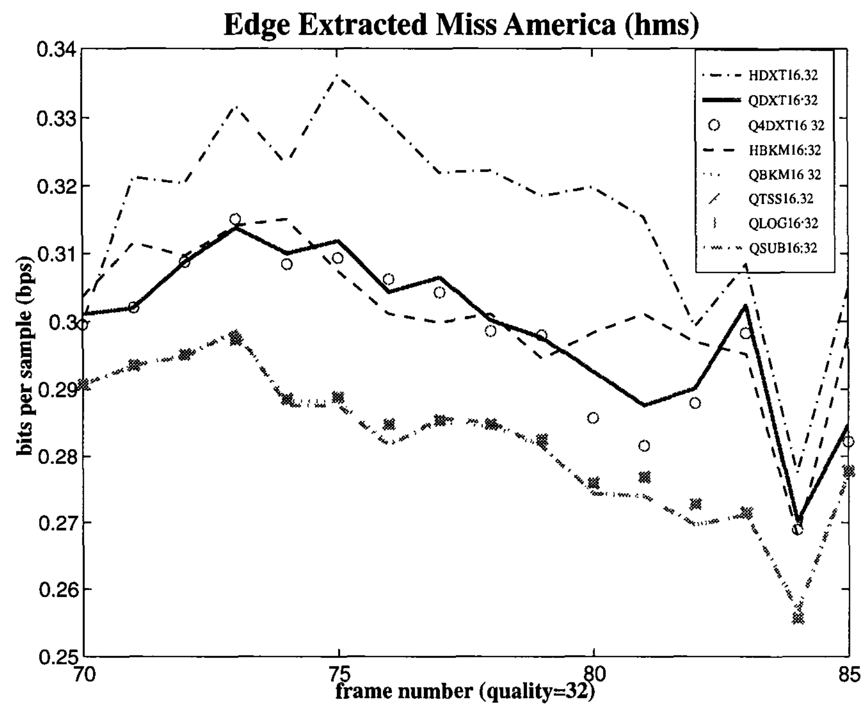




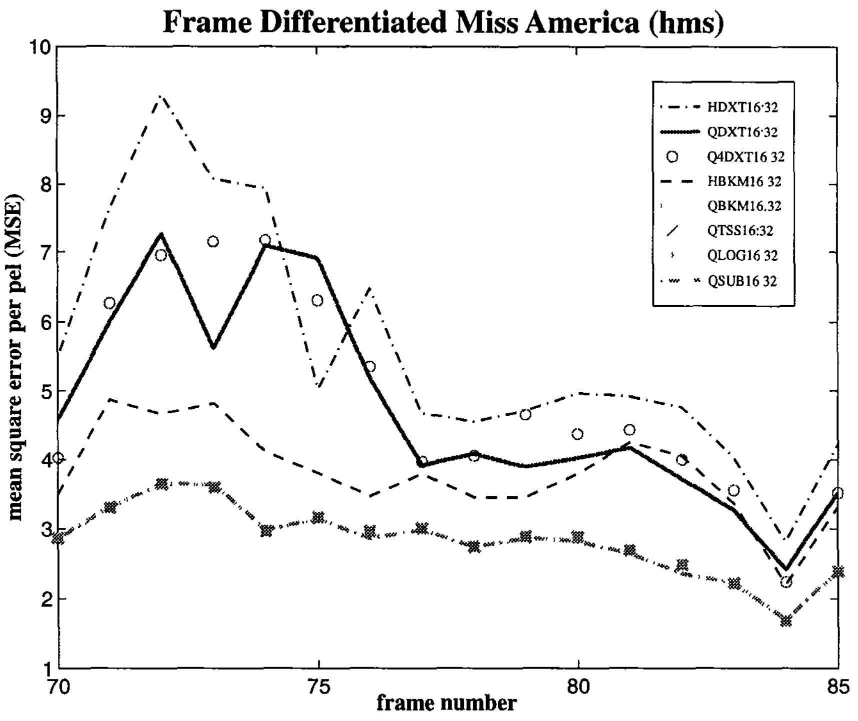
(b) Preprocessed by Frame Differentiation

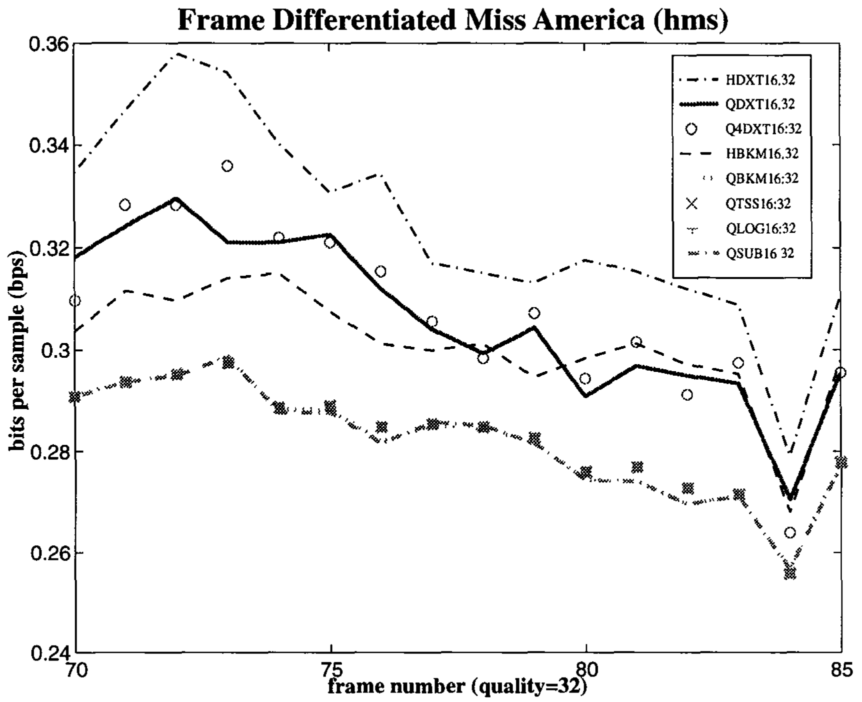
Figure 6.9: Simulation Results for motion estimation of half-pel accuracy on the Miss America sequence (HMS) in QCIF Format.





(a) Preprocessed by Edge Extraction





(b) Preprocessed by Frame Differentiation

Figure 6.10: Simulation Results for motion estimation of quarter-pel accuracy on the Miss America sequence (HMS) in QCIF Format.

Approach	MSE	MSE difference	MSE ratio	BPF	BPS	BPS ratio
INTEGER-PEL ACCURACY						
BKM	7.187	0.000	0.0%	8686	0.343	0.0%
frame differentiated DXT	7.851	0.664	9.2%	8855	0.349	1.9%
edge extracted DXT	9.363	2.176	30.3%	9200	0.363	5.9%
TSS	7.862	0.675	9.4%	8910	0.352	2.6%
LOG	7.862	0.675	9.4%	8910	0.352	2.6%
SUB	7.202	0.015	0.2%	8684	0.343	0.0%
HALF-PEL ACCURACY						
HBKM	3.807	0.000	0.0%	7628	0.301	0.0%
frame differentiated HDXT	5.598	1.791	47.0%	8216	0.324	7.7%
edge extracted HDXT	5.116	1.308	34.4%	8000	0.316	4.9%
HTSS	3.877	0.070	1.8%	7676	0.303	0.6%
HLOG	3.877	0.070	1.8%	7676	0.303	0.6%
HSUB	3.810	0.002	0.1%	7628	0.301	0.0%
QUARTER-PEL ACCURACY						
QBKM	2.820	0.000	0.0%	7146	0.282	0.0%
frame differentiated QDXT	4.728	1.908	67.7%	7758	0.306	8.6%
edge extracted QDXT	3.899	1.079	38.3%	7578	0.299	6.0%
frame differentiated Q4DXT	4.874	2.054	72.8%	7785	0.307	8.9%
edge extracted Q4DXT	3.765	0.945	33.5%	7532	0.297	5.4%
QTSS	2.843	0.023	0.8%	7162	0.283	0.2%
QLOG	2.843	0.023	0.8%	7162	0.283	0.2%
QSUB	2.825	0.005	0.2%	7144	0.282	0.0%

Table 6.2: Performance summary of the DXT-ME algorithm with either frame differentiation or edge extraction as preprocessing against full search and fast search block matching approaches (BKM, TSS, LOG, SUB) and their half-pel (HBKM, HTSS, HLOG, HSUB) and quarter-pel (QBKM, QTSS, QLOG, QSUB) counterparts over the sequence "Miss America" (HMS) in QCIF format. MSE difference is the difference from the MSE value of full search block matching method (BKM) and MSE ratio is the ratio of MSE difference to the MSE of BKM.

function is used to avoid removal of uniform areas while suppressing the aperture effect [68].

Fig. 6.9 and 6.10 display in terms of the MSE and BPS values the performances of the block matching approaches and the DXT-ME algorithm preprocessed by frame differentiation and edge extraction with half-pel and quarter-pel accuracy respectively. These performances are summarized by averaging over the sequence in Table 6.2 in which the edge extracted HDXT-ME, QDXT-ME and Q4DXT-ME are 34.4%, 38.3%, and 33.5% worse than the full search block matching counterparts. Even though the frame differentiated DXT-ME is better than the edge extracted DXT-ME by achieving only 9.2% worse than BKM-ME, edge extraction seems to provide better improvement to the DXT-ME algorithm than frame differentiation for subpixel motion estimation. The coding gain from subpixel motion estimation is obvious when we compare how much improvement we can have from integer-pel accuracy to half-pel and even quarter-pel accuracy:

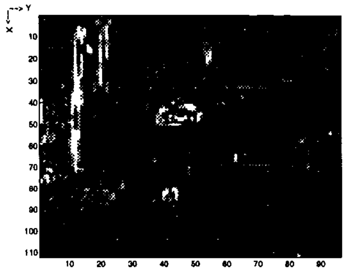
- HBKM-ME has 47.03% less of MSE value or 12.24% less of BPS value than BKM-ME whereas QBKM-ME has 60.76% less of MSE or 17.78% less of BPS than BKM-ME.
- Edge extracted HDXT-ME has 45.36% less of MSE value or 12.95% less of BPS value than edge extracted DXT-ME whereas edge extracted QDXT-ME has 59.79% less of MSE or 18.18% less of BPS.

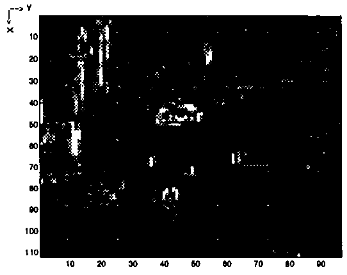
The other sequence in our simulation is "Infrared Car" which has 96×112 pixels and a moving car along the curved road viewed from a slightly shaking infrared camera. The MSE and BPS values for both the block matching approaches and the DXT-ME algorithm are plotted in Fig. 6.12 for half-pel estimation and Fig. 6.13 for

