# ESTIMATION OF MOTION FOR MOVEMENT COMPENSATED IMAGE PROCESSING

by

Philip Olurotimi Ogunbona, B.Sc.

March, 1987

A thesis submitted for the degree of Doctor of Philosophy of the University of London and for the Diploma of Membership of the Imperial College.

Electrical Engineering Department
Imperial College
London SW7 2BT

ABSTRACT

Estimation of motion from a sequence of images for use in motion compensated image processing has been investigated. Two motion estimation algorithms were developed and applied to motion compensated frame interpolation, temporal filtering and interframe differential pulse code modulation (DPCM) coding, respectively.

Using moment invariants as feature descriptors, a feature based motion estimation algorithm was developed. A means of segmenting the image frame into stationary and moving areas, based on the r.m.s value of the image frame difference signal, was developed. Feature vectors were generated and used in a two-stage block matching scheme. Experiments were performed using varying vector sizes. A vector size of 2 was found to be optimal, in the sense of computational efficiency and accuracy. Varying block sizes were also experimented with. A radial and angular moment invariant version of the scheme was also developed. This will be useful in circularly sampled images.

In a second development the differential technique of motion estimation was investigated in the projection domain. Using projections of the spatial and temporal differences an expression for estimating the 2-dimensional translational motion was derived (Projection Domain Differential Technique :PDDT1). Its performance was compared to an algorithm by Limb and Murphy [47]. A least square formulation (PDDT2) of the problem was done to improve the performance of the initial algorithm. Two more variations (PDDT3 and PDDT4) were developed and their performance tested using the variance of the frame difference as criterion.

PDDT3 assigns motion vectors to individual pixels, PDDT4 is a recursive least square formulation which operates on rows or columns of pixels recursively.

Lastly the algorithms were used in motion compensated image processing applications (motion compensated frame interpolation, motion compensated temporal filtering and motion compensated DPCM coding). A segmentation scheme was developed to solve the problem of partitioning the image frame into covered and uncovered background, moving area and stationary background in the frame interpolation application. Results of computer simulations show that the algorithms are efficient for motion compensation.

DEDICATION

This piece of work is dedicated to those human beings who, in one way or another, have shaped my outlook on life, especially;

**Yetunde Ogunbona,**

**Olusiji Ogunbona,**

who provided the right environment for the development of my personality;

**Beatriz Torres,**

through whom I learnt that twosome could be a fulfilling way of life;

**The Nigerian Society,**

whose rich culture provided a basis for my philosophy about life.

ACKNOWLEDGEMENTS

The choice and supervision of this research work owe a great deal to my supervisor, Dr. R. A. King. His pastoral disposition was a source of inspiration during this work and the congenial environment created by his personality greatly enhanced the crystallization of my thoughts. The useful criticism from members of Dr. R. A. King's research group was very helpful.

Most of the work was done using the facilities of the Centre for Remote Sensing of Imperial College. I am very grateful to Dr. C. England, the Centre's system manager, for his help throughout my stay in the Centre. The help of other reseachers in the Centre is acknowledged.

This work was sponsored by the Federal Government of Nigeria through a postgraduate scholarship awarded to me. In this respect, the labour of the taxpayers of the Federal Republic of Nigeria is acknowledged.

Finally, I am indebted to all my colleagues at Imperial College, who in one way or another contributed to the successful completion of this research work.
Muchisimas Gracias !

CONTENTS


| | | Page |
|---|---|---|
| ABSTRACT | | 2 |
| DEDICATION | | 4 |
| ACKNOWLEDGEMENTS | | 5 |
| CONTENTS | | 6 |
| LIST OF FIGURES | | 8 |
| LIST OF TABLES | | 10 |
| PROLOGUE | | 12 |
| CHAPTER 1 | INTRODUCTION AND SURVEY | 14 |
| 1.1 | Introduction | 14 |
| 1.2 | A Survey of Motion Estimation Techniques | 24 |
| 1.2.2 | 3-Dimensional Motion Estimation | 26 |
| 1.2.3 | 2-Dimensional Motion Estimation | 32 |
| CHAPTER 2 | A FEATURE BASED TECHNIQUE USING MOMENT INVARIANTS | 45 |
| 2.1 | Introduction | 45 |
| 2.2 | Moments and Moment Invariants | 47 |
| 2.3 | Feature Space Vectors | 55 |
| 2.4 | Matching Scheme and Motion Estimation | 58 |
| 2.5 | Smiulations and Results | 64 |
| 2.6 | Discussion of Results and Conclusion | 75 |
| CHAPTER 3 | DIFFERENTIAL TECHNIQUE IN THE PROJECTION DOMAIN | 78 |
| 3.1 | Introduction | 78 |
| 3.2 | The Projection Approach | 79 |
| 3.2.2 | Implementation Considerations | 85 |
| 3.2.3 | Experiments, Results and Conclusion | 88 |
| 3.3 | Least Square Estimation in the Projection Domain | 94 |
| 3.3.2 | Experiments, Results and Conclusion | 99 |
| 3.4 | Algorithm PDDT3 and Results of | |

| | | | |
|---|---|---|---|
| | | Experiments | 104 |
| | 3.5 | Recursive Projection Domain Estimation | 111 |
| | 3.5.2 | Experiments, Results and Conclusion | 116 |
| CHAPTER 4 | | APPLICATIONS OF MOTION ESTIMATION | 121 |
| | 4.1 | Introduction | 121 |
| | 4.2 | Motion Compensated Frame Interpolation | 121 |
| | 4.2.1 | Introduction | 121 |
| | 4.2.2 | Segmentation and Motion Estimation | 125 |
| | 4.2.3 | Interpolation Filter | 129 |
| | 4.2.4 | Experiments, Results and Conclusion | 132 |
| | 4.3 | Motion Compensated Temporal Filtering | 140 |
| | 4.3.1 | Introduction | 140 |
| | 4.3.2 | Temporal Filtering | 141 |
| | 4.3.3 | Experiments, Results and Conclusion | 144 |
| | 4.4 | Motion Compensated Interframe DPCM Coding | 148 |
| | 4.4.1 | Introduction | 148 |
| | 4.4.2 | Interframe Predictive Coding | 150 |
| | 4.4.3 | The Predictor | 152 |
| | 4.4.4 | The Quantizer | 154 |
| | 4.4.5 | Experiments, Results and Conclusion | 156 |
| CHAPTER 5 | | CONCLUSION | 177 |
| APPENDIX A-1 | | MOMENT INVARIANTS | 180 |
| APPENDIX A-2 | | LISTING OF COMPUTER PROGRAMS DEVELOPED FOR SIMULATION OF ALGORITHMS | 185 |
| REFERENCES | | | 236 |

LIST OF FIGURES


PAGE

1.1. Sequence "Walking man". 19
1.2. Sequence "Alan". 21
1.3. Sequence "Box". 23
1.4. Basic geometry for 3-D motion estimation. 26
2.1. Template matching. 61
2.2. Flowchart of two-stage, feature based matching scheme for motion estimation. 63
3.1. Illustrative pixel arrangement. 86
4.1. Block diagram of a motion-compensated frame interpolator. 122
4.2. Motion-adaptive interpolation of Haskell et al. 124
4.3. Areas for segmentation in frame interpolation. 127
4.4. Result of motion-compensated frame interpolation. 134
4.5. Result of uncompensated frame interpolation. 135
4.6. Result of motion-compensated frame interpolation 136
4.7. Result of uncompensated frame interpolation. 137
4.8. Result of motion compensated frame interpolation. 138
4.9. Result of uncompensated frame interpolation. 139
4.10. Differential pulse code modulation (DPCM) codec system. 151
4.11. Motion-compensated (DPCM) codec system. 154
4.12. Quantizer characteristics. 155
4.13. DPCM mean square error, using feature based technique on "Box". 159
4.14. DPCM mean square error, using feature based technique on "Alan". 160
4.15. DPCM mean square error, using feature based technique on "Walking man". 161
4.16. DPCM signal to noise ratio (SNR), using feature based technique on "Box". 162
4.17. DPCM SNR, using feature based

technique on "Alan". 163
4.18. DPCM SNR, using feature based technique on "Walking man". 164
4.19. DPCM prediction gain using feature based technique on "Box". 165
4.20. DPCM prediction gain using feature based technique on "Alan". 166
4.21. DPCM prediction gain using feature based technique on "Walking man". 167
4.22. DPCM mean square error, using differential technique on "Box". 168
4.23. DPCM mean square error, using differential technique on "Alan". 169
4.24. DPCM mean square error, using differential technique on "Walking man". 170
4.25. DPCM signal to noise ratio (SNR),using differential technique on "Box". 171
4.26. DPCM SNR, using differential technique on "Alan". 172
4.27. DPCM SNR, using differential technique on "Walking man". 173
4.28. DPCM prediction gain using differential technique on "Box". 174
4.29. DPCM prediction gain using differential technique on "Alan". 175
4.30. DPCM prediction gain using differential technique on "Walking man". 176

LIST OF TABLES

PAGE

T2.1. Variance of frame difference, using vector FV3 on "Walking man". 66
T2.2. Variance of frame difference, using vector FV3 on "Box". 67
T2.3. Variance of frame difference, using vector FV3 on "Alan". 68
T2.4. Comparison of frame difference variance using vectors FV1, FV2, FV3 and FV4. 69
T2.5. Prediction gain variation with varying block size. 70
T2.6. Variance of frame difference, using vector FV7 on "Walking man". 71
T2.7. Variance of frame difference, using vector FV7 on "Box". 72
T2.8. Comparison of frame difference variance using vectors FV5, FV6 and FV7. 73
T2.9. Prediction gain variation with varying block size. 74
T2.10. Comparison between prediction gains obtained using ordinary and radial/angular moment invariants. 75
T3.1. Frame difference variance, using algorithm PDDT1 on "Box" for varying displacements. 89
T3.2. Frame difference variance, using algorithm LM on "Box" for varying displacements. 90
T3.3. Comparison of frame difference variance, using PDDT1 and LM on "Walking man". 92
T3.4. Comparison of frame difference variance, using PDDT1 and LM on "Box". 93
T3.5. Comparison of frame difference variance, using PDDT1 and LM on "Alan". 94
T3.6. Frame difference variance, using algorithm PDDT2 on "Box" for varying displacements. 101

T3.7. Comparison of frame difference variance, using PDDT1 and PDDT2 on "Walking man". 102
T3.8. Comparison of frame difference variance, using PDDT1 and PDDT2 on "Box". 103
T3.9. Comparison of frame difference variance, using PDDT1 and PDDT2 on "Alan". 104
T3.10. Frame difference variance, using algorithm PDDT3 on "Box" for varying displacements. 106
T3.11. Frame difference variance, using algorithm PDDT4 on "Box" for varying displacements. 117
T4.1. Comparison of signal to noise ratio of temporally filtered images, using mean filter. 145
T4.2. Comparison of signal to noise ratio of temporally filtered images, using median filter. 146
T4.3. Quantizer for DPCM coding experiment.
T5.1. Comparison of prediction gains, using feature based and differential techniques. 179

# PROLOGUE

The sensation of seeing, in itself, is boring. It is the information which it brings with it that makes it interesting. The information conveyed also depends on the interpretation given to it. However there cannot be any new information if there is no change in what we are seeing. The origin of the change being perceived could be from two sources. On the one hand the observer could be moving and thus gaining information about the various stationary objects in a scene. On the other hand the objects in the scene could be moving and the observer is stationary. Whatever the origin of the change, it is clear that it is relative motion that brings it about.

In a very general sense, one can say that motion gives a lot of information about a scene. A means of measuring motion is thus a means of acquiring information. The usage to which the information acquired is put will depend largely on the application at hand. It is still fundamental to address the question of how humans perceive motion, or how any being that perceives motion does it. This may help in building vision models for image processing systems.

The pixel points that make up a digital image in themselves cannot convey any information except when taken in the context of a neighbourhood of pixels. A neighbourhood of pixels will form a hyperplane which will constitute the texture of some parts of an object. It may be doubted if, the object having moved, one can ascribe motion to an individual pixel. It appeals more to intuition that a group of pixels will move together. Humans, for example, still recognise an object after some motion, not because the pixels are individually recognisable but because features of the object

are recognisable over a time interval. This will explain why it is possible to recognise objects even when lighting conditions are changing from frame to frame. When the features already known disappear or new one suddenly appear it is still the context of other features that allow recognition. In order words, pixels are recognised within the context of features and features are in their turn recognised within the context of other features.

It is thus a natural course, to seek to solve the problem of motion estimation in an image sequence from the point of view of feature matching. Even when features are not being matched directly, efficient techniques must use a group of pixels. A technique that seeks to ascribe motion to individual pixels without a context can only be fraught with errors. The proper method of measuring motion will in the end be based upon reliable methods of describing object features uniquely and using the description so acquired in tracking objects. The various applications to which the motion vectors acquired can be put are numerous. It is interesting that all the methods which exist to date on motion estimation can be resolved, in some ways, into a means of measuring correlation between the frames in an image sequence.

# CHAPTER 1

# INTRODUCTION AND SURVEY

## 1.1 INTRODUCTION

There are a lot of application areas that depend for their efficient implementation, on the knowledge about motion taking place in a recorded scene. This requirement has given rise to research into techniques of estimating motion from a sequence of images. These application areas include among many others, robotics, navigation, i mage sequence coding, temporal filtering and interpolation of image sequences and cloud tracking. While the general problem is that of motion estimation, it is possible to identify two distinct needs.

In one the 3-dimensional translational and possibly rotational motion of an object in a real world space is required. This may be necessary to analyse the contents of a real scene, as in reconnaisance images. In robotics and navigation the need is also similar to this. Various attempts at solving this problem can be found in the literature [see for example 4, 9, 23, 77, 78, 79, 86].

The other need is the estimation of 2-dimensional translational motion in the image space. This is often needed in motion adaptive processing of image sequences. It is known that processing of image sequences containing moving objects is best done along the trajectory of motion of the objects. One application area that has received much attention with respect to motion estimation is image sequence coding [6, 11, 12, 13, 14, 16, 18, 21, 25, 41, 55, 59, 71, 73].

The work reported in this thesis is concerned with the latter kind of motion estimation. The various

mathematical formulations for the estimation of 2-dimensional motion from image sequences can be classified into three, namely [38];

(1) Fourier methods.
(2) Matching methods.
(3) Differential methods.

The methods will be described briefly as follows (more detailed discussion will appear in the survey):

(1) The Fourier method essentially takes the image plane as a two dimensional function undergoing some shift. Every pixel that makes up the image is assumed to be appearing in all the image frames that make up the sequence, but with a shift. Given two such image frames that are displaced with respect to each other, a Fourier transform of the images will yield similar magnitude values but different phase values. The displacement can be calculated from the phase terms. Techniques based on this principle have been reported with good performance [29,68].

(2) The matching method is in general a pattern matching process. Given two image frames from a sequence, a similarity function is set up, which is then used to find areas of best match between the two frames. The most commonly used similarity function is the cross correlation between segments of the pair of image frames. A search is carried out over positions in a search area, each time computing the cross correlation function, and the position giving the maximum correlation yields the displacement of the segments. Techniques based upon this approach have been employed in [39, 73, 83]. The feature based approach can be thought of as belonging to this class, however with the difference that the pixels in the image segments are first transformed into a feature space using some feature descriptors or extractors. The matching is then done in the feature space.

(3) Whenever there is interframe motion in an image sequence, there is generated a frame difference signal. In other words, any two consecutive image frames in a sequence will look alike but for the changes arising from motion. Thus when the difference between the two images is taken a frame difference signal approximately proportional to the relative motion between the two frames is obtained. The differential technique essentially relates the frame difference signal and the spatial gradients derived from an image pair to the motion between them. Since the initial formulation of Limb and Murphy [47] several variations have appeared which in some ways bear relationship to the formulation of [47]. Bergmann [6] subdivided the variations into three groups. Those in which;

"1. one displacement estimate is calculated for the entire moving area.
2. the displacement estimates are calculated within the moving area for a small rectangular image segment.
3. the displacement estimates are calculated recursively pel by pel"

The development of these variations are, to a large extent, well established, especially the types 2 and 3. Type 1 requires segmentation of the image frame into the moving objects and the stationary background. This in itself, called image analysis, is another research area requiring motion estimation.

The work reported herein can be divided into two, namely (1) the development of novel motion estimation algorithms, albeit classifiable into the traditional groupings of matching and differential techniques respectively and (2) the use of the developed algorithms in applications requiring motion compensation. Two major algorithms with several variations are developed. A feature

based matching algorithm using momemt invariants as feature descriptors is developed in chapter 2. In order to segment the image frames into moving and stationary areas, a simple segmentation scheme that uses the r.m.s value of the frame difference between two image frames was developed. For each moving area segment a feature vector based upon moment invariants was derived. This vector is used in the matching scheme. A two-stage matching scheme was developed to speed up the search, using the same segmentation threshold in selecting areas worthy of search. Varying feature vector sizes and segment sizes are experimented with. Radial and angular moment invariant feature vectors are also investigated.

In chapter 3, the concept of the projection domain differential technique of motion estimation is developed. Projections onto the x and y axes are used to resolve the motion into the x and y directions. Equations for estimating the x and y displacements are derived. Improvement to the basic algorithm is provided by reformulating the problem as a least square estimation problem. A recursive least square estimation is implemented to allow for misclassification of rows or columns of pixels as moving area pixels. In another variation, instead of assigning motion to a group of pixels, the centre pixel in a block is assigned the block motion estimate.

The developed techniques are applied to some image processing schemes. In chapter 4, motion compensated frame interpolation, temporal filtering and differential pulse code modulation (DPCM) coding are implemented using the motion estimation techniques developed. In the final chapter a comparison of the two techniques developed and conclusion are presented. In figures 1.1, 1.2 and 1.3 we present the series of images used as test sequence in the experiments. In figure 1.1 is shown the sequence "Walking man". Figure 1.2 shows the

sequence "Alan" and figure 1.3 shows some frames from the sequence "Box".



FRAME 1



FRAME 2



FRAME 3



FRAME 4



FRAME 5



FRAME 6



FRAME 7



FRAME 8



FRAME 9



FRAME 10



FRAME 11

Figure 1.1: Sequence "Walking man".
Frames 1 through 11.



FRAME 1



FRAME 2



FRAME 3



FRAME 4



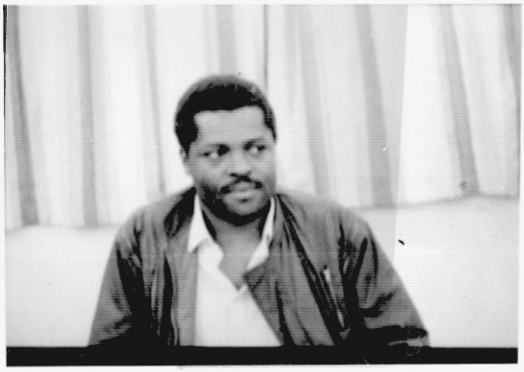
FRAME 5



FRAME 6



FRAME 7



FRAME 8



FRAME 9

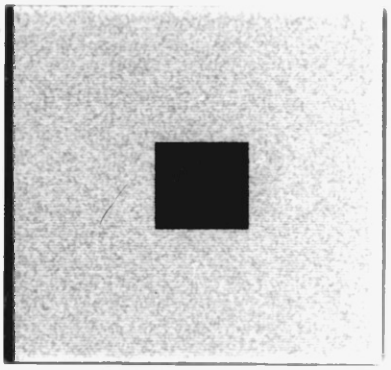


FRAME 10

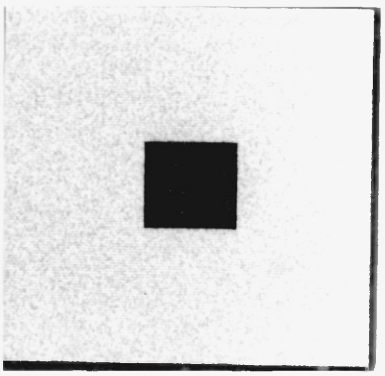


FRAME 11

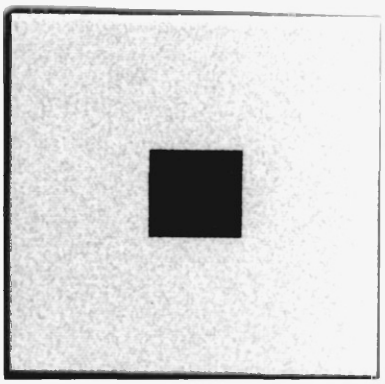
Figure 1.2: Sequence "Alan".
Frames 1 through 11.



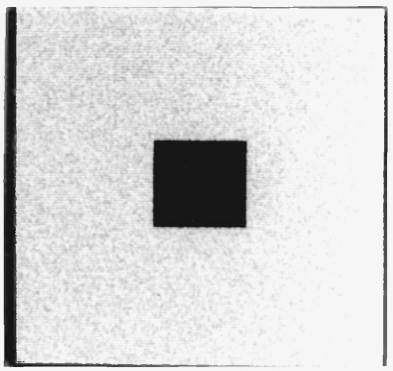
FRAME 1



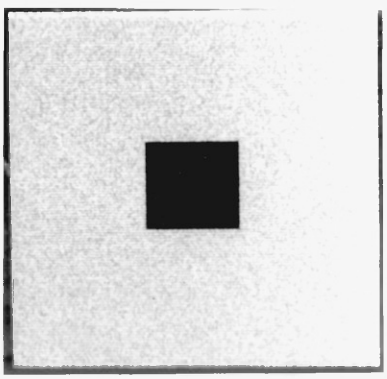
FRAME 2



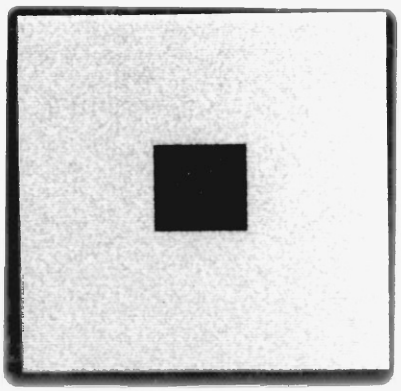
FRAME 3



FRAME 4



FRAME 5



FRAME 6

Figure 1.3: Sequence "Box"; Displacements of (1,1) through (5,5) from frame 1.

## 1.2 A SURVEY OF MOTION ESTIMATION TECHNIQUES

Owing to advancements in semiconductor and computer technologies, large memory sizes are now available at affordable prices and computational power has increased considerably over the past decade. The thrust in semiconductor technology has brought into the realm of real-time processing, operations which hitherto would have taken hours of processing time. The analysis of image sequences is one of the application areas that has benefited from the recent advancements in semiconductor technology.

The effectiveness of visual displays in conveying information to the human user and the use to which computer vision can be put, have given a fillip to research into image sequence analysis [2, 9, 54, 56, 57, 72] . Image sequences are derivable in a number of ways, including sequences of images that arise from recording different views of the same object and those that arise from recording a scene in which the constituent objects are undergoing motion. In general one type is space sequential while the other is time sequential. It is the image sequence of the time sequential type that is particularly interesting, the analysis of which has diverse applications. Huang and Tsai [38] enumerated some problem areas in which image sequence analysis has found application:

"1, Military problems- Tracking of multitargets from video data. Measuring missile dynamics from video data. Target detection and recognition in forward looking infrared image sequences.
2, Industrial problems- Dynamic monitoring of industrial processes. Dynamic robot vision.
3, Commercial problems- Bandwidth compression of TV conference and picture phone video signals.

4, Medical problems- Study of cell motion by microcinematography. Study of heart motion from x-ray movies.

5, Meteorology- Cloud tracking.

6, Transportation- Highway traffic monitoring".

The various techniques that have been developed to process/analyse image sequences are largely problem dependent. However, fundamental to these processes is the estimation of motion taking place over the image sequence. Furthermore it is possible to differentiate between two kinds of motion which have attracted attention. Thus we have motion estimation techniques which seek to measure 3-dimensional translation and rotation of moving objects. Coupled with the problem of 3-dimensional motion estimation is the determination of 3-dimensional stucture of the objects [86]. In these techniques the object plane and the image plane are mapped in order to obtain real world motion estimates of the moving objects. Examples of areas of application in which these techniques are used include military, industrial and transportation. The other group of techniques are geared towards the estimation of 2-dimensional motion of objects in the image plane. The point about image planes has been stressed because in most of the applications in which these techniques are used (medical and bandwidth compression application) the real world motion parameters of the objects are not of particular interest. However it should be noted that the 3-dimensional motion estimation techniques are in most cases a generalization of the 2-dimensional techniques and methods of formulation of 3-dimensional techniques can be carried over with some approximations into 2-dimensional cases. Some techniques that have been developed will now be given. Since the work reported in this research is on 2-dimensional motion estimation techniques we briefly review the 3-dimensional case before dwelling on the 2-dimensional case.

1.2.2 Estimation of 3-dimensional motion.

As aforementioned, the problem of estimation of 3-dimensional motion is that of obtaining the real world (object space) motion parameters of moving objects from the motion recorded and observable in the image space of the image sequence. Hence the image space and the object space have to be mapped to obtain correspondence between points in the two spaces. A basic geometry for 3-dimensional motion estimation is given in figure 1.4

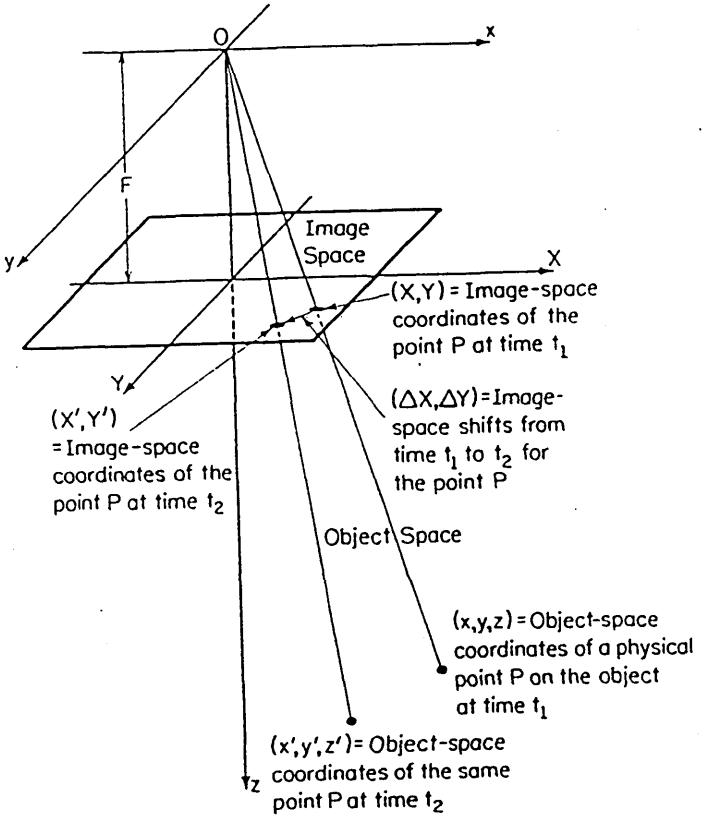



Figure 1.4: Basic geometry for 3-dimensional motion estimation.
(After Huang and Tsai [38]).

The object-space coordinates are denoted by (x,y,z) and the

image-space (assuming perspective view) coordinates by (X,Y). It is assumed that we have two time-sequential perspective views (i.e two frames) taken at times $t_1$ and $t_2$, of a scene containing moving objects. It is also important to include the assumption of rigidity of the moving objects. For a point P lying on the surface of the rigid body we have the following;

$(x,y,z)$= object-space coordinates of P at time $t_1$.
$(x',y',z')$= object-space coordinates of P at time $t_2$.
$(X,Y)$= image-space coordinates of P at time $t_1$.
$(X',Y')$= image-space coordinates of P at time $t_2$.

The image-space shifts or translations are defined thus;

$DX= X'- X$
$DY= Y'- Y$

for X and Y coordinates respectively. The relationship between the object and image spaces are given, from the geometry of figure 1.4, as;

$$X= Fx/z$$
$$Y= Fy/z \qquad \ldots.(1.1)$$

F can be thought of as the focal length of the imaging device.

Huang and Tsai [38] developed a two-step method of estimating 3-dimensional motion parameters of a rigid body. In the first step the image-space shifts DX and DY are obtained using any of the methods of 2-dimensional motion estimation. Then the motion parameters of the rigid body are determined from these image-space shifts. Assuming that the rigid body has translated by $(\Delta x,\Delta y,\Delta z)$ in the object space over a time interval $t_2- t_1$, then;

$$x' = x + \Delta x$$
$$y' = y + \Delta y$$
$$z' = z + \Delta z$$

....(1.2)

Equations (1.1) and (1.2) together will yield;

$$X' = \frac{F(x+\Delta x)}{z+\Delta z} = \frac{Xz + F\Delta x}{z+\Delta z}$$

$$Y' = \frac{F(y+\Delta y)}{z+\Delta z} = \frac{Yz + F\Delta y}{z+\Delta z}$$

and

$$DX = X' - X = \frac{F\Delta x - X\Delta z}{z+\Delta z} = \frac{\Delta x' - X}{\bar{z}+1}$$

$$DY = Y' - Y = \frac{F\Delta y - Y\Delta z}{z+\Delta z} = \frac{\Delta y' - Y}{\bar{z}+1}$$

....(1.3)

where $\Delta x' \triangleq F\Delta x/\Delta z$ ; $\Delta y' \triangleq F\Delta y/\Delta z$ ; $\bar{z} \triangleq z/\Delta z$.

Equations (1.3) are the basic equations relating the object-space displacement to the image-space translations. Defining r as;

$$r \triangleq DY/DX = (\Delta y' - Y)/(\Delta x' - X)$$

....(1.4)

a means of solving for the displacement $\Delta x$ and $\Delta y$ is provided. The image-space shifts at two points corresponding

to object-space points on the same rigid body are estimated and used to obtain two equations in two unknowns $\Delta x'$ and $\Delta y'$. These can be solved for $\Delta x'$ and $\Delta y'$. The range of the points $\bar{z}$ can be obtained from equations (1.3).

In a generalization to incorporate rotation, a similar line of thought was pursued and equations similar to (1.3) were obtained. Thus in general, assuming the rotation suffered by the rigid body is slight, we have;

$$DX = \frac{-[FY\phi_3 - (F^2 + X^2)\phi_2 + XY\phi_1]\bar{z} + F\Delta x' - FX}{-(\phi_2 X - \phi_1 Y)\bar{z} + F\bar{z} + F}$$

$$DY = \frac{-[FX\phi_3 - (F^2 + Y^2)\phi_1 + XY\phi_2]\bar{z} + F\Delta y' - FY}{-(\phi_2 X - \phi_1 Y)\bar{z} + F\bar{z} + F}$$

....(1.5)

where $\phi_1$, $\phi_2$, $\phi_3$ are the rotations about the three coordinate axes x, y, z respectively and the other variables are as defined before.

Given the displacement at an image-space point, we obtain (from equation (1.5)) two nonlinear equations in six unknowns: $\Delta x'$, $\Delta y'$, $\bar{z}$ , $\phi_1$, $\phi_2$, $\phi_3$. For every new image-space point we generate two equations and a new unknown, the range $\bar{z}$ for the point. Hence 5 image-space points will be needed to generate 10 equations with 10 unknowns. More points could be used thus generating more equations than unknowns and the least square technique employed in solving for the unknowns.

A direct method based upon the relationship between temporal and spatial differentials of image intensity was also developed in the same paper [38]. This technique requires knowledge about the geometry of the object surface.

However by assuming that all the image-space points lie on a plane describable by the equation;

$$ax + by + cz = 1 \qquad \ldots.(1.6)$$

and using this with equation (1.1) yields;

$$\tilde{z} = 1/(a'X + b'Y + c') \qquad \ldots.(1.7)$$

where,

$a' = a\Delta z$
$b' = b\Delta z$
$c' = c\Delta z$

This method yielded a nonlinear equation containing 8 unknowns; $\phi_1$, $\phi_2$, $\phi_3$, $\Delta x'$, $\Delta y'$, , $a'$, $b'$, $c'$. Hence using estimates at 8 image-space points, at least, will yield a solution.

In another contribution, Tsai and Huang [77] defined a set of 8 "pure parameters" which are uniquely determined for any given mapping from the two-space (X,Y) (coordinate space of the image frame at time $t_1$) onto the two-space (X',Y') (coordinate space of the image frame at time $t_2$). These 8 parameters are obtained by solving a set of linear equations, and are used in computing the real motion parameters, via the solution of a sixth order polynomial in one variable. They also addressed the problem of uniqueness of solution of the nonlinear equations of [38]. For the method discussed in [77] it was shown that but for a scale factor in the translation parameters, the number of real solutions never exceeded two.

The foregoing discussion are with the assumption

that the rigid body is planar. Consideration about rigid objects with curved surfaces was given in [79]. Using the "pure parameters" defined in [77], a 3x3 matrix whose singular value decomposition (SVD) gave the motion parameters was formed. The motion parameters are unique up to a scale factor for the translations. It was also established that seven point correspondences are sufficient to determine uniquely, the 3-dimensional motion parameters from two perspective views of a rigid object with curved surfaces. Roach and Aggarwal [69] have approached the problem differently. Assuming that the correspondence of points in the image-space is known a priori, they formulated a model of the problem using a 4x4 matrix in homogeneous coordinates. Based on a central projection assumption, they derived from 5 observed points in two views, a system of 18 nonlinear, transcendental equations in 18 unknowns.

Given that edge correspondences are easier to find than point correspondences, Tsai [78] offered a solution to the problem of point correspondence by using conic arc segments. The same 8 "pure parameters" of [77] were used in forming a 3x3 matrix whose singular value decomposition yielded the 3-dimensional parameters. The validity of simultaneously representing a pair of corresponding conic arc segments using rational quadratic parametrization was proved, and thus the uniqueness of the solution for 3-dimensional motion parameters and object surface orientation established. Attempts were made to use straight line correspondence in [86].

In the algorithms that involve the solution of nonlinear equations, the question of uniqueness of solution is a problem. The minimum number of point correspondences that will ensure unique solution has to be ascertained and even with this a good initial guess is essential for the iterative solution to converge quickly. The use of the 8

"pure parameters" and singular value decomposition method offers a solution to this problem. In the next section we discuss the techniques of 2-dimensional motion estimation.

### 1.2.3 Estimation of 2-dimensional motion.

From the discussions of sections 1.1 and 1.2.2 it will be apparent that the problem of 2-dimensional and 3-dimensional motion estimation are inter-related. The problem of finding point correspondence in the image-space, which constitute the first stage in the estimation of 3-dimensional motion parameters is in fact an image-space 2-dimensional motion estimation problem.

The techniques of estimating 2-dimensional translation can be classified into three, namely; (1) Fourier methods, (2) Matching methods and (3) Differential methods. These methods in one variation or another can be used in establishing the point correspondence needed in 3-dimensional motion estimation. We shall now discuss these methods in turn.

**1. Fourier method.**

This method derives from the translation property of the Fourier transformation. Thus if two functions f(x,y) with a Fourier pair F(u,v) and g(x,y) with Fourier pair G(u,v) are related in the following manner;

$$g(x,y) = f(x-DX,\ y-DY) \qquad \ldots.(1.8)$$

where DX and DY are spatial displacements along the x and y spatial coordinates and u, v are the spatial frequencies, then,

$$G(u,v) = F(u,v)\ \exp[-j\,2\pi(uDX+vDY)] \qquad \ldots.(1.9)$$

Thus if f(x,y) and g(x,y) are two consecutive frames from an image sequence, it is possible to relate their relative displacement to the phase angle of their Fourier transforms. In a technique used in [38], the difference of the phase angles was used in computing the displacements.

The difference of the phase angles is given as,

$$\Delta\phi = \angle G(u,v) - \angle F(u,v) = -2\pi(uDX + vDY) \qquad \ldots.(1.10)$$

Computing (1.10) at 2 frequency pairs will yield 2 equations in 2 unknowns, the solution of which is a simple matter. An alternative is to use projections of the functions f(x,y) and g(x,y), and we only need do computation at one frequency. However owing to noise the problem will not be solved as easily as computing at one or two points, we shall need to compute the phase angle at more than two points. Least square method can then be used to solve for the displacements. Two remarks were made about the use of the Fourier method.

In the first place, since phase angles are not single valued, care must be taken when using them in (1.10). Using principal values may not result in a correct $\Delta\phi$. A means of ensuring the uniqueness of $\Delta\phi$ must be sought [38]. Secondly, the Fourier method as described above works only if we have an isolated moving object with a uniform background. If the background is not uniform, the movement of the object will reveal new background pattern and g(x,y) will no longer be a translated version of f(x,y); the method then breaks down.

In a development by Riddle and Rajala [68], the

Fourier transform was used in conjunction with a cosine-area transform (defined by the author in [68]) to estimate the velocity of moving objects. The method also allows object recognition.

An area transform was defined, which involves convolving an image function with a weighting function. When the weighting function is a cosine function, a cosine-area transform results. A one-dimensional situation will help in explaining the fundamentals. For a one-dimensional time sequence f(x,t) the cosine-area transform is defined as,

$$g(t) = \int_{O}^{A} f(x,t)\,\mathrm{Cos}(2\pi k x)/A \; dx \qquad \ldots.(1.11)$$

x and t are spatial and time variables respectively. k is the order of the weighting function and A, the spatial extent of the function. It can be appreciated how stationary (with respect to time) areas will generate a constant value for g(t) and moving areas will generate a sinusoid.

On taking the Fourier transform of g(t), a moving area will generate frequency proportional to its velocity, while stationary area will generate zero frequency component. The frequencies generated by the moving areas can be obtained through peak detecting algorithms. They gave the two-dimensional case, for a time sequence f(x,y,t) as,

$$g_x(t) = \int_{O}^{B}\int_{O}^{A} f(x,y,t)\,\mathrm{Cos}(2\pi k x)/A \; dxdy$$

$$g_y(t) = \int_{O}^{B}\int_{O}^{A} f(x,y,t)\,\mathrm{Cos}(2\pi k y)/B \; dxdy$$

$$0<x<A \quad ; \quad 0<y<B \qquad \ldots.(1.12)$$

The velocity components can be extracted from the Fourier transforms of the cosine-area transforms.

$$G_X(u) = \int_{-\infty}^{+\infty} g_X(t)\ \exp(-j2\pi ut)dt$$

$$G_Y(v) = \int_{-\infty}^{+\infty} g_Y(t)\ \exp(-j2\pi vt)dt \qquad \ldots.(1.13)$$

A generalization of the area transform was given in a way that will allow for both the estimation and the identification of moving objects from an image sequence.

A motion detection transform [29] whose development was based on the generalized area transform of [68] was used in determining motion of objects from a sequence of images. In the technique described in [29], the image plane was partitioned into numerous blocks and motion detection applied to each block. In effect the motion of the object was derived by filtering the motion components of blocks.

We remark here that while the approach described in [38] may not have a general application it can be performed in real time, since only two frames from the image sequence are needed. However in the techniques of [68] and [29] several frames need to be analysed hence a large memory is required for frames storage ( in [68] they analysed 16 frames).

## 2. **Matching method.**

Generally, the matching methods set up a matching criterion which is a measure of similarity or dissimilarity between two given image frames, and a search is conducted in a parameter space for a point maximising (or minimising as the case may be) the matching criterion. Thus given two image frames $f_t(x,y)$ and $f_{t-1}(x,y)$ which are displaced relative to each other, that is,

$$f_t(x,y) = f_{t-1}(x-DX, y-DY) \qquad \ldots.(1.14)$$

where DX and DY are the displacements, and t represents time, we set up a matching criterion,

$$C(DX,DY) = I(f_t(x,y), f_{t-1}(x,y)) \qquad \ldots.(1.15)$$

where I is some function of the two image frames.

The most commonly used matching criterion is the normalized cross correlation function (NCCF), given as,

$$NCCF(i,j) = \frac{\sum_{m=1}^{M}\sum_{n=1}^{N} f_t(m,n) f_{t-1}(m+i,n+j)}{\left[\sum_{m=1}^{M}\sum_{n=1}^{N} f_t^2(m,n)\right]^{1/2} \left[\sum_{m=1}^{M}\sum_{n=1}^{N} f^2_{t-1}(m+i,n+j)\right]^{1/2}}$$

$$-md <= i,j <= +md \qquad \ldots.(1.16)$$

md is the maximum displacement in both directions. Usually to find the displacement of a point (x,y), a block of pixels, size MxN, centred at the point (x,y) is taken from the frame $f_t$ and correlated with pixels in a search area of frame $f_{t-1}$. This computation is done for all points (i,j) in the search area. The process is done for all points deemed to be moving. Since the maximum displacement expected in both directions (i.e x and y) is md, then the search area will be (M+2md)(N+2md) and the number of NCCF computations to be done for each moving area point is $(2md+1)^2$. The (i,j) point yielding the maximum NCCF value gives the displacement. However the peak may not be detectable in some cases owing to image noise. Furthermore the amount of computation that need to be done makes this technique unattractive. Various means have been suggested to improve the efficiency of the matching

technique based on the NCCF matching criterion. The discovery of the fast Fourier transform (FFT) led some researchers to use the FFT in computing the correlation function, for use in motion estimation work. Leese et al [44, 45] developed algorithms based on correlation matching for the estimation of cloud pattern motion. The cross correlation function were obtained using the FFT.

Other means of improvement are, the use of different matching criteria, the use of efficient search techniques and the assumption of block movement as opposed to point movement. The alternative matching criteria seek to reduce the computational burden incurred when using NCCF. The efficient search techniques are aimed at minimizing the amount of search points where computation of the matching criterion will be done. The block movement assumption operates by partitioning the image into small blocks of pixels which are undergoing similar motion.

Jain and Jain [39] proposed a mean square error, matching criterion, which does not involve as much computation as the NCCF. The criterion is given as

$$MSE(i,j)= 1/MN \sum_{m=1}^{M} \sum_{n=1}^{N} [f_t(m,n)-f_{t-1}(m+i,n+j)]^2 \quad ....(1.17)$$

Koga et al were reported [53] to have proposed a mean absolute frame difference matching criterion given as,

$$MAD(i,j)= 1/MN \sum_{m=1}^{M} \sum_{n=1}^{N} | f_t(m,n)- f_{t-1}(m+i,n+j)| \quad ....(1.18)$$

$$-md<=i,j<=+md$$

In the experiments reported by Srinivasan and Rao [73] the

two matching criteria of equations (1.17) and (1.18) were compared and their performance found to be very close. They adopted the criterion of equation (1.18) being computationally simpler than equation (1.17).

Attempts at finding efficient search methods have produced three significant methods, namely (1) 2-dimensional logarithmic search, (2) three-step search and (3) the modified conjugate direction search. The 2-dimensional logarithmic search proposed by Jain and Jain [39] is based upon the assumption that their matching criterion (see equation (1.17)) increases monotonically as the search moves away from the point of best match. The point of best match is defined by (i,j) such that MSE(i,j) is minimum. The best match point is found by successively reducing the area of search. At each step, 5 locations which contain the centre of the search area and four boundaries of the area along the axes passing through the centre are searched. The distance between the points is reduced if the minimum is found at the centre of the search points or on the boundary of the search area. The area of search is reduced until it is of side 3 pixels by 3 pixels, when all the nine points are searched and the minimum gives the desired displacement.

The three-step method of Koga et al reported in [53] is similar to the 2-dimensional search in that the area of search is progressively reduced. In the first step 8 points (excluding the centre point) are coarsely spaced around the centre and a first approximation to the displacement found using the matching criterion of (1.18). For the second step, the first approximation point becomes the centre point with 8 points less coarsely spaced around it. The procedure continues until the point which minimizes the criterion (1.18) is found. The modified conjugate direction search was proosed by Srinivasan and Rao [73]. In this technique two sets of searches are done. The first set, searches for the

point of best match in the x direction. Starting from this point the search for the best match point in the y direction is then obtained. This point which incorporates both best points gives an estimate of the displacement.

The block movement assumption has been implemented with success by some researchers [10, 39, 51, 73]. Some modification has appeared [83] to algorithm of [39]. Wells [83] used motion estimated for adjacent blocks to speed up the search algorithm. Some other application oriented block matching schemes are reported in [64] and [10].

Feature based matching schemes can be considered as members of the matching methods of motion estimation. Instead of using groups of pixels ( in their "raw" form) from the image frames for matching, features like edges, corners, arc segments, etc. are derived from the image for subsequent matching. The pixel-space so to speak is first transformed into a feature-space before matching is done. This possibility is now being investigated by reseachers and promises to be fruitful [67].

3. **Differential method.**

In an image sequence, a moving object generates interframe difference signals, which bear some relationship to the movement causing it. Limb and Murphy [47] derived a motion estimation algorithm that depends on this principle. In fact all the differential techniques of motion estimation are based upon the same principle. In the paper of Bergmann [6] the differential based algorithms were grouped into three, namely,

" 1, One displacement estimate is calculated for the entire moving area.

2, The displacement estimates are calculated within the moving area for small rectangular image

segments.

3, The displacement estimates are calculated recursively"

In this grouping, it is possible to identify the recursive and the nonrecursive algorithms. The algorithms proposed by Limb and Murphy [47] and Cafforio and Rocca [11] are nonrecursive and of the type 1 and 2 in the above grouping. A recursive technique was first proposed by Netravali and Robbins [59] and there have appeared various other approaches to the recursive method in the literature [6, 16, 34, 40, 53, 81].

The algorithm of Limb and Murphy [47] computes the displacement from a ratio of the frame difference signal and the element difference (i.e intraframe horizontal difference between pixel values) signal within a moving area. In order to specify the direction of motion, the frame differences were accumulated according to the sign of the frame difference and element difference signals. Thus if FDS denotes the frame difference and EDS, the element difference, the accumulated frame differences were given as,

$$FX1 = \sum_{MA} |FDS|\, sb(FDS) * sb(EDS)$$

$$FX2 = \sum_{MA} |FDS|\, \overline{sb(FDS) * sb(EDS)} \qquad \ldots.(1.19)$$

where sb(.) is a binary signal with value 1 if the argument is positive and 0 if it is negative. * denotes the exclusive OR operation and the bar denotes the logical complement operation. The horizontal component of motion would be given as,

$$DX = (FX1 - FX2) / \sum_{MA} |EDS| \qquad \ldots.(1.20)$$

The subscript MA, indicates that the summation is to be taken over the moving area pixels. The advantage of this derivation is that it lends itself to easy hardware implementation. Experimental results given by the authors indicate good performance up to displacements of 2.5 pixels/frame.

The derivation of Cafforio and Rocca [11] is similar to that of [47], however they started by expanding a displaced two-dimensional function in a Taylor's series. Given two image functions $f_t(x,y)$ and $f_{t-1}(x,y)$ which are acquired at different times t and t-1. If the displacements between the two images over a time interval dt are DX and DY in the x and y directions respectively then,

$$f_t(x,y) = f_{t-1}(x-Dx, y-DY) \qquad \ldots.(1.21)$$

Equation (1.21) can be expanded in a Taylor's series as follows,

$$f_t(x,y) = f_{t-1}(x,y) - DX\frac{\partial}{\partial x}f_{t-1}(x,y) - DY\frac{\partial}{\partial y}f_{t-1}(x,y) - e(x,y) \qquad \ldots.(1.22)$$

where e(x,y) represents the higher terms of the expansion, which will be neglected. If we take the case of a one dimensional horizontal motion (i.e DY=0) then (1.22) can be written as,

$$f_t(x,y) - f_{t-1}(x,y) = -DX\frac{\partial}{\partial x}f_{t-1}(x,y)$$

whereupon DX, the horizontal motion is given as

$$DX = (f_{t-1}(x,y) - f_t(x,y)) / \frac{\partial}{\partial x} f_{t-1}(x,y) \qquad \ldots.(1.23)$$

The numerator is the frame difference (FDS) at the point

$(x,y)$ and $\frac{\partial}{\partial x} f_{t-1}(x,y)$ is the element difference (EDS) at the same point. This equation when compared to equation (1.20) shows the similarity between the two techniques.

Cafforio and Rocca [11] used linear regression on equation (1.22) assuming that the error term $e(x,y)$ is not correlated with the signal and has an even probabilty density function, to obtain,

$$DX = -\left(\sum_{MA}(FDS.EDS)\right) / \sum_{MA}(EDS)^2$$

$$DY = -\left(\sum_{MA}(FDS.LDS)\right) / \sum_{MA}(LDS)^2 \qquad \ldots.(1.24)$$

where LDS is the line difference (i.e intraframe vertical difference between pixel values) signal. The summations are to be carried out over the moving area pixels. In [12] they investigated the effect of noise on the technique and concluded that the best way to offset the effect of noise is to estimate the noise variance and subtract it with proper weighting from the various pixel value functions. A hardware structure was also given for the estimator. In [13] they showed that a finite difference approximation of the image differential is mandatory for good performance. In a modification [14] they derived a stochastic model for the motion field in order to obtain a recursive algorithm. The component of the motion parallel to the local image gradient is updated on pixel by pixel basis. They reported experiments on coding which indicated good enough performance. In an analysis by Bergmann [6] the techniques of Cafforio and Rocca were investigated with respect to horizontal motion and good results were reported up to about 1.5 pixel/frame. Larger displacements could be estimated if a lowpass spatial filtering of the images is done as preprocessing.

The recursive algorithms assume that an initial

estimate of the motion vector $D_i$, can be used to produce an improved estimate $D_{i+1}$ according to the equation,

$$D_{i+1} = D_i + U_i \qquad \ldots.(1.25)$$

$D_i = [DX_i \; DY_i]^T$ is the ith displacement vector. T denotes transpose. $U_i$ is the update term of the ith iteration. The displaced frame difference (DFD) can be defined as,

$$DFD(x,y,D_i) = f_t(x,y) - f_{t-1}(x-DX_i, y-DY_i) \qquad \ldots.(1.26)$$

Using a gradient type of approach Netravali and Robbins [59] recursively minimized the squared value of the displaced frame difference (DFD) at each moving area pixel point. Thus,

$$D_{i+1} = D_i - 1/2k \; \nabla_{D_i} [DFD(x,y,D_i)]^2 \qquad \ldots.(1.27)$$

where $\nabla_{D_i}$ is the gradient operator with respect to $D_i$ and k is a positive constant. They developed the algorithm for use in a motion-compensated coding scheme, choosing k as 1/1024. The choice of k however, requires a compromise, a large value giving quick convergence at the expense of accuracy. In order to reduce the computational complexity of the scheme, they proposed simplified versions of the algorithm in [59] and [70].

The work of Netravali and Robbins has since stimulated other researchers and various recursive algorithms have appeared in the literature (see for example references in [53]). The improvements have been geared towards increasing the speed of convergence and thus the accuracy in the face of limited allowable number of iterations. In a review by Musmann et al. [53] it was shown that the various

recursive algorithms only differ in the denominator of the update term. They listed the x component of the displacement estimates derived by various authors. The effect of the update term denominator is found in the various rates of convergence. It was also demonstrated that minimization of the square of the displaced frame difference corresponds to maximization of the cross correlation function. An important feature of the recursive technique is the range of stability in which the algorithm converges to the actual displacement. The stability constraint is given by,

$$|D - D_{i+1}| < |D - D_i| \qquad \ldots.(1.28)$$

D is the actual displacement vector. The requirement is that the update term vector is always directed towards the actual displacement. The quantization noise in the image signal can seriously hamper the convergence rate of the algorithms, hence presmoothing has been proposed to enhance the performance.

# CHAPTER 2

## A FEATURE BASED TECHNIQUE USING MOMENT INVARIANTS.

### 2.1 INTRODUCTION.

In a dynamic scene, the objects are constantly changing their relative positions with respect to each other and possibly with respect to a fixed background. However there are parts of the object which though undergoing translation, will not change in shape. When a feature description exists that incorporates these parts, such feature descriptors can be used in tracking the object through the dynamic scene. With a rigid body (no shape change) assumption it is possible to track an occluding body provided there is no ambiguity about its distinguishing features. In the technique to be described, features are extracted from objects in a scene and used in estimating the motion of the objects. The motion thus estimated can be used in motion-adaptive image processing. An example of such motion-adaptive image processing is motion compensated interframe DPCM-coding. In this instance the image frame is segmented into stationary and moving areas and motion of the moving areas is computed. For the stationary areas it is sufficient only to repeat the already transmitted pixels. In the moving areas the differential (frame difference) signal to be transmitted is computed from motion-adaptively selected pixels.

The choice of feature descriptor for a particular application will depend on the features to be described. Image feature descriptors can be categorised either as regional or relational descriptors. Both types of descriptors attempt to give some characterisation to parts or whole of an object, in such a way that under linear transformation, like

translation, rotation and scaling, the object of interest is still recognisable. Examples of regional descriptors are Fourier, topological and moment invariant descriptors. A syntactic descriptor is a relational descriptor: this finds wide use in artificial intelligence.

A Fourier descriptor is used in describing the boundary of an object, given that it has been segmented and identified. Points on the boundary of the object are considered as points in a complex plane, representing the x-y coordinates of each point as complex numbers x + jy. The contour of the boundary will thus yield a sequence of complex numbers whose Fourier transform is referred to as the Fourier descriptor of the shape. A Fourier descriptor is also called a frequency domain descriptor, in the sense that image plane features are expressed in phase and magnitude form. It is also possible to have description in the image domain; topological descriptors are examples of image domain descriptors.

Given a figure with a number of holes (H) and a number of connected components (C), the Euler number (E), a topological property, can be defined as,

$$E = C - H$$

Figures represented by straight line segments (also referred to as polygonal networks) can also be described using the Euler number. The interior of such figures can be classified as a face or a hole. If W is the number of vertices, Q the number of edges and F the number of faces, we have a relationship, called the Euler formula [27],

$$W - Q + F = C - H$$

Thus the Euler number is written as,

$$E = C - H = W - Q + F$$

While the Fourier and topological descriptors can be said to be giving information about the external structural pattern of the object, moment invariants afford one the possibility of having information about the internal structure of the object. In other words the spatial arrangement of the pixel values which represent the object is described when moment invariants are used as descriptors. When used in a motion estimation scheme, this describes a pixel arrangement (the template) and one then searches within a given search region, for a pixel arrangement which most resembles the template. The technique to be described uses moment invariants as the components of a feature vector, which are used in the search. It is thus a variation of the matching technique of motion estimation.

## 2.2 MOMENTS AND MOMENT INVARIANTS.

A digital image is an array of pixel values which shall be called the density distribution function, f(x,y), of the image in digital form. Now, for a continuous density distribution function g(x,y) the two-dimensional moments of order (p,q), $m_{pq}$, are defined in terms of Riemann integrals as,

$$m_{pq} = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} x^p\, y^q\, g(x,y)\,dx\,dy$$

$$p,q = 0,1,2,\ldots \qquad \ldots.(2.1)$$

Assuming that g(x,y) is piecewise continuous, therefore a bounded function and that it can have non-zero values only in the finite part of the x-y plane [35], then moments of all orders exist. Furthermore the moment sequence $\{m_{pq}\}$ is

uniquely determined by g(x,y) and conversely g(x,y) is uniquely determined by the sequence $\{m_{pq}\}$.

For a digital image f(x,y) the moment of order (p,q) is defined as,

$$m_{pq} = \sum_{x=-\infty}^{+\infty} \sum_{y=-\infty}^{+\infty} f(x,y)\ x^p\ y^q$$

$$p,q = 0,1,2,\ldots$$

....(2.2)

The central moments referred to the image centroid are defined as,

$$u_{pq} = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} (x-\bar{x})^p\ (y-\bar{y})^q\ g(x,y)\,dx,dy$$

where $\bar{x} = m_{10}/m_{00}$ and $\bar{y} = m_{01}/m_{00}$

....(2.3)

For a digital image equation (2.3) becomes,

$$u_{pq} = \sum_{x=-\infty}^{+\infty} \sum_{y=-\infty}^{+\infty} (x-\bar{x})^p\ (y-\bar{y})^q\ \ f(x,y)$$

where $\bar{x} = m_{10}/m_{00}$ and $\bar{y} = m_{01}/m_{00}$

....(2.4)

The centralised moments are invariant under translation of objects within the image. It is possible to express the centralised moments in terms of the ordinary moments.

Expressions for central moments up to order 2 are listed below;

$$u_{00} = m_{00}$$
$$u_{10} = m_{01} = 0$$
$$u_{20} = m_{20} - u_{00}\,\bar{x}^2$$
$$u_{02} = m_{02} - u_{00}\,\bar{y}^2$$
$$u_{11} = m_{11} - u_{00}\,\bar{x}\bar{y}$$

....(2.5)

Drawing heavily from the algebra of quantics, Hu [35], derived the normalised form of the central moments. Thus the normalised central moments are defined as,

$$n_{pq} = u_{pq}/u_{00}^{r}$$

where $r=(p+q)/2 \quad p+q = 2,3,\ldots$

....(2.6)

From the second- and third-order normalised central moments Hu [35] derived a set of seven invariant moments. These are invariant to translation, size change, rotation and reflection. The seventh changes sign under reflection. They are listed below;

$$MI1 = n_{20} + n_{02}$$

$$MI2 = (n_{20} - n_{02})^2 + 4n_{11}^2$$

$$MI3 = (n_{30} - 3n_{12})^2 + (3n_{21} + n_{03})^2$$

$$MI4 = (n_{30} + n_{12})^2 + (n_{21} + n_{03})^2$$

$$MI5 = (n_{30} - 3n_{12})(n_{30} + n_{12})[\,(n_{30} + n_{12})^2 - 3(n_{21} + n_{03})^2\,] + (3n_{21} - n_{03})(n_{21} + n_{03})[\,3(n_{30} + n_{12})^2 - (n_{21} + n_{03})^2\,]$$

$$MI6 = (n_{20} - n_{02})[\,(n_{30} + n_{12})^2 - (n_{21} + n_{03})^2\,] + 4n_{11} \times (n_{30} + n_{12})(n_{21} + n_{03})$$

$$MI7= (3n_{12} - n_{30})(n_{30} + n_{12}) [ (n_{30} + n_{12})^2 - 3(n_{21} + n_{03})^2 ] + (3n_{21} - n_{03})(n_{21} + n_{03})[ 3(n_{30}+ n_{12})^2- (n_{21}+ n_{03})^2] \quad ....(2.7)$$

Attempts have been made by other reseachers to derive invariant moments from other polynomials, namely Zernike and Legendre polynomials [75]. Using complex moments, which are a generalization of the ordinary moments, Abu-Mostafa and Psaltis [1] investigated the notion of "information content" of moment invariants. Reddi [66] offered an alternative method of deriving the invariant moments of Hu, using radial and angular moments. His derivation, together with the discussions of [1] give an insight into the contribution of various parts of an image to the features extracted by invariant moments. An important point worth mentioning is the way in which moment invariants implicitly embody the phase information about images. For a one-dimensional function the relationship between the moments and the Fourier transform is given as [1],

$$F(w)= \sum_{n=o}^{+\infty} (-j)^n M_n w^n/n!$$

where $M_n$ is the nth ordinary moment and F(w) is the Fourier transform. Moment invariants being a combination of ordinary moments thus bear some relationship to the Fourier transform of the image: a phase and magnitude description. Every image pixel contributes to each moment in a way determined by the location and magnitude of the pixel in the image. Phase information pertains to the positioning of the pixel values that make up an image. Theoretical and possibly practical advantage of this kind of description is that in the event of occlusion, the most likely patch in the sense of minimum frame difference value will be used in making up the "mosaic" of the image.

Recall from equation (2.1) that in the continuous

form the moment of order (p,q) is given as,

$$m_{pq} = \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} g(x,y)\ x^p\ y^q\ dxdy$$

$$p,q = 0,1,2,...$$

A polar coordinate representation is given as,

$$M_{pq}(r,\theta) = \int_{-\pi}^{\pi}\int_{r=0}^{\infty} r^{p+q+1}\ \mathrm{Cos}^p\theta \mathrm{Sin}^q\theta\ L(r,\theta)drd\theta \qquad ....(2.8)$$

$$g(x,y) \equiv L(r,\theta)$$

where $x = r\mathrm{Cos}\theta$ $y = r\mathrm{Sin}\theta$ $dxdy = rdrd\theta$

For a given order of moment, (p+q+1) is a constant. The variables r and θ determine the fineness of polar sampling, but owing to the spatial correlation between adjacent pixel values in an image the angular variation θ will not produce much fluctuation in the pixel contributions. However the radial variation, together with the (p+q+1) index show the heavy dependence of the contribution of each pixel on its radial location from the centroid of the image.

Abu-Mostafa and Psaltis [1] have discussed the information properties of moments in terms of loss, suppression and redundancy.

The complex moment, $C_{pq}$, of order (p,q), which is related to ordinary moments is defined in [1], by,

$$C_{pq} = \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} (x+jy)^p(x-jy)^q\ g(x,y)\ dxdy \qquad ....(2.9)$$

for an image function g(x,y). The polar coordinates version is given by,

$$C_{pq} = \int_{0}^{2\pi}\int_{0}^{\infty} r^{p+q}\ L(r,\theta)\ \exp[j(p-q)\theta]rdrd\theta \qquad ....(2.10)$$

The circular harmonic expansion of the image function $L(r,\theta)$ is given by,

$$L(r,\theta) = \sum_{n=-\infty}^{+\infty} k_n(r) \exp(jn\theta) \qquad \ldots.(2.11)$$

with

$$k_n(r) = 1/2\pi \int_0^{2\pi} L(r,\theta) \exp(-jn\theta) d\theta \qquad \ldots.(2.12)$$

The coefficient $k_n(r)$ is a measure of angular variation of frequency n cycles/cycle. Using (2.10) and (2.12), an expression relating the complex moments and the circular harmonic expansion can be written. Thus,

$$C_{pq} = 2\pi \int_0^{\infty} r^{p+q+1} k_{q-p}(r) \, dr \qquad \ldots.(2.13)$$

The complex moment $C_{pq}$, thus depends only on one coefficient of the circular harmonic expansion of $L(r,\theta)$; $k_{q-p}(r)$. Thus if moments through an order N constitute a feature space, then they depend on harmonic expansion coefficients $k_{-N}(r)$ through $k_{+N}(r)$. Features of the image corresponding to angular variations of N+1 cycles/cycle or more will be lost [1]. This is not peculiar to complex moments alone, Zernike moments and ordinary moments suffer the same deficiency.

Let an image whose discriminating features lies close to the centroid (or central parts) be considered. The radial dependence of ordinary moments which lays more emphasis on pixel values farther away from the central parts will suppress the information derivable from these features. This will render the recognition properties of moment invariants almost useless. Since the radial suppression/emphasis is a property of ordinary moments whose

combinations constitute moment invariants, features derived from moments up to a given order will suppress or emphasize parts of the image in more or less the same way. This is a pointer to the possibility of redundant features being extracted. These points, regarding the information properties of moments invariants must be taken into consideration when choosing the size of the feature vector for description.

Radial and angular moments afford us the opportunity of investigating the radial dependence of moment invariants and the possibility of rectifying, as it were, the information suppression deficiency mentioned earlier. Let $L(r,\theta)$ be the pixel value at position $(r,\theta)$ in polar coordinates. Define [66] the radial and angular moments as follows,

$$\Psi_r(k,L)=\int_0^{\infty} r^k\, L(r,\theta)\, dr \qquad \ldots.(2.14)$$

and

$$\Psi_\theta(p,q,L)=\int_{-\pi}^{\pi} \mathrm{Cos}^p\theta \mathrm{Sin}^q\theta\, L(r,\theta)d\theta \qquad \ldots.(2.15)$$

with $\Psi_\theta(0,0,L)= \Psi_\theta(L)$

Equations (2.14) and (2.15) respectively express radial and angular moments. Combination of (2.14) and (2.15) will give,

$$\Psi(k,p,q,L)=\int_0^{\infty}\int_{-\pi}^{\pi} r^k\, L(r,\theta)\, \mathrm{Cos}^p\theta\, \mathrm{Cos}^q\theta\, drd\theta \qquad \ldots.(2.16)$$

Comparing equations (2.8) and (2.16) and assuming that $m_{pq}$ in (2.8) are centralised, then the ordinary moments can be expressed in terms of radial and angular moments as,

$$m_{pq}= \Psi(p+q+1,p,q,L) \qquad \ldots.(2.17)$$

The seven moment invariants of Hu are derivable in terms of radial and angular moments of equation (2.16). The first, second and third moment invariants are given below,

$$MI1 = \Psi_r(3, \Psi_\theta(L))$$
$$MI2 = \Psi_r(3, \Psi_\theta(L.e^{j2\theta}))^2$$
$$MI3 = \Psi_r(4, \Psi_\theta(L.e^{j3\theta}))^2$$

....(2.18)

An example of the derivation together with a listing of the expressions for for the other invariant moments are given in appendix A-1. The index k is positive for all the seven moment invariants, thus radial dependence emphasises the pixel points farther away from the central portion of image. The central part features are suppressed.

The properties of rotation and reflection invariance are easily demonstrated using radial and angular moment invariants. Let us consider MI4.

$$MI4 = \Psi_r(4, \Psi_\theta(L.e^{j\theta}))^2$$

....(2.19)

For an image going through a rotation of $\phi$ radians

$$MI4' = \Psi_r(4, \Psi_\theta(L.e^{j(\theta+\phi)}))^2$$
$$= \Psi_r(4, \Psi_\theta(L.e^{j\theta}.e^{j\phi}))^2$$
$$= MI4.$$

since $|e^{j\phi}| = 1$.

For reflection we replace $\theta$ by $-\theta$ and

$$MI4' = \Psi_r(4, \Psi_\theta(L.e^{-j\theta}))^2$$
$$= \Psi_r(4, \Psi_\theta(L.e^{j\theta}))^2$$
$$= MI4.$$

since $|e^{-j\theta}| = |e^{j\theta}|$

A general expression for radial and angular moment invariants is thus given as,

$$MI = \Psi_r(K, \Psi_\theta(L.e^{jm\theta}))^2 \qquad \ldots.(2.20)$$

where K= radial index.
L= image function in polar coordinates.
m= angular factor.

The radial index K can be chosen, in the sign of its value, to suit our purpose without impairing those properties which make moment invariants a good choice for recognition work. If K is chosen to possess negative value, then the information suppression with increasing radial distance is reversed. A fractional K could also be chosen. Equation (2.20) thus gives a simple way of generating moment invariants with the advantage of being able to choose K and $\theta$ to suit the application at hand.

## 2.3 FEATURE SPACE VECTORS.

The pixel values that make up an image can be thought of as the elements of the image description space. The specific spatial arrangement of these pixels gives the salient features needed for recognition. The desire to reduce the number of elements in the description space and still retain the essential features of the image is the motivation behind feature descriptors. While as much information as possible is needed for discrimination, redundant information must be avoided by using the minimum number of features necesary. The features so extracted constitute the feature space vector.

The knowledge attained through the analysis of the last section will enable us to construct various feature space vectors, using moment invariants.

These are the feature vectors we shall be using in subsequent sections for motion estimation. From the expressions for the moment invariants in terms of radial and angular moments, we can list the radial dependences as follows

$$MI1 \propto r^3 \quad : \quad MI2 \propto r^3$$
$$MI3 \propto r^4 \quad : \quad MI4 \propto r^4$$
$$MI5 \propto r^4 \quad : \quad MI6 \propto r^3 \text{ and } r^4$$
$$MI7 \propto r^4$$

....(2.21)

The varying angular factors (see appendix A-1) only emphasise the sampling pattern. MI1 and MI2 have the same radial index 3, MI3, MI4, MI5 and MI7 have radial dependence of the order 4. MI6 is jointly dependent on radial index of order 3 and 4. All the moment invariants having the same radial dependence may not be telling different stories about the image, but for the difference in the angular factor. As stated earlier on MI7 is a skew invariant moment and its sign changes upon reflection. For our purpose we may drop this moment invariant without loss of generality because we do not envisage a reflected image. Thus we have constructed the following feature vectors;

$$FV1 = \begin{bmatrix} MI1 \\ MI3 \\ MI5 \\ MI6 \end{bmatrix} \qquad FV2 = \begin{bmatrix} MI2 \\ MI4 \\ MI5 \\ MI6 \end{bmatrix}$$

$$FV3 = \begin{bmatrix} MI1 \\ MI2 \\ MI3 \\ MI4 \\ MI5 \\ MI6 \end{bmatrix}$$

In his experiment Hu [35] used only the first and

second moment invariants for recognition of alphabets and numbers with high sucess. We have thus included a feature vector consisting only of MI1 and MI2.

$$FV4 = \begin{bmatrix} MI1 \\ MI2 \end{bmatrix}$$

The result of simulations carried out using these feature vectors will be presented in section 2.5. Recall that these features have the property of suppressing the central parts of the image under consideration. We shall now construct feature vectors that will emphasize both radially inwardly and outwardly using radial and angular moment invariants

From equation (2.15),

$$MI = |\Psi_r(K, \Psi_\theta(L \exp(jm\theta))|^2$$

Arbitrarily we choose,

$$\begin{aligned} MI1' &= |\Psi_r(1, \Psi_\theta(L \exp(j\theta))|^2 \\ MI2' &= |\Psi_r(-1, \Psi_\theta(L \exp(j\theta))|^2 \\ MI3' &= |\Psi_r(2, \Psi_\theta(L \exp(j\theta))|^2 \\ MI4' &= |\Psi_r(-2, \Psi_\theta(L \exp(j\theta))|^2 \end{aligned} \qquad \ldots.(2.22)$$

Three feature vectors will be constructed from these moment invariants. They are,

$$FV5 = \begin{bmatrix} MI1' \\ MI2' \end{bmatrix} \qquad FV6 = \begin{bmatrix} MI3' \\ MI4' \end{bmatrix}$$

$$FV7 = \begin{bmatrix} MI1' \\ MI2' \\ MI3' \\ MI4' \end{bmatrix}$$

## 2.4 MATCHING SCHEME AND MOTION ESTIMATION

In this section we shall develop a motion estimation scheme based on moment invariant-derived feature matching. It is effectively a variation of the usual matching scheme based on correlation. However the features extracted via the method of moment invariants will be employed in this scheme. Given a sequence of images containing moving objects, the image frame is segmented into moving areas and stationary areas. The features in the moving area are extracted and used for tracking and subsequent motion estimation. A simple thresholding of the frame difference signal derived from a pair of images can be used to segment the image into moving and stationary areas. However this will entail setting the threshold "manually" from one image pair to another, since variations in imaging conditions may not facilitate the use of a constant threshold. A technique has been developed to set the segmentation threshold automatically, thus facilitating adaptation to variations in imaging conditions.