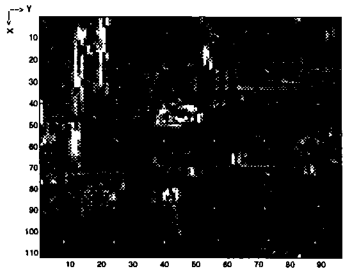
(a) Frame 10

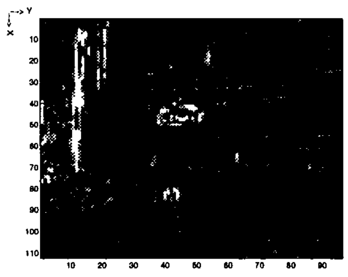
(b) Edge Extracted Frame 10

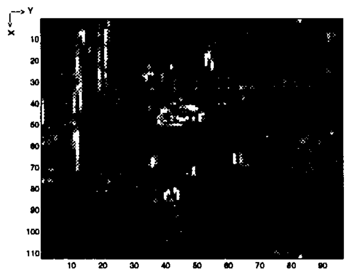
(c) Frame Differentiated Frame 10


(d) HBKM-ME


(e) HDXT-ME after edge extraction


(f) HDXT-ME after frame differentiation


(g) QBKM-ME


(h) QDXT-ME after edge extraction

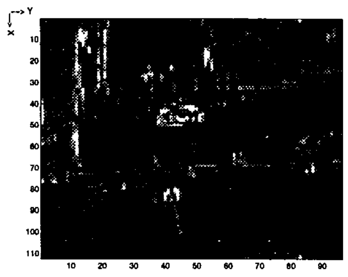
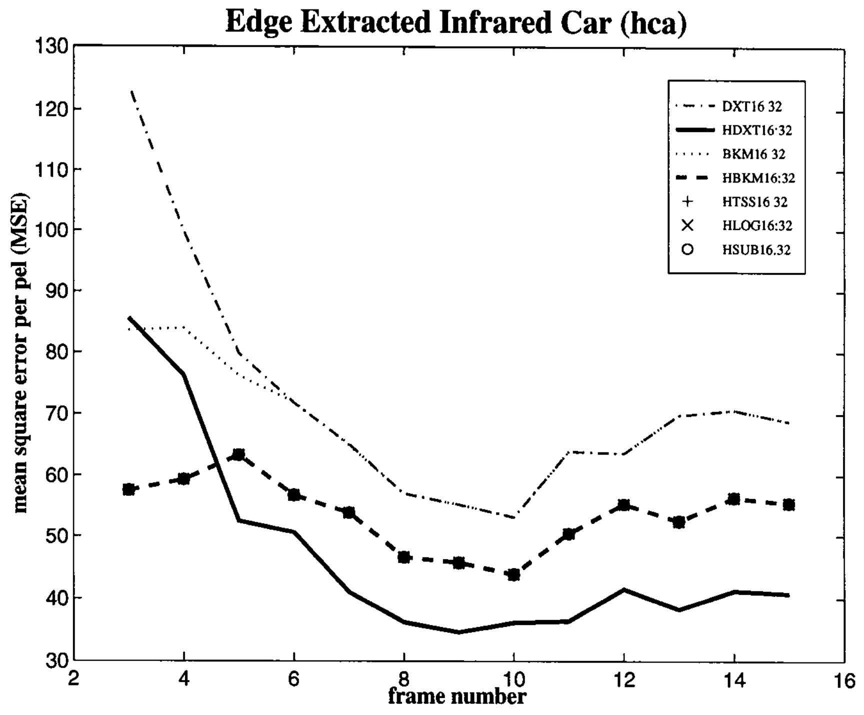
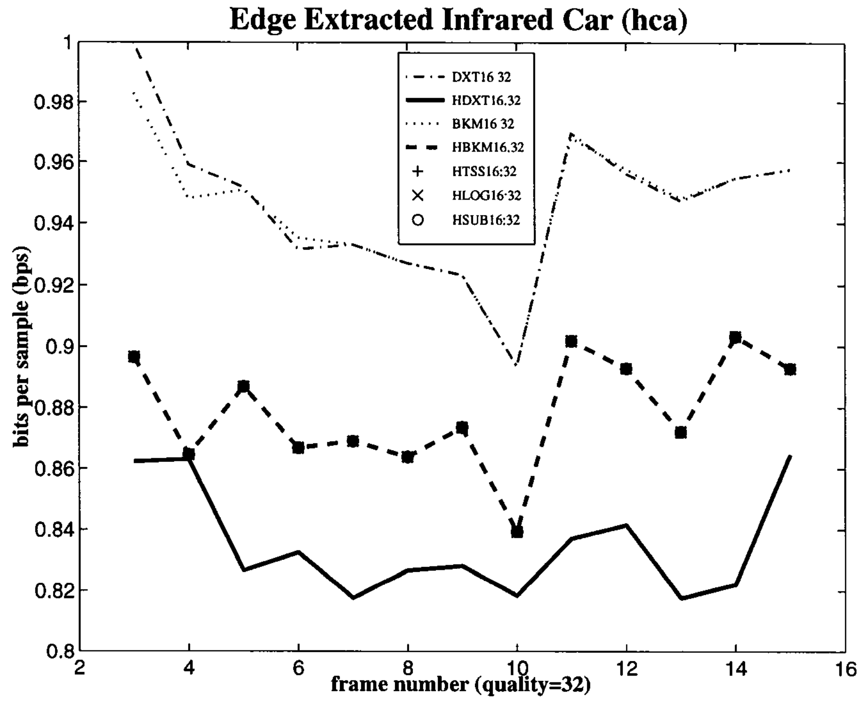
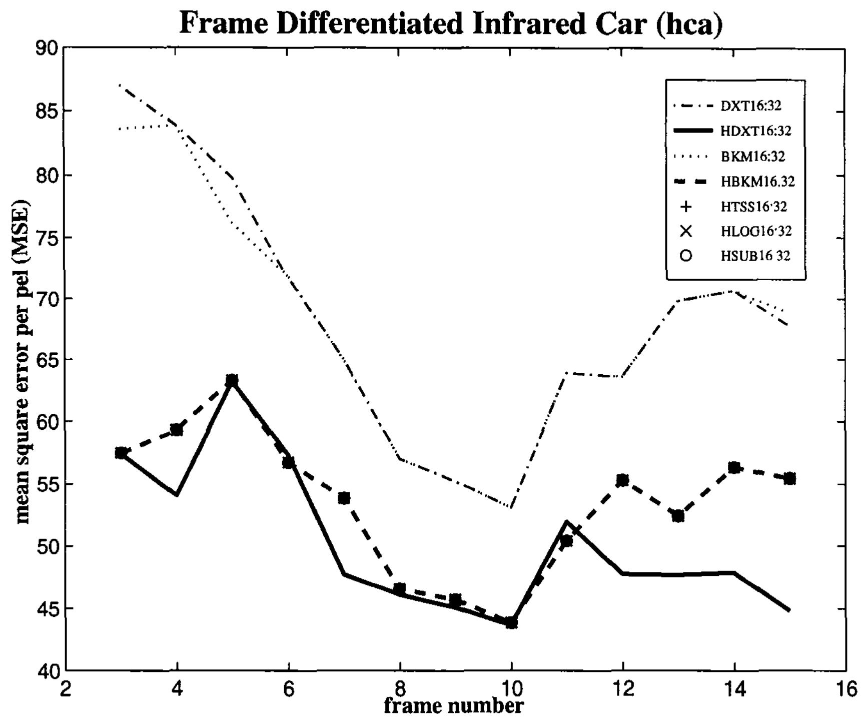

(i) QDXT-ME after frame differentiation

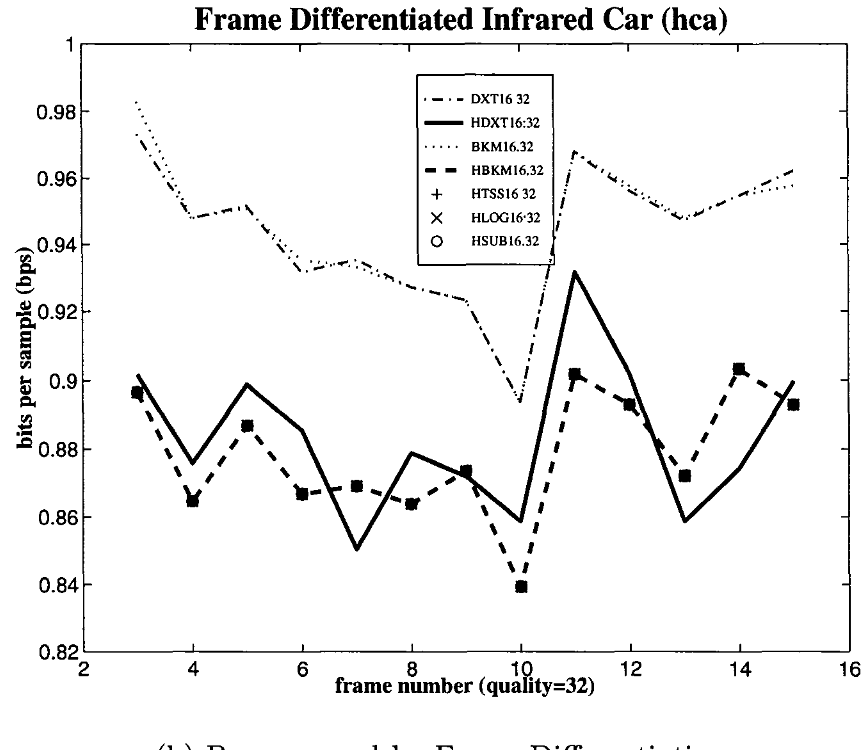
Figure 6.11: Comparison of different approaches on Frame 10 of Infrared Car sequence (HCA) for block size 16×16 and search size 32×32. Visualization is applied to (c) by setting the saturation level to 25.





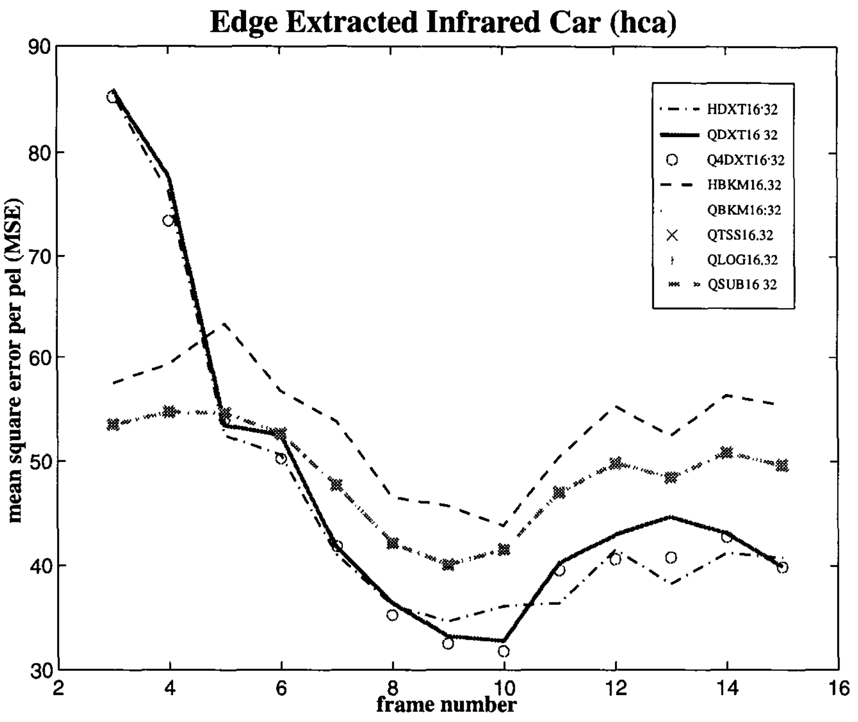
(a) Preprocessed by Edge Extraction

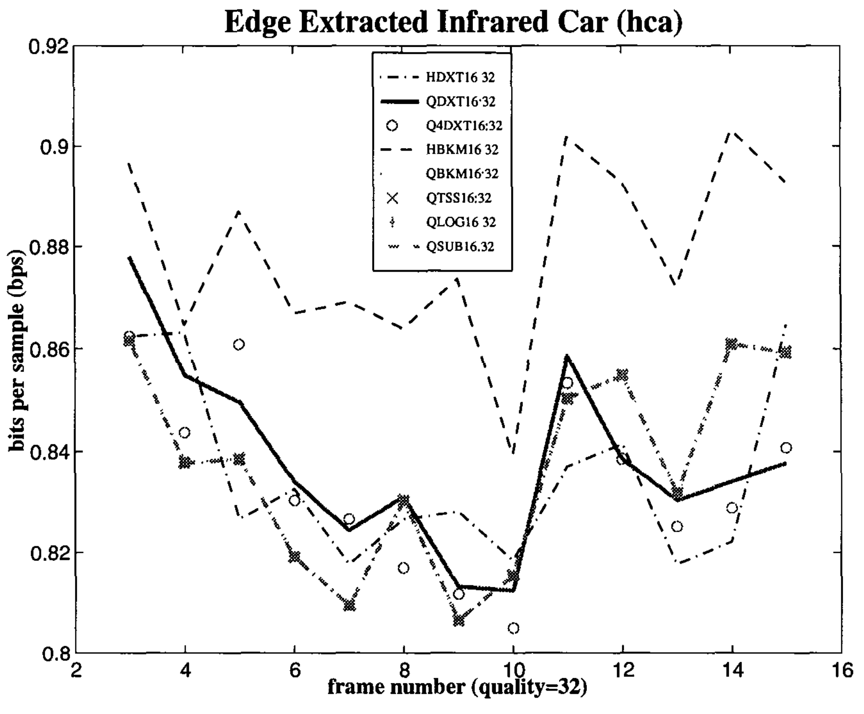




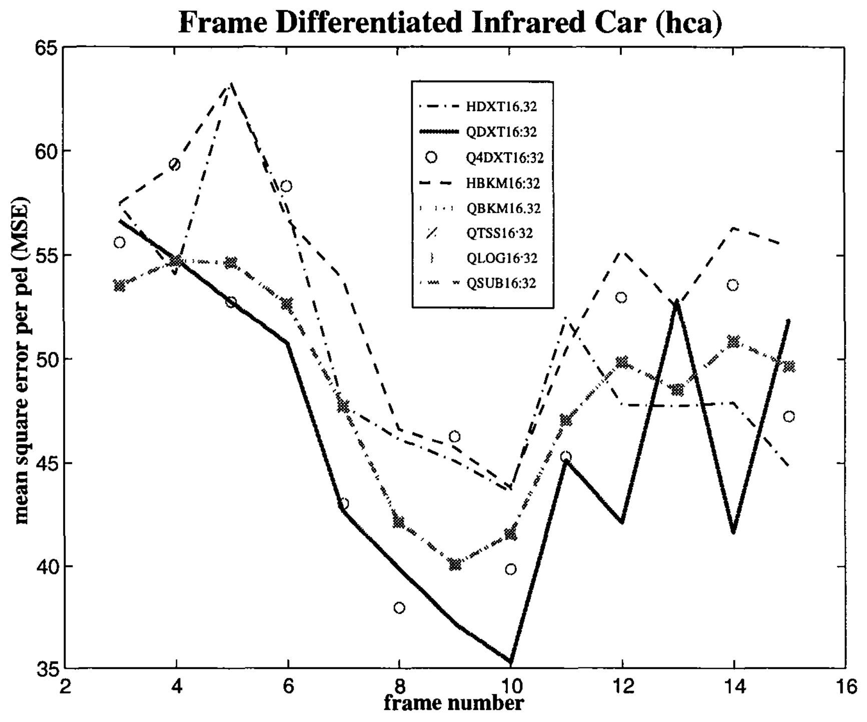
(b) Preprocessed by Frame Differentiation