Given two image frames $f_1(x,y)$ and $f_2(x,y)$, with x and y representing the coordinates of the pixel values: $f_2$ is the reference image while $f_1$ is the search image. The frame difference signal at the position (x,y) is defined as,

$$FD(x,y) = f_1(x,y) - f_2(x,y) \qquad \ldots.(2.23)$$

The segmentation threshold is defined as the root-mean-square (r.m.s) value of the frame difference signal computed over the whole image frame. Thus, the threshold is written as,

$$FDTH = \{1/(MN)\sum_{x=1}^{M}\sum_{y=1}^{N} FD^2(x,y)\}^{1/2} \qquad \ldots.(2.24)$$

where M and N are the dimensions of the image frames. The

reference image $f_2$, is partitioned into blocks of dimension mxn such that m < M and n < N. The r.m.s value of the frame difference signal over the block area m x n is computed for each block and compared with the global threshold FDTH.

For the (k,l)th block the local r.m.s value is defined as,

$$LTH(k,l) = \{1/(mn)\sum_{j=nl}^{n(l+1)}\sum_{i=mk}^{m(k+1)} FD^2(i,j)\}^{1/2}$$

$$k=0,1,2,\ldots,M/m-1$$
$$l=0,1,2,\ldots,N/n-1$$

....(2.25)

In areas where there is considerable change in average pixel value, it is reasonable to assume that this is due to the relative motion between the frame pair and otherwise for areas of slight change. In the stationary areas it is expected that the frame difference is zero, but as aforementioned, variations in the imaging conditions from one frame to the other will not make this possible. Thus if LTH is greater than or equal to FDTH for a given block, the block is classified as a moving area. For LTH less than FDTH, the corresponding block is classified as a stationary area.

$$LTH \begin{cases} \geq FDTH : \text{moving area.} \\ < FDTH : \text{stationary area.} \end{cases}$$

The advantage of this method of segmentation, besides the simplicity, is in the adaptation of the threshold to variation in the average pixel value from frame to frame. The set of image blocks {(k,l)} for which LTH is greater than or equal to FDTH will be passed on to the next processing stage: computation of feature vectors. Any of the feature vectors derived in section 2.3 can be used at this stage. In

order to locate the position, in the search image frame, of the moving area block, we match the feature vector of the block with that of blocks of similar dimension in the search frame. This is done within a limited region of the search frame called the search area, usually given as horizontal (p) and vertical (q) displacements. A metric is used to measure the extent of similarity between the vectors. In the work reported here the following metric has been used.

Supposing we have two vectors V1 and V2 of equal dimension M, a measure of similarity (SM) based on the mean square error (mse) is defined as,

$$SM = 1/M \sum_{i=1}^{M} (V1_i - V2_i)^2 \qquad \ldots.(2.26)$$

where the $V1_i$ represents the ith component of the feature vector V1, and $V2_i$ that of vector V2. Other similarity measures could be defined, e.g,

$$D(V1,V2) = |V1-V2| \qquad \ldots.(2.27)$$

The similarity measure of equation (2.26) is computed for every shift in the x- and y-axes of the reference image plane, within the search area. The shift corresponding to the minimum SM gives an estimate of the location of the block under consideration in the reference image. Figure 2.1 depicts the matching scheme.

The computational effort required at each search point could be tremendous when using a feature vector of large size, like the seven moment invariants. Reduction of the feature vector size using arguments of section 2.3 are efforts in reducing the unnecessary computation. Furthermore, with ordinary template matching we require (2p+1)(2q+1)

searches for an (m+2p)(n+2q) search area (p and q are the bounds on horizontal and vertical displacements respectively). However apart from feature vector size reduction more could be done at improving the computational efficiency of matching schemes. Various two-stage matching schemes have been reported in the literature [5, 24, 28, 80, 84, 85] aimed at improving the computational efficiency. Within the framework of this research a simple scheme is required. We have thus devised a two-stage matching scheme based on the same threshold used in segmenting the image blocks into moving and stationary areas.

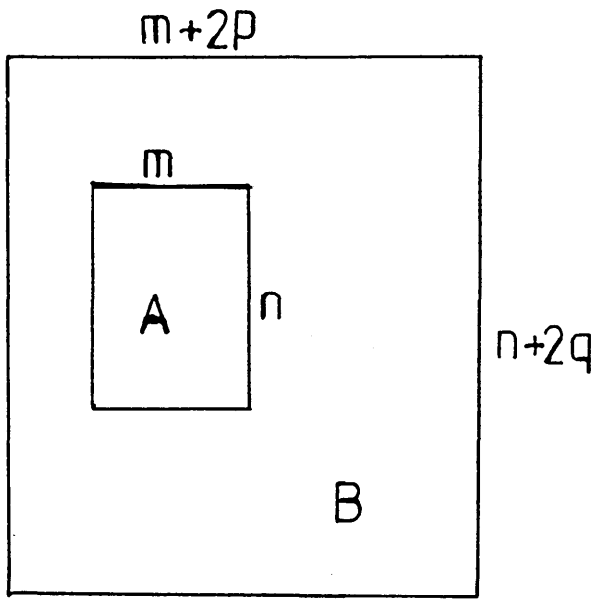
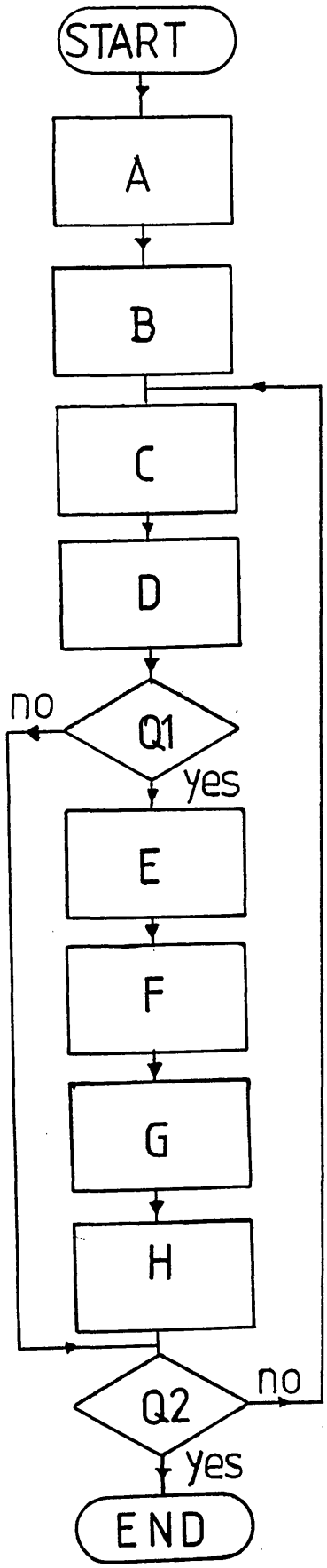



Figure 2.1: Template matching.
Reference block A(mxn) is searched within search area B(m+2p x n+2q). Total number of searches= (2p+1)(2q+1).

At each search point the r.m.s value of the frame difference between the reference block and the corresponding search block is computed. Blocks with any close resemblance to the reference block will have small values of difference. Using the global threshold (FDTH) already computed, a

discriminating function could be set up. We shall consider as candidates for further search, those blocks within each search area with r.m.s value of frame difference less than FDTH. At these selected search points the feature vectors for the corresponding blocks are computed and the mean square error (m.s.e) between them and the reference block accumulated. The block with the minimum m.s.e gives an estimate of the displacement of the reference block. The process is continued for all moving area blocks. A considerable saving ( as much as 30%) could be achieved in terms of computational effort. Figure 2.2 shows the flowchart of the two-stage feature based matching algorithm for motion estimation.




LEGEND

A: COMPUTE GLOBAL R.M.S VALUE OF FRAME DIFFERENCE SIGNAL, FDTH.

B: DIVIDE IMAGE INTO N BLOCKS.

C: COUNT 1,...,N

D: CLASSIFY BLOCK AS MOVING OR STATIONARY AREA USING FDTH.

E: COMPUTE FEATURE VECTOR FOR BLOCK.

F: SELECT SEARCH AREA USING FDTH.

G: PERFORM MINIMUM MEAN SQUARE ERROR SEARCH OVER SELECTED AREA.

H: ASSIGN MOTION ESTIMATE TO BLOCK.

Q1: DOES BLOCK BELONG TO MOVING AREA?

Q2: IS COUNT = N?

Figure 2.2: Flowchart of two-stage feature based matching scheme for motion estimation. Features are derived from moment invariants.

## 2.5 SIMULATIONS AND RESULTS.

A number of simulations were carried out to demonstrate the efficiency of the techniques proposed. Given a pair of images, the frame difference signal in the stationary areas is equal to zero. This is so, provided the pixels in the stationary areas have similar values. However, in reality the values differ owing to noise that arise from various factors like changing illumination condition, camera parameters, quantization error and other imaging errors. Under these conditions the frame difference in the stationary areas does not equal zero, but some minimum. In the moving areas the frame difference signal will be much larger than in the stationary areas. The purpose of motion estimation is to compute motion vectors for moving areas and hence use estimates to achieve a minimization of the frame difference signal in the moving areas.

A measure of performance of the proposed schemes would be the extent to which the frame difference signal in the moving areas has been reduced after motion compensation. The variance of the frame difference signal before and after motion compensation would serve a very good purpose. Also applicable is the prediction error gain, $G_p$, given as,

$$G_p = 10 \times \log_{10} \frac{\text{(variance of image signal)}}{\text{(variance of frame difference)}}$$

Image signal variance is computed over the whole image, while the frame difference variance is computed in the moving areas only. This performance index will be used in comparing different variations of the scheme.

Test images of varying properties were acquired to test the schemes out. The image sequence "Walking man"

represents an outdoor scene of well textured background with slow to medium motion. Motion between frames could be made appreciable by omitting some frames from the sequence. Maximum motion is about 8 pixels/frame. Each frame was originally 512x512 pixels digitized at 8bits/pixel. Using neighbourhood averaging the size was reduced to 256x256 and treated as noninterlaced grid image. Sequence "Walking man" consists of frames from a television image sequence of a man walking in front of a camera. Detailed information about the presence or absence of camera motion, (panning or zooming) is not available from officials of Student Television Services of Imperial College (STOIC), who gave the sequence. There are 30 frames, out of which 11 have been chosen for these experiments and others omitted to allow appreciable interframe motion. All experiments are off-line, by which is meant that the whole sequence of frames were not processed at one time. Pairwise processing was done.

The image sequence "Box" represents artificially generated image. The "Box" is made up of gaussian random numbers and moved over a background generated from gaussian random numbers, however with differing mean and variance. A routine which moves the "Box" around is used in generating a sequence of frames, such that motion through the sequence is simulated. Each frame is 256x256 pixels and digitized at 8bits/pixel. Eleven frames were worked with in order to obtain 10 pairs of frames.

The sequence "Alan" represents an indoor sequence. It is derived from 35mm head and shoulder snapshot of a subject imitating a newscaster. Snapshots were taken at intervals to allow motion of subject. Acquired image frames were digitized at 8bits/pixel and 700x480 pixels/frame. Neighbourhood averaging was applied to reduce resolution to 350x240.

Feature vectors FV1, FV2, FV3, FV4, FV5, FV6 and FV7 were used in motion estimation with varying block sizes:2x4, 4x4, 4x8, 8x8, 8x16. The feature vectors derived from ordinary moment invariants were tried out on the test sequences. Table T2.1 shows the result, in terms of frame difference signal reduction, of using the feature vector FV3 (six-component vector) on the sequence "Walking man". Tables T2.2 and T2.3 show similar result on the sequences "Box" and "Alan" respectively.

Table T2.1:
Variance of frame difference in moving areas.
Block size:4X4; Feature vector:FV3.
Image Sequence:"Walking man".

| IMAGE PAIR. | COMPENSATED VARIANCE. | UNCOMPENSATED VARIANCE. |
|---|---|---|
| 1 | 27.29 | 33.48 |
| 2 | 27.26 | 34.27 |
| 3 | 25.58 | 31.99 |
| 4 | 28.11 | 34.76 |
| 5 | 29.03 | 34.22 |
| 6 | 27.01 | 33.58 |
| 7 | 28.84 | 40.85 |
| 8 | 13.74 | 30.97 |
| 9 | 33.67 | 40.13 |
| 10 | 36.61 | 48.35 |

Table T2.2:

Variance of frame difference in moving areas.
Block size:4X4; Feature vector:FV3.
Image Sequence:"Box".

| IMAGE PAIR. | COMPENSATED VARIANCE. | UNCOMPENSATED VARIANCE. |
|---|---|---|
| 1 | 11.75 | 18.49 |
| 2 | 13.81 | 32.05 |
| 3 | 28.00 | 39.00 |
| 4 | 21.87 | 39.65 |
| 5 | 3.54 | 10.67 |
| 6 | 27.84 | 39.68 |
| 7 | 24.31 | 45.87 |
| 8 | 28.96 | 78.28 |
| 9 | 25.60 | 51.77 |
| 10 | 33.34 | 64.98 |

As a means of comparing the gains achieved in using the various feature vector sizes the feature vectors FV1, FV2, FV3, and FV4 were applied to the sequence "Walking man". The frame difference variances obtained are tabulated in table T2.4.

Table T2.3:

Variance of frame difference in moving areas.
Block size:4X4; Feature vector:FV3.
Image Sequence:"Alan".

| IMAGE PAIR. | COMPENSATED VARIANCE. | UNCOMPENSATED VARIANCE. |
|---|---|---|
| 1 | 89.71 | 1130.18 |
| 2 | 141.77 | 937.63 |
| 3 | 80.94 | 589.52 |
| 4 | 193.80 | 818.95 |
| 5 | 34.25 | 696.06 |
| 6 | 41.82 | 284.07 |
| 7 | 39.71 | 643.45 |
| 8 | 30.39 | 222.02 |
| 9 | 33.04 | 222.02 |
| 10 | 97.51 | 642.11 |

Table T2.4:

Comparison of frame difference variance using feature vectors FV1, FV2, FV3 and FV4.

Block size:2X4.

Image Sequence:"Walking man".

| IMAGE PAIR. | FV1 | FV2 | FV3 | FV4 |
|---|---|---|---|---|
| 1 | 25.54 | 25.58 | 25.55 | 25.55 |
| 2 | 25.62 | 25.64 | 25.67 | 25.67 |
| 3 | 24.56 | 25.59 | 24.59 | 24.59 |
| 4 | 26.20 | 26.13 | 26.13 | 26.13 |
| 5 | 27.40 | 27.36 | 27.36 | 27.36 |
| 6 | 25.53 | 25.55 | 25.55 | 25.55 |
| 7 | 26.65 | 26.43 | 26.43 | 26.43 |
| 8 | 10.77 | 10.79 | 10.79 | 10.79 |
| 9 | 30.44 | 30.39 | 30.39 | 30.40 |
| 10 | 32.57 | 32.44 | 32.44 | 32.43 |

Since block size will affect the performance of the scheme, simulations were carried out to investigate the effect of block size. Using the two-component feature vector FV4, the prediction gains achieved using various block sizes are compared in table 2.5.

Table T2.5:

Prediction Gain (dB) variation with varying block size

Image Sequence:"Walking man".

Feature vector:FV4.

| IMAGE PAIR. | BLOCK SIZE. | | | | |
|---|---|---|---|---|---|
| | 2 X 4 | 4 X 4 | 4 X 8 | 8 X 8 | 8 X 16 |
| 1 | 16.62 | 16.33 | 16.19 | 15.92 | 15.82 |
| 2 | 16.36 | 16.02 | 15.90 | 15.54 | 15.49 |
| 3 | 16.45 | 16.28 | 16.14 | 15.86 | 15.89 |
| 4 | 16.47 | 16.15 | 16.04 | 15.82 | 15.76 |
| 5 | 16.39 | 16.13 | 15.97 | 15.75 | 15.67 |
| 6 | 16.71 | 16.47 | 16.28 | 16.01 | 16.08 |
| 7 | 16.15 | 15.77 | 15.37 | 15.22 | 15.05 |
| 8 | 19.92 | 18.87 | 17.92 | 16.65 | 16.72 |
| 9 | 15.86 | 15.42 | 15.20 | 14.98 | 14.97 |
| 10 | 15.11 | 14.59 | 14.25 | 14.00 | 14.07 |

Simulations similar to those used with ordinary moment invariants were carried out using radial and angular moment invariants. Tables T2.6 and T2.7 show the result of using feature vector FV7, the four-component vector, on the sequences "Walking man" and "Box" respectively. A block size of 4 x 4 has been used. In this situation a block size of 2 x 4 is not feasible because of the necessary interpolation that must be done in transforming from rectangular to polar coordinates. A comparison of the three feature vectors FV5, FV6, FV7 is done in table T2.8. The effect of varying the block size is shown in table T2.9. We have attempted comparing the ordinary moment invariants and radial and angular moment invariants. Table T2.10 shows the prediction gains (dB) obtained by using feature vectors FV1 (ordinary moment invariants) and FV7 (radial/angular moment invariants), on the sequence "Walking man".

Table T2.6:

Variance of frame difference in moving areas.
Block size: 4 X 4.
Sequence: "Walking man".
Feature vector: FV7

| IMAGE PAIR. | COMPENSATED VARIANCE. | UNCOMPENSATED VARIANCE. |
|---|---|---|
| 1 | 29.50 | 33.48 |
| 2 | 30.49 | 34.27 |
| 3 | 28.82 | 31.99 |
| 4 | 31.02 | 34.76 |
| 5 | 30.91 | 34.22 |
| 6 | 29.56 | 33.58 |
| 7 | 28.49 | 34.80 |
| 8 | 23.56 | 30.97 |
| 9 | 36.71 | 40.13 |
| 10 | 41.69 | 48.35 |

Table T2.7:

Variance of frame difference in moving areas.
Block size: 4 X 4.
Sequence: "Box".
Feature vector: FV7

| IMAGE PAIR. | COMPENSATED VARIANCE. | UNCOMPENSATED VARIANCE. |
|---|---|---|
| 1 | 12.01 | 18.49 |
| 2 | 13.81 | 32.05 |
| 3 | 13.16 | 39.00 |
| 4 | 23.93 | 39.65 |
| 5 | 3.54 | 10.67 |
| 6 | 29.49 | 39.68 |
| 7 | 28.03 | 45.87 |
| 8 | 38.30 | 78.28 |
| 9 | 31.54 | 51.77 |
| 10 | 42.10 | 64.98 |

Table T2.8:

Comparison of frame difference variance using feature vectors FV5, FV6 and FV7.

Block size: 4 x 4.

Sequence: "Walking man".

| IMAGE PAIR. | FV5 | FV6 | FV7 |
|---|---|---|---|
| 1 | 27.55 | 31.28 | 29.80 |
| 2 | 28.21 | 31.44 | 30.49 |
| 3 | 26.16 | 29.68 | 28.82 |
| 4 | 28.43 | 32.38 | 31.02 |
| 5 | 29.20 | 32.11 | 30.91 |
| 6 | 27.09 | 31.02 | 29.56 |
| 7 | 30.47 | 35.62 | 33.64 |
| 8 | 15.99 | 26.67 | 23.56 |
| 9 | 34.31 | 37.82 | 36.71 |
| 10 | 38.19 | 43.56 | 41.69 |

Table T2.9:

Prediction Gain (dB) variation with varying block size.

Sequence:"Walking man".

Feature vector: FV5.

| IMAGE PAIR. | BLOCK SIZE: 4 X 4 | 4 X 8 | 8 X 8 | 8 X 16 |
|---|---|---|---|---|
| 1 | 16.29 | 15.97 | 15.89 | 15.83 |
| 2 | 15.95 | 15.76 | 15.54 | 15.49 |
| 3 | 16.18 | 15.86 | 15.85 | 15.91 |
| 4 | 16.10 | 15.76 | 15.78 | 15.58 |
| 5 | 16.11 | 15.79 | 15.73 | 15.61 |
| 6 | 16.46 | 16.15 | 15.98 | 16.07 |
| 7 | 15.53 | 15.06 | 15.20 | 15.02 |
| 8 | 18.21 | 16.84 | 16.65 | 16.63 |
| 9 | 15.34 | 15.04 | 14.96 | 14.95 |
| 10 | 14.40 | 13.87 | 13.95 | 14.06 |

Table T2.10:

Comparison between prediction gains (dB) ordinary and radial/angular moment invariants.
Block size: 4 x 4.
Sequence: "Walking man".

| IMAGE PAIR. | FV1 | FV7 |
|---|---|---|
| 1 | 16.33 | 15.99 |
| 2 | 16.02 | 15.61 |
| 3 | 16.28 | 15.76 |
| 4 | 16.15 | 15.72 |
| 5 | 16.13 | 15.86 |
| 6 | 16.47 | 16.08 |
| 7 | 15.77 | 15.10 |
| 8 | 18.87 | 16.53 |
| 9 | 15.42 | 15.04 |
| 10 | 14.59 | 14.02 |

## 2.6 DISCUSSION OF RESULTS AND CONCLUSION.

A comparison of the amount of reduction in frame difference variance obtained with the three test sequences (tables T2.1, T2.2 and T2.3) show that moment invariants are susceptible to imaging noise. The greatest reduction has been achieved with the sequence "Alan" and the least with "walking man". The result of table T2.4 shows that there is not much gain in using more invariant moments in the feature vector. This is attributable to the fact that all these features are derived from invariant moments of order up to three. In other words the same information is being expressed in different ways and no new information is gained by using six instead of four feature vectors. A situation where moment invariants derived from first through sixth order moments,say, are used in forming the feature vectors will yield similar trend. A

way out is to make each component of the feature vector derivable from invariant moments of varying orders. There is however a limitation in the extent to which it can be done. The computational effort increases as the order of moment increases, and usage of this scheme in realtime will then probably require more efficient computational procedures like parallel processing or concurrent programming.

Block size will have a tremendous effect on the overall performance of this scheme. A very large block size will not allow for accurate partitioning of the image into moving and stationary areas. Also a very small block size will limit the discriminating efficiency of the feature vectors. The likelihood of finding more than one two-pixel block with nearly similar configuration as a two-pixel template is higher than it is for an eight-pixel template or 16-pixel block, with an eight-pixel or 16-pixel template. The simulation carried out has shown that a very large block size does not give high prediction gain (table T2.5). The possibility of having a poor prediction gain with small block size will be heightened with high level of noise in the image. In this situation a not too large block size will be useful, for a given feature vector set.

The trend shown in the performance of the radial and angular moment invariants is similar to that of the ordinary moment invariants. The effect of noise is shown when the amount of reduction in the frame difference variance, with "Walking man" is compared to the sequence "Box" (tables T2.6 and T2.7). The feature vectors FV5, FV6 and FV7 are compared in table T2.8 using the sequence "Walking man". FV5 has radial indices of 1 and -1 combined to give a "balance-out" effect on the feature supression property of moment invariants. However its radial dependence is to be seen as "linear" while FV6, with radial indices of 2 and -2, has a "quadratic" dependence. FV7 combines the "linear" and

"quadratic" dependencies. Table T2.8 indicates that FV5 performs better than either of FV6 and FV7. In table T2.10 the performance of the radial and angular moment invariant has been compared with the ordinary moment invariants. The feature vectors FV1 and FV7 though have similar components count are nevertheless not equivalent. Radial and angular moment invariants are out-performed by the ordinary moment invariants, by about 1dB on the average.

The effect on the occlusion has not been investigated in isolation. Some part (the arm) of the sequence "Walking man" is undergoing occlusion; more of the arm became exposed as the sequence progress in time. In the next chapter we shall consider the differential technique developed in the projection domain.

# CHAPTER 3

## DIFFERENTIAL TECHNIQUE IN THE PROJECTION DOMAIN.

### 3.1 INTRODUCTION.

The differential technique, as introduced in the paper of Limb and Murphy [47] essentially relates the spatial and temporal changes in a sequence of images to the translational motion causing the changes. The temporal change or the frame difference signal as it is aptly called, is derived from the difference of a pair of succesive image frames. The spatial change (element or line difference) is derived from intra-frame differences between pixel values, and could either be vertical or horizontal. Vertical spatial difference or line difference being normally derived from vertically adjacent pixels and horizontally adjacent pixels give rise to horizontal spatial difference or element difference. In their derivation of the displacement estimates of moving objects in a television scene [47], frame differences having similar sign to the corresponding element (or line) differences were accumulated separately from those having dissimilar sign. The difference between the magnitude-sum of the accumulated frame-differences was normalised by the magnitude-sum of the element (or line) differences to obtain the horizontal (or vertical) displacement estimate. Thus if FDS denotes frame difference signal, and EDS denotes element difference the accumulated frame differences will be given as,

$$FX1 = \sum_{MA} |FDS| sb(FDS) * sb(EDS)$$

$$FX2 = \sum_{MA} |FDS| \overline{sb(FDS) * sb(EDS)}$$

where sb(.) is a binary signal with value 1 if the argument

is positive and 0 if it is negative, * denotes the exclusive OR operation and the bar denotes the logical complement. The horizontal component of motion, DX, would be given as,

$$DX = (FX1 - FX2) / \sum_{MA} |EDS| \qquad \ldots.(3.1)$$

The subscript MA, indicates that the summation is to be taken over the moving area pixels. Various modifications to the basic idea have appeared in the literature [12, 13, 14, 53, 59, 71]. It is interesting to note that some of the modifications upon simplification reduce to equation (3.1). Starting off from a Taylor's series expansion of a displaced two-dimensional function Cafforio and Rocca [12] derived displacement estimate expressions which upon approximation is equivalent to equation (3.1).

The technique (a differential technique variation) being proposed in this chapter operates on the projections in the moving areas. Projections have been used to separate displacement into horizontal and vertical components. Also in taking projections, a low pass filtering process is implicitly introduced. In the differential technique, the gradients at the edges of images have large values which lead to erroneous estimation. This problem is usually solved by a low pass filtering preprocessing. The projection approach being proposed will not require this since taking projections implicitly smooothens the data. In section 3.2, the fundamental equations for the differential technique will be derived and the projection approach presented.

## 3.2 THE PROJECTION APPROACH.

The image sequence (time sequential) being considered consists of image frames which are two dimensional function of spatial variables x and y. The changes from frame

to frame are caused by the motion in time, of the objects in the scene recorded in the image sequence. The pixel value at spatial coordinates (x,y) in a given image frame will be denoted by $f_t(x,y)$. The subscript, t, is the temporal variable indicating the time order in which the frames were acquired. Furthermore let the displacements in the x (horizontal) and y (vertical) directions of the image plane, over a time interval, dt, be given as DX and DY respectively.

If we assume that all the pixels in a frame appear in a succeeding frame with the same value, except for some displacement in the case of moving objects [38], then,

$$f_t(x,y) = f_{t-dt}(x-DX,\ y-DY) \qquad \ldots.(3.2)$$

Equation (3.2) represents a basic assumption which in reality is not the case, because in newly decovered and covered areas of the image frame we are having new pixels values which were hitherto not present. This is a major drawback, because for such areas the differential approach will breakdown.

If we are considering small amount of motion from one frame to another and the image function possesses continuous nth derivative, it is possible to expand the right hand side of equation (3.2) in a Taylor's power series. Thus,

$$f_{t-dt}(x-DX,\ y-DY)$$

$$= f_{t-dt}(x,y) - DX\frac{\partial}{\partial x} f_{t-dt}(x,y) - DY\frac{\partial}{\partial y} f_{t-dt}(x,y) - e(x,y) \qquad \ldots.(3.3)$$

where e(x,y), represents higher order terms of the expansion.

For small motion, the expansion can be approximated by taking only up to the linear terms. In which case,

$$f_{t-dt}(x-DX,\ y-DY) - f_{t-dt}(x,y)$$

$$= -DX\frac{\partial}{\partial x} f_{t-dt}(x,y) - DY\frac{\partial}{\partial y} f_{t-dt}(x,y) \qquad \ldots.(3.4)$$

Equation (3.4) is the basic equation for estimating two-dimensional translations, using the differential approach [38]. Let equation (3.4) be written for the sake of compactness as,

$$FD(x,y) = -\ DX\ GRX(x,y) - DY\ GRY(x,y) \qquad \ldots.(3.5)$$

where,

$$FD(x,y) \triangleq f_{t-dt}(x-DX,\ y-DY) - f_{t-dt}(x,y)$$

$$= f_t(x,y) - f_{t-dt}(x,y)$$

is the frame difference value computed for two succesive image frames at the spatial point (x,y).

$$GRX(x,y) \triangleq \frac{\partial}{\partial x} f_{t-dt}(x,y)$$

is the partial derivative with respect to x, evaluated at the point (x,y).

$$GRY(x,y) \triangleq \frac{\partial}{\partial y} f_{t-dt}(x,y)$$

is the partial derivative with respect to y, evaluated at the point (x,y).

Since we are dealing with digital images, the partial derivatives are approximated as differences between

adjacent pixels along the x or y image axis. It is also possible to implement the partial derivatives as the sobel operator or any of its variations.

If the current image frame (IMAGE 2) is denoted by $f_t(x,y)$ and the previously acquired image frame (IMAGE 1) as $f_{t-dt}(x,y)$. It becomes apparent that the partial derivatives must be computed from IMAGE 1. It is as though IMAGE 2, the current image, is being predicted from IMAGE 1.

At each moving area point (x,y) we can compute FD(x,y), GRX(x,y) and GRY(x,y) and get one linear equation with two unknowns. In an implementation of the basic equation Huang and Hsu [54] created a system of equations by computing equation (3.4) at points deemed to have similar motion and solved for the displacement estimates. In matrix form, the system of equations can be written as,

$$[F] = [G]\ [S] \qquad \ldots.(3.6)$$

Where [F] is the frame difference column vector, [G] is the gradient matrix and [S] the displacement vector. The matrix [G] is usually not square, hence the method of pseudoinverse is used to solve (3.6). Assuming that the moving area could be partitioned into rectangular blocks of pixels or in the case of multiple objects moving differently, that the different moving areas could be partitioned into rectangular blocks, then similar motion estimates would be obtained for all points within a moving area block.

We proceed with the presentation of the projection approach by recalling that equation (3.3) is a convergent power series and hence term by term integration is possible. This is true as long as the integration is within the interval of convergence under consideration [76]. The

integration will however be taken partially, in other words, integration will be with respect to the variables x and y separately. Let us assume that integration will be taken over the interval;

$$a <= x <= b \quad ; \quad c <= y <= d$$

which is within the interval of convergence of the series expansion.
Integration of equation (3.4) with respect to x yields,

$$\int_a^b f_{t-dt}(x-DX,y-DY)dx = \int_a^b f_{t-dt}(x,y)dx - \int_a^b DX\frac{\partial}{\partial x}f_{t-dt}(x,y)dx - \int_a^b DY\frac{\partial}{\partial y}f_{t-dt}(x,y)dx \quad ....(3.7)$$

Similarly integration with respect to y, yields;

$$\int_c^d f_{t-dt}(x-DX,y-DY)dy = \int_c^d f_{t-dt}(x,y)dy - \int_c^d DX\frac{\partial}{\partial x}f_{t-dt}(x,y)dy - \int_c^d DY\frac{\partial}{\partial y}f_{t-dt}(x,y)dy \quad ....(3.8)$$

from equation (3.7) we can write,

$$\int_a^b [f_{t-dt}(x-DX,y-DY) - f_{t-dt}(x,y)]dx = -DX\int_a^b \frac{\partial}{\partial x} f_{t-dt}(x,y)dx - DY\int_a^b \frac{\partial}{\partial y} f_{t-dt}\, dx \quad ....(3.9)$$

or

$$\int_a^b [f_t(x,y)- f_{t-dt}(x,y)]dx = -DX\int_a^b \frac{\partial}{\partial x}f_{t-dt}(x,y)dx - DY\int_a^b \frac{\partial}{\partial y}f_{t-dt}(x,y)dx \quad ....(3.10)$$

When equation (3.2) is substituted into (3.9) equation (3.10) can be viewed as the y-projection of the integrands. However we are dealing with digital images, hence integrals will be replaced with summations.
Thus equation (3.10) will be written as;

$$\sum_{x=a}^{b}\left(f_t(x,y) - f_{t-dt}(x,y)\right)$$

$$= -DX\sum_{x=a}^{b}\frac{\partial}{\partial x}f_{t-dt}(x,y) \; -DY\sum_{x=a}^{b}\frac{\partial}{\partial y}f_{t-dt}(x,y) \qquad \ldots.(3.11)$$

which when written in the manner of equation (3.5) becomes;

$$\sum_{x=a}^{b}FD(x,y) = -DX\sum_{x=a}^{b}GRX(x,y) - DY\sum_{x=a}^{b}GRY(x,y) \qquad \ldots.(3.12)$$

Equation (3.8)represents the x-projection and using similar reasoning as used on equation (3.9) we can write

$$\sum_{y=c}^{d}FD(x,y) = -DX\sum_{y=c}^{d}GRX(x,y) - DY\sum_{y=c}^{d}GRY(x,y) \qquad \ldots.(3.13)$$

The implicit assumption is that all the pixels within the region of integration are undergoing similar displacement. Hence DX and DY are constant for these pixels.
Equations (3.12) and (3.13) can be solved simultaneously for DX and DY to yield

$$DX(x,y)=\frac{\sum_{y=c}^{d} FD(x,y)\sum_{x=a}^{b} GRY(x,y)-\sum_{x=a}^{b} FD(x,y)\sum_{y=c}^{d} GRY(x,y)}{\sum_{y=c}^{d} GRY(x,y)\sum_{x=a}^{b} GRX(x,y)-\sum_{x=a}^{b} GRY(x,y)\sum_{y=c}^{d} GRX(x,y)} \quad \ldots.(3.14)$$

$$DY(x,y)=\frac{\sum_{x=a}^{b} FD(x,y)\sum_{y=c}^{d} GRX(x,y)-\sum_{y=c}^{d} FD(x,y)\sum_{x=a}^{b} GRX(x,y)'}{\sum_{y=c}^{d} GRY(x,y)\sum_{x=a}^{b} GRX(x,y)-\sum_{x=a}^{b} GRY(x,y)\sum_{y=c}^{d} GRX(x,y)} \quad \ldots.(3.15)$$

These displacements may be calculated at each individual pixel within the window. Under ideal conditions they should be identical. We shall therefore define an average value for each as

$$DX=1/(b-a)(d-c)\sum_{x=a}^{b}\sum_{y=c}^{d} DX(x,y)$$
$$DY=1/(b-a)(d-c)\sum_{x=a}^{b}\sum_{y=c}^{d} DY(x,y)$$

The expressions of equation (3.14) and (3.15) give the x-and y-displacements respectively. We shall call them algorithm 1 of projection domain differential technique (PDDT1). This algorithm will be compared to the algorithm of Limb and Murphy [47], which we shall call algorithm LM.

### 3.2.2 Implementation Considerations.

The segmentation algorithm developed in chapter 2 has been used in partitioning the image into moving and stationary blocks. Implicitly we have assumed that the objects in the image frames can be divided into blocks of moving and stationary area pixels. Thus given two image frames, the r.m.s value of the frame difference over the image frame is computed and set as the threshold of segmentation. Blocks of various sizes are then created and the r.m.s value of frame difference within them tested

against the set threshold. The blocks with r.m.s value greater than the set threshold are considered as moving area blocks and otherwise for blocks with lower r.m.s value.

The frame difference FD(x,y) to be used in equation (3.14) and (3.15) are computed by point for point differencing between two image frames. The partial derivatives are approximated by the Sobel function at each pixel point. Thus given an arrangement of pixels as shown in figure 3.1, the partial derivative

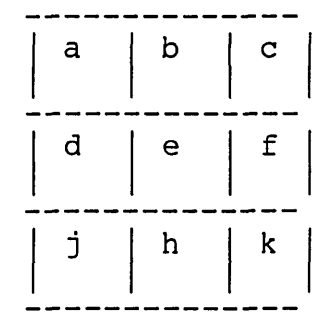



Figure 3.1: Illustrative pixel arrangement.

with respect to x (pdx) at the pixel point e is given by,

$$pdx=\{(2f + c + k)- (2d + a + j)\}/4.0$$

and that with respect to y (pdy) is given by,

$$pdy=\{(2b + a + c)- (2h + j + k)\}/4.0$$

For a given block of pixels, say as given in figure 3.1, the projections are computed by summing up pixels

along the appropriate columns and rows. The x projections will be given by,

$$x1 = a + d + j$$
$$x2 = b + e + h$$
$$x3 = c + f + k$$

Similarly the y projections are given by,

$$y1 = a + b + c$$
$$y2 = d + e + f$$
$$y3 = j + h + k$$

The projections could be normalized by the frame sizes in the two directions. Thus if NX and NY are the x and y dimensions of the image frames, the projections become x1/NY, x2/NY, e.t.c and y1/NX, y2/NX, e.t.c. Results obtained using the normalized projections are not different from those obtained using unnormalized projections. This is expected since any normalizing factor will cancel out from equations (3.12) and (3.13).

The motion estimates derived from this algorithm are usually non-integers, hence some interpolation will be necessary in computing the corresponding predicted pixel value. The following method has been used. For a given displacement estimate DX and DY in the x and y directions respectively, we write,

$$I1 = I - INT(DX)$$
$$J1 = J - INT(DY)$$
$$I2 = I - NINT(DX)$$
$$J2 = J - NINT(DY)$$

where INT(.) and NINT(.) are the integer part and nearest integer, respectively, of the real numbers enclosed in the

brackets. I and J are the undisplaced coordinates of a moving area pixel. The predicted pixel value will thus be given by,

$$0.75 \times f_{t-dt}(I1,J1) + 0.25 \times f_{t-dt}(I2,J2)$$

3.2.3 Experiments, Results and Conclusion.

In a set of experiments the artificially generated image "Box" was used, with the box given displacements of 1, 2, 3, 4 and 5 pixels per frame respectively in both horizontal and vertical directions. In all cases the variance of the frame difference in the moving areas was measured.

The segmentation of the images into moving and stationary areas was performed using the method developed for the feature based technique. Thus, the r.m.s value of the frame difference signal over the image is set as the segmentation threshold. The image frame is partitioned into blocks which are then classified as either moving or stationary areas. For each moving area block we apply equations (3.14) and (3.15) to all the pixels in the block and assign the average of the displacement obtained to the whole block. In order to obtain the motion compensated frame difference variance the computed displacement is applied to equation (3.2) for moving area block pixels. The implementation considered in section 3.2.2 are used for non-integral displacement estimates. For the various displacements we have tabulated (Tables T3.1 and T3.2) the compensated and uncompensated frame difference variance in moving areas using different sizes of blocks.

Table T3.1

Frame difference variance in moving areas, using algorithm PDDT1 on "Box" for varying displacements DX and DY.

| BLOCK SIZE | COMPENSATED VARIANCE | UNCOMPENSATED VARIANCE |
|---|---|---|
| (I) DX= 1, DY= 1. | | |
| 2x4 | 18.62 | 17.99 |
| 4x4 | 18.49 | 18.00 |
| 8x8 | 17.83 | 18.01 |
| 8x16 | 17.87 | 18.01 |
| (II) DX= 2, DY= 2 | | |
| 2x4 | 32.67 | 31.91 |
| 4x4 | 33.02 | 32.00 |
| 8x8 | 32.30 | 31.12 |
| 8x16 | 32.51 | 32.01 |
| (III) DX= 3, DY= 3 | | |
| 2x4 | 45.56 | 45.21 |
| 4x4 | 46.57 | 45.26 |
| 8x8 | 45.71 | 45.58 |
| 8x16 | 46.03 | 45.79 |
| (IV) DX= 4, DY= 4 | | |
| 2x4 | 59.28 | 58.39 |
| 4x4 | 59.76 | 58.20 |
| 8x8 | 58.17 | 58.29 |
| 8x16 | 59.22 | 58.72 |
| (V) DX= 5, DY= 5 | | |
| 2x4 | 71.99 | 71.36 |
| 4x4 | 71.78 | 71.14 |
| 8x8 | 71.26 | 71.16 |
| 8x16 | 72.40 | 71.87 |

Table T3.2

Frame difference variance in moving areas, using algorithm LM on "Box" for varying displacements DX and DY.

| BLOCK SIZE | COMPENSATED VARIANCE | UNCOMPENSATED VARIANCE |
|---|---|---|
| (I) DX= 1, DY= 1 | | |
| 2x4 | 21.85 | 17.99 |
| 4x4 | 19.68 | 18.00 |
| 8x8 | 18.16 | 18.00 |
| 8x16 | 18.00 | 18.01 |
| (II) DX= 2, DY= 2 | | |
| 2x4 | 31.66 | 31.91 |
| 4x4 | 30.48 | 32.00 |
| 8x8 | 31.08 | 32.11 |
| 8x16 | 31.82 | 32.01 |
| (III) DX= 3, DY= 3 | | |
| 2x4 | 44.99 | 45.21 |
| 4x4 | 44.55 | 45.26 |
| 8x8 | 44.92 | 45.58 |
| 8x16 | 45.68 | 45.79 |
| (IV) DX= 4, DY= 4 | | |
| 2x4 | 57.94 | 58.39 |
| 4x4 | 57.98 | 58.20 |
| 8x8 | 57.63 | 58.29 |
| 8x16 | 58.48 | 58.72 |
| (V) DX= 5, DY= 5 | | |
| 2x4 | 68.27 | 71.36 |
| 4x4 | 70.47 | 71.14 |
| 8x8 | 70.60 | 71.16 |
| 8x16 | 71.24 | 71.87 |

From the entries of table T3.1 and T3.2 it can be concluded that the basic algorithms (i.e PDDT1 and LM) do not perform as expected in predicting the motion of the segments

of the image "Box". In cases where there is reduction of the frame difference variance, such reductions are really insignificant. These reduction of frame difference variance when seen from the point of view of quantization in DPCM coding, will be found not to give appreciable gain. Only when there is significant reduction in frame difference variance can motion compensation be justified. Experiments with the sequences "Walking man", "Box", and "Alan" are also tabulated in tables T3.3, T3.4, and T3.5. The results tabulated for the natural scenes (tables T3.3 and T3.5) indicate that algorithm PDDT1 is an improvement over algorithm LM. This can be indicative of the advantage of using projections. In subsequent sections various means of improving the basic algorithm (PDDT1) are explored.

Two means of improvement have been suggested. A reformulation of the problem as a least square estimation problem has been proposed, leading to two algorithms (1) least square estimation in the projection domain, PDDT2 and (2) recursive least square estimation in the projection domain, PDDT4. The second improvement will be to center a block of pixels over each of the pixels in a moving area block and assign computed displacement to the center pixel. In this way we obtain separate estimate for each of the pixels in a moving area block. This variation will be referred to as algorithm PDDT3. Section 3.3 presents algorithm PDDT2.

Table T3.3

Comparison of frame difference variance in moving areas, using algorithms PDDT1 and LM.

Sequence: "Walking Man".

Block size: 8 X 16

| IMAGE PAIR | PDDT1 | LM | UNCOMPENSATED VARIANCE. |
|---|---|---|---|
| 1 | 36.65 | 43.74 | 34.77 |
| 2 | 40.25 | 47.90 | 35.54 |
| 3 | 33.95 | 44.37 | 32.35 |
| 4 | 36.20 | 41.45 | 34.80 |
| 5 | 35.50 | 39.99 | 34.86 |
| 6 | 35.15 | 43.15 | 34.12 |
| 7 | 46.35 | 52.49 | 41.83 |
| 8 | 35.45 | 52.00 | 29.53 |
| 9 | 42.64 | 50.97 | 39.60 |
| 10 | 56.75 | 61.36 | 48.41 |

Table T3.4

Comparison of frame difference variance in moving areas, using algorithms PDDT1 and LM.

Sequence: "Box".

Block size: 8 X 16

| IMAGE PAIR | PDDT1 | LM | UNCOMPENSATED VARIANCE. |
|---|---|---|---|
| 1 | 19.00 | 18.19 | 18.49 |
| 2 | 33.67 | 32.58 | 32.15 |
| 3 | 41.86 | 39.35 | 39.26 |
| 4 | 41.18 | 40.05 | 39.91 |
| 5 | 10.55 | 10.54 | 10.50 |
| 6 | 40.85 | 39.94 | 39.91 |
| 7 | 49.36 | 45.92 | 46.27 |
| 8 | 82.82 | 78.60 | 78.45 |
| 9 | 53.72 | 54.57 | 52.18 |
| 10 | 70.03 | 64.84 | 64.81 |

Table T3.5

Comparison of frame difference variance in moving areas, using algorithms PDDT1 and LM.
Sequence: "Alan".
Block size: 8 X 8

| IMAGE PAIR. | PDDT1 | LM | UNCOMPENSATED VARIANCE. |
|---|---|---|---|
| 1 | 1156.32 | 1523.82 | 1054.54 |
| 2 | 938.62 | 1135.32 | 874.75 |
| 3 | 668.46 | 980.13 | 556.11 |
| 4 | 877.92 | 1240.00 | 772.84 |
| 5 | 714.28 | 1139.95 | 652.22 |
| 6 | 299.51 | 519.91 | 264.71 |
| 7 | 675.63 | 1082.26 | 627.40 |
| 8 | 249.12 | 397.78 | 223.45 |
| 9 | 254.71 | 393.02 | 223.45 |
| 10 | 662.60 | 908.17 | 601.40 |

## 3.3 LEAST SQUARE ESTIMATION IN THE PROJECTION DOMAIN (PDDT2).

In the considerations of this section we shall regard equations (3.12) and (3.13) as models for the displacement estimation problem. The linear least square estimator will be used in estimating the motion parameters DX and DY. In general the least square estimator minimizes the mean-square error criterion. Before applying the estimation technique to our problem we shall give the general "problem statement" and solution.

Given a process described by the equation,

$$y = Cx + v \qquad \ldots.(3.16)$$

in which y is the measurement and x the parameter to be estimated from the noisy measurement. The noise v, is additive, zero-mean independent gaussian random sequence. The covariance of the noise is Rv. Let us assume that x is an n-parameter vector and that we have N measurements from which estimation is to be made. C is an Nxn full-rank matrix, that is, rank of C is n ($n < N$). y is an N-vector of measurements.

The weighted least square estimate is obtained by minimization of the mean square error criterion,

$$J = [y-Cx]^T W^{-1} [y-Cx] \qquad \ldots.(3.17)$$

with respect to x.
Where W is NxN symmetrical, positive definite and is called the weighting matrix. The superscript T denotes transpose. Differentiation of J with respect to x and equating to zero yields,

$$C^T W^{-1} [y-Cx] = 0 \qquad \ldots.(3.18)$$

and the weighted least-squares estimate of x is given by

$$x = [C^T W^{-1} C]^{-1} C^T W^{-1} y \qquad \ldots.(3.19)$$

In practice the requirement for C to be full rank (in which case the columns are linearly independent) is usually satisfied. With this, $[CW^{-1}C]$ is a nonsingular matrix since W is positive definite. The inverse of $[CW^{-1}C]$ thus exists. This condition, referred to as the observability condition, implies that in the absence of noise we can solve for the unknown parameters exactly. We have assumed that our model is correct in the first instance.

If W is chosen to be an identity matrix or differ from it by an arbitrary multiplicative constant then, we have the ordinary least square estimation [58]. In the case where the noise is additive white gaussian the least square estimate is also the maximun likelihood estimate. It is interesting to note also that when the weighting matrix W is chosen to be equal to the covariance of the noise, Rv, then the least square estimator is identical to the minimum variance estimator. Both of them in this case satisfy the Cramer-Rao bound with equality, in estimating a random vector from noisy data[15].

This method, whereby we have all the N measurements available at a time is called the batch approach. A recursive approach which allows the estimate to be updated as data comes in , will be developed after we have applied least-square estimation to our problem.

Thus recalling the model equations (3.12) and (3.13) we have,

$$\sum_{x=a}^{b} FD(x,y) = -DX\sum_{x=a}^{b} GRX(x,y) - DY\sum_{x=a}^{b} GRY(x,y) \qquad \ldots.(3.20)$$

$$\sum_{y=c}^{d} FD(x,y) = -DX\sum_{y=c}^{d} GRX(x,y) - DY\sum_{y=c}^{d} GRY(x,y) \qquad \ldots.(3.21)$$

Each displacement will be determined from the projection onto the corresponding axis. Thus the y-displacement, DY, will be obtained from equation (3.20), the y-projection equation. A similar method is used for the x-displacement employing equation (3.21). The rationale behind this is understood when we consider the shadow cast by a moving object onto horizontal and vertical planes. The component of the motion in the horizontal is embodied in the shadow cast (projection) onto the horizontal plane. The same

is true for motion in the vertical plane.

Rotational motion may pose some problem depending on how large it is. Recording of scenes by the vidicon or ordinary camera entails a mapping process. A three-dimensional world is mapped onto a two-dimensional plane. Obviously, certain information (e.g depth) will be lost as a result of this process. A small rotational motion will be recorded as translational motion in the appropriate planes. A large rotational motion which takes one part of the object out of view of the imaging equipment results in self occlusion. It will be assumed however that rotational motion, if any, is small enough to be modelled as translational motion.

For the purpose of brevity, let equations (3.20) and (3.21) be written respectively as,

$$FDY = DX\ YGX + DY\ YGY \qquad ....(3.22)$$

and

$$FDX = DX\ XGX + DY\ XGY \qquad ....(3.23)$$

where

$$FDY \triangleq \sum_{x=a}^{b} FD(x,y) \ ; \quad FDX \triangleq \sum_{y=c}^{d} FD(x,y)$$

$$YGX \triangleq -\sum_{x=a}^{b} GRX(x,y) \ ; \quad YGY \triangleq -\sum_{x=a}^{b} GRY(x,y)$$

$$XGX \triangleq -\sum_{y=c}^{d} GRX(x,y) \ ; \quad XGY \triangleq -\sum_{y=c}^{d} GRY(x,y)$$

If for the x-projection we assume there are N measurements we can write (3.23) in matrix form as,

$$F = GD \qquad ....(3.24)$$

where D is the 2-component motion parameter to be estimated from F, the N-vector of measurements. G is the Nx2 matrix

that is assumed to be of rank 2.

The noisy process is,

$$F = GD + v \qquad \ldots.(3.25)$$

where v is an independent, zero-mean gaussian random noise sequence ($v \sim N(0, R_v)$). This noise will be attributed to various imaging noises and Taylor's series truncation.

$$F = [FDX_1 \ldots\ldots FDX_N]^T$$

$$D = [DX\ DY]^T$$

$$G = \begin{bmatrix} XGX_1 & XGY \\ \cdot & \cdot \\ \cdot & \cdot \\ \cdot & \cdot \\ XGX_N & XGY_N \end{bmatrix} = \begin{bmatrix} g_1^T \\ \cdot \\ \cdot \\ \cdot \\ g^T{}_n \end{bmatrix}$$

Applying (3.16) through (3.19) to (3.25) will yield an estimate for the displacement vector D,

$$D = [G^T W^{-1} G]^{-1} G^{-1} W^{-1} F \qquad \ldots.(3.26)$$

The weighting matrix W will be chosen to be the unit matrix or some matrix which differ from it by a multiplicative factor. In which case D becomes

$$D = [G^T G]^{-1} G^T F \qquad \ldots.(3.27)$$

A similar equation can be obtained using equation (3.22) for the y-displacement. These equations will constitute algorithm PDDT2 of the projection domain differential technique.

### 3.3.2 Experiments, Results and Conclusion.

The set of images "Box" with displacements 1, 2, 3, 4 and 5 pixels per frame have been used for experiments with algorithm PDDT2. Since this algorithm can use different block sizes, experiments have been performed with varying block sizes. Thus for each displacement the frame difference variance is recorded for block sizes of 2x4, 4x4, 8x8 and 8x16. Table T3.6 presents the compensated and uncompensated frame difference variances, for various block sizes and displacements. From the entries, one finds that the amount of reduction in frame difference variance and hence the accuracy of the estimation depends on the block size used for segmentation and computation and the displacement being considered.

The amount of reduction obtained at a displacement of 2 pixel per frame is not as much as that obtained at a displacement of 1 pixel per frame. At displacements of 4 and 5 pixels per frame the reduction obtained increased. This phenomenon is attributable to two factors. The segmentation accuracy becomes poor as the block size increases for any given displacement and even more so for small displacements. This is because some of the stationary area pixels will be included in the moving area block. Such pixels will lead to erroneous estimation. The second factor is a property of the least square technique; the more observations that are used in the computation the more accurate is the estimation of the parameters. Hence it is expected that larger block sizes should give higher reductions. However, the effect of more observations tails off after an optimum block size.

Using these two factors, it is possible to explain why the reduction at higher displacement is more than at small displacement. At a small displacement of say 1 pixel per frame the only way of obtaining an accurate segmentation

is to use a block size of less than 1 on a side. This is however not feasible with the implementation of algorithm PDDT2. Using a large block size will introduce misclassification of pixels, but this will be compensated by the provision of large observation data. The net effect is that for small displacement a poor performance is recorded. In the case of displacements of 4 and 5 pixels per frame, block size of 8x8 gave good reduction. This block size provided a good enough segmentation as well as enough data for computation. As a conclusion, the least square estimation formulation improved the accuracy of the basic algorithm (PDDT1). A disadvantage lies in the situation when the amount of motion is not known a priori, then choice of optimum segmentation and computation block size will be a problem. From the experiments carried out, it can be said that 8x8 is an optimum block size. The algorithm has been applied to the sequences "Walking man", "Box" and "Alan". The results of comparison with the performance of PDDT1 are tabulated in tables T3.7, T3.8 and T3.9 respectively. An alternative means of improvement was suggested in section 3.2.3: centering a block of pixels on each of the pixels within a moving area block and estimating a displacement for each pixel. In section 3.4 the technique (PDDT3) will be described and results of experiments presented.

Table T3.6

Frame difference variance in moving areas, using algorithm PDDT2 on "Box" for varying displacements DX and DY.

| BLOCK SIZE | COMPENSATED VARIANCE | UNCOMPENSATED VARIANCE |
|---|---|---|
| (I) DX= 1, DY= 1 | | |
| 2x4 | 15.03 | 17.99 |
| 4x4 | 14.97 | 18.00 |
| 8x8 | 13.54 | 18.01 |
| 8x16 | 13.64 | 18.01 |
| (II) DX= 2, DY= 2 | | |
| 2x4 | 30.37 | 31.91 |
| 4x4 | 31.60 | 32.00 |
| 8x8 | 30.31 | 31.12 |
| 8x16 | 30.00 | 32.01 |
| (III) DX= 3, DY= 3 | | |
| 2x4 | 40.08 | 45.21 |
| 4x4 | 39.55 | 45.26 |
| 8x8 | 41.46 | 45.58 |
| 8x16 | 40.54 | 45.79 |
| (IV) DX= 4, DY= 4 | | |
| 2x4 | 51.99 | 58.39 |
| 4x4 | 51.45 | 58.20 |
| 8x8 | 44.02 | 58.29 |
| 8x16 | 54.20 | 58.72 |

(V) DX= 5, DY= 5

| | | |
|---|---|---|
| 2x4 | 64.19 | 71.36 |
| 4x4 | 62.15 | 71.14 |
| 8x8 | 53.51 | 71.16 |
| 8x16 | 63.08 | 71.87 |

Table T3.7

Comparison of frame difference variance in moving areas, using algorithms PDDT1 and PDDT2.
Sequence: "Walking Man".
Block size: 8 X 16

| IMAGE PAIR. | PDDT1 | PDDT2 | UNCOMPENSATED VARIANCE. |
|---|---|---|---|
| 1 | 36.65 | 33.36 | 34.77 |
| 2 | 40.25 | 34.13 | 35.54 |
| 3 | 33.95 | 31.55 | 32.35 |
| 4 | 36.20 | 34.50 | 34.80 |
| 5 | 35.50 | 34.05 | 34.86 |
| 6 | 35.15 | 33.10 | 34.12 |
| 7 | 46.35 | 41.71 | 41.83 |
| 8 | 35.45 | 23.18 | 29.53 |
| 9 | 42.64 | 37.89 | 39.60 |
| 10 | 56.75 | 46.60 | 48.41 |

Table T3.8

Comparison of frame difference variance in moving areas, using algorithms PDDT1 and PDDT2.

Sequence: "Box".

Block size: 8 X 16

| IMAGE PAIR. | PDDT1 | PDDT2 | UNCOMPENSATED VARIANCE. |
|---|---|---|---|
| 1 | 19.00 | 17.27 | 18.49 |
| 2 | 33.67 | 29.76 | 32.15 |
| 3 | 41.86 | 34.46 | 39.26 |
| 4 | 41.18 | 36.33 | 39.91 |
| 5 | 10.55 | 9.16 | 10.50 |
| 6 | 40.85 | 38.57 | 39.91 |
| 7 | 49.36 | 45.87 | 46.27 |
| 8 | 82.82 | 72.55 | 78.45 |
| 9 | 53.72 | 50.19 | 52.18 |
| 10 | 70.03 | 60.90 | 64.81 |

Table T3.9

Comparison of frame difference variance in moving areas, using algorithms PDDT1 and PDDT2.
Sequence: "Alan".
Block size: 8 X 8

| IMAGE PAIR. | PDDT1 | PDDT2 | UNCOMPENSATED VARIANCE. |
|---|---|---|---|
| 1 | 1156.32 | 919.12 | 1054.54 |
| 2 | 938.62 | 822.63 | 874.75 |
| 3 | 668.46 | 458.52 | 556.11 |
| 4 | 877.92 | 723.63 | 772.84 |
| 5 | 714.28 | 557.70 | 652.22 |
| 6 | 299.51 | 256.72 | 264.71 |
| 7 | 675.63 | 511.60 | 627.40 |
| 8 | 249.12 | 207.80 | 223.45 |
| 9 | 254.71 | 201.74 | 223.45 |
| 10 | 662.60 | 573.74 | 601.40 |

## 3.4 ALGORITHM PDDT3 AND RESULTS OF EXPERIMENTS.

In the previous variation of the basic algorithm (PDDT2) each block of moving area pixels is assigned a displacement. In this variation, while the segmentation block gives moving area pixels, each pixel within the block is considered as having its own unique displacement. Thus at each pixel point in the moving area block we construct a "computation block" with the pixel under consideration at the center. The pixels within the "computation block" are used in computing the displacement of the center pixel. For each center pixel algorithm PDDT1 was applied to the pixels within the "computation block". This variation of the algorithm increases the parameters that can be manipulated to obtain a displacement estimate. In addition to the segmentation block

that can be chosen from a given set, it is possible to chose the "computation block" as well. The "computation block" can be chosen from the set of 3x3, 5x5, 7x7, 9x9 and 11x11. Higher block sizes have not been used because of their ineffectiveness.

Experiments to test the algorithm at varying displacements were performed using displacements of 1, 2, 3, 4 and 5 pixels per frame. At each displacement the segmentation block and the "computation block" are also varied. The results obtained are given in table T3.10. A comparison of table T3.6 and T3.10 reveals that on average a better performance is obtainable with PDDT3 than with PDDT2. This can only be attributed to the higher degree of flexibility in terms of choice of block size, that PDDT3 affords. The amount of reduction given are in most of the cases comparable. In an attempt to increase the flexibility obtainable with the least square estimation variation (PDDT2) a recursive least square algorithm PDDT4 has been developed. In this algorithm a starting block size is given and by adding more columns (or rows) an updated displacement estimate is obtained. In this way it is expected that only the columns (or rows) that actually belong to the moving area will be used in estimating the displacement. This technique is developed in section 3.5.

A few remarks about the results of table T3.10 are in order. For a given segmentation block size a "computation block" size of 3x3 gave the most reduction of frame difference variance, apart from the case of 1 pixel per frame. This case can only be explained by the smallness of the displacement. Furthermore, as the "computation block" size increases from 4x4 through to 8x8 there is a general decrease in the reduction of the frame difference variance. Recall that in the formulation of algorithm PDDT1 the assumption is that all the pixels are undergoing similar displacement, hence in applying PDDT1 to each of the pixel in

the segmentation block we implicitly make this assumption about the "computation block" pixels. This however is not the case. In using the technique on an image sequence where the displacement is not known a priori, it is recommended that a small size is an optimum choice of computational and segmentation blocks. In general this choice may be difficult to make as the image type in terms of detail and texture is likely to affect the performance of the algorithm.

Table T3.10

Frame difference variance in moving areas, using algorithm PDDT3 on "Box" for varying displacements DX and DY.

(I)

DX= 1, DY= 1.

Uncompensated variance= 17.99

Segmentation block size= 2x4

| COMPUTATION BLOCK | COMPENSATED VARIANCE. |
|---|---|
| 3x3 | 16.81 |
| 5x5 | 15.29 |
| 7x7 | 15.21 |
| 9x9 | 14.00 |
| 11x11 | 15.16 |

(II)
DX= 1, DY= 1
Uncompensated variance= 18.00
Segmentation block size= 4x4

| COMPUTATION BLOCK | COMPENSATED VARIANCE. |
|---|---|
| 3x3 | 16.85 |
| 5x5 | 15.69 |
| 7x7 | 15.99 |
| 9x9 | 14.60 |
| 11x11 | 15.68 |

(III)
DX= 1, DY= 1
Uncompensated variance= 18.01
Segmentation block size= 8x8

| COMPUTATION BLOCK | COMPENSATED VARIANCE. |
|---|---|
| 3x3 | 16.86 |
| 5x5 | 15.72 |
| 7x7 | 16.00 |
| 9x9 | 14.92 |
| 11x11 | 16.30 |

(IV)
DX= 2, DY= 2
Uncompensated variance= 31.91
Segmentation block size= 2x4

| COMPUTATION BLOCK | COMPENSATED VARIANCE. |
|---|---|
| 3x3 | 27.29 |
| 5x5 | 32.41 |
| 7x7 | 34.06 |
| 9x9 | 34.71 |
| 11x11 | 33.34 |

(V)

DX= 2, DY= 2

Uncompensated variance= 32.00

Segmentation block size= 4x4

| COMPUTATION BLOCK | COMPENSATED VARIANCE. |
|---|---|
| 3x3 | 27.55 |
| 5x5 | 33.59 |
| 7x7 | 36.53 |
| 9x9 | 36.54 |
| 11x11 | 35.65 |

(VI)

DX= 2, DY= 2

Uncompensated variance= 32.11

Segmentation block size= 8x8

| COMPUTATION BLOCK | COMPENSATED VARIANCE. |
|---|---|
| 3x3 | 27.79 |
| 5x5 | 33.92 |
| 7x7 | 36.76 |
| 9x9 | 37.67 |
| 11x11 | 38.32 |

(VII)

DX= 3, DY= 3

Uncompensated variance= 45.21

Segmentation block size= 2x4

| COMPUTATION BLOCK | COMPENSATED VARIANCE. |
|---|---|
| 3x3 | 37.55 |
| 5x5 | 39.40 |
| 7x7 | 42.08 |
| 9x9 | 44.10 |
| 11x11 | 42.32 |

(VIII)

DX= 3, DY= 3

Uncompensated variance= 45.26

Segmentation block size= 4x4

| COMPUTATION BLOCK | COMPENSATED VARIANCE. |
|---|---|
| 3x3 | 38.76 |
| 5x5 | 41.22 |
| 7x7 | 44.80 |
| 9x9 | 46.52 |
| 11x11 | 45.56 |

(IX)

DX= 3, DY= 3

Uncompensated variance= 45.58

Segmentation block size= 8x8

| COMPUTATION BLOCK | COMPENSATED VARIANCE. |
|---|---|
| 3x3 | 39.36 |
| 5x5 | 41.83 |
| 7x7 | 46.53 |
| 9x9 | 49.07 |
| 11x11 | 49.01 |

(X)

DX= 4, DY= 4

Uncompensated variance= 58.39

Segmentation block size= 2x4

| COMPUTATION BLOCK | COMPENSATED VARIANCE. |
|---|---|
| 3x3 | 49.19 |
| 5x5 | 51.79 |
| 7x7 | 52.21 |
| 9x9 | 56.65 |
| 11x11 | 50.64 |

(XI)

DX= 4, DY= 4

Uncompensated variance= 58.20

Segmentation block size= 4x4

| COMPUTATION BLOCK | COMPENSATED VARIANCE. |
|---|---|
| 3x3 | 48.96 |
| 5x5 | 52.08 |
| 7x7 | 53.84 |
| 9x9 | 58.14 |
| 11x11 | 53.06 |

(XII)

DX= 4, DY= 4

Uncompensated variance= 58.29

Segmentation block size= 8x8

| COMPUTATION BLOCK | COMPENSATED VARIANCE. |
|---|---|
| 3x3 | 49.93 |
| 5x5 | 53.69 |
| 7x7 | 56.36 |
| 9x9 | 61.37 |
| 11x11 | 56.54 |

(XIII)

DX= 5, DY= 5

Uncompensated variance= 71.36

Segmentation block size= 2x4

| COMPUTATION BLOCK | COMPENSATED VARIANCE. |
|---|---|
| 3x3 | 57.80 |
| 5x5 | 62.49 |
| 7x7 | 62.35 |
| 9x9 | 64.39 |
| 11x11 | 60.12 |

(XIV)

DX= 5, DY= 5

Uncompensated variance= 71.14

Segmentation block size= 4x4

| COMPUTATION BLOCK | COMPENSATED VARIANCE. |
|---|---|
| 3x3 | 58.04 |
| 5x5 | 62.66 |
| 7x7 | 63.55 |
| 9x9 | 66.20 |
| 11x11 | 62.70 |

(XV)

DX= 5, DY= 5

Uncompensated variance= 71.16

Segmentation block size= 8x8

| COMPUTATION BLOCK | COMPENSATED VARIANCE. |
|---|---|
| 3x3 | 59.81 |
| 5x5 | 65.49 |
| 7x7 | 66.66 |
| 9x9 | 68.98 |
| 11x11 | 66.48 |

## 3.5 RECURSIVE PROJECTION DOMAIN ESTIMATION (PDDT4).

The technique described in section 3.3 is based on many assumptions, among which is the accuracy of the segmentation into moving and stationary areas. The differential technique, unlike the feature based matching approach of Chapter 2, is very sensitive to improper segmentation. This is as a result of the underlying assumption in the Taylor´s series representation of the displaced image; every pixel in the previous frame is appearing in the subsequent frame with some displacement. If during segmentation some pixels which belong to the

stationary area are grouped as belonging to the moving area, this requirement will not be satisfied. Natural scenes do not always contain geometrical shapes like rectangles and squares, which form the basis of the segmentation schemes used in the simulations of Section 3.3. It is desirable to vary the block sizes used in the segmentation scheme and still maintain the advantage of simplicity and automatic setting of threshold. A recursive scheme has thus been developed and will be discussed in this section, to incorporate the aforementioned desirable feature.

The recursive least-square estimator does not give a different estimate from the "straight" method. However it affords the possibility of sequentially computing the motion estimate as a new measurement is taken, without having to perform a matrix inversion each time. In this we see the possibility of "growing" the moving area instead of using a fixed size of moving area block. The advantage is that a variable block size can be used through the image, depending on the progress of the recursion at each estimation. In algorithm PDDT2, a fixed size of block was designated as a moving area and the pixels within the block used in computing the displacement estimate. In the recursive scheme, which will be referred to as algorithm PDDT4, the segmentation scheme is used to designate an initial moving area block. The whole of the pixels in this block need not be used in computing the displacement estimate. Thus , given a block of moving area pixels of size r x c say, we proceed by taking a small block out of the given block and computing the motion estimate based on the pixels within the block. Next, either a row or a column is added to the block and an update of the motion estimate is obtained. The recursion can stop in two ways. Either when the addition of more rows or columns causes the new motion estimate to differ from a hitherto "stable" estimate by more than a threshold or when the initial block size has been used up. The recursion will be used in

computing the estimate along one direction, say, x-axis,and the resulting block will then be used in computing motion along the other direction (y-axis).

The recursive least square algorithm will be developed in what follows. Proceeding from equation (3.27), we use a subscript k to denote the estimate based on k measurements. Thus the estimate $D_k$ derived from k measurements is defined as,

$$D_k = [G^T_k G_k]^{-1} G^T_k F_k \qquad ....(3.28)$$

As already explained, equation (3.28) could refer to either the x- or y-motion estimate, since the method acts on rows (or columns) recursively to obtain the x-displacement (or y-displacement) and then uses the "straight" least-square estimation formula to obtain the y-displacement (or x-displacement). When the stopping criterion has been satisfied, the assumption is that the block created belongs to a homogeneous moving area. It remains only to obtain the block's y-displacement.