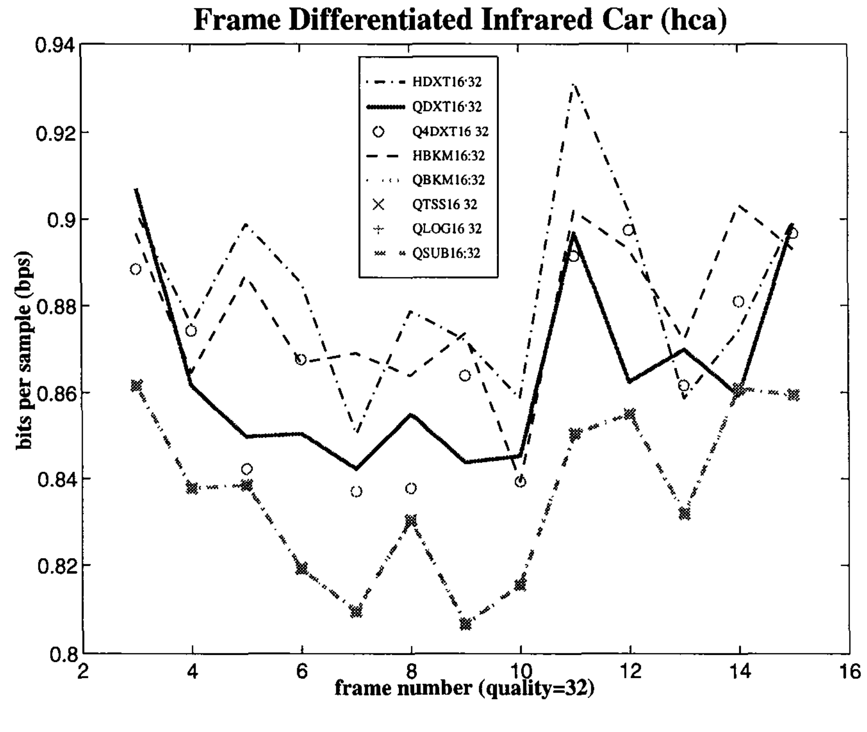
Figure 6.12: Simulation Results for motion estimation of half-pel accuracy on the Infrared Car sequence (HCA).





(a) Preprocessed by Edge Extraction





(b) Preprocessed by Frame Differentiation

Figure 6.13: Simulation Results for motion estimation of quarter-pel accuracy on the Infrared Car sequence (HCA).

Approach	MSE	MSE difference	MSE ratio	BPF	BPS	BPS ratio
INTEGER-PEL ACCURACY						
BKM	67.902	0.000	0.0%	10156	0.945	0.0%
frame differentiated DXT	68.355	0.453	0.7%	10150	0.944	-0.1%
edge extracted DXT	72.518	4.615	6.8%	10177	0.946	0.2%
TSS	68.108	0.206	0.3%	10159	0.945	0.0%
LOG	68.108	0.206	0.3%	10159	0.945	0.0%
SUB	68.493	0.591	0.9%	10159	0.945	0.0%
HALF-PEL ACCURACY						
HBKM	53.596	0.000	0.0%	9448	0.879	0.0%
frame differentiated HDXT	50.371	-3.224	-6.0%	9501	0.884	0.6%
edge extracted HDXT	47.013	-6.582	-12.3%	8981	0.835	-4.9%
HTSS	53.596	0.000	0.0%	9448	0.879	0.0%
HLOG	53.596	0.000	0.0%	9448	0.879	0.0%
HSUB	53.596	0.000	0.0%	9448	0.879	0.0%
QUARTER-PEL ACCURACY						
QBKM	48.677	0.000	0.0%	8996	0.837	0.0%
frame differentiated QDXT	46.426	-2.251	-4.6%	9298	0.865	3.4%
edge extracted QDXT	48.067	-0.611	-1.3%	9013	0.838	0.2%
frame differentiated Q4DXT	49.277	0.600	1.2%	9328	0.868	3.7%
edge extracted Q4DXT	46.769	-1.908	-3.9%	8969	0.834	-0.3%
QTSS	48.677	0.000	0.0%	8996	0.837	0.0%
QLOG	48.677	0.000	0.0%	8996	0.837	0.0%
QSUB	48.677	0.000	0.0%	8996	0.837	0.0%

Table 6.3: Performance summary of the DXT-ME algorithm with either frame differentiation or edge extraction as preprocessing against full search and fast search block matching approaches (BKM, TSS, LOG, SUB) and their half-pel (HBKM, HTSS, HLOG, HSUB) and quarter-pel (QBKM, QTSS, QLOG, QSUB) counterparts over the sequence "Infrared Car" (HCA). MSE difference is the difference from the MSE value of full search block matching method (BKM) and MSE ratio is the ratio of MSE difference to the MSE of BKM.

quarter-pel accuracy. These performance curves are summarized in Table 6.3 which shows surprisingly that the DXT-ME algorithm is better than the full search block matching algorithm for either half-pel or quarter-pel motion estimation in terms of MSE values. The edge extracted HDXT-ME is 12.3% better than HBKM-ME in terms of MSE values and 4.9% better in terms of BPS values. For quarter-pel accuracy, the edge extracted Q4DXT-ME is 3.9% better than QBKM-ME as shown in Table 6.3.

Chapter 7

DCT-Based Motion Compensation Schemes

Manipulation of compressed video data in DCT domain has been recognized as an important component in many advanced video applications [10, 11, 45, 69, 43]. In a video bridge where multiple sources of compressed video are combined and retransmitted in a network, techniques of manipulation and composition of compressed video streams entirely in DCT domain eliminate the need to build a decoding/encoding pair. Furthermore, manipulation in DCT domain provides flexibility to match heterogeneous Quality of Service requirements with different network or user resources such as prioritization of signal components from low order DCT coefficients to fit low-end communication resources. Finally, many manipulation functions can be performed in the DCT domain more efficiently than in the spatial domain [11] due to a much lower data rate and removal of the decoding/encoding pair. However, all the earlier works have been focused mainly on manipulation at the decoder side.

To serve the purpose of building a fully DCT-based motion compensated video coder, our aim is to develop the techniques of motion compensation in the DCT domain without converting back to the spatial domain before motion compensation. In [11], the method of pixelwise (integer-pel) translation in the DCT domain is proposed for extracting a DCT block out of four neighboring DCT blocks at an arbitrary position. Though addressing a different scenerio, this method can be applied after modification to integer-pel motion compensation in the DCT domain. For subpel motion compensation, we derive an equivalent form of bilinear interpolation in the DCT domain and then show that it is possible to perform other interpolation functions for achieving more accurate and visually better approximation in the DCT domain without increasing the complexity.

7.1 Integer-Pel DCT-Based Motion Compensation

As illustrated in Fig. 7.1(a), after motion estimation, the current block $\mathbf{C}$ of size $N \times N$ in the current frame I_t can be best predicted from the block displaced from the current block position by the estimated motion vector (d_u, d_v) in the spatial domain. This motion estimate determines which four contiguous predefined DCT blocks are chosen for the prediction of the current block out of eight surrounding DCT blocks and the block at the current block position. To extract the displaced DCT block in the DCT domain, a direct method is used to obtain separately from these four contiguous blocks four subblocks which can be combined together to form the final displaced DCT block as shown in Fig. 7.1(b) with the upper-left, lower-left, upper-right and lower-right blocks from the previous frame I_{t-1} labeled as $\mathbf{B}_1$, $\mathbf{B}_2$, $\mathbf{B}_3$ and $\mathbf{B}_4$ respectively [11]. Subblocks $\mathbf{S}_i$ are extracted in the spatial domain from these four blocks by premultiplication and postmultiplication of the windowing/shifting matrices, $\mathbf{H}_i$ and $\mathbf{V}_i$:

$$\mathbf{S}_k = \mathbf{H}_k \mathbf{B}_k \mathbf{V}_k, \quad \text{for } k = 1, \ldots, 4, \tag{7.1}$$

where $\mathbf{H}_k$ and $\mathbf{V}_k$ are the $N \times N$ windowing/shifting matrices defined as

$$\mathbf{H}_1 = \begin{bmatrix} \mathbf{0} & \mathbf{I}_{h_1} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}, \mathbf{V}_1 = \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{I}_{v_1} & \mathbf{0} \end{bmatrix}, \tag{7.2}$$

$$\mathbf{H}_2 = \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{I}_{h_2} & \mathbf{0} \end{bmatrix}, \mathbf{V}_2 = \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{I}_{v_2} & \mathbf{0} \end{bmatrix}, \tag{7.3}$$

$$\mathbf{H}_3 = \begin{bmatrix} \mathbf{0} & \mathbf{I}_{h_3} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}, \mathbf{V}_3 = \begin{bmatrix} \mathbf{0} & \mathbf{I}_{v_3} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}, \tag{7.4}$$

$$\mathbf{H}_4 = \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{I}_{h_4} & \mathbf{0} \end{bmatrix}, \mathbf{V}_4 = \begin{bmatrix} \mathbf{0} & \mathbf{I}_{v_4} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}. \tag{7.5}$$

Here $\mathbf{I}_n$ is the $n \times n$ identity matrix, i.e. $\mathbf{I}_n = \text{diag}\{1, \ldots, 1\}$ and n is determined by the height/width of the corresponding subblock. These pre-multiplication and post-multiplication matrix operations can be visualized in Fig. 7.1(c) where the overlapped grey areas represent the extracted subblock. Then these four subblocks are summed to form the desired translated block $\hat{\mathbf{B}}_{ref}$.