For a full rank matrix $[G_k]$, addition of more rows (new measurements) will not change its rank. Since the columns of $[G_k]$ are linearly independent (being full rank), then $[G^T_k G_k]$ is non-singular and its inverse will exist.

Let $[G^T_k G_k]^{-1}$ be denoted by $P_k$. We can then write (3.28) as

$$D_k = P_k G^T_k F_k \qquad ....(3.29)$$

Upon receiving a new measurement, the new estimate will be given by

$$D_{k+1} = P_{k+1}\, G^T_{k+1}\, F_{k+1} \qquad \ldots(3.30)$$

which may be written in partitioned form as

$$D_{k+1} = \rho_{k+1} \begin{bmatrix} G_k \\ \hline g^T(k+1) \end{bmatrix}^T \begin{bmatrix} F_k \\ \hline f(k+1) \end{bmatrix} \qquad \ldots.(3.31)$$

where,

$$g^T(k+1) = [XGX_{k+1} \quad XGY_{k+1}]$$

and

$$f(k+1) = FDX_{k+1}$$

Thus,

$$D_{k+1} = P_{k+1}[G_k{}^T F_k + g(k+1)f(k+1)] \qquad \ldots.(3.32)$$

Equation (3.29) can be written as

$$P_k{}^{-1}\, D_k = G_k{}^T F_k \qquad \ldots.(3.33)$$

Using this in (3.32) gives,

$$D_{k+1} = P_{k+1}P_k{}^{-1}D_k + D_k - D_k + P_{k+1}g(k+1)f(k+1)$$

where $(D_k - D_k)$ has been introduced to obtain

$$D_{k+1} = D_k + (P_{k+1}P_k{}^{-1} - I)D_k + P_{k+1}g(k+1)f(k+1) \qquad \ldots.(3.34)$$

I is the identity matrix.

Recall that

$$P_k = [G_k^T G_k]^{-1}$$

and thus

$$P_{k+1} = [G^T_{k+1}\, G_{k+1}]^{-1}$$

which could be written in partitioned form as

$$P_{k+1} = \begin{bmatrix} G_k \\ \hline g^T(k+1) \end{bmatrix}^T \begin{bmatrix} G_k \\ \hline g^T(k+1) \end{bmatrix}^{-1}$$

$$= [G^T_k G_k + g(k+1)g^T(k+1)]^{-1}$$

and

$$P_{k+1} = [P_k^{-1} + g(k+1)g^T(k+1)]^{-1} \qquad \ldots.(3.35)$$

since $P_k^{-1} = G^T_k G_k$

Equation (3.35) can be written as

$$P_{k+1}P_k^{-1} + P_{k+1}g(k+1)g^T(k+1) = I$$

and

$$P_{k+1}\, P_k^{-1} = I - P_{k+1}g(k+1)g^T(k+1) \qquad \ldots.(3.36)$$

Equation (3.34) can now be written as,

$$D_{k+1} = D_k + P_{k+1}g(k+1)[f(k+1) - g^T(k+1)D_k] \qquad \ldots.(3.37)$$

We now have an expression for computing the new estimate $D_{k+1}$ recursively. However $P_{k+1}$ must be put in a form that allows easy computation ( i.e no inversions ). Using the matrix inversion lemma [3] equation (3.35) can be written as

$$P_{k+1} = P_k - P_k g(k+1)[g^T(k+1)P_k g(k+1) + 1]^{-1}\, g^T(k+1)P_k$$

which is a recursive way of obtaining $P_{k+1}$. Since $g^T(k+1)P_k g(k+1) + 1$ is a scalar, we have

$$P_{k+1} = P_k - \frac{P_k g(k+1) g^T(k+1) P_k}{g^T(k+1) P_k g(k+1) + 1} \quad \ldots.(3.38)$$

Equations (3.37) and (3.38) will be used in algorithm PDDT4 to compute the motion estimate. The initial conditions, that is to say, $D_1$ and $P_1$ are obtainable once the observability condition (i.e nonsingularity of $G^T_k G_k$) is satisfied for some initial k. Thus, the initial $P_k$ is obtained by substitution into

$$P_k = [G^T_k G_k]^{-1}$$

for some k, for which $G^T_k G_k$ is nonsingular. The initial $D_k$ is obtained from

$$D_k = [G^T_k G_k]^{-1} G^T_k F_k$$

for the same k.
It should be noted that in this particular problem $[G^T_k G_k]$ is a 2 x 2 matrix and its inversion is a trivial matter, being only some simple algebraic manipulation.

### 3.5.2 Experiments, Results and Conclusion.

Algorithm PDDT4 has been tested on the image "Box" with displacements of 1, 2, 3, 4 and 5 pixels per frame. Results of two sets of experiments are presented. In the first we start the recursion using a block size of 8x8 and in the other a block size of 8x16. In both cases starting column sizes of 3, 4, and 5 are used. The results are tabulated in table T3.11.

Compared to algorithm PDDT2, there is not much improvement, however it is an improvement over the basic algorithm (PDDT1). In general it is dificult to ascribe a trend to the performance of the differential technique in the projection domain, because of the many parameters that could be varied. A delicate balance could however be obtained by using a block size of 8x8. A priori knowledge about the expected amount of motion in a particular application will be very helpful.

Table T3.11

Frame difference variance in moving areas, using algorithm PDDT4 on "Box" for varying displacements DX and DY.
Starting column size is given last in block size description.

(I)

DX= 1, DY= 1.

Uncompensated variance= 18.01

| BLOCK SIZE | COMPENSATED VARIANCE. |
|---|---|
| 8x8x3 | 19.17 |
| 8x8x4 | 14.81 |
| 8x8x5 | 13.25 |

(II)

DX= 2, DY= 2

Uncompensated variance= 32.12

| BLOCK SIZE | COMPENSATED VARIANCE. |
|---|---|
| 8x8x3 | 31.82 |
| 8x8x4 | 30.77 |
| 8x8x5 | 32.32 |

(III)
DX= 3, DY= 3
Uncompensated variance= 45.58

| BLOCK SIZE | COMPENSATED VARIANCE. |
|---|---|
| 8x8x3 | 43.85 |
| 8x8x4 | 41.30 |
| 8x8x5 | 42.37 |

(IV)
DX= 4, DY= 4
Uncompensated variance= 58.29

| BLOCK SIZE | COMPENSATED VARIANCE. |
|---|---|
| 8x8x3 | 51.00 |
| 8x8x4 | 52.82 |
| 8x8x5 | 46.74 |

(V)
DX= 5, DY= 5
Uncompensated variance= 71.16

| BLOCK SIZE | COMPENSATED VARIANCE. |
|---|---|
| 8x8x3 | 65.13 |
| 8x8x4 | 62.31 |
| 8x8x5 | 55.29 |

(VI)
DX= 1, DY= 1
Uncompensated variance= 18.01

| BLOCK SIZE | COMPENSATED VARIANCE. |
|---|---|
| 8x16x3 | 16.32 |
| 8x16x4 | 13.87 |
| 8x16x5 | 12.66 |

(VII)
DX= 2, DY= 2
Uncompensated variance= 32.01

| BLOCK SIZE | COMPENSATED VARIANCE. |
|---|---|
| 8x16x3 | 33.01 |
| 8x16x4 | 29.74 |
| 8x16x5 | 36.40 |

(VIII)
DX= 3, DY= 3
Uncompensated variance= 45.79

| BLOCK SIZE | COMPENSATED VARIANCE. |
|---|---|
| 8x16x3 | 43.86 |
| 8x16x4 | 45.15 |
| 8x16x5 | 46.97 |

(IX)

DX= 4, DY= 4

Uncompensated variance= 58.72

| BLOCK SIZE | COMPENSATED VARIANCE. |
|---|---|
| 8x16x3 | 57.32 |
| 8x16x4 | 55.43 |
| 8x16x5 | 54.37 |

(X)

DX= 5, DY= 5

Uncompensated variance= 71.87

| BLOCK SIZE | COMPENSATED VARIANCE. |
|---|---|
| 8x16x3 | 69.50 |
| 8x16x4 | 64.42 |
| 8x16x5 | 57.64 |

# CHAPTER 4

# APPLICATIONS OF MOTION ESTIMATION

## 4.1 INTRODUCTION.

Motion estimation schemes will in the end be judged by the improvement produced in the various application areas. Motion adaptivity has been found applicable to almost all image processing techniques such as interpolation, filtering and coding, when dealing with image sequences. The performance of such techniques is improved, especially in the moving areas of the image sequence because processing is along the trajectory of motion. The need for an accurate estimation of motion in the moving areas is emphasised when it is realised that the various techniques will be driven by the motion estimates.
In this chapter the motion estimation schemes developed in chapters two and three will be applied to some image sequence processing techniques, namely motion compensated frame interpolation, motion compensated filtering and motion compensated differential pulse code modulation (DPCM) coding. There is no doubt that the applications of motion estimation are numerous but these will suffice to portray the savings in terms of bit rate reduction of coding and improved intelligibity of reproduced image.

## 4.2 MOTION-COMPENSATED FRAME INTERPOLATION.

### 4.2.1 Introduction.

An attractive image sequence coding scheme is the concept of motion-compensated frame skipping/interpolation. A simple scheme is depicted in figure 4.1. Essentially the

scheme operates by dropping some frames within an image sequence and transmitting some "key" frames. At the receiver, where there will be the necessary information about the "key" frames, the missing frames are reconstructed using motion-compensated interpolation and extrapolation. The basic modules needed in the implementation of the scheme are (see figure 4.1) image segmenter and motion estimator, filter coefficient selector and the spatio-temporal interpolation filter.

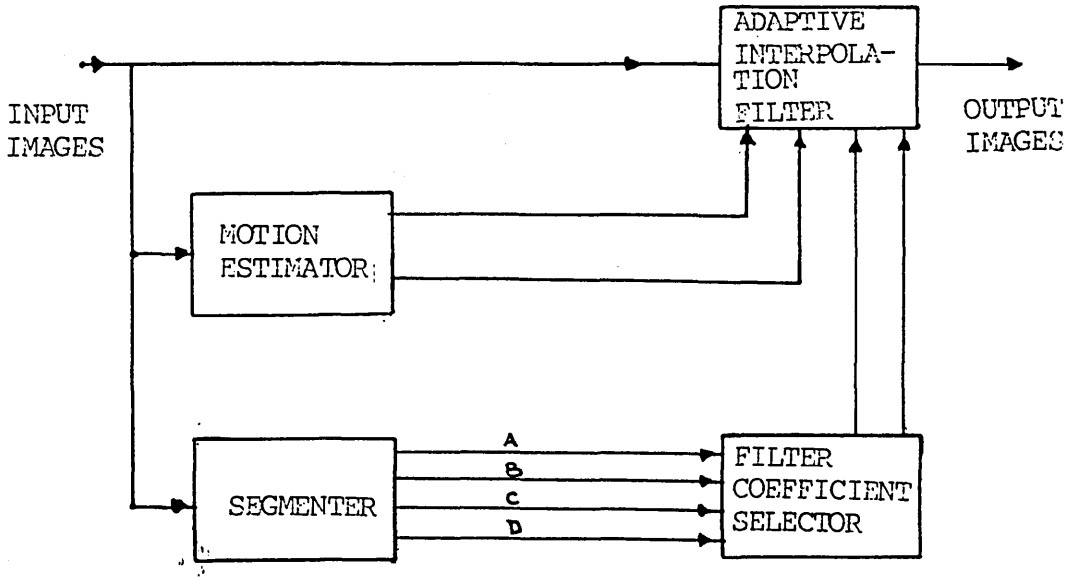



Figure 4.1: Block diagram of a motion-compensated frame interpolator.

When this scheme is combined with known coding techniques, tremendous bit rate reduction can be achieved, [7, 8, 20, 26, 31, 33, 46, 48, 49, 50, 53, 63]. Other techniques, such as frame repetition and pure linear interpolation, which attempt to achieve similar objectives are known to subject the displayed image sequence to some annoying degradation. Frame

repetition causes jerkiness in moving areas of images. The jerkiness is due to aliasing in the temporal axis frequency domain. In other words the input signal has significant power above the half sampling frequency [31]. Simple interpolation by temporal filtering introduces blurring in the moving areas. The degradation caused is usually proportional to the amount of interframe motion: being small for slow motion and severe for fast movements. If the interpolation is applied to the field, that is field skipping is performed rather than frame skipping, the degradation is less noticeable. This is due to the smallness of interfield motion compared to interframe motion. The price to be paid in this strategy is the reduction of vertical resolution in the stationary areas of the displayed image. Work that involves interpolation of fields or frames of images have been directed at videotelephone at 200Kbits/second [31], monochrome television at 1.5Mbits/second [33], videoconference [26, 46], and aerial image coding/transmission [8, 48, 49, 50].

Haskell et al [33] designed a motion-adaptive field interpolator, for use in an interframe coder, which is driven by a motion detector. The motion detection scheme consists of a simple logical comparison of adjacent pixels around a pixel to be interpolated. If the pixel is considered as belonging to a moving area, it is interpolated from a neighbourhood of pixels in previously reconstructed frame and current field. The scheme is shown in figure 4.2. The pixel marked X, belonging to a moving area is reconstructed by interpolation using pixels A, B, C and D. For a stationary area pixel the value is left unchanged.

Lippmann [48, 49] reported experiments with aerial scenes containing a single uniformly moving object. Estimating motion via a scene-content driven correlation scheme and taking the aerial imaging perspective into consideration a motion compensated interpolation scheme was implemented. In an experiment [49] a bit rate reduction

factor of 25 was reported. Application of the scheme to multiple objects will entail obtaining motion estimates for the various objects.

Bergmann [7] listed some requirements for an accurate implementation of motion compensated frame interpolation.

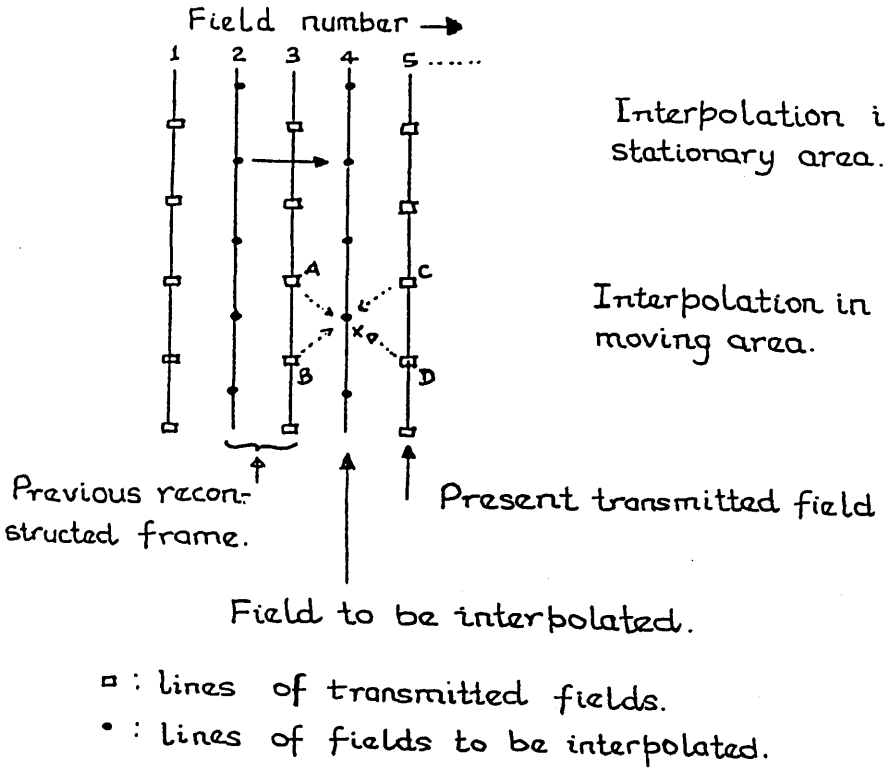



Figure 4.2: Motion-adaptive interpolation of Haskell et al.

They are as follows;

1. Segmentation of the transmitted frames into stationary areas, moving areas, uncovered background in the present frame and background to be covered in the next frame.

2. Improvement of displacement estimation algorithm to give accurate estimates for single pixel.

3. Development of appropriate interpolation filter and algorithm for adapting the coefficients.

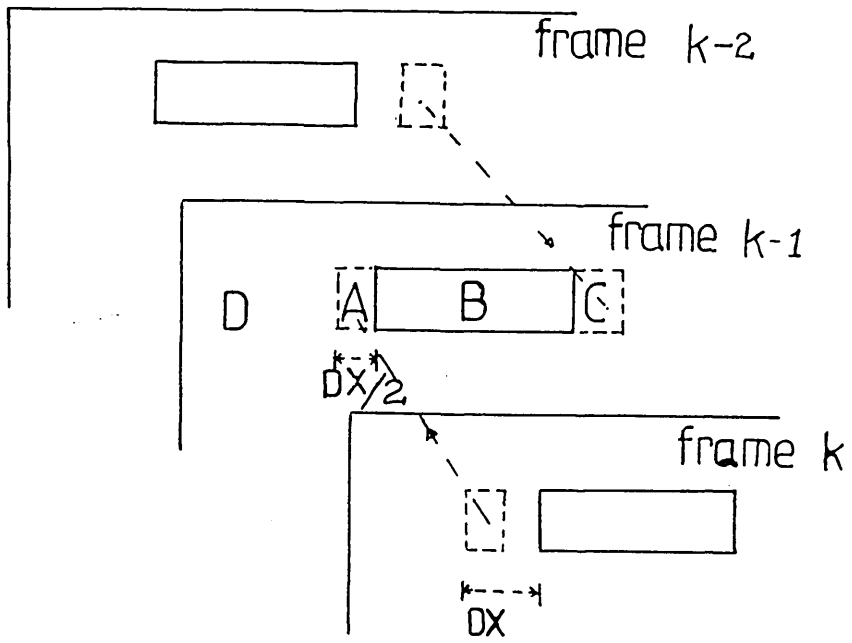
Lenz and Gerhard [46] developed an occlusion based image model in which the constituents of an image frame are assigned an occlusion priority level (OPL). A lower value of the OPL indicates that the object so labelled can be occluded by another with a higher OPL number. Using frame difference thresholding and region growing a mask was derived to isolate moving objects. In the final stage of the algorithm object features are extracted to track individual objects through the image sequence. The features extracted are object area, average greyvalue, centre of gravity, boundary line and occlusion priority level. In the work of Bergmann [7] the segmentation task was approached by computing the frame difference among three successive corresponding fields. Two logical variables were then obtained, to provide masks for the unchanged and changed areas, after thresholding and binary limiting. Using the logical combination, each pixel can be assigned to one of the four predefined image segments. Bergmann´s work is considered the most detailed motion adaptive frame interpolation scheme published up till 1985 [53].

### 4.2.2 Segmentation and Motion Estimation.

In this section we shall describe a new contribution to the solution of the segmentation problem in motion-compensated frame interpolation schemes. In the work of chapter 2, a segmentation scheme was developed to delineate the image frame into moving and stationary areas. In motion-compensated frame interpolation it is also necessary to further segment the moving area into background area to be covered and background area uncovered. This is essentially the problem of occlusion which arises as a result

of a moving object translating across a fixed background. An illustration of the areas to be segmented as given by Bergmann [7] is shown in figure 4.3. Segment A is an uncovered background area, and segment C is a covered background area. Segment D is the stationary background while segment B is the moving area proper.

The two "key" frames to be transmitted, k-2 and k, are buffered at the transmitter. A frame difference signal is generated from these two frames and the root-mean-square (r.m.s) value used as the segmentation threshold. This technique, as mentioned earlier (chapter 2) gives opportunity of adapting the segmentation threshold to variations in imaging conditions from frame to frame. During segmentation, the current frame (k) is partitioned into rectangular or square blocks and the r.m.s value of the frame difference (between frames k and k-2) within the block computed at each partitioned block. Blocks with r.m.s value less than the threshold are classified as stationary while those with greater r.m.s values are classified as moving areas. It should be noted that the areas classified as moving also include the decovered and covered background areas.




A: UNCOVERED BACKGROUND AREA.
B: MOVING IMAGE AREA.
C: BACKGROUND AREA TO BE COVERED.
D: STATIONARY BACKGROUND AREA.
DX: X-DISPLACEMENT OF IMAGE.

Figure 4.3:Motion-compensated frame interpolation. The areas to be segmented are marked.

In order to further segment the moving area into these occluded areas and moving area proper, a scheme based on the mean and variance of the pixel values of areas classified as stationary, was developed. In this segmentation scheme we are testing the partitioned blocks from left to right of the image frame. Let us assume that no part of the moving objects is at either edge of the frame. In which case some part of the background would be detected before reaching the object. A situation in which this assumption does not hold will not necessarily lead to a breakdown of the scheme.

During a segmentation pass, whenever a block is detected as corresponding to a stationary area, the mean and variance of the pixels from the two frames (k-2 and k) are

stored as reference description of the stationary background area. On detecting a moving area the corresponding blocks in the two frames are tested for classification as a decovered or covered background. The mean of the pixel values within the block in each of the frames k-2 and k are computed. If the mean falls within one standard deviation of the mean of the most recent stationary area, then the block is classified as a background. Otherwise it belongs to a moving image area. If for instance a back ground area was found in frame k and image area in frame k-2, then for this block we have a decovered background. A reverse situation will apply to a covered background. A moving area proper will correspond to the case in which blocks in both frames belong to image areas. The mean and variance of the stationary background stored, is updated each time a stationary area block is detected.

A situation in which some part of the object is occluded by the edge of the image frame would be resolved by a similar, but more time consuming scheme. On locating a moving area, the nearest stationary area is also located and used as the reference description of the background area. The scheme can then proceed from this point. A more rigorous approach to the problem of segmenting a moving area into decovered and covered background is the use of the Wilcoxon-Whitney-Mann test. In this case we assume that the stationary background area and the image area pixels have a normal distribution. Whenever a moving area block is detected we test the pixel values (from the blocks in frames k-2 and k) for membership of the same distribution as the background or image area pixels. This technique is involved computationally and also requires a look up table (memory requirement). Moreover, since segmentation is only a precursor to some more elaborate motion estimation algorithm it needs to be kept as simple as possible. Hardware implementation may not also be in favour of such elaborate

procedure.

Owing to the difficulty of accurate modelling of image signals, the differential technique of motion estimation does not perform as well as the feature-based matching technique. The results of experiments reported in chapters 2 and 3 corroborate this fact. After segmenting the image frame into the four segments (stationary, decovered, covered and moving areas) it is then necessary to estimate motion in the moving areas. In the experiments of this chapter we have employed the developed motion estimation algorithm (feature-based) of chapter two, with some modification in the segmentation. The segmentation must delienate four segments rather than two.

### 4.2.3 Interpolation Filter.

Without motion compensation the finite impulse response (FIR) linear interpolation filter output is given for image frames k-2, k-1 and k as,

$$I_{k-1}(x,y) = (1-\alpha) I_{k-2}(x,y) + \alpha I_k(x,y) \qquad .....(4.1)$$

where $I_{k-1}(x,y)$ represents pixel values of the eliminated frame k-1, $I_{k-2}(x,y)$ and $I_k(x,y)$ represent those of the transmitted frames k-2 and k respectively. $(1-\alpha)$ and $\alpha$ are coefficients of the interpolation filter. In a general case, if only every Nth frame is transmitted, then for interpolating the nth frame, where n= 1, 2, 3,...N-1, the filter coefficients have to be chosen according to the formula,

$$\alpha_n = (N-n)/N \qquad .....(4.2)$$

such that,

$$I_{k-n} = \alpha_n I_k + (1-\alpha_n) I_{k-N}$$

In the case of equation (4.1), where every second frame is transmitted, $N=2$, with only one missing frame,

$1-\alpha = 1/2$ and $\alpha = 1/2$

For the case of interpolation of 3 frames from 2 transmitted frames the coefficients will have values as follows;

interpolated frame n= 1 $1-\alpha = 3/4$ $\alpha = 1/4$.
" n= 2 $1-\alpha = 1/2$ $\alpha = 1/2$.
" n= 3 $1-\alpha = 1/4$ $\alpha = 3/4$.

Motion compensation allows the pixel values $I_{k-2}$ and $I_k$ to be chosen along the trajectory of motion of the objects. Hence, taking into consideration the fact that only motion estimates at frame k are available, the motion compensated filter output will be given as,

$$I_{k-1}(x,y) = (1-\alpha) I_{k-2}(x - DX/2, y - DY/2) + \alpha I_k(x + DX/2, y + DY/2) \quad .....(4.3)$$

where DX and DY are the motion estimates, in the x and y directions respectively, of the moving area pixels. For non-integral motion estimates, as would be obtained when using differential technique, a bilinear filter is employed. A neighbourhood of pixels will then be used in the interpolation. The basic assumption in equation (4.3) is that we can assign the computed motion at position (x,y) in frame k to the same pixel position in frame k-1 [7]. Furthermore a smooth linear motion assumption allows the placement of corresponding pixels in frames k-2 and k at (x- DX/2, y- DY/2) and (x+ DX/2, y+ DY/2) respectively. In order to generate more than one image in between transmitted frames the motion estimate has to be weighted in proportion to the position of the frame to be reconstructed.

From the segmentation and motion estimation

information the interpolation filter could be appropriately applied for reconstruction of missing frames. Referring to figure 4.3, the segmented areas A, B, C, D will be reconstructed as follows;

Segment B:

This is the moving object area and will be reconstructed using the motion estimates in equation (4.3) with the appropriate filter coefficients. The filter coefficients, depending on the number of skipped frames, will be chosen according to the formula given in equation (4.2).

Segment D:

This is a stationary background area and will be reconstructed by substituting a zero displacement in equation (4.3). Thus equation (4.3) reduces to simple linear interpolation filter of equation (4.1).

Segments A and C:

These are the areas to be covered (C) and decovered (A). In these areas an appropriate extrapolation is all that is needed. For the segment C, the coefficients are assigned values as follows,

$1-\alpha = 1$ and $\alpha = 0$ in equation (4.1).

That is,

$$I_{k-1}(x,y) = (1-\alpha) I_{k-2}(x,y) = I_{k-2}(x,y)$$

Similarly for segment A,

$$I_{k-1}(x,y) = \alpha I_k(x,y) = I_k(x,y)$$

with $1-\alpha = 0$ and $\alpha = 1$.

4.2.4 Experiments, Results and Conclusion.

Two pairs of images from the sequence "Alan" are used in testing the developed algorithm. "Alan" consists of images of 240 noninterlaced lines with 350 pixels per line. Other details concerning the acquisition of the images are as given in chapter 2.

Experiments with "Alan" consists of passing a pair of images from the sequence, through a "software system" developed for the algorithm (see appended programs). The "software system" consists essentially of the folllowing modules,

1. Segmentation module.
2. Motion estimation module.
3. Interpolation Coefficient selector.
4. Interpolation Filter module.

The segmentation module implements the segmentation algorithm described in section 4.2.2. The output being classification of the image frame into one of the four segment types of figure 4.3. Motion estimation is performed in the moving areas using the feature based scheme described in chapter 2. The feature vector size is two. Depending upon the number of interpolated images required, the interpolation coefficients are chosen as below. For a given image pair we have reconstructed two and three images respectively. This strategy has been used because a sequence like "Alan" does not conform to the standard 25 frames per second of normal television frames. The coefficients are thus chosen from situations;

N= 2 n= 1 $a_{k-2}$= 1/2 $a_k$= 1/2.

N= 3 n= 1 $a_{k-2}$= 2/3 $a_k$= 1/3.
n= 2 $a_{k-2}$= 1/3 $a_k$= 2/3.

N= 4 n= 1 $a_{k-2}$= 1/4 $a_k$= 3/4.
n= 2 $a_{k-2}$= 1/2 $a_k$= 1/2.
n= 3 $a_{k-2}$= 3/4 $a_k$= 1/4.

With these selections, reconstruction using motion adaptive interpolation and extrapolation are performed on image pairs from the sequence. Results from two image pairs from sequence "Alan" are shown together with uncompensated interpolation versions. See figures 4.4 through 4.9. Figure 4.4 and 4.5 show the case of N= 4 on one image pair. Figure 4.4 is the motion compensated case and 4.5 the uncompensated case. Figures 4.6, 4.7 show results with N= 4 on the second image pair. Figure 4.6 shows the motion compensated case while 4.7 shows the uncompensated case. Figures 4.8 and 4.9 show the compensated and the uncompensated cases for N=3.

The effect of motion compensation can be seen around the moving areas of the image. Comparison of the sequence in figure 4.4 with that in figure 4.5 will show the benefit of motion compensation. As more frames are required to be interpolated the effect of motion becomes evident. In the moving areas, motion compensation helped the intelligibilty of the image details. The same argument holds for the other sequences.



KEY FRAME 1



INTERPOLATED
FRAME n=1



INTERPOLATED
FRAME n=2



INTERPOLATED
FRAME n=3



KEY FRAME 2

Figure 4.4: Motion Compensated
Frame Interpolation.



KEY FRAME 1



INTERPOLATED
FRAME n=1



INTERPOLATED
FRAME n=2



INTERPOLATED
FRAME n=3



KEY FRAME 2

Figure 4.5: Uncompensated Frame Interpolation.



KEY FRAME 1



INTERPOLATED
FRAME n=1



INTERPOLATED
FRAME n=2



INTERPOLATED
FRAME n=3



KEY FRAME 2

Figure 4.6: Motion Compensated
Frame Interpolation.



KEY FRAME 1



INTERPOLATED
FRAME n=1



INTERPOLATED
FRAME n=2

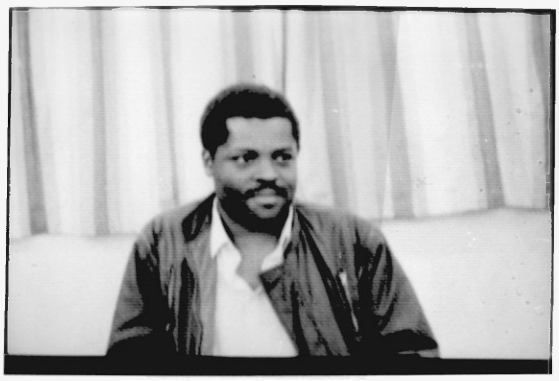


INTERPOLATED
FRAME n=3



KEY FRAME 2

Figure 4.7: Uncompensated Frame Interpolation.



KEY FRAME 1



INTERPOLATED
FRAME n=1



INTERPOLATED
FRAME n=2



KEY FRAME 2

Figure 4.8: Motion Compensated
Frame Interpolation.



KEY FRAME 1



INTERPOLATED
FRAME n=1



INTERPOLATED
FRAME n=2



KEY FRAME 2

Figure 4.9: Uncompensated Frame Interpolation.

## 4.3 MOTION-COMPENSATED TEMPORAL FILTERING.

### 4.3.1 Introduction.

In this section, experiments in motion-compensated temporal filtering using the motion estimation algorithms developed are reported. Due to imaging and transmission errors, image sequences acquired by visible, infrared and other scanners could be corrupted by random and burst noise including line dropout. An example of a case in which this situation arises is satellite imagery, in which images are acquired for the purpose of detecting or recognising a desired object. Random noise may make the recognition task difficult, given the land resolution of satellite images. In automatic tracking of objects through an image sequence, the task is facilitated by preprocessing, (e.g filtering), the image sequence. Other examples of areas of application in which one form of filtering or another will be needed are biomedical imagery, photo-reconnaisance images, forward looking infrared radar (FLIR), side looking radar (SLR) and multispectral scene mapping. In general the problem is that of enhancement or restoration of a noise corrupted image into a desirable one. The index of acceptability will depend on the use for which the final image is meant.

The theories of one- and two-dimensional digital filters have been developed and documented in the literature [36, 65]. However, while one-dimensional digital filters are relatively well developed, their extension to two dimensions are difficult; dimensionality and polynomial factorisation are some problems. The idea of processing an image sequence adds a third dimension to the problem of filtering; the temporal axis [37]. For image sequences, the motion taking place in the recorded scene is captured in this dimension. Hence any meaningful processing, involving this dimension must take into consideration the motion from one image frame

to another. This then necessitates the estimation of motion from frame to frame, for use in motion-compensated filtering.

The approaches to image filtering can be categorised as either frequency or pixel domain processing. In frequency domain processing the filter characteristics (magnitude, phase and cutoff frequency) are specified in forms amenable to modification of the Fourier transform of the image to achieve a desired goal. In pixel domain techniques, there is a direct manipulation of the pixels of the image. The application of these approaches to image sequences will entail three dimensional filters; spatio-temporal filters. The consideration of this section can only be seen from the point of view of pixel based processing, here restricted to the temporal axis alone. It should be stressed that, to achieve a full spatio-temporal filtering, one will have to do filtering in the x-y coordinates of the image plane followed by a temporal filtering. We shall be looking at temporal averaging and median filtering in the temporal domain.

### 4.3.2 Temporal filtering.

Let $I_k(x,y)$, denote the pixel value at coordinate $(x,y)$ in the kth frame of the noisy image sequence, and $I'_k(x,y)$ the corresponding pixel value in the filtered image sequence. The idea of temporal filtering is, to select pixels (temporally) from a stack of image frames and use such pixels in deriving the output of a filter function. The function could represent a recursive or nonrecursive filter.

Assuming we are using pixels from a stack of 2n+1 frames (n= 1, 2, 3,..) in computing the filtered pixel value, a nonrecusive temporal domain filter will be given as,

$$I'_k(x,y) = F\{I_{k-n}(x,y), I_{k-n+1}, \ldots I_k(x,y), \ldots I_{k+n}(x,y)\} \quad \ldots.(4.7)$$

The recursive, temporal domain filter is given as,

$$I'_k(x,y) = F\{I_{k-n}(x,y), I_{k-n+1}(x,y), \ldots I_k(x,y), \ldots I_{k+n}(x,y); I'_{k-m}(x,y), I'_{k-m+1}(x,y), \ldots \ldots, \ldots I'_{k-1}(x,y)\} \quad \ldots.(4.8)$$

F, represents the filter function. In equation (4.8) m is chosen to indicate the number of filtered pixels being put back into the computation. For a linear time-invariant filter the nonrecursive function of equation (4.7) could be written as,

$$I'_k(x,y) = \sum_{p=-n}^{n} a_p I_{k+p}(x,y) \quad \ldots.(4.9)$$

where $a_p$'s are the filter coefficients. When they are chosen to be equal we have a mean (averaging) filter. The function in equation (4.7) could have been chosen such that the median of a set of pixels is computed as the filtered output. In that case we have,

$$I'_k(x,y) = \text{Median}\{I_{k-n}(x,y), .., I_k(x,y), \ldots I_{k+n}(x,y)\} \quad \ldots.(4.10)$$

For the recursive case equation (4.9) becomes,

$$I'_k(x,y) = \sum_{p=-n}^{n} a_p I_{k+p}(x,y) + \sum_{q=-m}^{-1} b_q I'_{k+q}(x,y) \qquad \ldots.(4.11)$$

where $a_p$'s and $b_q$'s are the filter coefficients.