If we define the DCT operation on a $N \times N$ matrix $\mathbf{B}$ as

$$\text{DCT}\{\mathbf{B}\} = \mathbf{D}\mathbf{B}\mathbf{D}^T,$$

where the (k, m) element of $\mathbf{D}$ is the DCT-II kernel:

$$\mathbf{D}(k, m) = \frac{2}{N} C(k) \cos \frac{k\pi}{N}(m + \frac{1}{2}), \text{ for } k, m = 0, \ldots, N-1.$$

Therefore, $\mathbf{D}^T\mathbf{D} = \frac{2}{N}\mathbf{I}_N$. The formation of the DCT of $\hat{\mathbf{B}}_{ref}$ in the DCT domain can be described in this equation:

$$\text{DCT}\{\hat{\mathbf{B}}_{ref}\} = (\frac{N}{2})^2 \sum_{k=1}^{4} \text{DCT}\{\mathbf{H}_k\}\text{DCT}\{\mathbf{B}_k\}\text{DCT}\{\mathbf{V}_k\}. \tag{7.6}$$

This corresponds to pre and post multiplication of the DCT transformed $\mathbf{H}_k$ and $\mathbf{V}_k$ with the DCT of $\mathbf{B}_k$ since DCT is a unitary orthogonal transformation and is guaranteed to be distributive to matrix multiplications. The DCT of the motion-compensated residual (displaced frame difference or DFD) for the current block $\mathbf{C}$ is, therefore,

$$\text{DCT}\{DFD\} = \text{DCT}\{\hat{\mathbf{B}}_{ref}\} - \text{DCT}\{\mathbf{C}\}. \tag{7.7}$$

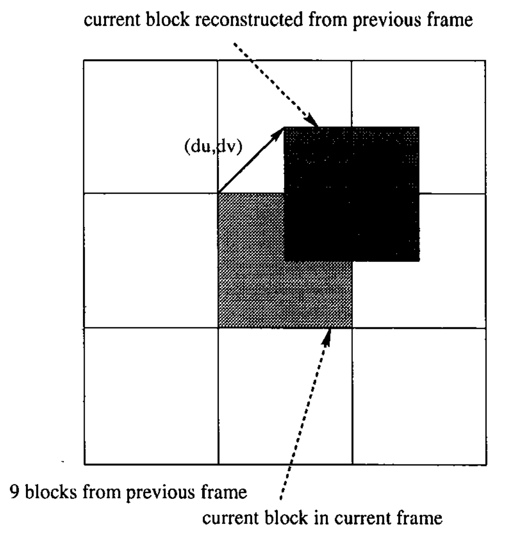
$\text{DCT}\{\mathbf{H}_k\}$ and $\text{DCT}\{\mathbf{V}_k\}$ can be precomputed and stored in the memory. Furthermore, many high-frequency coefficients of $\text{DCT}\{\mathbf{B}_k\}$ or displacement estimates are zeros (i.e., sparse and block aligned reference blocks), making the actual number of computations in (7.6) small. In [11], simulation results show that the DCT-domain approach is faster than the spatial-domain approach by about 10% to 30%. Further simplication is also possible as seen from Fig. 7.1(b) that

$$\mathbf{H}_U = \mathbf{H}_1 = \mathbf{H}_3,\ \mathbf{H}_L = \mathbf{H}_2 = \mathbf{H}_4, \tag{7.8}$$

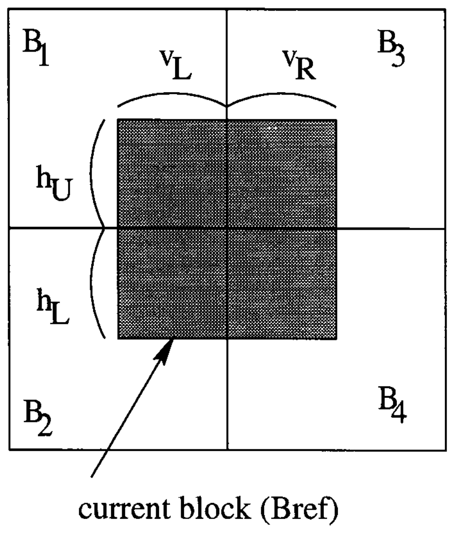
$$\mathbf{V}_L = \mathbf{V}_1 = \mathbf{V}_2,\ \mathbf{V}_R = \mathbf{V}_3 = \mathbf{V}_4. \tag{7.9}$$

Therefore, only four windowing/shifting matrices need to be accessed from the memory instead of eight.

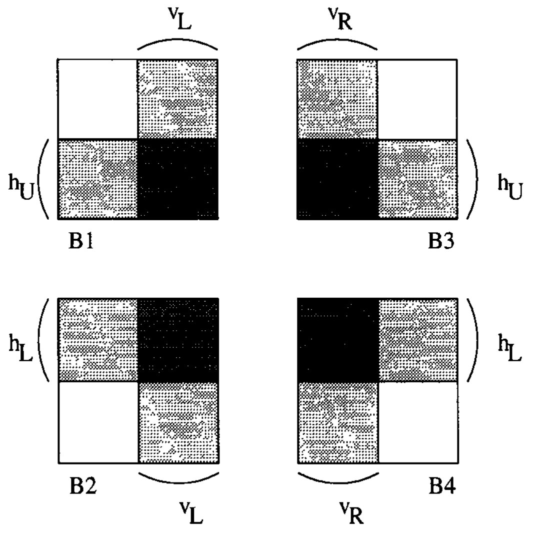
In [53], further savings in the computation of the windowing/shifting matrices is made by using fast DCT. It is reported that 47% reduction in computational complexity with fast DCT over the brute-force method without the assumption of sparseness and 68% with only the top-left 4×4 subblocks being nonzero can be achieved with the use of fast DCT.


current block reconstructed from previous frame
(du,dv)
9 blocks from previous frame
current block in current frame

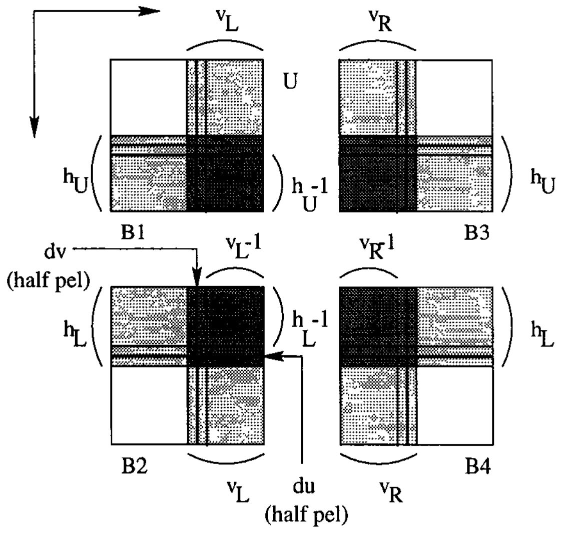
(a) DCT-Based Motion Compensation



(b) Pixelwise Translated DCT Block



(c) Integer-Pel Compensation



(d) Half-Pel Compensation

Figure 7.1: (a) Prediction of current block in current frame from four contiguous DCT blocks selected among nine neighboring blocks in previous frame based upon the estimated displacement vector for current block. (b) Schematic diagram of how a pixelwise translated DCT block is extracted from four contiguous DCT blocks. (c) Decomposition of integer-pel DCT-based translation as four matrix multiplication operations. (d) Decomposition of half-pel DCT-based translation as four matrix multiplication operations.

7.2 Subpixel DCT-Based Motion Compensation

For the case of subpixel motion, interpolation is used to predict interpixel values. According to the MPEG standards, bilinear interpolation is recommended for its simplicity in implementation and effectiveness in prediction [54, 55], though it is well known that a range of other interpolation functions, such as cubic, spline, Gaussian, and Lagrange interpolations, can provide better approximation accuracy and more pleasant visual quality [65, 66, 28, 24]. The complexity argument is true if the interpolation operation is performed in the spatial domain, but in the DCT domain, it is possible to employ better interpolation functions than the bilinear interpolation without any additional computational load increase.

7.2.1 Interpolation Filter

For simplicity of derivations, we start with the one dimensional half-pel bilinear interpolation and then proceed to the two dimensional case of quarter-pel accuracy with other interpolation functions. Consider two one dimensional adjacent blocks, $x_{1a}(n)$ and $x_{1b}(n)$ for $n = 0, \ldots, N-1$ as shown in Fig. 7.2. We want to extract a block $\{x_2(n)\}_{n=0}^{N-1}$ displaced u pixels to the right of $x_{1a}(0)$ where u is supposed to be an odd multiple of 0.5 (i.e. half-pel motion). Therefore, we can show that

$$x_2(n) = \begin{cases} \frac{1}{2}[x_{1a}(N+n-i) + x_{1a}(N+n-i+1)], & 0 \le n \le i-2, \\ \frac{1}{2}[x_{1a}(N-1) + x_{1b}(0)], & n = i-1, \\ \frac{1}{2}[x_{1b}(n-i) + x_{1b}(n-i+1)], & N-1 \ge n \ge i, \end{cases} \tag{7.10}$$

where $i = \lceil u \rceil$. In the matrix form,

$$\vec{\mathbf{x}}_2 = \mathbf{G}_{BL}(i)\vec{\mathbf{x}}_{1a} + \mathbf{G}_{BR}(i)\vec{\mathbf{x}}_{1b}, \tag{7.11}$$

where $\vec{\mathbf{x}}_2$, $\vec{\mathbf{x}}_{1a}$, and $\vec{\mathbf{x}}_{1b}$ are the column vectors of $x_2(n)$, $x_{1a}(n)$ and $x_{1b}(n)$ respectively, and $\mathbf{G}_{BL}(i)$ and $\mathbf{G}_{BR}(i)$ are defined as follows:

$$\mathbf{G}_{BL}(i) = \frac{1}{2}\{\begin{bmatrix} \mathbf{0} & \mathbf{I}_i \\ \mathbf{0} & \mathbf{0} \end{bmatrix} + \begin{bmatrix} \mathbf{0} & \mathbf{I}_{i-1} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}\}, \tag{7.12}$$

$$\mathbf{G}_{BR}(i) = \frac{1}{2}\{\begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{I}_{N-i} & \mathbf{0} \end{bmatrix} + \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{I}_{N-i+1} & \mathbf{0} \end{bmatrix}\}. \tag{7.13}$$

In the DCT domain,

$$\mathrm{DCT}\{\vec{\mathbf{x}}_2\} = \mathrm{DCT}\{\mathbf{G}_{BL}(i)\}\mathrm{DCT}\{\vec{\mathbf{x}}_{1a}\} + \mathrm{DCT}\{\mathbf{G}_{BR}(i)\}\mathrm{DCT}\{\vec{\mathbf{x}}_{1b}\}. \tag{7.14}$$

Here $\mathbf{G}_{BL}(i)$ and $\mathbf{G}_{BR}(i)$ can be regarded as bilinear interpolation filter matrices which act as a linear filter or transform. Therefore, $\mathbf{G}_{BL}(i)$ and $\mathbf{G}_{BR}(i)$ can be replaced by any FIR filter or interpolation function of finite duration (preferably with the length much smaller than the block size N).

7.2.2 Bilinear Interpolated Subpixel Motion Compensation

For the 2-D case, if (u, v) is the displacement of the reconstructed block $\hat{\mathbf{B}}_{ref}$ measured from the upper left corner of the block $\mathbf{B}_1$, then for $h_U = \lceil u \rceil$ and $v_L = \lceil v \rceil$,

$$\mathrm{DCT}\{\hat{\mathbf{B}}_{ref}\} = \sum_{k=1}^{4} \mathrm{DCT}\{\mathbf{H}_k\}\mathrm{DCT}\{\mathbf{B}_k\}\mathrm{DCT}\{\mathbf{V}_k\}, \tag{7.15}$$

where

$$\mathbf{H}_1 = \mathbf{H}_3 = \mathbf{H}_U = \mathbf{G}_{BL}(h_U), \tag{7.16}$$

$$\mathbf{H}_2 = \mathbf{H}_4 = \mathbf{H}_L = \mathbf{G}_{BR}(h_U), \tag{7.17}$$

$$\mathbf{V}_1 = \mathbf{V}_2 = \mathbf{V}_L = \mathbf{G}_{BL}^T(v_L), \tag{7.18}$$

$$\mathbf{V}_3 = \mathbf{V}_4 = \mathbf{V}_R = \mathbf{G}_{BR}^T(v_L). \tag{7.19}$$

Here

$$[\mathbf{G}_{BL}(h_U)\ \mathbf{G}_{BR}(h_U)] = \begin{bmatrix} 0 & \cdots & 0 & 0.5 & 0.5 & 0 & \cdots \\ \vdots & \ddots & \vdots & 0 & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & \cdots & 0 & 0.5 & 0.5 \end{bmatrix}. \tag{7.20}$$

Once again, $\mathbf{G}_{BL}(\cdot)$ and $\mathbf{G}_{BR}(\cdot)$ can be precomputed and stored in the memory as in the case of integer-pel motion compensation and thus the extra computational load for doing bilinear interpolation is eliminated.

7.2.3 Cubic Interpolated Subpixel Motion Compensation

Three different interpolation functions, namely cubic, cubic spline and bilinear interpolations, are plotted in Fig. 7.3(a). As can be seen, the bilinear interpolation has the shortest filter length and the cubic spline has a longest ripple but the cubic spline has the smallest approximation error among these three [24]. To compromise between filter length and approximation accuracy, we choose the cubic interpolation in the simulation. By choosing the resolution of the filter as half a pixel length, the bilinear interpolation $f_{hb}(n) = [0.5, 1, 0.5]$ and the cubic interpolation $f_{hc}(n) = [-0.0625, 0, 0.5625, 1.0000, 0.5625, 0, -0.0625]$. From Fig. 7.3(b), it is clear that the contributions at the half-pel position from all the pixel values are summed up and give rise to the bilinear filter matrices $\mathbf{G}_{BL}(\cdot)$ and $\mathbf{G}_{BR}(\cdot)$. In a similar way, as in Fig. 7.3(c), the cubic filter matrices $\mathbf{G}_{CL}(\cdot)$ and $\mathbf{G}_{CR}(\cdot)$ can be defined as

$$\mathbf{G}_{CL}(i) = \begin{bmatrix} \mathbf{0} & -0.0625\mathbf{I}_{i+1} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} + \begin{bmatrix} \mathbf{0} & 0.5625\mathbf{I}_i \\ \mathbf{0} & \mathbf{0} \end{bmatrix}$$

$$+\begin{bmatrix} \mathbf{0} & 0.5625\mathbf{I}_{i-1} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} + \begin{bmatrix} \mathbf{0} & -0.0625\mathbf{I}_{i-2} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}, \tag{7.21}$$

$$\mathbf{G}_{CR}(i) = \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ -0.0625\mathbf{I}_{N-i-1} & \mathbf{0} \end{bmatrix} + \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ 0.5625\mathbf{I}_{N-i} & \mathbf{0} \end{bmatrix}$$

$$+\begin{bmatrix} \mathbf{0} & \mathbf{0} \\ 0.5625\mathbf{I}_{N-i+1} & \mathbf{0} \end{bmatrix} + \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ -0.0625\mathbf{I}_{N-i+2} & \mathbf{0} \end{bmatrix}. \tag{7.22}$$

Here $\mathbf{G}_{CL}(\cdot)$ and $\mathbf{G}_{CR}(\cdot)$ can be precomputed and stored. Therefore, its computational complexity remains the same as both integer-pel and half-pel bilinear interpolated DCT-based motion compensation methods. The reconstructed DCT block and the corresponding motion-compensated residual can be obtained in a similar fashion:

$$\text{DCT}\{\hat{\mathbf{B}}_{ref}\} = \sum_{k=1}^{4} \text{DCT}\{\mathbf{H}_k\}\text{DCT}\{\mathbf{B}_k\}\text{DCT}\{\mathbf{V}_k\}, \tag{7.23}$$

$$\text{DCT}\{DFD\} = \text{DCT}\{\hat{\mathbf{B}}_{ref}\} - \text{DCT}\{\mathbf{C}\}, \tag{7.24}$$

where

$$\mathbf{H}_1 = \mathbf{H}_3 = \mathbf{H}_U = \mathbf{G}_{CL}(h_U), \tag{7.25}$$

$$\mathbf{H}_2 = \mathbf{H}_4 = \mathbf{H}_L = \mathbf{G}_{CR}(h_U), \tag{7.26}$$

$$\mathbf{V}_1 = \mathbf{V}_2 = \mathbf{V}_L = \mathbf{G}_{CL}^T(v_L), \tag{7.27}$$

$$\mathbf{V}_3 = \mathbf{V}_4 = \mathbf{V}_R = \mathbf{G}_{CR}^T(v_L). \tag{7.28}$$

This idea can be extended to other interpolation functions such as sharped Gaussian [66] and quarter-pel accuracy.

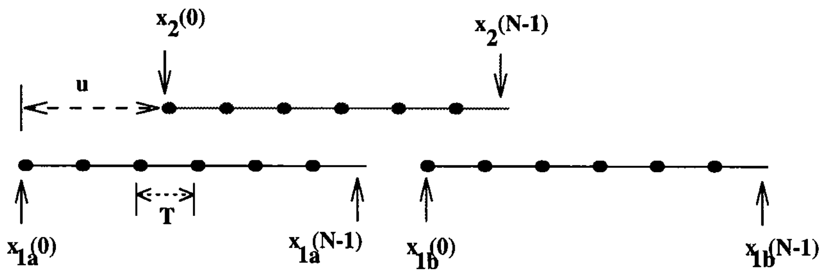


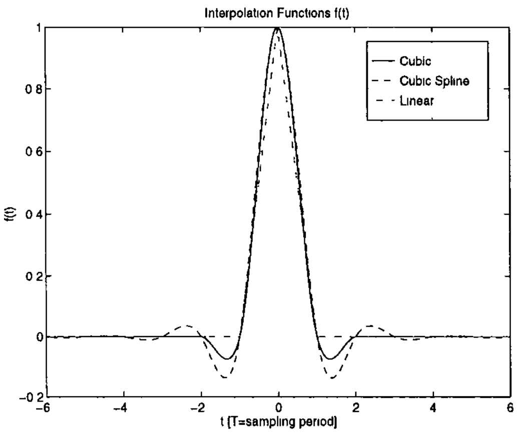
Figure 7.2: Illustration of extraction of the subpel displaced block $x_2(n)$ from two adjacent 1-D blocks $x_{1a}(n)$ and $x_{1b}(n)$ with bilinear interpolation.

7.2.4 Simulation

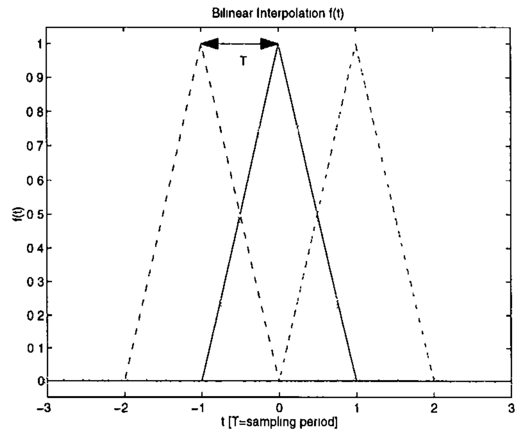
Simulation is performed on the Infrared Car and Miss America sequences to demonstrate the effectiveness of our bilinear and cubic motion compensation methods.

The first set of simulations subsamples each picture $I_t(i,j)$ from the sequences (i.e. $y(i,j) = I_t(2*i, 2*j)$) and then this shrinked picture $y(i,j)$ is displaced by a half-pel motion vector (arbitrarily chosen as $(2.5, 1.5)$) with both bilinear and cubic interpolated motion compensation methods. The mean square errors per pixel (MSE) are computed as $MSE = \frac{\sum_{i,j}[\hat{x}(i,j)-x(i,j)]^2}{N^2}$ by treating the original unsampled pixels $I_t(2*i+1, 2*j+1)$ as the reference picture $x(i,j) = I_t(2*i+1, 2*j+1)$ where $\hat{x}(i,j)$ is the predicted pixel value from $y(i,j)$. As shown in Fig. 7.4, the zero-order interpolation is also simulated for comparison. The zero-order interpolation, also called sample-and-hold interpolation, simply takes the original pixel value as the predicted half-pel pixel value [28]. As can be seen in Fig. 7.4, both the bilinear and cubic methods have much lower MSE values than the zero-order method and also the cubic method performs much better than the bilinear counterpart without increased computational load.

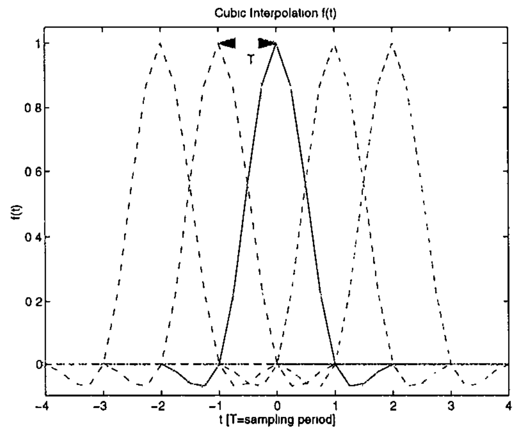
Fig. 7.5 shows the results of another set of simulations in which the subpixel



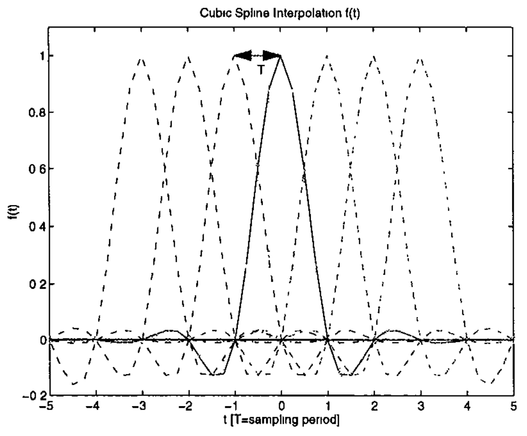
(a) Different Interpolation Functions



(b) Bilinear Interpolation

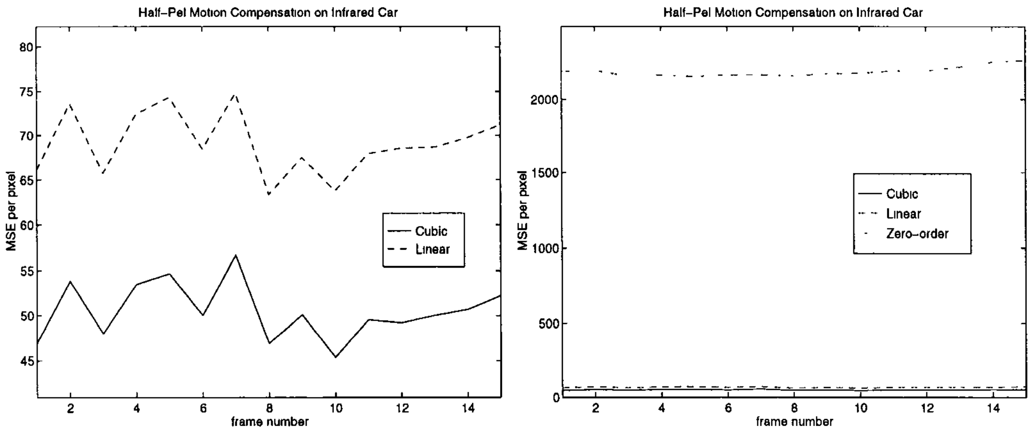


(c) Cubic Interpolation

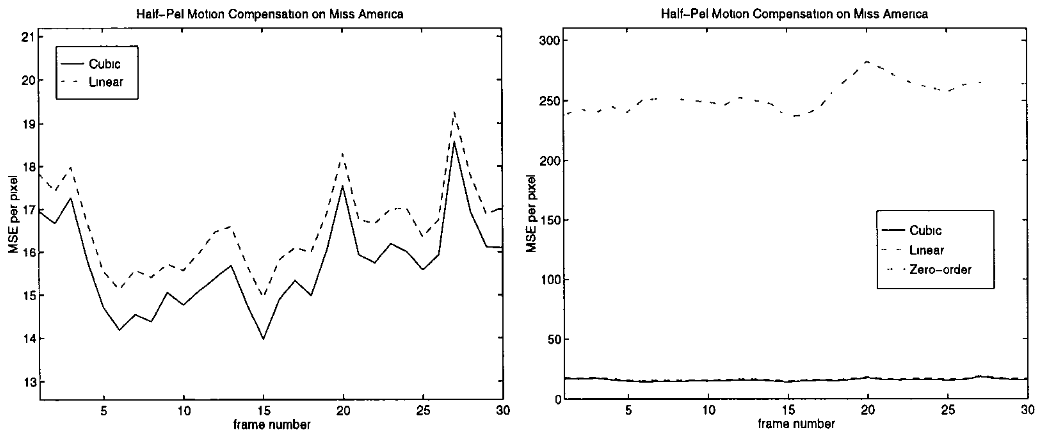


(d) Cubic Spline Interpolation

Figure 7.3: (a) plots different interpolation functions. (b), (c), and (d) depict how to form a pre or post multiplication matrix for half-pel or even quarter-pel DCT-based motion compensation.