The equivalent median filter is given as,

$$I'_k(x,y) = \text{Median}\{I'_{k-m}(x,y),..,I'_{k-1}(x,y),I_k(x,y)\} \qquad \ldots.(4.12)$$

The foregoing considerations are for the straight temporal filtering. When the motion from frame to frame is taken into consideration, the pixel values to be used in equations (4.9) to (4.12) must be adaptively chosen. In the moving areas the pixel values do not correspond in a straight temporal sense, there is a displacement of the pixels over the frame stack. The displacement has to be estimated and used in motion adaptivity. As an example, let us consider the linear time invariant filter of equation (4.9), the motion compensated expression is given as,

$$I'_k(x,y) = \sum_{p=-n}^{n} a_p \; I_{k+p}(x+ DX_p \; , \; y+ \; DY_p) \qquad \ldots.(4.13)$$

where $DX_p$'s and $DY_p$'s are the x and y displacements of the pixel at coordinate (x,y) (in the kth frame) over the 2n+1 frames to be used in the filter computation. The pixel at coordinate (x,y) has to be tracked over the whole 2n+1 frames. In order to do this we apply the motion estimation algorithms developed in earlier chapters.

### 4.3.3 Experiment, Results and conclusion.

Two temporal filtering experiments will be reported.
(1) Nonrecursive linear time invariant mean filtering, using the feature based and the differential technique motion estimators for compensation.
(2) Nonrecursive motion compensated median filtering using the two motion estimators.

The test sequence "Box" will be used. Furthermore in the experiments 2n+1 will be set to 3. In which case we only need to track the pixel points over 2 image frames. Noisy test sequence were generated by adding normally distributed random numbers with zero mean to the images. In order to characterize the noise, the signal-to-noise-ratio in dB between the original and noisy images were computed. Hence the noise added to the various images in the sequence will be quantified in terms of the signal-to-noise-ratios; a large value indicating less noise than a small value. Also in measuring the performance of the schemes, the signal-to-noise-ratios after temporal filtering will be computed and compared to the original and the uncompensated filtering cases. We have done filtering in the temporal domain alone and the added noise will manifest in both temporal and spatial domains, hence the results tabulated show the effect of filtering on the temporal component of the noise. Spatial filtering will still be necessary to clean all the noise added to the images. Results are presented in tables 4.1 and 4.2.

Table T4.1

Comparison of signal-to-noise-ratios (SNR in dB) of temporally filtered images, using feature based motion estimator.

Sequence: "Box".

Block size:2x2.

Filter: Mean filter.

| FRAME NUMBER | INITIAL SNR | COMPENSATED SNR | UNCOMPENSATED SNR |
|---|---|---|---|
| 2 | 26.81 | 28.18 | 27.87 |
| 3 | 29.30 | 25.54 | 24.97 |
| 4 | 24.88 | 25.85 | 24.90 |
| 5 | 23.33 | 25.29 | 25.39 |
| 6 | 24.92 | 24.61 | 24.43 |
| 7 | 29.33 | 27.04 | 26.00 |
| 8 | 26.81 | 27.00 | 24.20 |
| 9 | 29.33 | 26.81 | 24.46 |
| 10 | 23.30 | 25.74 | 24.18 |

Table T4.2

Comparison of signal-to-noise-ratios (SNR in dB) of temporally filtered images, using feature based motion estimator.

Sequence: "Box".

Block size:2x2.

Filter: Median filter.

| FRAME NUMBER | INITIAL SNR | COMPENSATED SNR | UNCOMPENSATED SNR |
|---|---|---|---|
| 2 | 26.81 | 16.95 | 17.00 |
| 3 | 29.30 | 23.03 | 22.86 |
| 4 | 24.88 | 22.72 | 22.03 |
| 5 | 23.33 | 16.72 | 16.71 |
| 6 | 24.92 | 23.56 | 23.78 |
| 7 | 29.33 | 23.49 | 23.94 |
| 8 | 26.81 | 17.94 | 17.23 |
| 9 | 29.33 | 21.05 | 20.92 |
| 10 | 23.30 | 17.73 | 17.07 |

The entries in tables 4.1 and 4.2 respectively show the effect of motion compensation on temporal mean and median filtering. The "initial SNR" entries quantify the amount of noise added to the original image and the other entries are the compensated and uncompensated SNR´s. The mean filter performs better than the median filter. It can be noted that it is not in all the cases that the compensated mean filter has outperformed the uncompensated case. This can be attributed to the sensitivity to noise of the motion estimation algorithm. The median filter´s performance is very poor and in cases where motion compensation gave some improvement, it is very slight. In general some gains can be made using motion compensated filters for image sequences

instead of straight temporal filters. The problem with filtering is that there is usually a maximum noise that can be added to an image with any hope of faithful reconstruction. It is only in cases of very slight noise that an accurate reconstrution can be made.

## 4.4 MOTION-COMPENSATED INTERFRAME DPCM CODING.

### 4.4.1 Introduction.

Image sequences derived from various sources (e.g radar, infrared, acoustic, television, x-ray etc.) require an enormous amount of data bits for the purpose of storage or transmission. For instance the NTSC (National Television System Committee) colour signal sampled at 10.7MHz and linearly quantized to 256 levels gives a data rate of 85.6Mb/second. In high resolution satellite image sequence, 1024 pixels for each of 1024 lines in a frame, and digitized at 8 bits/pixel say, the amount of storage capacity needed for a sequence acquired over a period of 48 hours is of the order of 800Mbits. However it is well known [43] that television signals manifest a lot of redundancies and this is also true of image sequences derived from other sources. In an image sequence there are redundancies in both the spatial and temporal domains. These are due to the spatial and temporal correlation which exist between and among the pixels in an image frame.

It is the natural presence of these redundancies that has made coding of images a possibility. Intraframe coders exploit the spatial correlation while interframe coders exploit both spatial and temporal redundancies. Generalizing, the interframe coding of image sequences exploits both image domain redundancies and the human user's psychovisual deficiencies in achieving a data rate or memory requirement reduction. For example a great amount of coding degradation can be allowed in those areas of the picture where human visual sensitivity is low and accurate reproduction in the areas where human visual sensitivity is high. After a change in a scene the human eye takes some time to appreciate full picture resolution. This property can be exploited in coding schemes.

The problem of coding can be stated as that of data rate or memory requirement reduction subject to some fidelity constraints. It is thus a trade off between data compression and intelligibility of the reproduced (decoded) images. This makes the coding problem a task oriented endeavour, because various applications will be tolerant to different amounts and kinds of degradation. Degradations allowable in video teleconferencing are not allowable in broadcast television transmission applications. Subjecting x-ray images to the same kind of degradation as satellite images may render the output x-ray images useless.

The rate distortion theory of Shannon has placed the problem of source coding on a mathematical basis. The rate distortion function, R(D), of a discrete source, which measures the amount of information needed to specify the output with certain distortion, D, is defined as the expectation of fidelity loss. For a source of known probability distribution the expectation is easily obtained. However the probability distribution of image signals are seldom known a priori, hence an assumed distribution is usually used. The rate distortion theory provides a bound on the attainable performance of coding schemes, and is very useful in evaluating the performance of such schemes. At zero distortion, the rate approaches that of the source. The practical usage of the theory has been hampered by the fact that image signal statistics are not precisely known. Furthermore it is difficult to find mathematically tractable objective fidelity criteria which also have an acceptable subjective meaning.

In this section we are considering motion-compensated interframe predictive coding systems and emphasis is on the use of motion estimation algorithms developed earlier, in achieving some data compression.

### 4.4.2 Interframe Predictive Coding.

In a typical interframe predictive coding of an image sequence, every pixel value in a present frame under consideration is predicted from a combination of some pixels in a previous frame. The aim is to produce prediction errors (that is, the difference between the actual pixel value and the predicted value), whose variance is smaller compared to the variance of the original pixel values. The prediction errors are quantized, coded and transmitted (or stored). Since the prediction errors have a smaller variance compared to the original image pixel values, it is possible to roughly quantize them and thereby achieve a reduction in the required bit rate. If the coding is for reduced storage capacity, information about the prediction scheme is stored along with the compressed image for subsequent reconstruction. On the other hand if the coding is for reduced transmission bit rate, a similar prediction scheme is used at the receiver to produce the predicted value. Addition of the decoded received prediction error yields the original pixel value. The most common form of predictive coding is differential pulse code modulation (DPCM) and the block diagram is given in figure 4.10.

From the block diagram of the DPCM codec shown in figure 4.10, it is apparent that in the absence of transmission errors, the only error introduced in the system is the quantization error. If a high performance predictor is used, thus decreasing the variance of the prediction error, then for a fixed number of quantization levels, the variance of the quantization error can be reduced, leading to an improved fidelity of reconstruction. For a given fidelity criterion, reduction of prediction error variance will lead to a reduction in transmission rate.

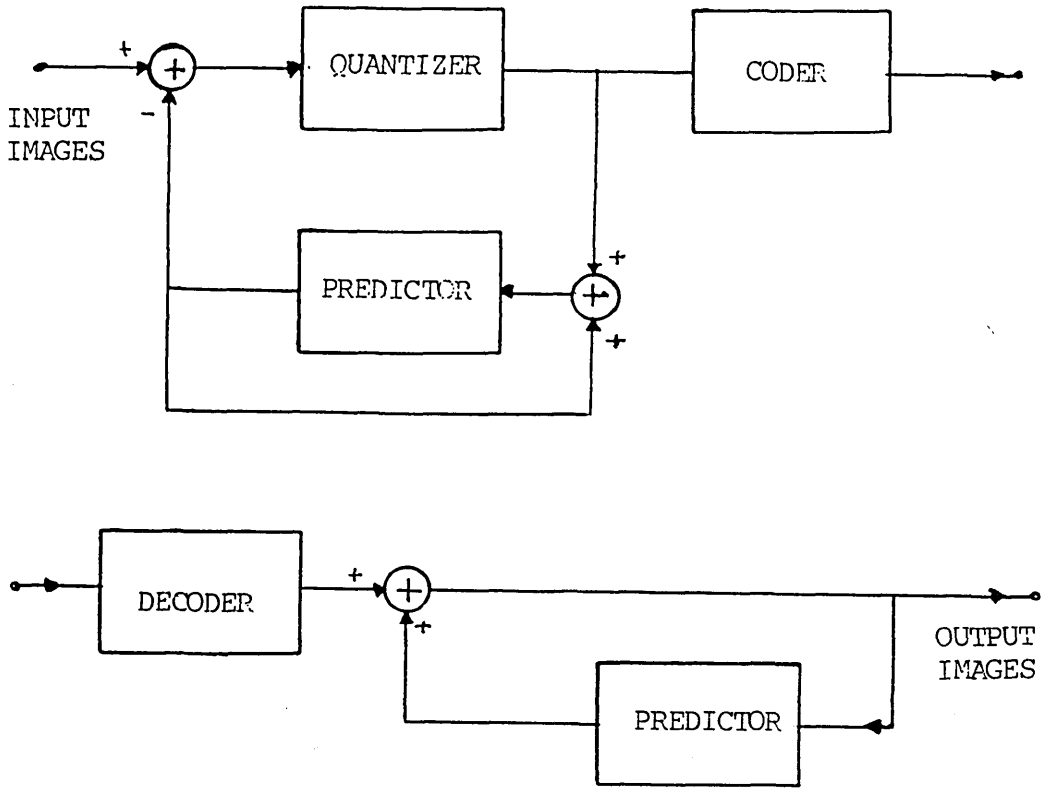



Figure 4.10: Differential Pulse Code Modulation (DPCM) Codec System.

A DPCM system can be broken down for design purpose into (a) Predictor (b) Quantizer (c) Coder and due consideration given to each subsystem in the light of the application at hand. The work reported here is geared towards the design of efficient predictors. The optimal system is obtained by optimizing the predictor and the quantizer given the error criterion. Theoretically the predictor and the quantizer should not be optimized separately since the quantization error depends on the probability distribution of the prediction error. However in the case of a mean-square error criterion the mean-square value of the quantizing noise is approximately proportional to the mean-square value of the prediction error [62]. Under this approximation, it is possible to optimize the prediction error and the quantizer independently. In our case simple previous frame pixel

prediction is employed with motion compensation in moving areas. This scheme allows a full appraisal of the benefit of motion compensation.

### 4.4.3 The Predictor.

There are a lot of alternatives open, when designing the predictor. Prediction could be done in the transform domain or in the image domain [22]. When it is done in the transform domain, a unitary transform is performed on the image and prediction done on the transform coefficients. The alternative is to use a set of pixels in the image domain for the prediction. Predictors could further be classified as either fixed or adaptive. In a fixed predictor some pixels in the previous frame which bear a fixed spatial relationship to the pixel being predicted are used for prediction. A set of such predictors, which are in common use is given in [18], and irrespective of interframe changes, the same set of pixels are used. However, in an image sequence there is the possibility of interframe motion, which will render fixed predictors inefficient. For example for two image frames in which there is relative motion of the objects, using a previous frame pixel prediction in a newly covered area will yield a high prediction error. This is due to the fact that the pixel being used for prediction has no statistical relationship with the pixel being predicted. In such cases where motion could be detected and measured, it is better to compensate for the motion by doing prediction along the trajectory of motion. Such predictors which compensate for the interframe motion are called motion-compensated predictors. Similar concept has been implemented in [18, 22, 30, 32, 39, 42, 53, 59, 61, 73].

It is also possible to design predictors that achieve adaptivity by using such image characteristics as edges and contour direction [67, 87]. Richard et al. [67]

extracted edges and used their derived characteristics to select pixels used in an adaptive DPCM coding of T.V pictures. Zschunke [87] derived information about contour directions from a neighbourhood of pixels and used such information in selecting a suitable prediction value. The predictors of interest in our discussion are the motion-compensated predictors. Their use in a motion-compensated interframe DPCM coding scheme will be highlighted. If these predictors will be classified, it is the method of motion estimation which will differentiate them. There are the pel (pixel)-recursive motion estimation techniques [10, 59, 60, 74, 81, 82] in which no information about the motion estimates need be sent. Matching based schemes (see [39] and [73]) require additional information about the motion estimates to be transmitted. However, the latter kinds of scheme give better performance as evidenced in the discussions of chapters 2 and 3. We shall be implementing a simple motion-compensated interframe DPCM codec in which the predictor is driven by a motion estimator. Figure 4.11 shows such a scheme. In this scheme, the image is segmented into stationary and moving areas. In the stationary areas the best predictor (for a perfect image) is the corresponding previous frame pixels, while in the moving areas the prediction pixel value is adaptively selected depending on the motion estimates. Hence for the moving areas the motion vectors must be derived.

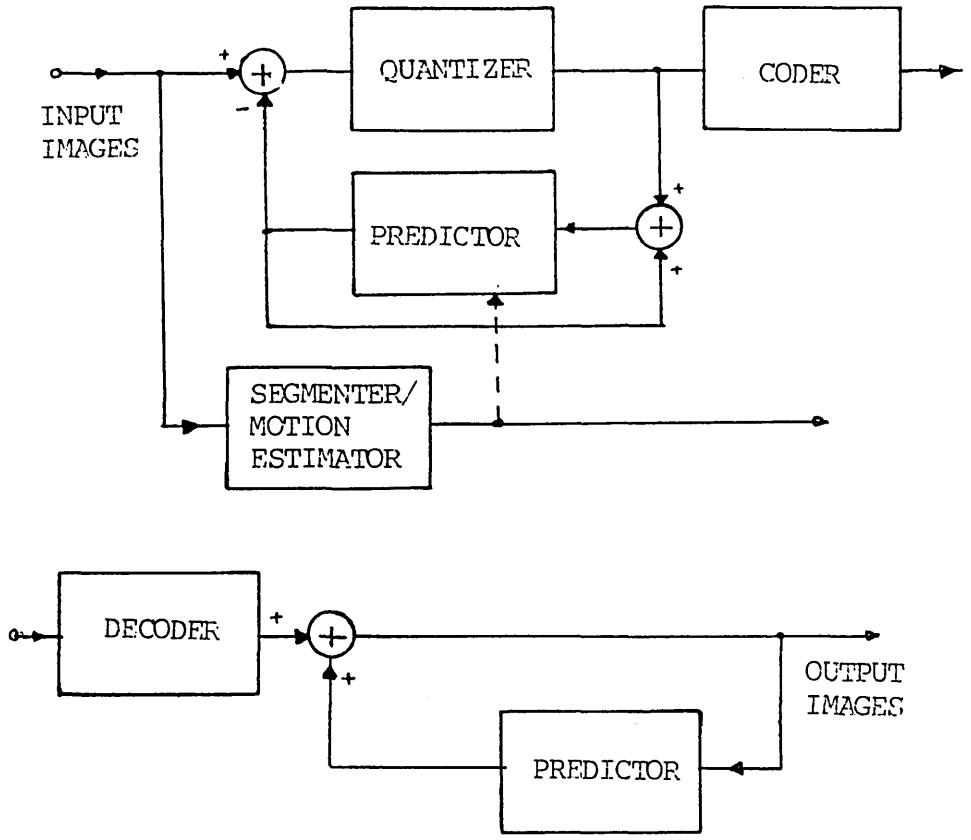



Figure 4.11: Motion-Compensated DPCM Codec.

4.4.4 The Quantizer.

Having obtained the prediction errors for each block of pixels, stationary or moving, the next step is the quantization of the error signals. Quantizers could be either fixed or adaptive, but the general rule of design is the minimization of image distortion subject to a transmission rate constraint. The most widely used distortion criterion is the mean-square error, $\langle (Q(x) - x)^2 \rangle$, with $Q(x)$ being the quantizer output and $x$ the input prediction error signal. A typical quantizer input-output characteristic is shown in figure 4.12.

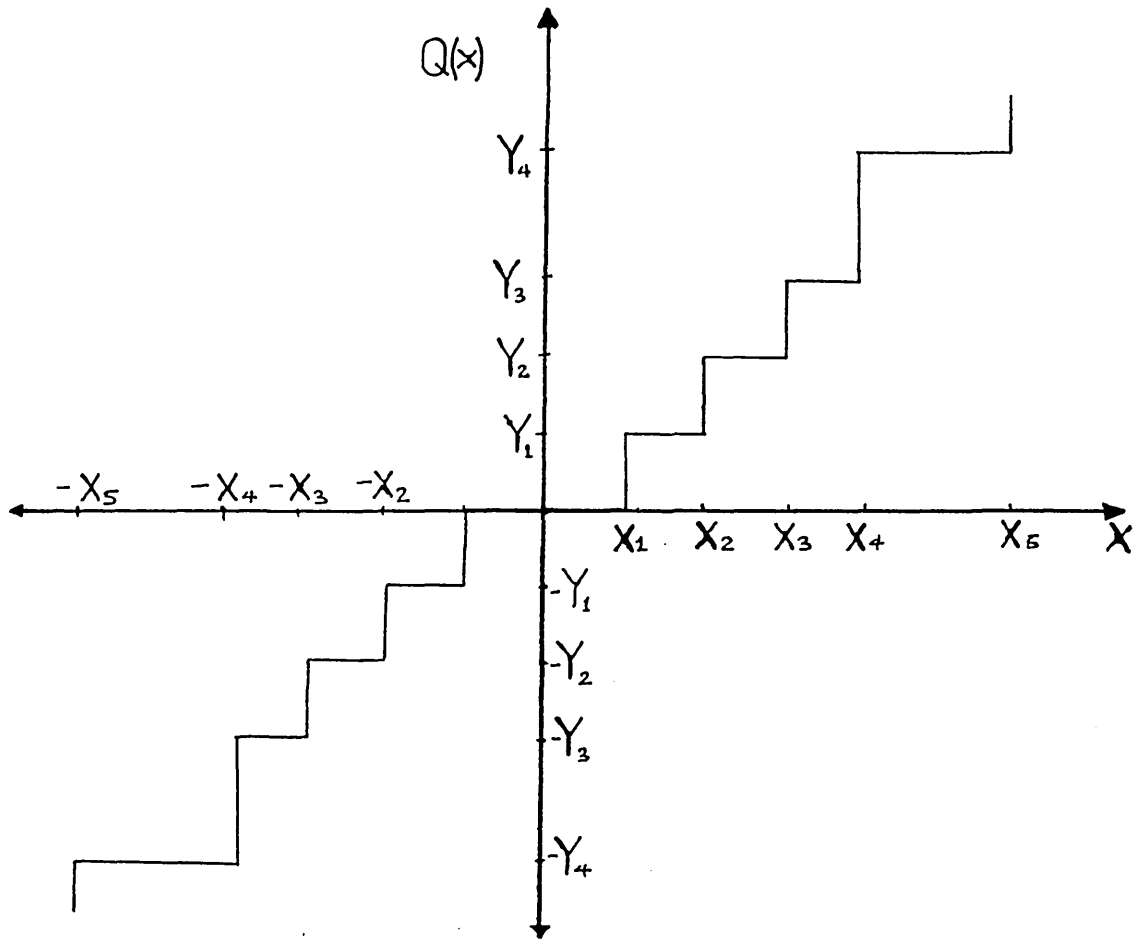



Figure 4.12: Quantizer Characteristics.

The output $Y_i$, referred to as representative level, take on a discrete set of values such that,

$$Q(x) = Y_i \text{ where } x_i < x <= x_{i+1}$$

The $x_i$'s are referred to as the decision levels. For a fixed number of quantizer levels Max [52] has shown that the optimal quantizer is specified by,

$$x_i = (Y_{i-1} + Y_i)/2$$

and

$$Y_i = \int_{x_i}^{x_{i+1}} x\, p(x)dx \Big/ \int_{x_i}^{x_{i+1}} p(x)dx$$

p(x) is the probability density of the input error signal. Quantizers which adapt to statistical and or psychovisual characteristics of images give improvement over the fixed quantizers.

In the experiments carried out we have implemented

the quantizer given by Netravali and Robbins [59]. The use of this quantizer gave good picture quality. The positive side of the 25-level symmetric quantizer (they actually used a 35-level symmetric quantizer) is shown in table T4.3.

Table 4.3: Quantizer for DPCM Coding Experiment. Only positive side of the symmetric quantizer is shown.

| | |
|---|---|
| DECISION LEVELS | 1,7,13,21,30,39,49,60,71,82,95,108,255 |
| REPRESENTATIVE LEVELS. | 0,4,10,17,26,35,44,55,66,77,87,102,182 |

No code assingment has been done, since the effect of motion adaptive prediction can be known without coding the quantized prediction error. For reconstruction, we only need to add the quantized prediction error to the prediction produced at the receiver.

4.4.5 Experiments, Results and Conclusion.

The simulations carried out were simply to implement the modules of the motion compensated DPCM codec. The segmentation into stationary and moving areas was done using the method developed in earlier chapters. Thus we have blocks of moving and stationary areas whose pixels are predicted, depending on the classification. For the stationary blocks, the corresponding block of pixels in the previous frame is used in generating the prediction error. For a moving area block, we apply a motion estimation algorithm to the block, obtain displacement estimates which are used in adapting the predictor. Thus, in the stationary areas the prediction error is given as ,

$$e = I_t(x,y) - I_{t-1}(x,y)$$

while in the moving areas the prediction error is given as,

$$e = I_t(x,y) - I_{t-1}(x-DX,\ y-DY)$$

where e, denotes prediction error, $I_t(x,y)$ is the pixel value in a current frame, $I_{t-1}(x,y)$ is the corresponding pixel value in a previous frame. DX and DY are the displacement estimates in the x and y directions. The method of applying nonintegral estimates given in section 3.2.2 must be taken into consideration when computing the prediction errors. The displacements have been estimated using the motion estimation algorithms of chapters 2 and 3. The prediction errors are quantized using the quantizer of table T4.3.

The effect of motion compensation has been quantified by the prediction gain in the moving areas. This has been compared to the uncompensated case. The signal-to-noise ratio of the output image has been measured and comparison made between the compensated and the uncompensated cases. Also measured is the mean square prediction error in the moving areas before and after motion compensation.

The results of the experiments are plotted in figures 4.13 through 4.30. Figures 4.13, 4.14 and 4.15 show the mean square error in the moving areas with and without motion compensation for the test sequences "Box", "Alan" and "Walking man" respectively. The feature based motion estimation technique has been used. For the sequence "Box" and "Walking man" a block size of 2x4 was used while a block size of 5x6 was used for the sequence "Alan". The graphs show good performance with all the test sequences, apart from "Walking man" where the performance is not very good. Figure 4.16, 4.17 and 4.18 show the respective signal-to-noise-ratios over the output images. While there is good performance with "Box" and "Alan", the motion compensation did not improve considerably the

signal-to-noise-ratio of the output in the case of sequence "Walking man". The respective prediction gains are shown in figures 4.19, 4.20 and 4.21. The same trend as depicted with mean square error and signal-to-noise-ratio can be seen here. The nature (in terms of pattern and hence pixel distribution) of the images tend to affect the technique of motion estimation. The smoooth images like "Box" tend to be well suited to this technique while a highly patterned image sequence like "Walking man" did not perform well. On the average side is the sequence "Alan" which is not too detailed. This image sequence however represents a typical video conference or video telephone image and it is here that this application (image sequence coding) might be most needed. Some medical images will also fall into this image category.

Figures 4.22 through 4.30 show the result of the experiments carried out using the differential technique PDDT2 as the motion estimator. The difficulty with these experiments is in the choice of block size, since this, as shown in chapter 3, combines with the displacement in giving various degree of accuracy of prediction. However in all the experiments we have used a block size of 8x8 since this appears to be an optimum block size (using information gathered from experiments of chapter 3). Figure 4.22, 4.23 and 4.24 show the mean square error in the moving areas with and without motion compensation. The mean square errors have not been reduced considerably when compared to the reduction obtained using the feature based technique. Figures 4.25, 4.26 and 4.27 show the signal-to-noise-ratios while figures 4.28, 4.29 and 4.30 show the prediction gains.

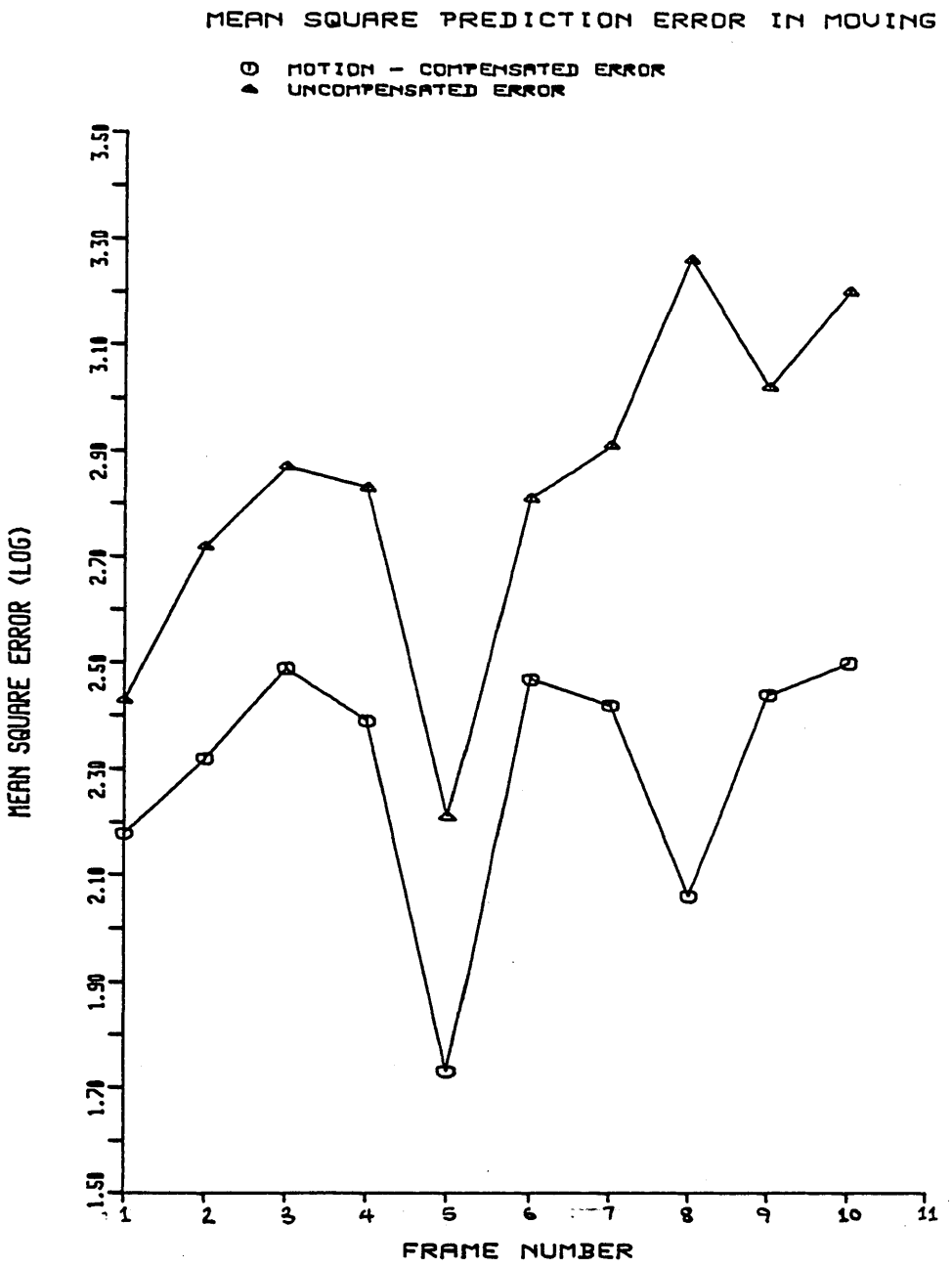



**Figure 4.13:** Motion-compensated interframe DPCM coding using feature based motion estimation technique. Sequence "Box".

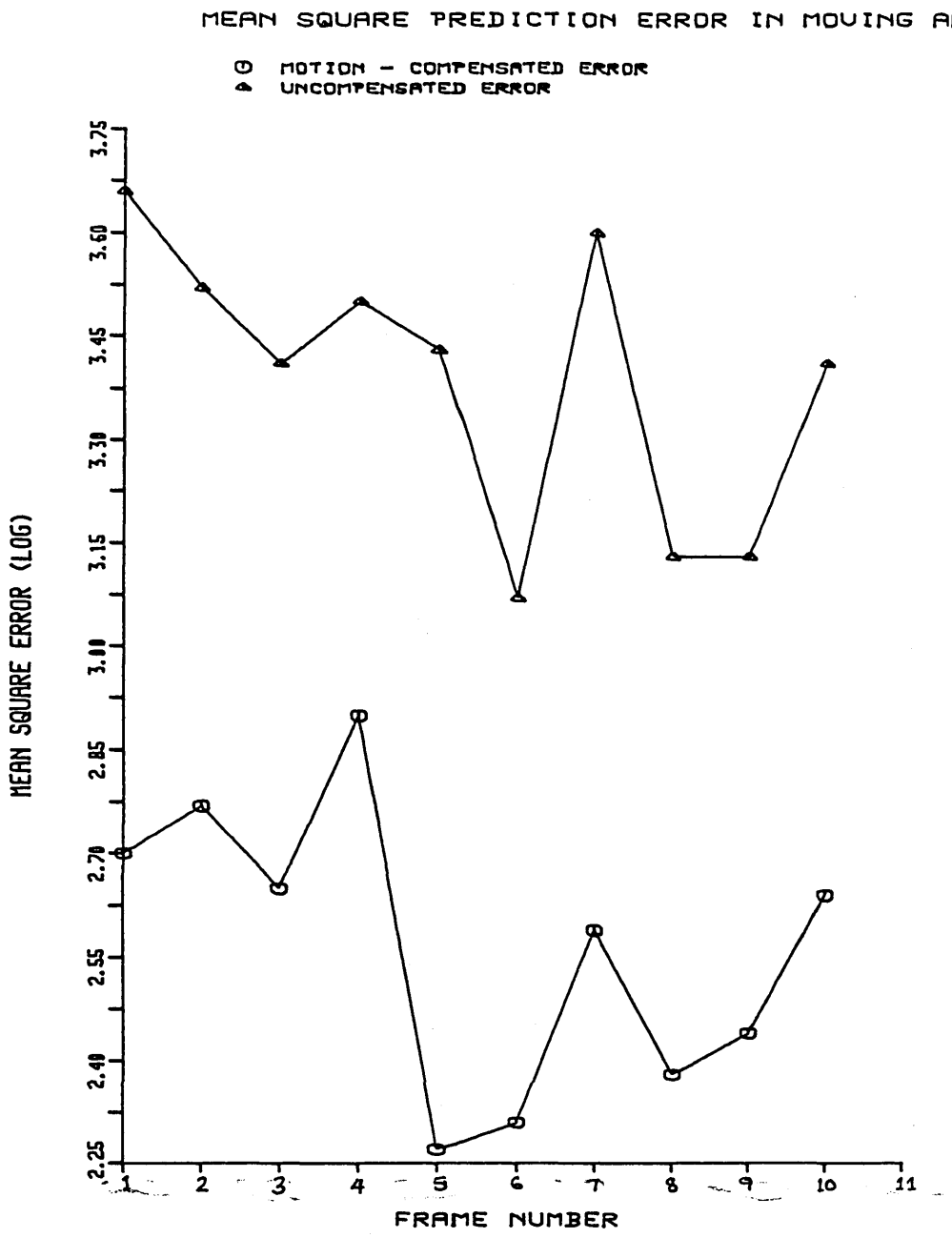



**Figure 4.14:** Motion-compensated interframe DPCM coding using feature based motion estimation technique.
Sequence "Alan".