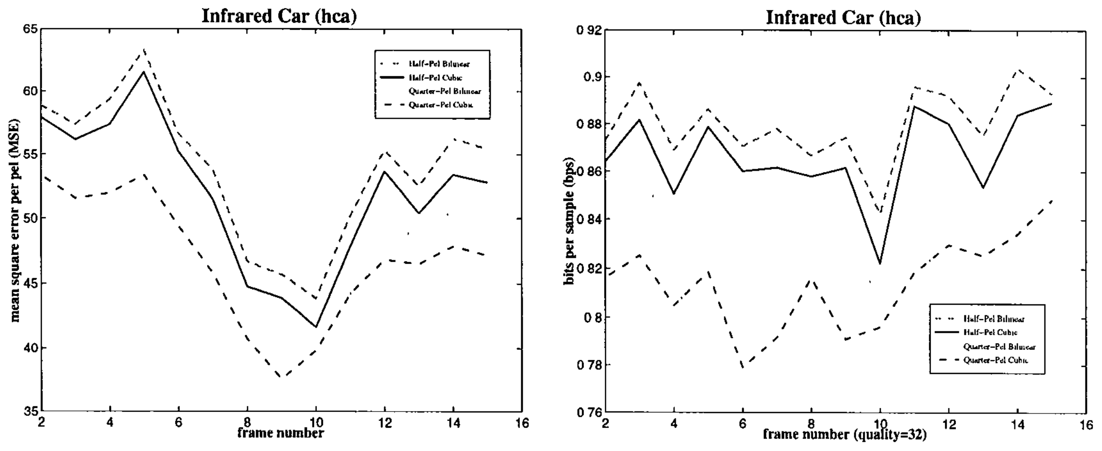


(a) Infrared Car sequence

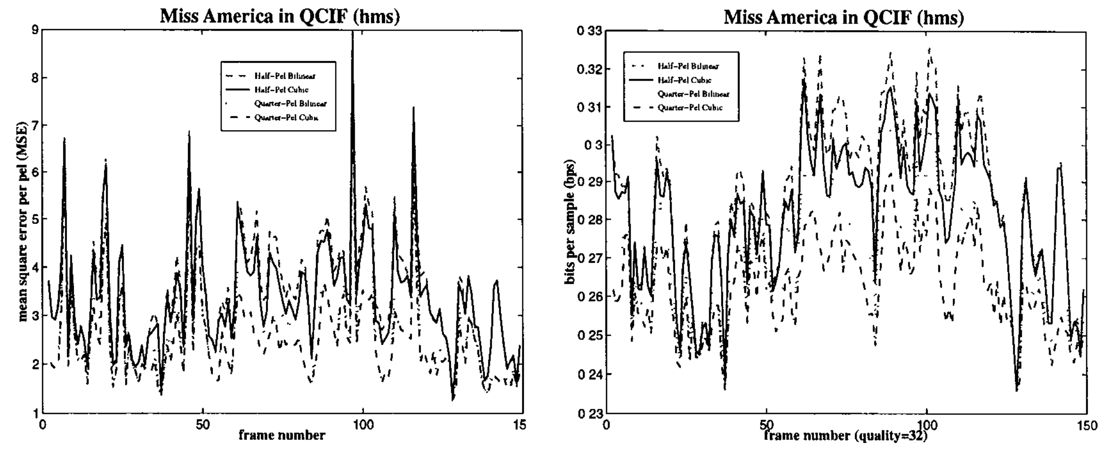


(b) Miss America sequence

Figure 7.4: Pictures from the Infrared Car and Miss America sequences are subsampled and displaced by a half-pel motion vector with different motion compensation methods. The MSE-per-pixel values are obtained by comparing the original unsampled pixel values with the predicted pixel values of the motion compensated residuals. Zero-order interpolation means replication of sampled pixels as the predicted pixel values.



(a) Infrared Car sequence



(b) Miss America sequence

Figure 7.5: Pictures from the Infrared Car and Miss America sequences are subsampled and displaced by a half-pel motion vector with different motion compensation methods. The MSE-per-pixel values are obtained by comparing the original unsampled pixel values with the predicted pixel values of the motion compensated residuals. Zero-order interpolation means replication of sampled pixels as the predicted pixel values.

DCT-based motion compensation algorithms generate motion compensated residuals of the sequences "Infrared Car" and "Miss America" based on the displacement estimates of the full search block matching algorithm, where the residuals are used to compute the MSE and BPS values for comparison. It can be seen that the cubic interpolation approach achieves lower MSE and BPS values than the bilinear interpolation.

7.3 Interpolation By DCT/DST

Discrete Cosine Transform of type I (DCT-I) or type II (DCT-II) have successfully been applied to discrete interpolation applications [76, 77, 2, 25]. Interpolation using DST or the more general W transform has also been studied over the years [81, 78, 80, 79]. Interpolation using DCT/DST is found to surpass the usual DFT interpolation method, especially for sinusoidal inputs, and, by taking into account of the boundary condition, very accurate interpolated values can be generated [25]. In the following, we will relate DCT-I interpolation with the Nyquist sampling theorem and show that, by means of DCT-I interpolation, the DCT-II coefficients of a half-pel shifted block can be directly obtained from the DCT-I cofficients of the original block.

7.3.1 DCT-I Interpolated Sequence

The procedure of interpolation using DCT-I is given as follows:

1. Compute the modified DCT-I defined as:

$$Y(m) = \text{DCT-I}\{y(n)\}$$

$$= \frac{2}{N} K_m \sum_{n=0}^{N} K_n y(n) \cos(\frac{mn\pi}{N}); \text{ for } m = 0, \ldots, N, \tag{7.29}$$

where

$$K_m = \begin{cases} \frac{1}{2}, & m = 0, \text{ or } N, \\ 1, & \text{otherwise.} \end{cases}$$

2. Append zeros to the DCT-I sequence $Y(m)$ to form another DCT-I sequence $Z(m)$:

$$Z(m) = \begin{cases} Y(m), & m = 0, \ldots, N, \\ 0, & m = N+1, \ldots, MN. \end{cases} \tag{7.30}$$

3. Obtain the interpolated sequence $z(n)$ by calculating the modified inverse DCT-I transform of the zero-padded $Z(m)$ as below:

$$\begin{aligned} z(n) &= \text{DCT-I}^{-1}\{Z(m)\} \\ &= \sum_{m=0}^{MN} Z(m) \cos(\frac{mn\pi}{MN}); \text{ for } n = 0, \ldots, MN. \end{aligned} \tag{7.31}$$

The above interpolation procedure of using DCT-I can be shown to be equivalent to upsampling of the reconstructed bandlimited signal from the sequence $y(n)$ by a pair of sinc-like η functions. By defining

$$Q_c(n, \nu) \triangleq \frac{2}{N} K_n \sum_{m=0}^{N} K_m \cos(\frac{mn\pi}{N}) \cos(\frac{m\nu\pi}{N}), \tag{7.32}$$

the interpolated sequence is

$$z(n) = \sum_{n'=0}^{N} y(n') Q_c(n', \frac{n}{M}). \tag{7.33}$$

Notice that

$$Q_c(n, \nu) = \frac{K_n}{N}[\sum_{m=0}^{N} K_m \cos\frac{m\pi}{N}(n - \nu) + \sum_{m=0}^{N} K_m \cos\frac{m\pi}{N}(n + \nu)] \tag{7.34}$$

$$= \frac{K_n}{N}\{\eta(n - \nu) + \eta(n + \nu)\} \tag{7.35}$$

where

$$\eta(x) \triangleq \sum_{m=0}^{N} K_m \cos\frac{m\pi}{N}(x). \tag{7.36}$$

It can be shown that

$$\eta(x) = \frac{1}{2}\sin\pi x \cdot \frac{\cos(\frac{\pi x}{2N})}{\sin(\frac{\pi x}{2N})} \tag{7.37}$$

$$\approx N\text{sinc}(x) \text{ if } \pi x \ll 2N. \tag{7.38}$$

As a matter of fact, $\eta(x)$ is the last term of the ξ function in Chapter 6 as illustrated in Fig. 6.2. Since the orthogonal equation is

$$\frac{2}{N}K_m \sum_{n=0}^{N} K_n \cos(\frac{m'n\pi}{N})\cos(\frac{mn\pi}{N}) = \delta(m'-m), \tag{7.39}$$

we can show that

$$Q_c(n,\nu) = \delta(n-\nu) \text{ if } \nu \text{is an integer.} \tag{7.40}$$

Therefore, $z(nM) = y(n)$ for $n = 0, \ldots, N$. This satisfies the requirement of being an interpolation function.

From the Nyquist sampling theorem, a continuous bandlimited signal, $f(t)$, can be perfectly reconstructed from its sampled sequence, $f(nT)$ for the sampling interval T by a series of sinc functions [59, 33]:

$$f(t) = \sum_{n=-\inf}^{\inf} f(nT)\text{sinc}(n - \frac{t}{T}). \tag{7.41}$$

The reconstructed $f(t)$ can then be resampled at a different sampling rate, $T_1 = T/M$, to generate an interpolated sequence $f(mT_1)$.

$$f(mT_1) = f(mT/M) = \sum_{n=-\infty}^{\infty} f(nT)\text{sinc}(n - \frac{m}{M}). \tag{7.42}$$

Therefore, interpolation using DCT-I expressed in (7.33) is the truncated version of (7.42) for a symmetric sequence $f(nT)$.

7.3.2 DCT-II of DCT-I Interpolated Half-Pel Motion Compensated Block

Given a sequence of length $N+1$, $\{y(n); n=0,\ldots,N\}$, the half-pel shifted sequence $\{w(n); n=0,\ldots,N-1\}$ is generated by sampling the DCT-I interpolated sequence of length $2N+1$, $z(n)$ such that

$$w(i) = z(2i+1) \text{ for } i = 0,\ldots,N-1. \tag{7.43}$$

The DCT-II coefficients $W(k)$ of $w(i)$ are found to have simple relationship with modified DCT-I cofficients $Y(m)$ of the original sequence $y(n)$ as follows:

$$\begin{aligned} W(k) &\triangleq DCT-II\{w(i)\} = \frac{2}{N}C(k)\sum_{i=0}^{N-1} z(i)\cos\frac{k\pi}{N}(i+\frac{1}{2}) \\ &= C(k)\sum_{m=0}^{N-1} Y(m)[\delta(m+k)+\delta(m-k)] \text{ for } k=0,\ldots,N-1. \end{aligned} \tag{7.44}$$

Therefore,

$$W(0) = \sqrt{2}Y(0);\ W(k) = Y(k) \text{ for } k=1,\ldots,N-1. \tag{7.45}$$

With this simple relationship, once the modified DCT-I coefficients of a $N+1$ sequence are obtained, the DCT-II of the DCT-I interpolated half-pel shifted block will easily be obtained via (7.45).

Chapter 8

Conclusions and Future Research

In this thesis, we propose a fully DCT-based motion-compensated video coder structure which is not only compliant with the hybrid motion-compensated DCT video coding standards (H.261, MPEG, HDTV) but also has a higher system throughput and a lower overall complexity than the conventional video coder structure due to removal of DCT and IDCT from the feedback loop. Therefore, it is more suitable for high quality and high bit rate video applications such as HDTV. To realize such a fully DCT-based coder, we develop DCT-based techniques and algorithms.

Due to the fact that the DCT pseudo phase techniques and DXT-ME algorithm can estimate motion in the DCT domain, their immediate application is video coding, realizing the fully DCT-based motion-compensated video coder structure which contains, in the performance-critical feedback loop of the coder, only one major component, the transform-domain motion estimation unit, instead of three major components as in the conventional hybrid DCT motion-compensated video coder design, and thus achieves higher throughput and lower system complexity. In addition to this advantage, the DXT-ME algorithm has low computational complexity: $O(N^2)$ as compared to $O(N^4)$ for the full search block matching approach (BKM-ME) or 75776 operations versus 130816 operations for BKM-ME, depending on the actual implementation. Even though the DXT-ME algorithm is not in the category of the fast search block matching schemes, we compare its performance in application to video coding with BKM-ME and some fast search approaches such as three step search (TSS), logarithmic search (LOG) and subsampled search (SUB), and find that for the Flower Garden and Infrared Car sequences, the DXT-ME algorithm achieves fewer bits per sample of the motion-compensated residual images (DFD) than all the other fast search approaches. Furthermore, its DCT-based nature enables us to incorporate its implementation with the DCT codec design to gain further savings in complexity and take advantage of advances in research on the DCT codec design. Finally the DXT-ME algorithm has inherently highly parallel operations in computing the pseudo phases and thus it it very suitable for VLSI implementation.

We also develop the DCT-based subpixel motion estimation techniques based on the subpel sinusoidal orthogonal principles and preservation of subpixel motion information in DCT coefficients under the Nyquist condition. These techniques can estimate subpixel motion in DCT domain without any inter-pixel interpolation at a desired level of accuracy. Equally applicable to other areas as well, the proposed techniques are applied to video coding and result in DCT-based half-pel and quarter-pel motion estimation algorithms (HDXT-ME, QDXT-ME, Q4DXT-ME) which estimate motion with half-pel or quarter-pel accuracy without interpolation of input images. This results in significant savings in computational complexity for interpolation and far less data flow compared to the conventional block matching methods on interpolated images. Also, the resulting algorithms are more suitable for VLSI implementation [12, 50]. Furthermore, it avoids the deterioration of estimation precision caused by interpolation required in most current subpixel motion estimation schemes. In addition, the proposed DCT-based subpixel motion estimation technique and the resulting algorithms are scalable in the sense that higher estimation accuracy can be provided easily by applying the same subpel sinusoidal orthogonal principles without re-computing pseudo phases. Therefore, flexible fully DCT-based codec design is possible because the same hardware can support different levels of required accuracy. Meanwhile, the computational complexity of the DCT-based algorithms is only $O(N^2)$ compared to $O(N^4)$ for BKM-ME or its subpixel versions. Finally, HDXT-ME, QDXT-ME and Q4DXT-ME are DCT-based, enabling us to build a low-complexity and high-throughput fully DCT-based video coder.

Finally, we discuss the integer-pel DCT-based motion compensation method and develop the subpel DCT-based motion compensation schemes using the bilinear and cubic interpolation functions. We show that without increasing the number of computations, the cubic interpolated half-pel and quarter-pel schemes exhibit higher coding gain in terms of smaller MSE and BPS values than the bilinear interpolated counterparts. Furthermore, the benefits of performing motion compensation in the DCT domain come from the facts that

(1) Many DCT coefficients are zero and thus the actual number of computations required is cut down.

(2) Fast DCT algorithms can be employed to save computation;

(3) Better interpolation functions, other than the commonly used bilinear function, can be utilized to increase coding gain and provide better visual quality

without increasing complexity.

(4) There is no need for an IDCT-DCT pair, further reducing overall complexity, especially important for the case of a video bridge or a fully DCT-based video coder.

(5) The DCT (DCT-II) of a half-pel motion compensated block can be obtained directly from the DCT-I coefficients of the original block.

Further research can focus on the direction of direct motion compensation from DCT coefficients using DCT/DST interpolation and the effects of this approach on the coding gain and visual quality of the resulting coder.

Future research efforts can be directed to the improvement of the DCT-based motion estimation approach, possibly through the multiresolution approach and wavelets, and the analysis of estimation accuracy of the DCT-based approach over quantized/dequantized DCT coefficients. A parallel architecture can be derived and a fixed-point analysis can be studied on the derived architecture. Issues of software and hardware implementation of the fully DCT-based video coder require further investigation. Adaptation of the fully DCT-based coder to different DCT-based video coding standards can also be considered. Incorporating scalability in MPEG-2 with DCT-based algorithms can enable us to design an efficient MPEG-2 encoder. Similar DCT-based algorithms and techniques can be applied to the design of an efficient decoder or video bridge.

In parallel to this DCT-based video coder research, several directions of low bitrate video coding can be attempted:

- Subspace approach to efficiently encode spatial images or residuals — DCT may not be the best transform to encode motion-compensated residuals which have different statistical characteristics from spatial images;

- Incorporation with Human Visual System (HVS) to improve the visual quality of heavily quantized DCT coefficients;

- Investigation of the possibility of frame interpolation benefitted from DCT-based techniques.

- With the increasing interests in wavelet-based image/video compression, it is important to investigate how we can estimate motion directly in the wavelet transform domain.

Appendix A

Determinant of the System Matrix

It is interesting to investigate the property of the determinant of the system matrix $|\mathbf{Z}_{t-1}(k,l)|$ in (4.23). A zero or near-zero value of $|\mathbf{Z}_{t-1}(k,l)|$ may jeopardize the performance. However, it can be shown analytically that this determinant will rarely be zero. Some algebraic manupilations on the determinant of $\mathbf{Z}_{t-1}(k,l)$ give us this close form of $|\mathbf{Z}_{t-1}(k,l)|$:

$$
\begin{aligned}
|\mathbf{Z}_{t-1}(k,l)| &= \begin{vmatrix} Z^{cc}_{t-1}(k,l) & -Z^{cs}_{t-1}(k,l) & -Z^{sc}_{t-1}(k,l) & Z^{ss}_{t-1}(k,l) \\ Z^{cs}_{t-1}(k,l) & Z^{cc}_{t-1}(k,l) & -Z^{ss}_{t-1}(k,l) & -Z^{sc}_{t-1}(k,l) \\ Z^{sc}_{t-1}(k,l) & -Z^{ss}_{t-1}(k,l) & Z^{cc}_{t-1}(k,l) & -Z^{cs}_{t-1}(k,l) \\ Z^{ss}_{t-1}(k,l) & Z^{sc}_{t-1}(k,l) & Z^{cs}_{t-1}(k,l) & Z^{cc}_{t-1}(k,l) \end{vmatrix} \\
&= (Z^{cs}_{t-1}(k,l)^2 - Z^{sc}_{t-1}(k,l)^2)^2 + (Z^{cc}_{t-1}(k,l)^2 - Z^{ss}_{t-1}(k,l)^2)^2 \\
&\quad +2(Z^{cc}_{t-1}(k,l)Z^{cs}_{t-1}(k,l) + Z^{sc}_{t-1}(k,l)Z^{ss}_{t-1}(k,l))^2 \\
&\quad +2(Z^{cc}_{t-1}(k,l)Z^{sc}_{t-1}(k,l) + Z^{cs}_{t-1}(k,l)Z^{ss}_{t-1}(k,l))^2 \qquad \text{(A.1)}
\end{aligned}
$$

in which $|\mathbf{Z}_{t-1}(k,l)| = 0$ implies that

$$
\begin{aligned}
Z^{cs}_{t-1}(k,l) &= \pm Z^{sc}_{t-1}(k,l), \\
Z^{cc}_{t-1}(k,l) &= \pm Z^{ss}_{t-1}(k,l),
\end{aligned}
$$

$$Z_{t-1}^{cc}(k,l)Z_{t-1}^{cs}(k,l) = -Z_{t-1}^{sc}(k,l)Z_{t-1}^{ss}(k,l),$$

$$Z_{t-1}^{cc}(k,l)Z_{t-1}^{sc}(k,l) = -Z_{t-1}^{cs}(k,l)Z_{t-1}^{ss}(k,l).$$

However, the last two equalities allow only two possible conditions:

$$\textit{either}\ \ Z_{t-1}^{cs}(k,l) = Z_{t-1}^{sc}(k,l),\ \ Z_{t-1}^{cc}(k,l) = -Z_{t-1}^{ss}(k,l), \tag{A.2}$$

$$\textit{or}\ Z_{t-1}^{cs}(k,l) = -Z_{t-1}^{sc}(k,l),\ Z_{t-1}^{cc}(k,l) = Z_{t-1}^{ss}(k,l). \tag{A.3}$$

Alternatively, in their explicit compact forms,

$$\sum_{m,n=0}^{N-1} x_{t-1}(m,n)\sin[\frac{\pi}{N}(km \mp ln)] = 0, \tag{A.4}$$

$$\sum_{m,n=0}^{N-1} x_{t-1}(m,n)\cos[\frac{\pi}{N}(km \mp ln)] = 0. \tag{A.5}$$

Here the minus signs in (A.4) and (A.5) correspond to the first condition in (A.2) and the plus signs correspond to (A.3). Satisfying either condition requires $x_{t-1}(m,n) \equiv 0$. Therefore it is very unlikely that this determinant is zero and as such, the DXT-ME is a stable estimator. Even so, if $\mathbf{Z}_{t-1}(k,l) = 0$ really happens or $\mathbf{Z}_{t-1}(k,l)$ is less than a threshold, then we can let $f(k,l) = g(k,l) = 1$, which is equivalent to the situation when $x_{t-1}(m,n) \equiv 0$. In this way, the catastrophic effect of computational precision of a certain implementation on the stability of DXT-ME will be kept to minimum or even eliminated.

Bibliography

[1] "Special Issue on Time Delay Estimation", *Signal Processing*, vol. 29, no. 3, 1981.

[2] J. I. Agbinya, "Interpolation using the discrete cosine transform", *Electronics letters*, vol. 28, no. 20, pp. 1927, September 1992.

[3] J. K. Aggarwal and N. Nandhakumar, "On the computation of motion from sequences of images – a review", *Proceedings of the IEEE*, vol. 76, no. 8, pp. 917–935, August 1988.

[4] H. K. Aghajan, C. D. Schaper, and T. Kailath, "Machine vision techniques for subpixel estimation of critical dimensions", *Optical Engineering*, vol. 32, no. 4, pp. 828–39, April 1993.

[5] T. Akiyama, H. Aono, K. Aoki, K. W. Ler, B. Wilson, T. Araki, T. Morishige, H. Takeno, A. Sato, S. Nakatani, and T. Senoh, "MPEG2 video codec using image compression DSP", *IEEE Transactions on Consumer Electronics*, vol. 40, pp. 466–72, 1994.

[6] B. F. Alexander and K. C. Ng, "Elimination of systematic error in subpixel accuracy centroid estimation", *Optical Engineering*, vol. 30, no. 9, pp. 1320–31, September 1991.

[7] Grand Alliance, *Grand Alliance HDTV System Specification*, April 1994.

[8] D. Brinthaupt, L. Letham, V. Maheshwari, J. Othmer, R. Spiwak, B. Edwards, C. Terman, and N. Weste, "A video decoder for H.261 video teleconferencing and MPEG stored interactive video applications", in *1993 IEEE International Solid-State Circuits Conference*, San Francisco, CA, USA, 1993, pp. 34–5.

[9] J. B. Burl, "A reduced order extended kalman filter for sequential images containing a moving object", *IEEE Trans. Image Processing*, vol. 2, no. 3, pp. 285–295, July 1993.

[10] S. F. Chang and D. G. Messerschmitt, "A new approach to decoding and compositing motion-compensated DCT-based images", in *Proc. IEEE Int. Conf. Acoustics, Speech, Signal Processing*, 1993, vol. V, pp. 421–424.

[11] S.-F. Chang and D. G. Messerschmitt, "Manipulation and compositing of MC-DCT compressed video", *IEEE Journal on Selected Areas in Communications*, vol. 13, no. 1, pp. 1, January 1995.

[12] C. T. Chiu and K. J. R. Liu, "Real-time parallel and fully pipelined two-dimensional DCT lattice structures with applications to HDTV systems", *IEEE Trans. Circuits and Systems for Video Technology*, vol. 2, no. 1, pp. 25–37, March 1992.

[13] I. J. Cox, J. B. Kruskal, and D. A. Wallach, "Predicting and estimating the accuracy of a subpixel registration algorithm", *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 12, no. 8, pp. 721, August 1990.

[14] G. de Haan and W. A. C. Biezen, "Sub-pixel motion estimation with 3-D recursive search block-matching", *Signal Processing: Image Communication*, vol. 6, no. 3, pp. 229–39, June 1994.

[15] F. Dufaux and F. Moscheni, "Motion estimation techniques for digital TV: a review and a new contribution", *Proceedings of the IEEE*, , no. 6, pp. 858–876, June 1995.

[16] J. P. Fillard, "Subpixel accuracy location estimation from digital signals", *Optical Engineering*, vol. 31, no. 11, pp. 2465–71, November 1992.

[17] J. P. Fillard, J. M. Lussert, M. Castagne, and H. M'timet, "Fourier phase shift location estimation of unfocused optical point spread functions", *Signal Processing: Image Communication*, vol. 6, no. 4, pp. 281–7, August 1994.

[18] M. Ghanbari, "the cross-search algorithm for motion estimation", *IEEE Trans. Communications*, vol. 38, no. 7, pp. 950–953, July 1990.

[19] B. Girod, "Motion-compensating prediction with fractional-pel accuracy", *IEEE Trans. Communications*, vol. 41, no. 4, pp. 604, April 1993.

[20] B. Girod, "Motion compensation: Visual aspects, accuracy, and fundamental limits", in *Motion Analysis and Image Sequence Processing*, M. I. Sezan and R. L. Lagendijk, Eds., chapter 5. Kluwer Academic Publishers, 1993.

[21] CCITT Recommendation H.261, *Video Codec for Audiovisual Services at p* $\times$ *64 kbit/s*, CCITT, August 1990.

[22] Draft CCITT Recommendation H.263, *Line transmission of non-telephone signals: video coding for low bitrate communication*, CCITT, July 1995.

[23] D. Heeger, "A model for extraction of image flow", in *Proc. First Int'l Conf. Computer Vision*, London, 1987, pp. 181–190.

[24] H. Hou and H. C. Andrews, "Cubic splines for image interpolation and digital filtering", *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 26, no. 6, pp. 508–517, 1978.

[25] C.-Y. Hsu and S.-M. Chen, "Discrete interpolation using the discrete cosine transform with the mapping of the boundary conditions", *IEEE signal processing letters*, vol. 2, no. 10, pp. 185, October 1995.