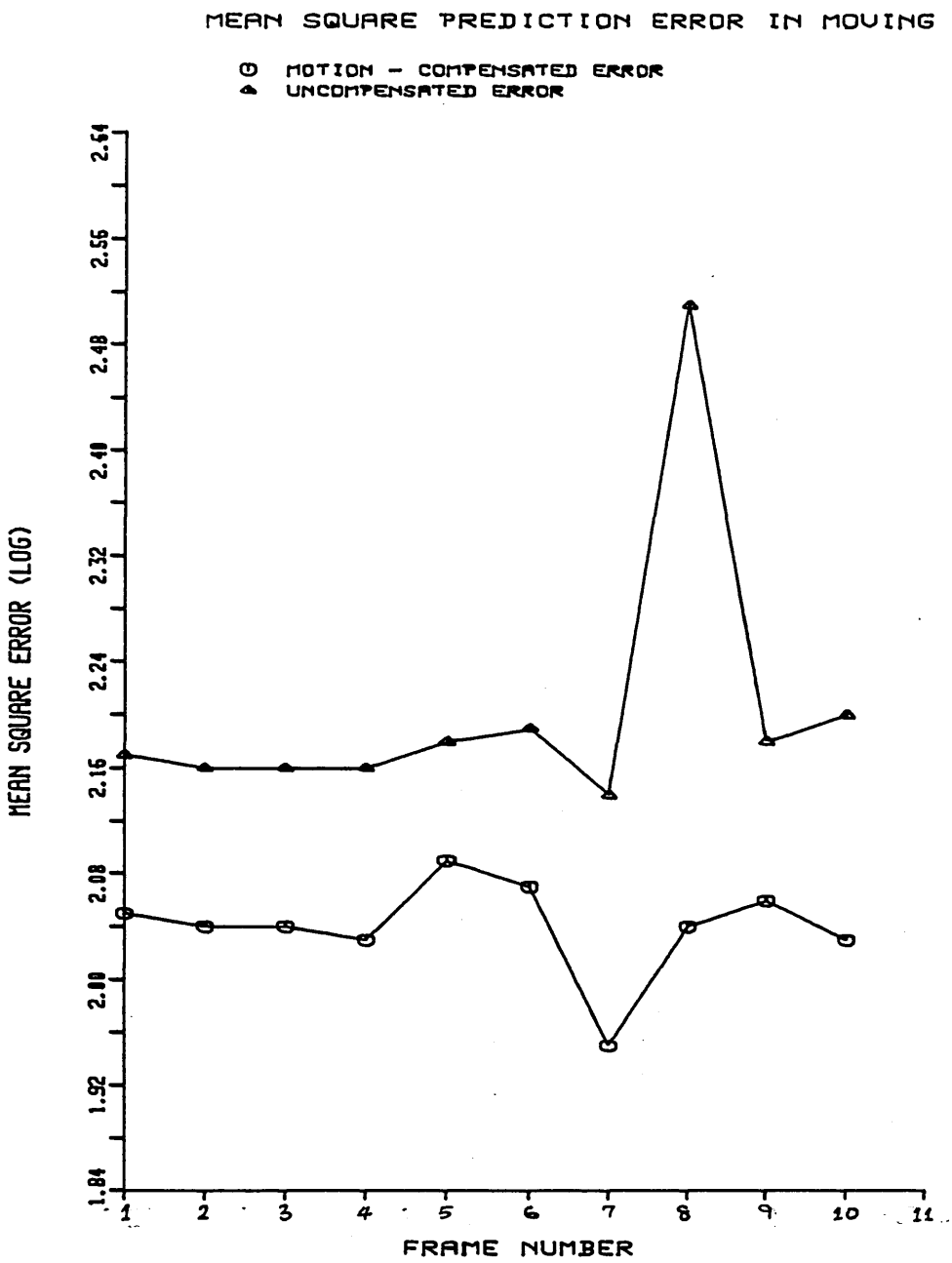



**Figure 4.15:** Motion-compensated interframe DPCM coding using feature based motion estimation technique.
Sequence "Walking man".

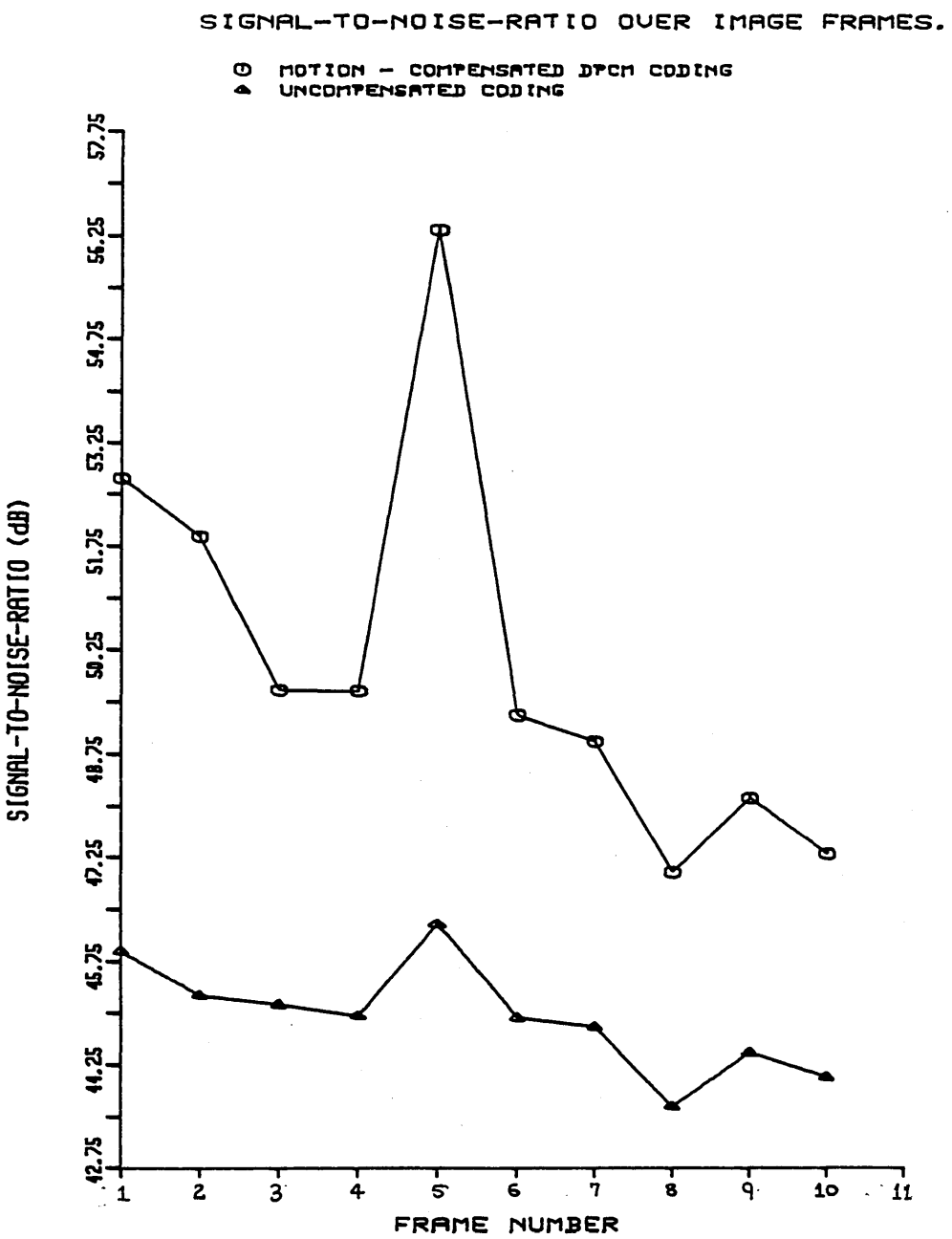



**Figure 4.16:** Motion-compensated interframe DPCM coding using feature based motion estimation technique.
Sequence "Box".

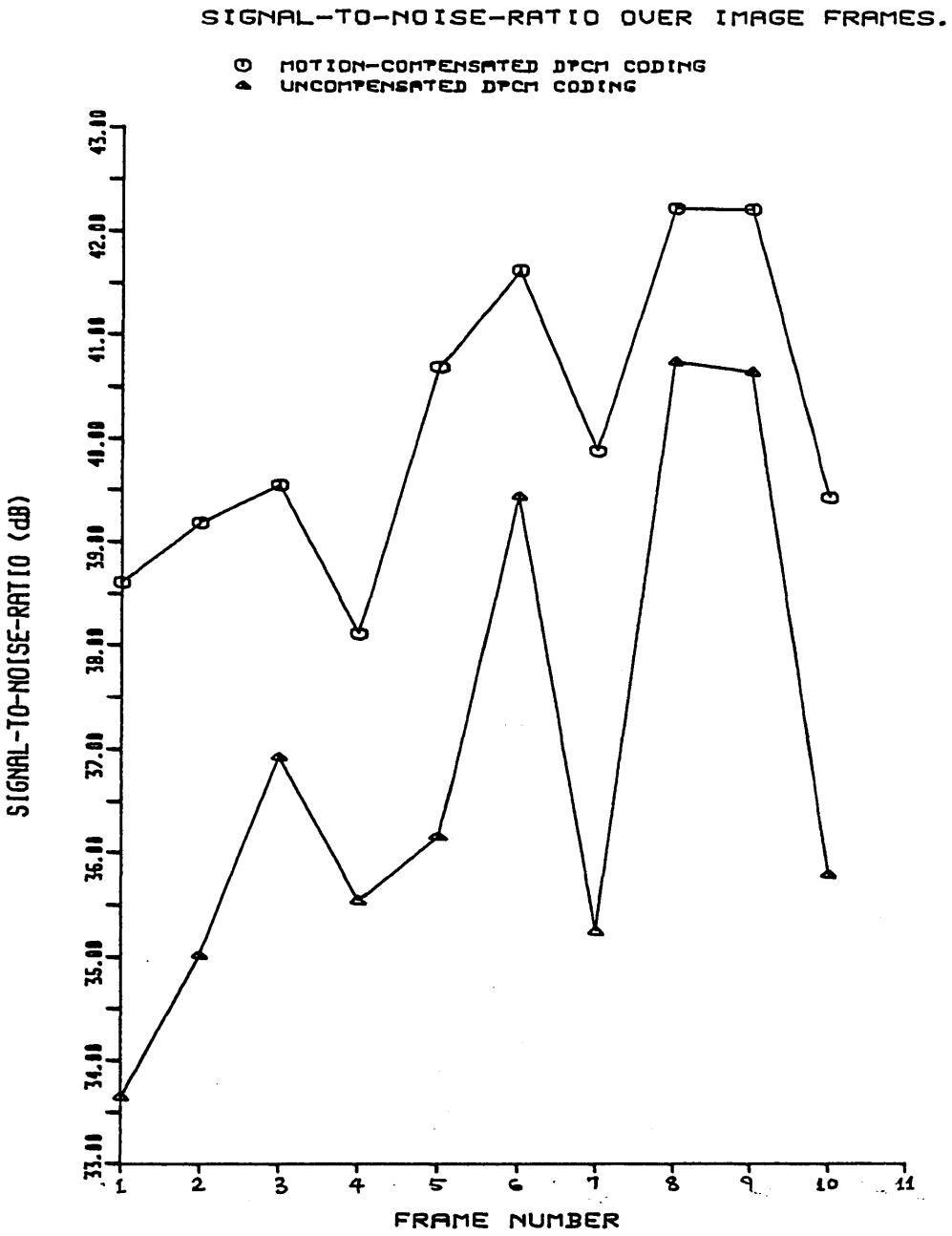



**Figure 4.17:** Motion-compensated interframe DPCM coding using feature based motion estimation technique.
Sequence "Alan".

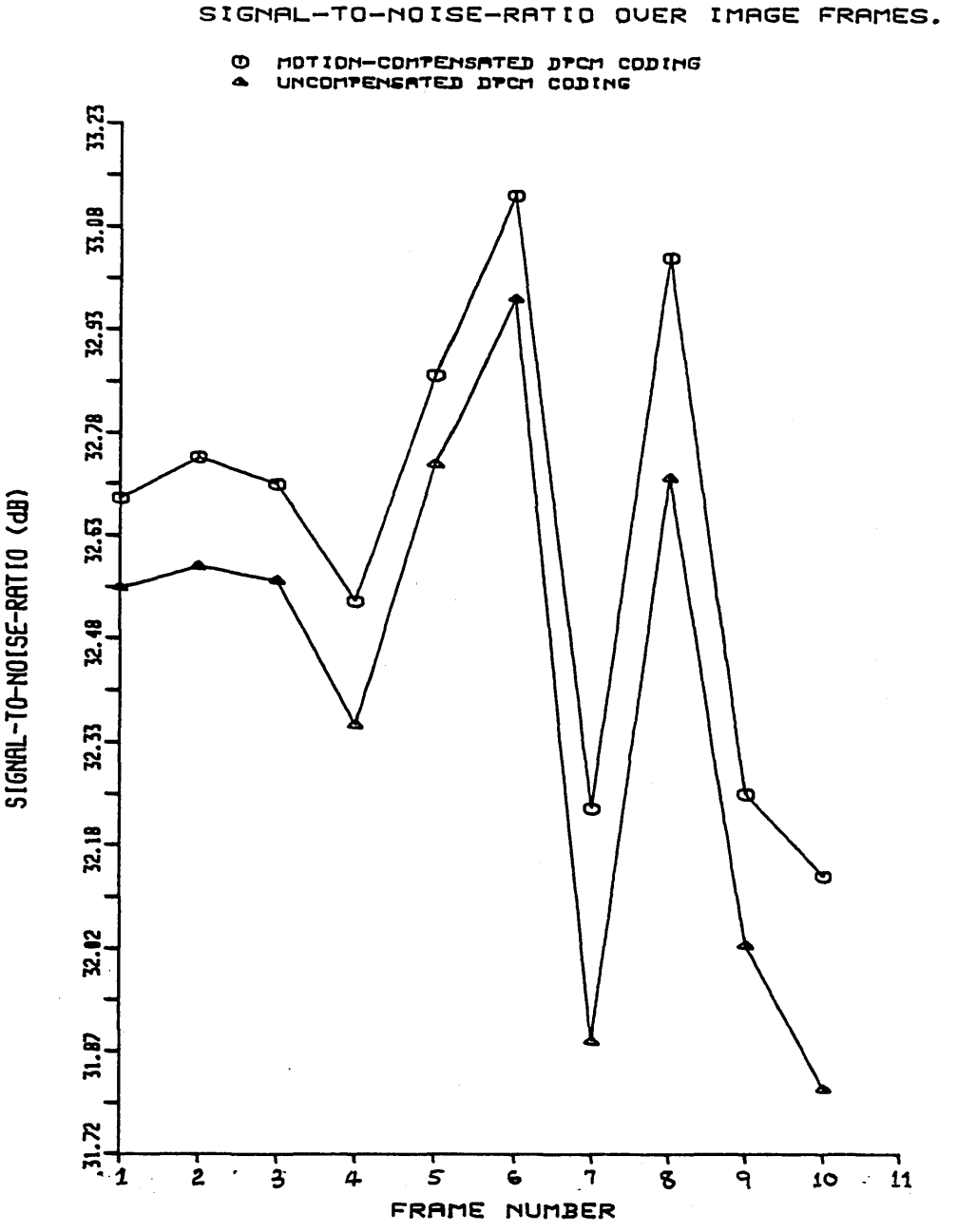



**Figure 4.18:** Motion-compensated interframe DPCM coding using feature based motion estimation technique.
Sequence "Walking man".

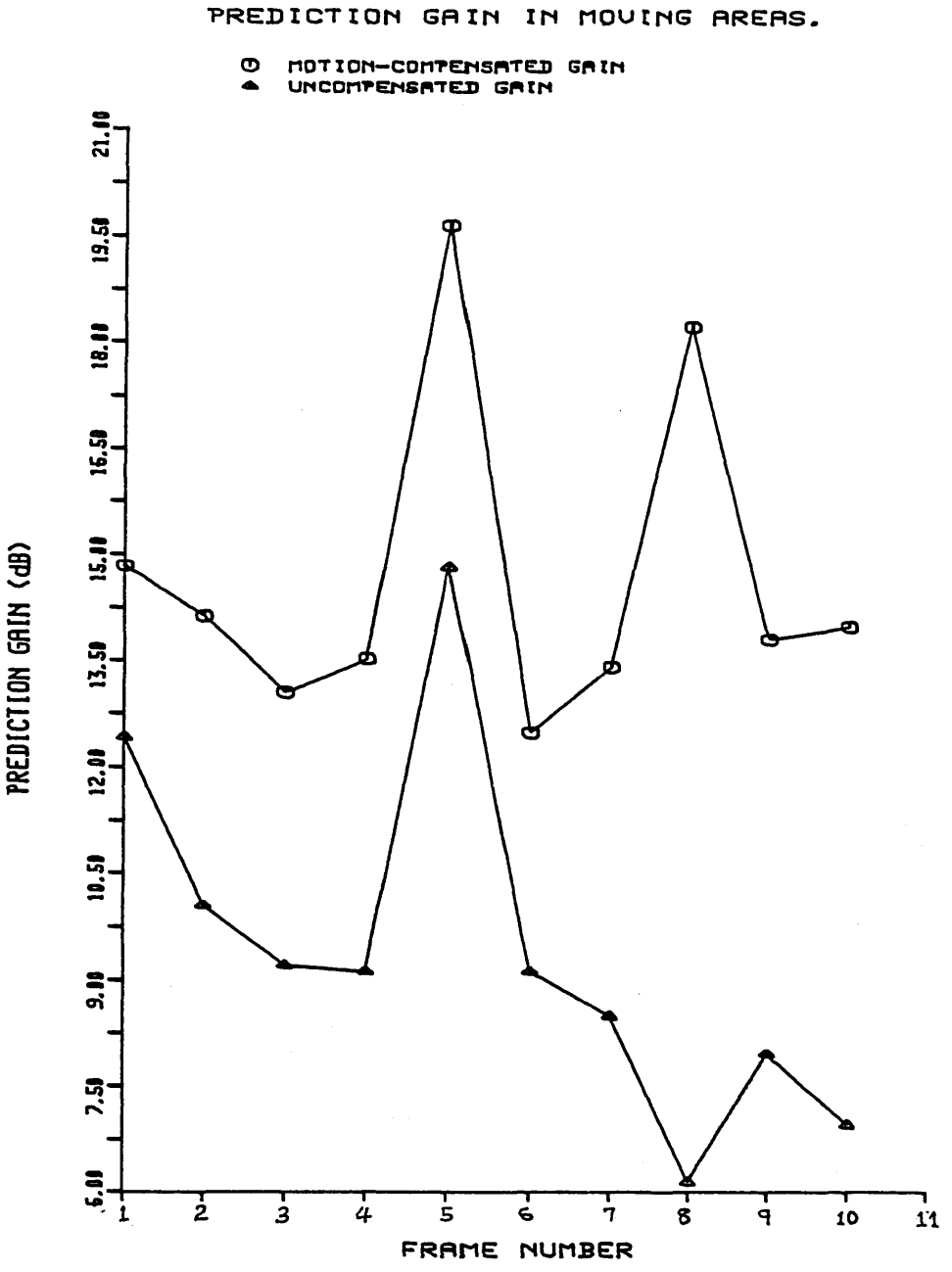



Figure 4.19: Motion-compensated interframe DPCM coding using feature based motion estimation technique.
Sequence "Box".

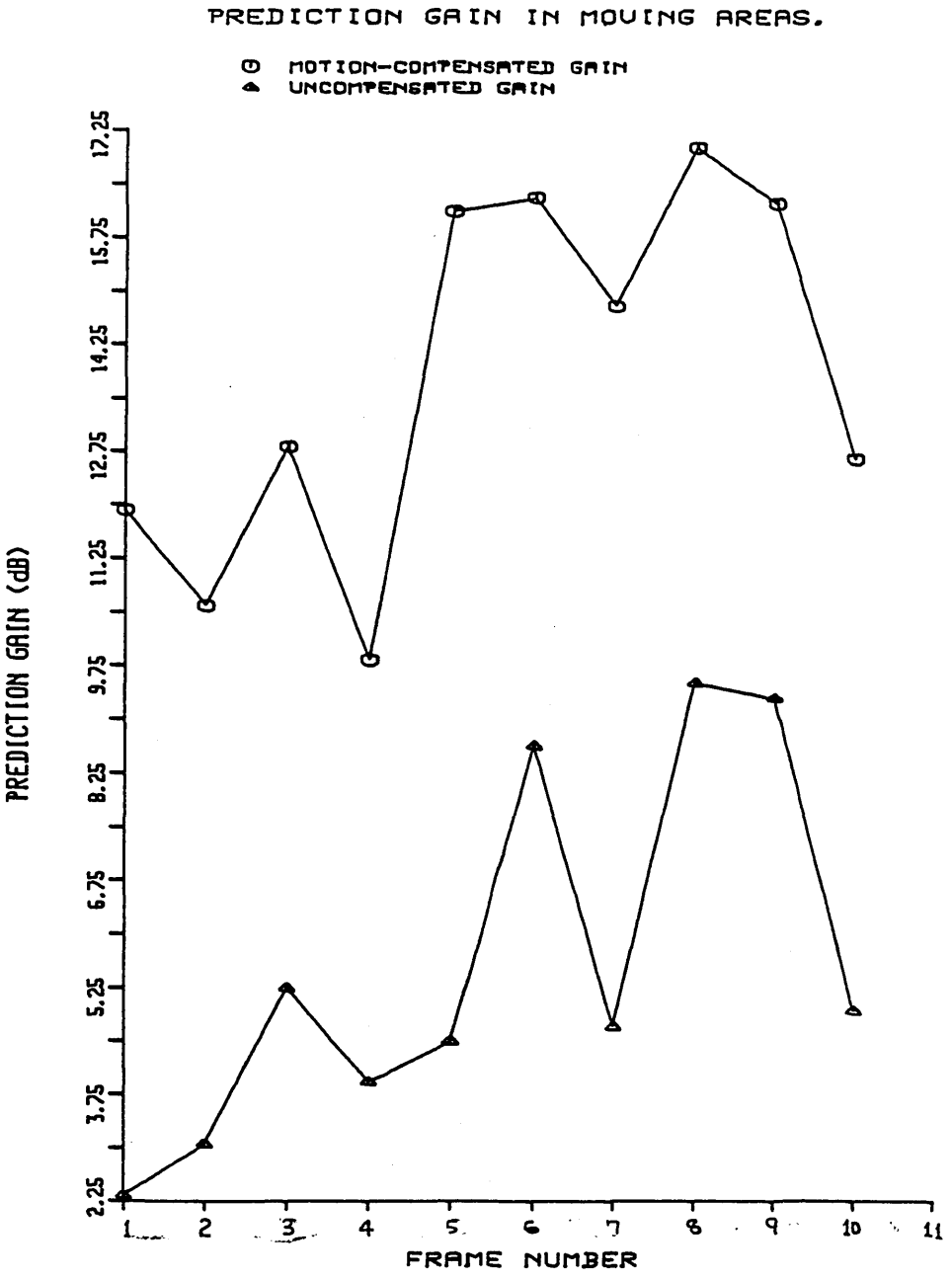



**Figure 4.20:** Motion-compensated interframe DPCM coding using feature based motion estimation technique.
Sequence "Alan".

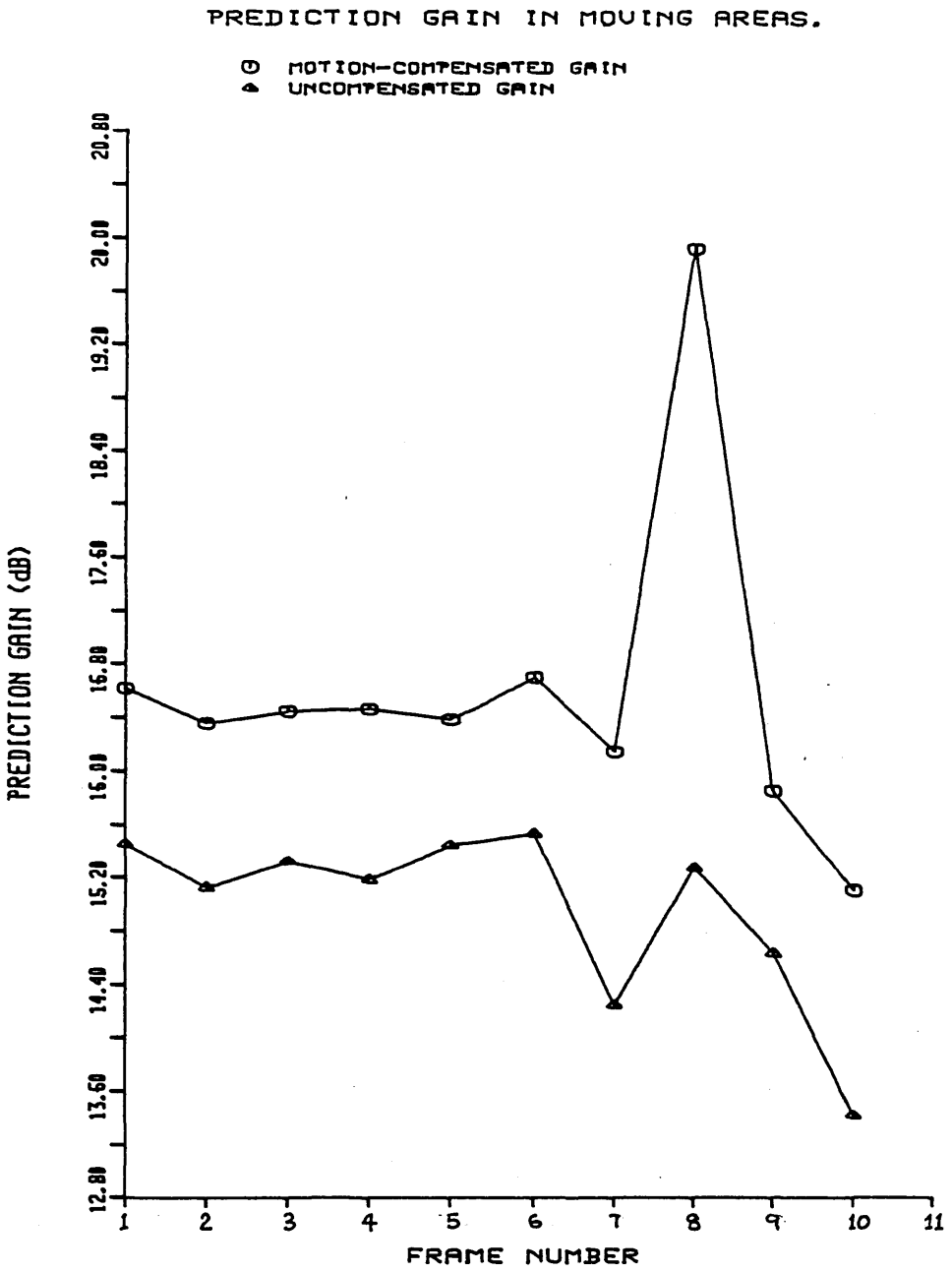



**Figure 4.21**: Motion-compensated interframe DPCM coding using feature based motion estimation technique.
Sequence "Walking man.

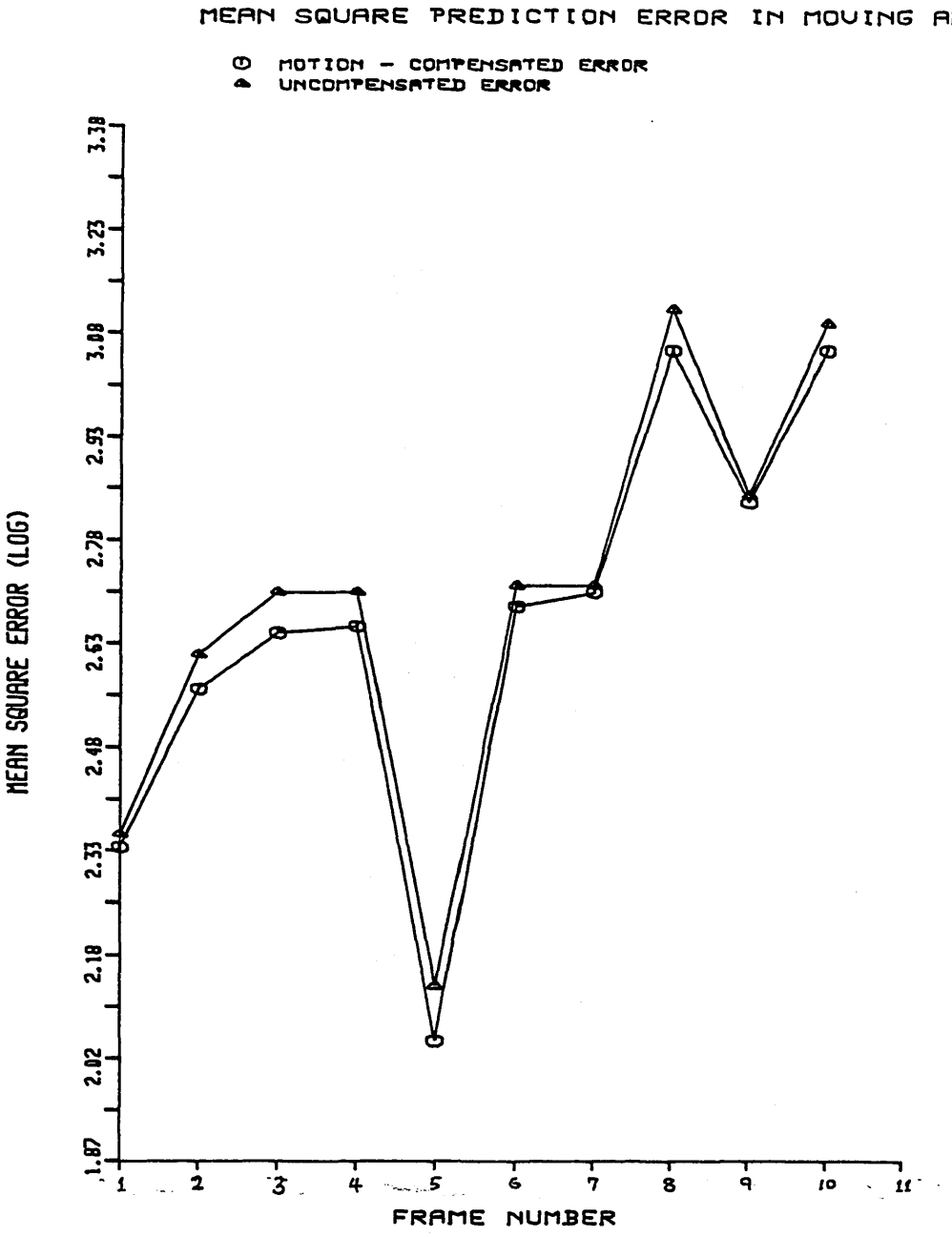



Figure 4.22: Motion-compensated interframe DPCM coding using the differential technique of motion estimation.
Sequence "Box".

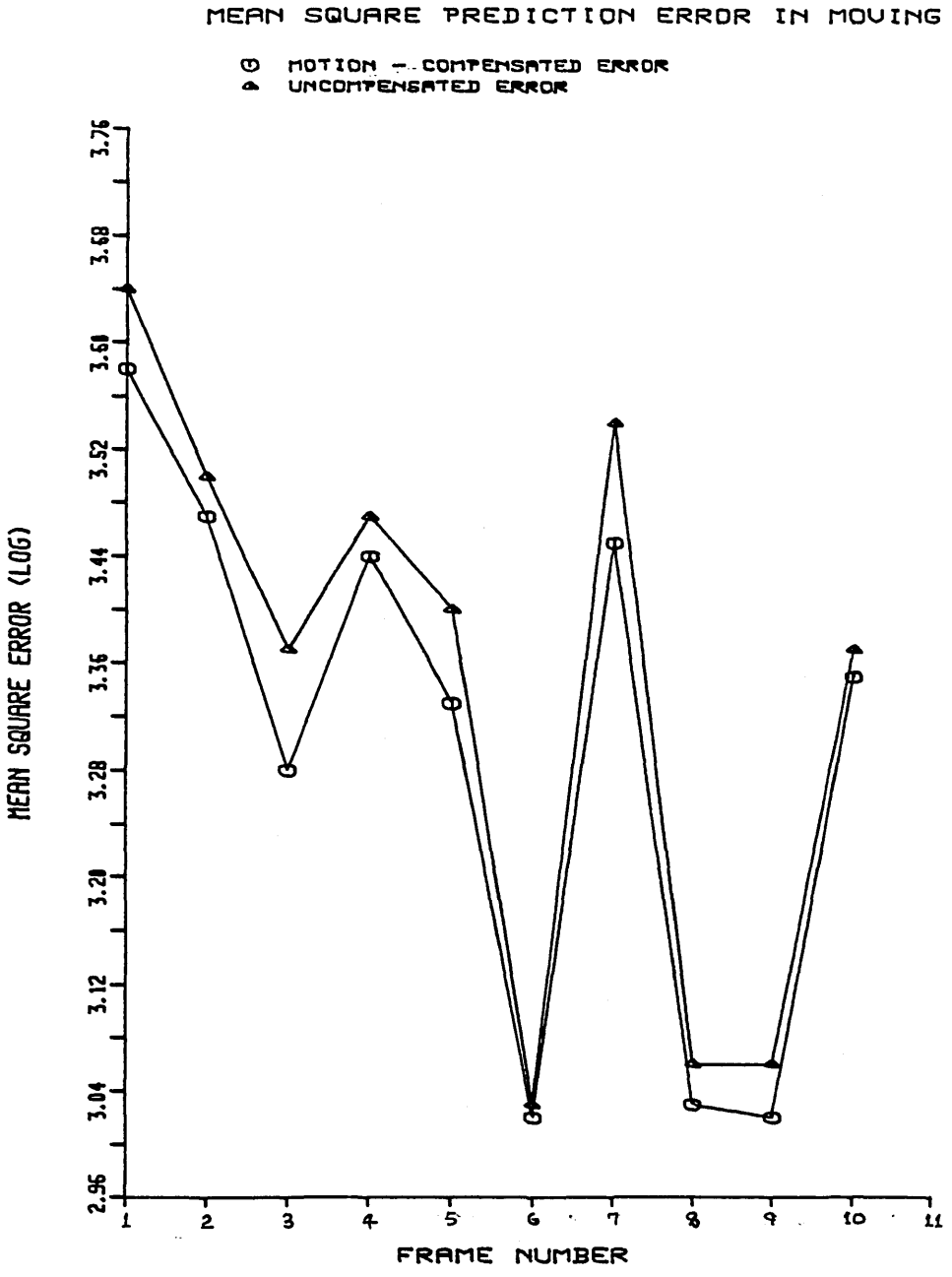



**Figure 4.23:** Motion-compensated interframe DPCM coding using the differential technique of motion estimation.
Sequence "Alan".