[26] S.-L. Iu, "Comparison of motion compensation using different degrees of sub-pixel accuracy for interfield/interframe hybrid coding of HDTV image sequences", in *1992 IEEE International Conference on Acoustics, Speech and Signal Processing*, San Francisco, CA, USA, 1992, vol. 3, pp. 465–8.

[27] G. Jacovitti and G. Scarano, "Discrete-time techniques for time-delay estimation", *Signal Processing*, vol. 41, no. 2, pp. 525–533, 1993.

[28] A. K. Jain, *Fundamental of Digital Image Processing*, Prentice-Hall, 1989.

[29] J. R. Jain and A. K. Jain, "Displacement measurement and its application in interframe image coding", *IEEE Trans. Communications*, vol. COM-29, pp. 1799–1806, December 1981.

[30] N. Jayant, "Signal compression: Technology targets and research directions", *IEEE Journal on Selected Areas in Communications*, vol. 5, no. 10, pp. 796–818, June 1992.

[31] N. Jayant, J. Johnston, and R. Safranek, "Signal compression based on models of human perception", *Proceedings of the IEEE*, vol. 81, no. 10, pp. 1385–422, October 1993.

[32] R. C. Kim and S. U. Lee, "A VLSI architecture for a pel recursive motion estimation algorithm", *IEEE Trans. Circuits and Systems*, vol. 36, no. 10, pp. 1291–1300, October 1989.

[33] S. P. Kim and W.-Y. Su, "Direct image resampling using block transform coefficients", *Signal Processing: Image Communication*, vol. 5, no. 3, pp. 259–272, May 1993.

[34] S. P. Kim and W. Y. Su, "Subpixel accuracy image registration by spectrum cancellation", in *1993 IEEE International Conference on Acoustics, Speech, and Signal Processing*, Minneapolis, MN, USA, 1993, vol. 5, pp. 153–6.

[35] T. Koga, K. Iinuma, A. Hirano, Y. Iijima, and T. Ishiguro, "Motion-compensated interframe coding for video conferencing", in *Proc. Nat. Telecom. Conf.*, New Orleans, LA, December 1981, pp. G.5.3.1–G.5.3.5.

[36] A. Kojima, N. Sakurai, and J. Kishigami, "Motion detection using 3D-FFT spectrum", in *ICASSP-93*, Minnesota, April 1993, IEEE, vol. V, pp. V213–V216.

[37] T. Komarek and P. Pirsch, "Array architectures for block-matching algorithms", *IEEE Trans. Circuits and Systems*, vol. 36, no. 10, pp. 1301–1308, October 1989.

[38] W. D. Kou and T. Fjallbrant, "A direct computation of DCT coefficients for a signal block taken from 2 adjacent blocks", *Signal Processing*, vol. 39, no. 7, pp. 1692–1695, 1991.

[39] C. D. Kuglin and D. C. Hines, "The phase correlation image alignment method", in *Proceedings of the 1975 IEEE International Conference on Systems, Man and Cybernetics*, September 1975, pp. 163–165.

[40] M. Kunt, M. Benard, and R. Leonardi, "Recent results in high-compression image coding", *IEEE Trans. Circuits and Systems*, vol. 34, no. 11, pp. 1306–1336, November 1987.

[41] M. Kunt, A. Ikonomopoulos, and M. Kocher, "Second-generation image-coding techniques", *Proceedings of the IEEE*, vol. 73, no. 4, pp. 549–574, April 1985.

[42] E. Lantz, "Subpixel signal centering and shift measurement using a recursive spectral phase algorithm", *Signal Processing*, vol. 17, no. 4, pp. 365, August 1989.

[43] J. B. Lee and B. G. Lee, "Transform domain filtering based on pipelining structure", *IEEE Trans. on Signal Processing*, vol. 40, pp. 2061–2064, August 1992.

[44] X. Lee, "A fast feature matching algorithm of motion compensation for hierarchical video codec", in *Proceedings of the SPIE: Visual Communications and Image Processing '92*, Boston, MA, USA, 1992, vol. 1818, pp. 1462–74.

[45] Y. Y. Lee and J. W. Woods, "Video post-production with compressed images", *SMPTE Journal*, vol. 103, pp. 76–84, February 1994.

[46] H. Li, A. Lundmark, and R. Forchheimer, "Image sequence coding at very low bitrates: A review", *IEEE Trans. Image Processing*, vol. 3, no. 5, pp. 589–608, September 1994.

[47] R. Li, B. Zeng, and M. L. Liou, "A new three-step search algorithm for block motion estimation", *IEEE Trans. Circuits and Systems for Video Technology*, vol. 4, no. 4, pp. 438–442, August 1994.

[48] B. Liu and A. Zaccarin, "New fast algorithms for the estimation of block motion vectors", *IEEE Trans. Circuits and Systems for Video Technology*, vol. 3, no. 2, pp. 148–157, April 1993.

[49] K. J. R. Liu and C. T. Chiu, "Unified parallel lattice structures for time-recursive Discrete Cosine/Sine/Hartley transforms", *IEEE Trans. Signal Processing*, vol. 41, no. 3, pp. 1357–1377, March 1993.

[50] K. J. R. Liu, C. T. Chiu, R. K. Kologotla, and J. F. JaJa, "Optimal unified architectures for the real-time computation of time-recursive Discrete Sinusoidal Transforms", *IEEE Trans. Circuits and Systems for Video Technology*, vol. 4, no. 2, pp. 168–180, April 1994.

[51] G. Madec, "Half pixel accuracy in block matching", in *Picture Coding Symp.*, Cambridge, MA, March 1990.

[52] J. S. McVeigh and S.-W. Wu, "Comparative study of partial closed-loop versus open-loop motion estimation for coding of HDTV", in *Proc. IEEE Workshop on Visual Signal Processing and Communications*, New Brunswick, September 1994, pp. 63–68.

[53] N. Merhav and V. Bhaskaran, "A fast algorithm for DCT-domain inverse motion compensation", in *Proc. IEEE Int. Conf. Acoustics, Speech, Signal Processing*, 1996, vol. IV, pp. 2309–2312.

[54] CCITT Recommendation MPEG-1, *Coding of Moving Pictures and Associated Audio for Digital Storage Media at up to about 1.5 Mbit/s*, ISO/IEC 11172, Geneve Switzerland, 1993.

[55] CCITT Recommendation MPEG-2, *Generic Coding of Moving Pictures and Associated Audio*, ISO/IEC 13818, Geneve Switzerland, 1994, H.262.

[56] H. G. Musmann, P. Pirsch, and H.-J. Grallert, "Advances in picture coding", *Proceedings of the IEEE*, vol. 73, no. 4, pp. 523–548, April 1985.

[57] A. N. Netravali and J. D. Robbins, "Motion compensated television coding – part 1", *Bell Syst. Tech. J.*, vol. 58, pp. 631–670, March 1979.

[58] A. Nosratinia and M. T. Orchard, "Discrete formulation of pel-recursive motion compensation with recursive least squares updates", in *1993 IEEE International Conference on Acoustics, Speech, and Signal Processing*, Minneapolis, MN, USA, 1993, vol. 5, pp. 229–232.

[59] A. V. Oppenheim and R. W. Schafer, *Discrete-Time Signal Processing*, Signal Processing. Prentice Hall, New Jersey, 1989.

[60] A. Papoulis, *Signal Analysis*, McGraw-Hill, Inc., 1977.

[61] P. Pirsch, N. Demassieux, and W. Gehrke, "VLSI architecture for video compression – a survey", *Proceedings of the IEEE*, , no. 2, pp. 220–246, February 1995.

[62] B. Porat and B. Friedlander, "A frequency domain algorithm for multiframe detection and estimation of dim targets", *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 12, no. 4, pp. 398–401, April 1990.

[63] X. Ran and C. Y. Choo, "Syntax-based arithmetic video coding for very low bitrate visual telephony", in *Proceedings of IEEE International Conference on Image Processing (ICIP)*, Washington, DC, October 1995, vol. 2, pp. 410–413.

[64] J. D. Robbins and A. N. Netravali, "Recursive motion compensation: A review", in *Image Sequence Processing and Dynamic Scene Analysis*, T. S. Huang, Ed., pp. 76–103. Springer-Verlag, Berlin, Germany, 1983.

[65] R. W. Schafer and L. R. Rabiner, "A digital signal processing approach to interpolation", *Proceedings of the IEEE*, pp. 692–702, June 1973.

[66] W. F. Schreiber, *Fundamentals of Electronic Imaging Systems – Some Aspects of Image Processing*, Springer-Verlag, 3rd edition, 1993.

[67] M. R. Shortis, T. A. Clarke, and T. Short, "A comparison of some techniques for the subpixel location of discrete target images", in *Proceedings of the SPIE: Videometrics III*, Boston, MA, USA, 1994, vol. 2350, pp. 239–50.

[68] A. Singh, *Optic Flow Computation – A Unified Perspective*, IEEE Computer Society Press, 1991.

[69] B. C. Smith and L. Rowe, "Algorithms for manipulating compressed images", *IEEE Comput. Graph. Appl.*, pp. 34–42, September 1993.

[70] R. Srinivasan and K. R. Rao, "Predictive coding based on efficient motion estimation", *IEEE Trans. Communications*, vol. COM-33, no. 8, pp. 888–896, August 1985.

[71] M. A. Tekalp, *Digital Video Processing*, Prentice-Hall, 1st edition, August 1995.

[72] G. A. Thomas, "Television motion measurement for DATV and other applications", Tech. Rep. 11, BBC Research Department, 1987.

[73] Q. Tian and M. N. Huhns, "Algorithms for subpixel registration", *Computer Vision, Graphics and Image Processing*, vol. 35, pp. 220–233, 1986.

[74] S.-I. Uramoto, A. Takabatake, and M. Yoshimoto, "A half-pel precision motion estimation processor for NTSC-resolution video", *IEICE Transactions on Electronics*, vol. 77, no. 12, pp. 1930, December 1994.

[75] L. D. Vos and M. Stegherr, "Parametrizable VLSI architectures for the full-search block-matching algorithm", *IEEE Trans. Circuits and Systems*, vol. 36, no. 10, pp. 1309–1316, October 1989.

[76] Z. Wang, "Interpolation using type i discrete cosine transform", *Electronics letters*, vol. 26, no. 15, pp. 1170, July 1990.

[77] Z. Wang, "Interpolation using the discrete cosine transform: Reconsideration", *Electronics letters*, vol. 29, no. 2, pp. 198, January 1993.

[78] Z. Wang, G. A. Jullien, and W. C. Miller, "Interpolation using the discrete sine transform with increased accuracy", *Electronics letters*, vol. 29, no. 22, pp. 1918, October 1993.

[79] Z. Wang, G. A. Jullien, and W. C. Miller, "The generalized discrete W transform and its application to interpolation", *Signal processing*, vol. 36, no. 1, pp. 99, March 1994.

[80] Z. Wang, J. J. Soltis, and W. C. Miller, "Accurate interpolation via the discrete W transform", *Electronics letters*, vol. 29, no. 14, pp. 1282, July 1993.

[81] Z. Wang and L. Wang, "Interpolation using the fast sine transform", *Signal Processing*, vol. 26, no. 2, pp. 131–137, January 1992.

[82] G. A. W. West and T. A. Clarke, "A survey and examination of subpixel measurement techniques", in *Proceedings of the SPIE: Close-Range Photogrammetry Meets Machine Vision*, Zurich, Switzerland, 1990, vol. 1395, pp. 456–63.

[83] K. M. Yang, M. T. Sun, and L. Wu, "A family of VLSI designs for the motion compensation block-matching algorithm", *IEEE Trans. Circuits and Systems*, vol. 36, no. 10, pp. 1317–1325, October 1989.

[84] P. Yip and K. R. Rao, "On the shift property of DCT's and DST's", *IEEE Trans. Acoustics, Speech, and Signal Processing*, vol. ASSP-35, no. 3, pp. 404–406, March 1987.

[85] R. W. Young and N. G. Kingsbury, "Frequency-domain motion estimation using a complex lapped transform", *IEEE Trans. Image Processing*, vol. 2, no. 1, pp. 2–17, January 1993.

[86] A. Zakhor and F. Lari, "Edge-based 3-D camera motion estimation with application to video coding", *IEEE Trans. Image Processing*, vol. 2, no. 4, pp. 481–498, October 1993.

[87] M. Ziegler, "Hierarchical motion estimation using the phase correlation method in 140 mbit/s HDTV-coding", in *Signal Processing of HDTV, II*, Turin, Italy, 1990, pp. 131–137.