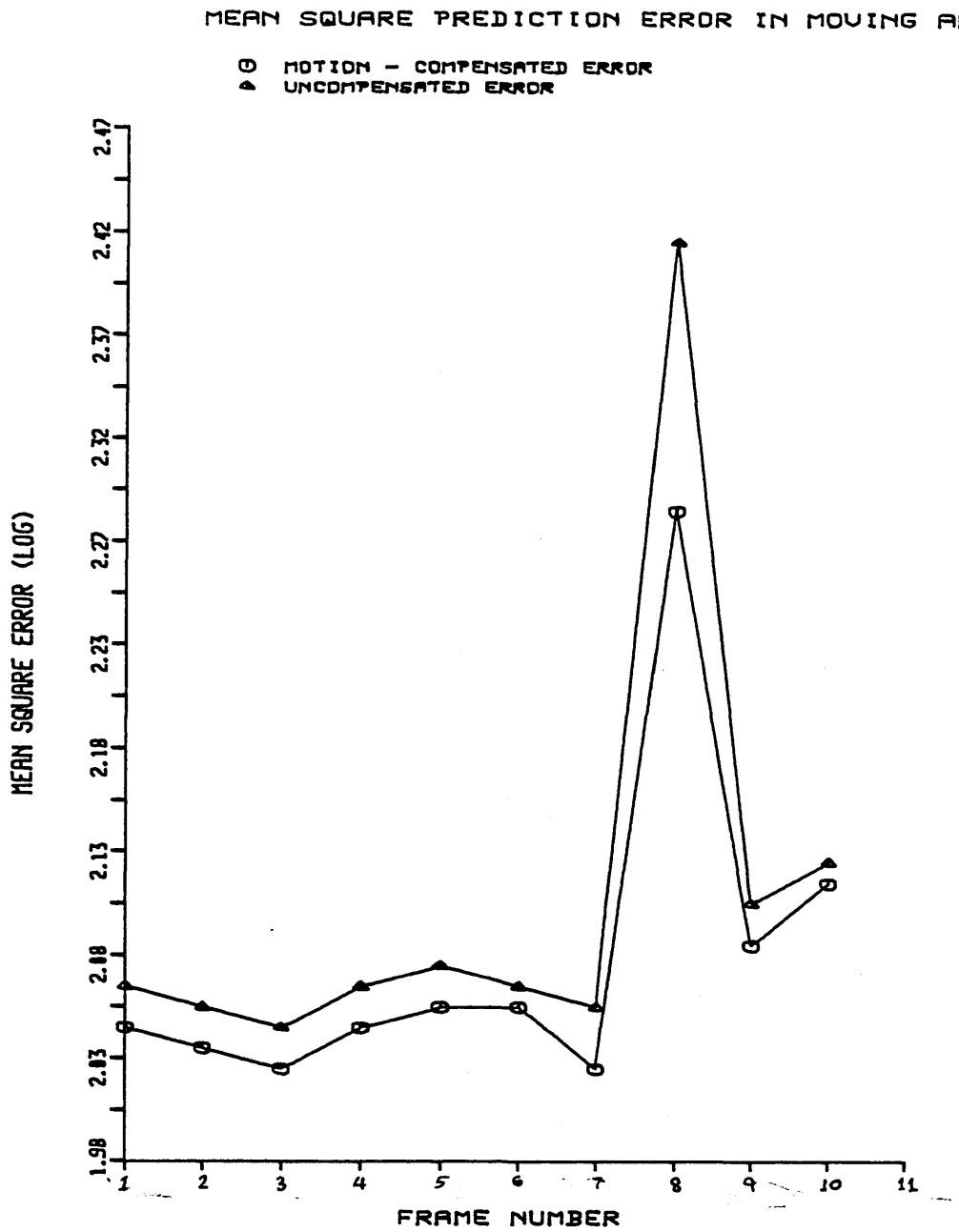



**Figure 4.24:** Motion-compensated interframe DPCM coding using the differential technique of motion estimation.
Sequence "Walking man".

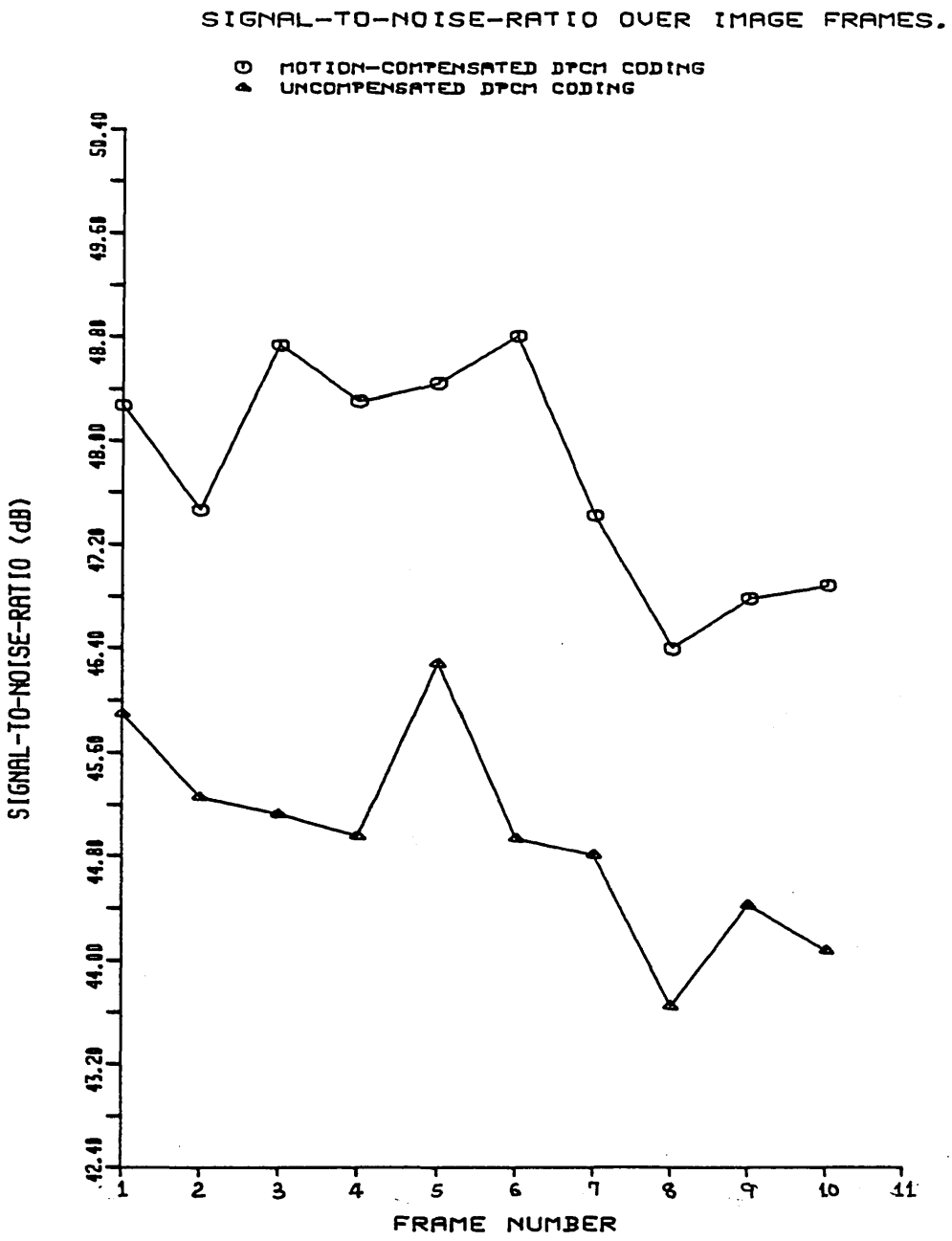



**Figure 4.25:** Motion-compensated interframe DPCM coding using the differential technique of motion estimation.
Sequence "Box".

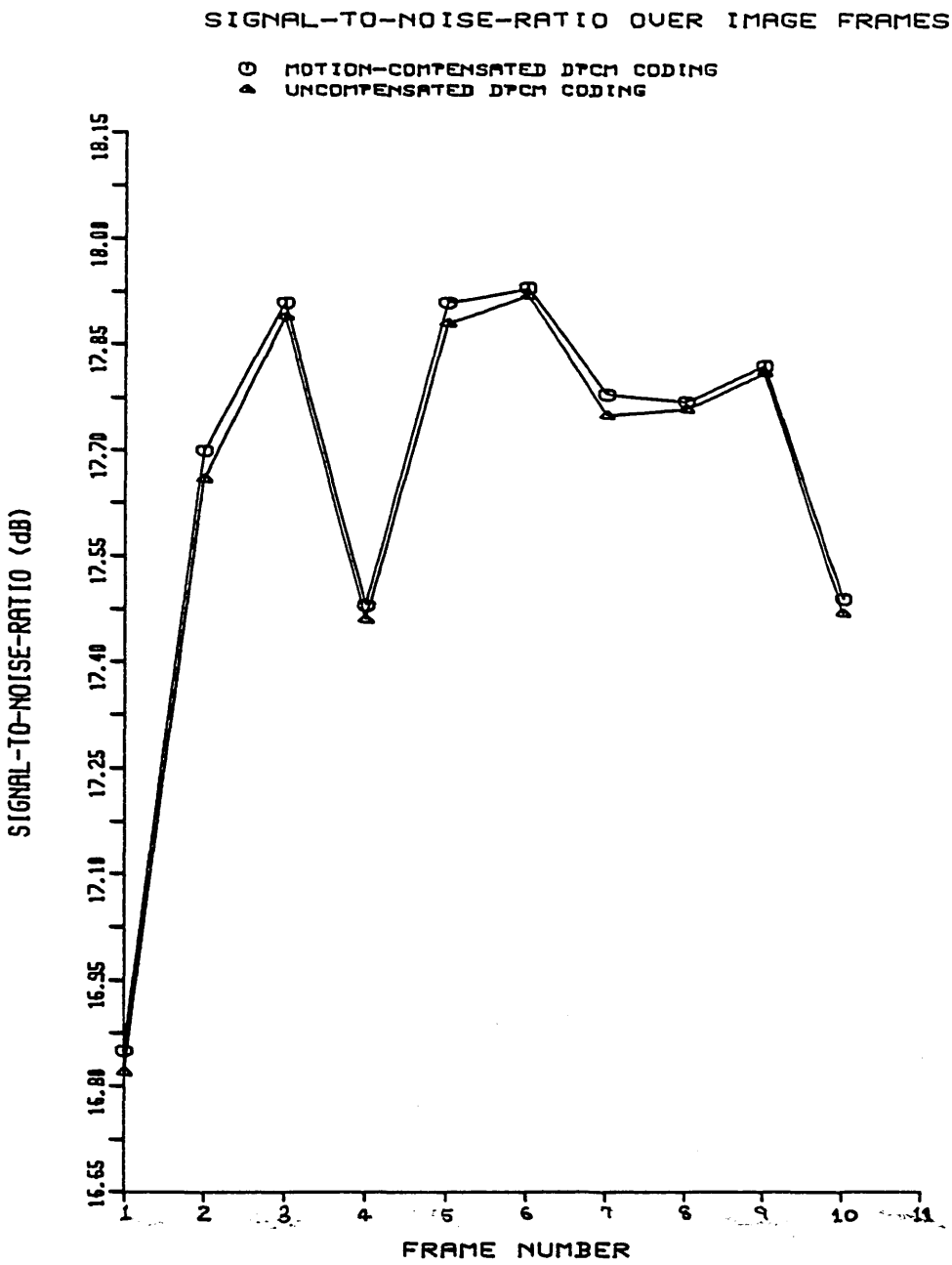



**Figure 4.26:** Motion-compensated interframe DPCM coding using the differential technique of motion estimation.
Sequence "Alan".

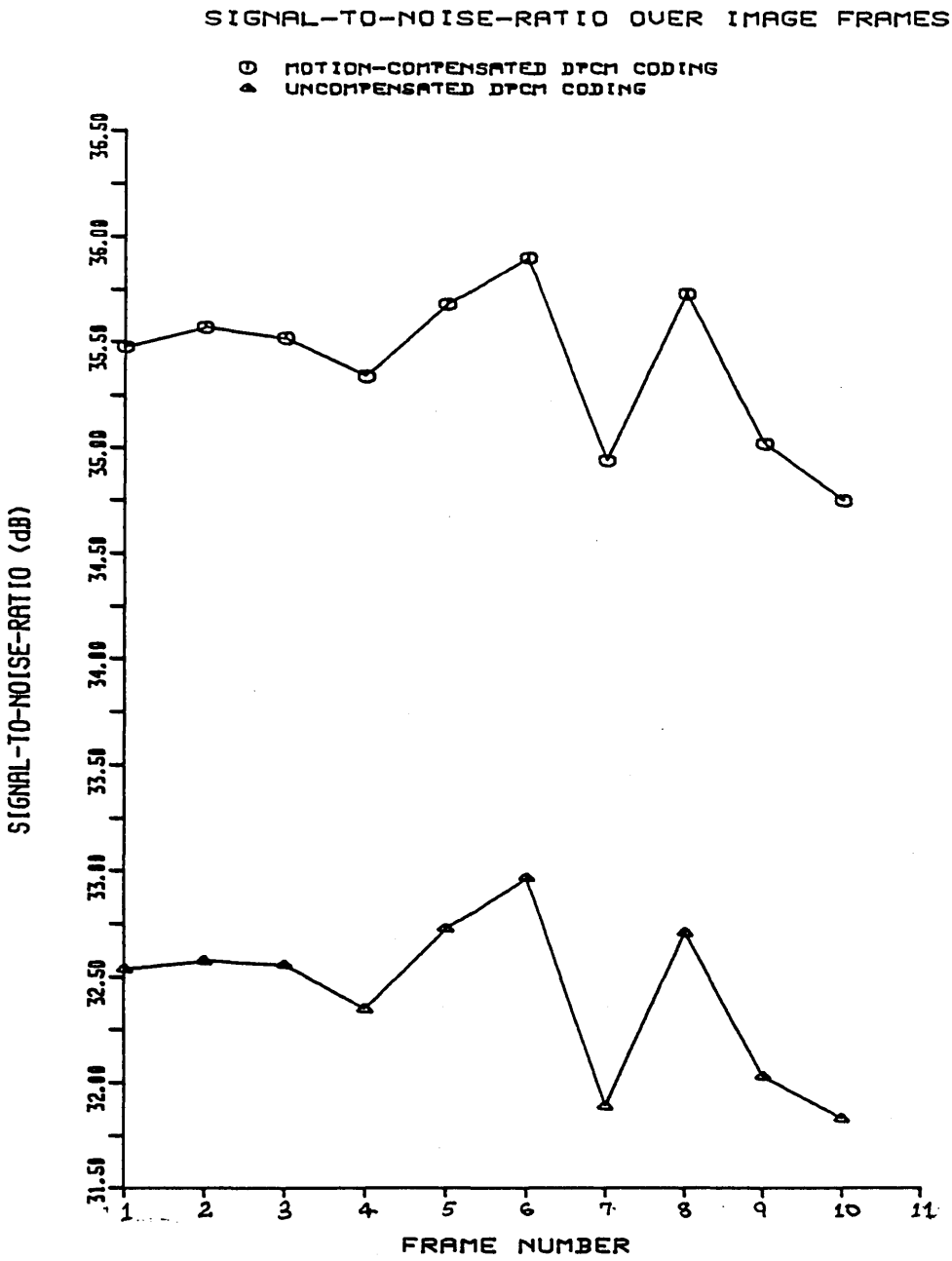



**Figure 4.27:** Motion-compensated interframe DPCM coding using the differential technique of motion estimation.
Sequence "Walking man".

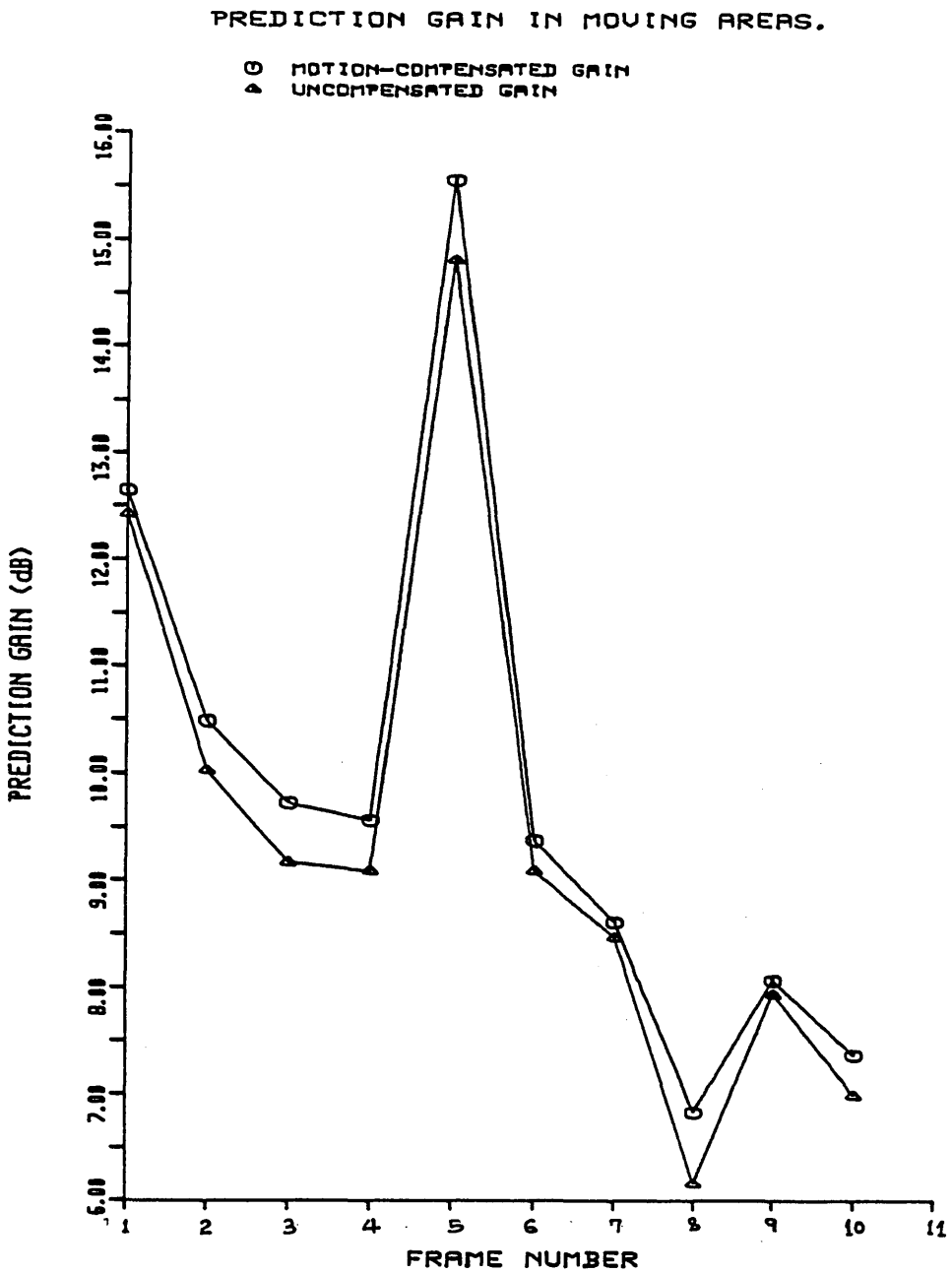



**Figure 4.28:** Motion-compensated interframe DPCM coding using the differential technique of motion estimation.
Sequence "Box".

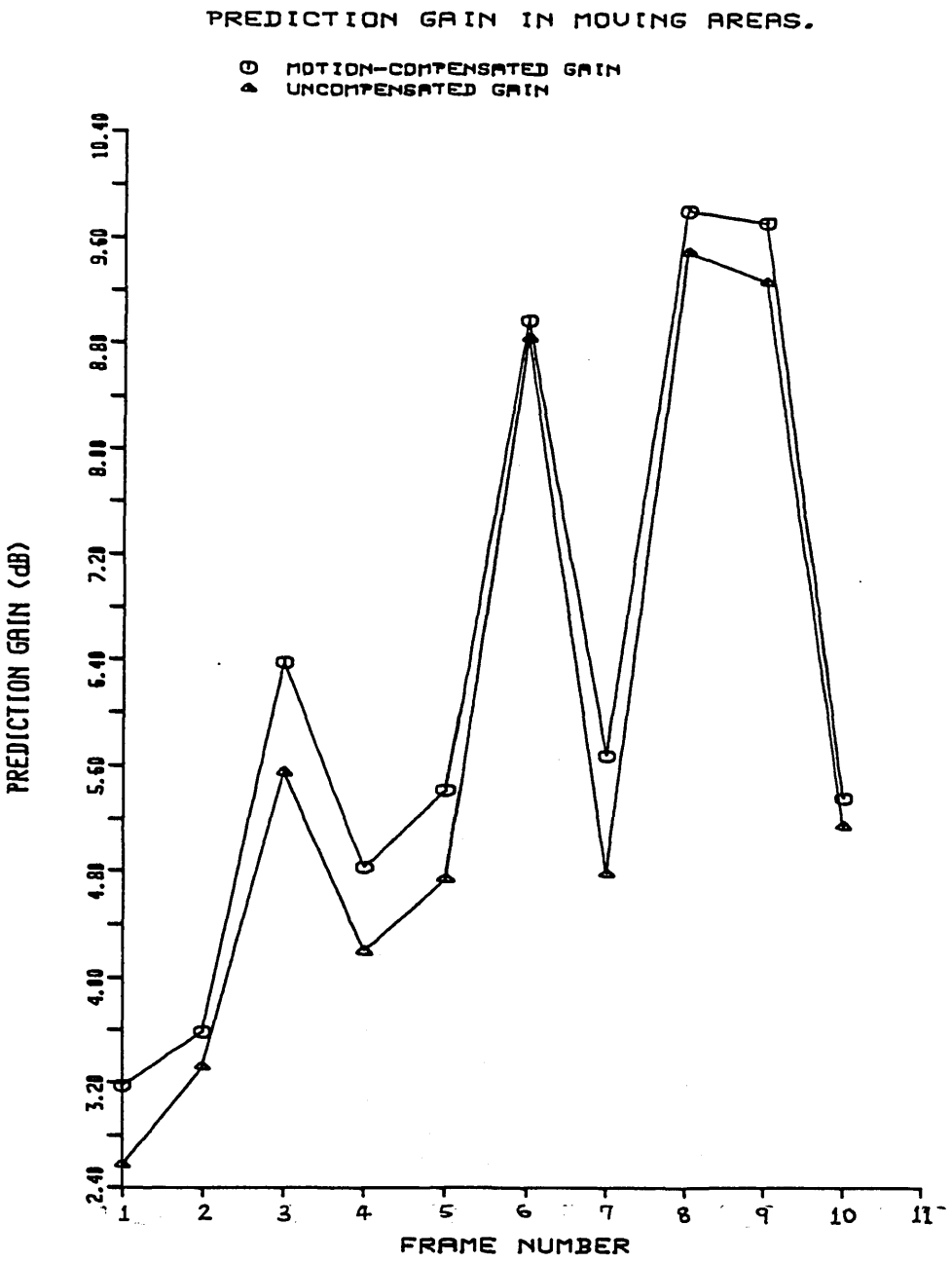



**Figure 4.29:** Motion-compensated interframe DPCM coding using the differential technique of motion estimation.
Sequence "Alan".

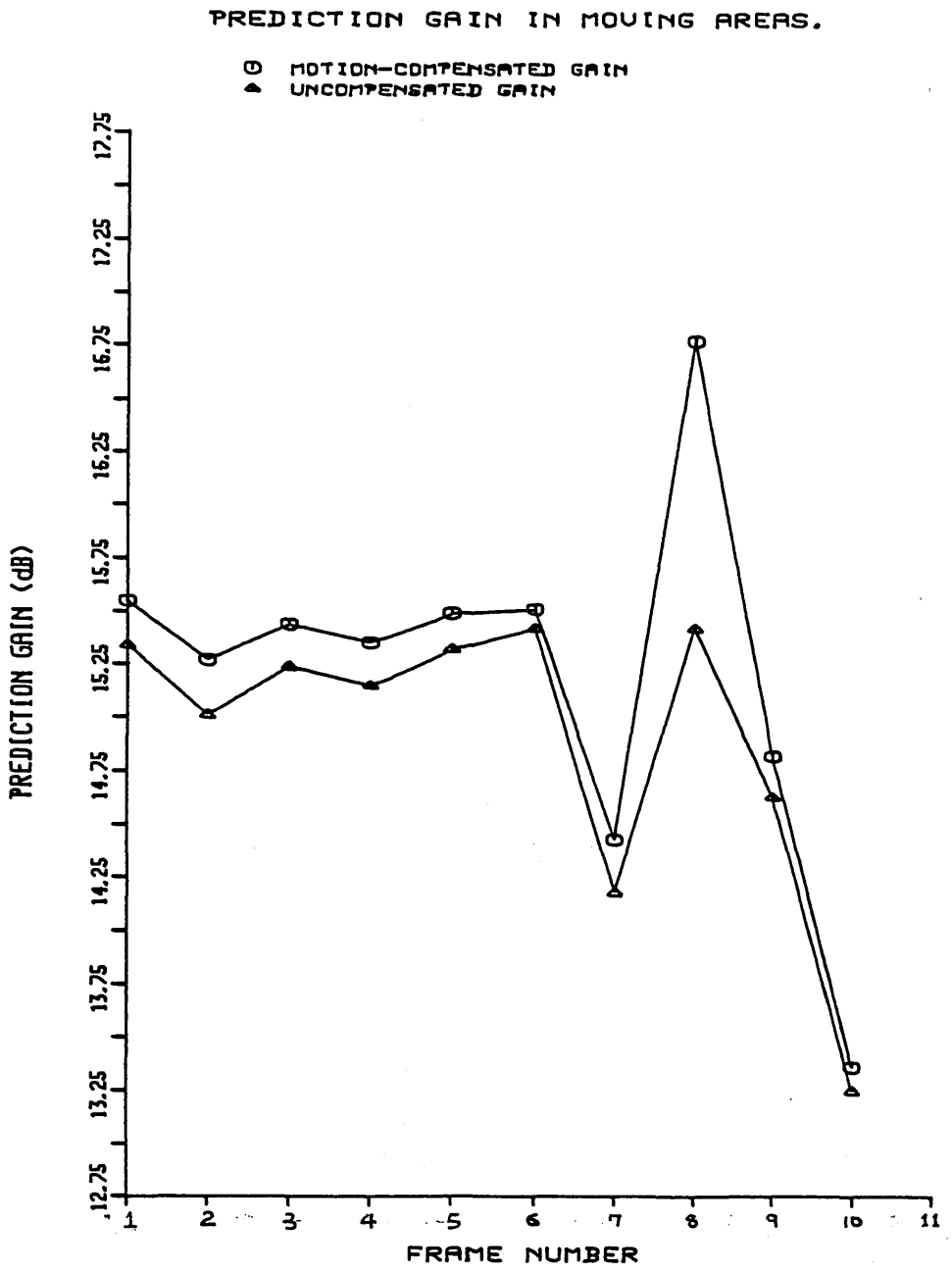



**Figure 4.30**: Motion-compensated interframe DPCM coding using the differential technique of motion estimation.
Sequence "Walking man".

# CHAPTER 5

## CONCLUSION

The problem of motion estimation has been investigated. Two approaches which have led to the development of new algorithms were investigated. In one approach image frames were partitioned into rectangular blocks and moment invariants used to obtain their feature description. The feature vectors derived were then used in a two-stage matching scheme for motion estimation. The results obtained with the test images indicate that this approach to motion estimation is feasible and efficient. The effect of noise was not investigated explictly, however the results obtained in the temporal filtering application are an indication of the effect of noise. In a noisy environment the moment invariants will not perform wonderfully. A good point for this approach is that, given a noise free sequence of images, we are sure that the blocks that match, will do so approximately in the sense of minimum frame difference variance between the pixels that make up the block. This conclusion follows from the uniqueness theorem of moments [35]. The radial and angular moment invariants were also used in motion estimation. The performance of this variation is not comparable to the rectangular coordinate counterpart. This is attributable to the sampling pattern of the test images used; rectangular grid sampling. In deriving a circular sampling from these images, an implicit averaging (blurring) was introduced, thus blurring out discriminating features of the image.

In the other approach the differential technique was investigated in the projection domain. An improvement was obtained over the technique of Limb and Murphy [47]. The

results obtained from the simulations indicate that this variation of the differential technique is useful even when the displacement is greater than 2.5 pixels per frame. The application of this technique to motion compensated DPCM coding gave results that indicate its usefulness in image compression work.

The techniques developed were used in compensating for motion in some image processing work; motion compensated interpolation, filtering and coding were investigated. Good performances were obtained.

A comparison of the feature based technique and the projection domain differential technique is given in table T5.1. The image sequence "Walking man" has been used with a segmentation block size of 4x4. The comparision is in terms of the prediction gain, a measure of the accuracy of the displacement estimate. The feature based technique gave a better prediction gain than the projection domain differential technique. The projection domain differential technique is an improvement over the algorithm of Limb and Murphy [47].

Table T5.1:

Comparision of prediction gain (dB) using feature based and projection domain differential technique.

Block size: 4 X 4.

Sequence: "Walking man".

Feature vector: FV4 ; Algorithm PDDT2.

| IMAGE PAIR. | FEATURE BASED TECHNIQUE | PROJECTION DOMAIN TECHNIQUE. |
|---|---|---|
| 1 | 16.33 | 15.66 |
| 2 | 16.02 | 15.18 |
| 3 | 16.28 | 15.48 |
| 4 | 16.15 | 15.11 |
| 5 | 16.33 | 15.44 |
| 6 | 16.47 | 15.46 |
| 7 | 15.77 | 14.12 |
| 8 | 18.87 | 16.73 |
| 9 | 15.42 | 14.70 |
| 10 | 14.59 | 13.28 |

As a summary it can be said that the use of features as a means of estimating motion deserves more attention than it has been given hitherto. Recalling the argument proffered in the prologue, features will serve as a better means of tracking moving objects than the individual pixels making up the object.

Further research should be geared towards the design of efficient feature descriptors which are also computationally inexpensive. The use of research results in the area of artificial intelligence should be brought to bear on the problem of motion estimation for movement compensated image processing. With adequate knowledge base the problem of occlusion could be tackled, yielding interesting results.

APPENDIX A-1

MOMENT INVARIANTS.

(I) DERIVATION IN TERMS OF RADIAL AND ANGULAR INVARIANTS.

The derivation of Hu's moment invariants in terms of radial and angular moments are given below.

The ordinary 2-dimensional moment of order (p,q), referred to the centroid (i.e central moment) is given as,

$$u_{pq}=\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} x_1^p y_1^q \, f(x,y)\,dx\,dy \qquad \ldots.(A1\text{-}1)$$

with $x_1= x- \bar{x}$ and $y_1= y- \bar{y}$

$\bar{x}= m_{10}/ m_{00}$ $\bar{y}= m_{01}/ m_{00}$ and $f(x,y)$ is the image function.

$$m_{00}=\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y)\,dx\,dy \quad m_{01}=\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} y f(x,y)\,dx\,dy$$
$$m_{10}=\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} x f(x,y)\,dx\,dy.$$

The radial and angular moments are defined respectively as,

$$\Psi_r(k,L)= \int_{o}^{\infty} r^k L(r,\theta)\,dr \qquad \ldots.(A1\text{-}2)$$

and

$$\Psi_\theta(p,q,L)= \int_{-\pi}^{\pi} \cos^p\theta \sin^q\theta \; L(r,\theta)\,d\theta \qquad \ldots.(A1\text{-}3)$$

where $L(r,\theta)$ is the image function in polar coordinates, k is a radial index.

From (A1-2) and (A1-3) we can write

$$\Psi(k,p,q,L) = \int_{0}^{\infty}\int_{-\pi}^{\pi} r^{k}\cos^{p}\theta\sin^{q}\theta L(r,\theta)\, d\theta dr$$

$$\triangleq \Psi_{r}(k,\Psi_{\theta}(p,q,L))$$

....(A1-4)

In polar form the (A1-1) can be expressed as,

$$u_{pq}=\int_{0}^{\infty}\int_{-\pi}^{\pi} r^{p+q+1}\cos^{p}\theta\sin^{q}\theta L(r,\theta)\, d\theta dr$$

....(A1-5)

From (A1-4) and (A1-5) the moments $u_{pq}$ can be written as,

$$u_{pq}= \Psi(p+q+1,\ p,\ q,\ L)$$

....(A1-6)

The first moment invariant of Hu is given as,

$$MI1= u_{20} + u_{02}$$

....(A1-7)

Using (A1-6) in (A1-7) we have,

$$\begin{aligned} MI1 &= \Psi(3,2,0,L) + \Psi(3,0,2,L) \\ &= \int_{0}^{\infty}\int_{-\pi}^{\pi} [r^{3}\cos^{2}\theta L(r,\theta) + r^{3}\sin^{2}\theta L(r,\theta)]\, dr d\theta \\ &= \int_{0}^{\infty}\int_{-\pi}^{\pi} r^{3}L(r,\theta)[\cos^{2}\theta + \sin^{2}\theta]\, dr d\theta \\ &= \int_{0}^{\infty}\int_{-\pi}^{\pi} r^{3}L(r,\theta)\, dr d\theta = \Psi_{r}(3,\Psi_{\theta}(L)) \end{aligned}$$

The derivation of MI5 is presented.

$$\begin{aligned} MI5 = {} & (u_{30}-3u_{12})(u_{30}+u_{12})\,[(u_{30}+u_{12})^{2} - 3(u_{21}+u_{03})^{2}] + \\ & (3u_{12}-u_{03})(u_{21}+u_{03})\,[3(u_{30}+u_{12})^{2} - (u_{21}+u_{03})^{2}] \end{aligned}$$

The following can be shown easily

$$u_{30}-3u_{12}=\int_{-\pi}^{\pi}\int_{0}^{\infty} r^4\cos 3\theta L(r,\theta)\,dr\,d\theta$$

$$3u_{21}-u_{03}=\int_{-\pi}^{\pi}\int_{0}^{\infty} r^4\sin 3\theta L(r,\theta)\,dr\,d\theta$$

$$u_{30}+u_{12}=\int_{-\pi}^{\pi}\int_{0}^{\infty} r^4\cos\theta L(r,\theta)\,dr\,d\theta$$

$$u_{21}+u_{03}=\int_{-\pi}^{\pi}\int_{0}^{\infty} r^4\sin\theta L(r,\theta)\,dr\,d\theta$$

Substitution into MI5 gives,

$$\begin{aligned}
MI5=&\int_{-\pi}^{\pi}\int_{0}^{\infty} r^4\cos 3\theta L(r,\theta)\,dr\,d\theta \int_{-\pi}^{\pi}\int_{0}^{\infty} r^4\cos\theta L(r,\theta)\,dr\,d\theta\\
&\{[\int_{-\pi}^{\pi}\int_{0}^{\infty} r^4\cos\theta L(r,\theta)\,dr\,d\theta]^2\\
&-3[\int_{-\pi}^{\pi}\int_{0}^{\infty} r^4\sin\theta L(r,\theta)\,dr\,d\theta]^2\}+\int_{-\pi}^{\pi}\int_{0}^{\infty} r^4\sin 3\theta L(r,\theta)\,dr\,d\theta\\
&\cdot\int_{-\pi}^{\pi}\int_{0}^{\infty} r^4\sin\theta L(r,\theta)\,dr\,d\theta\cdot\{3[\int_{-\pi}^{\pi}\int_{0}^{\infty} r^4\cos\theta L(r,\theta)\,dr\,d\theta]^2\\
&-[\int_{-\pi}^{\pi}\int_{0}^{\infty} r^4\sin\theta L(r,\theta)\,dr\,d\theta]^2\}\\
=&\int_{-\pi}^{\pi}\int_{0}^{\infty} r^4\cos 3\theta L(r,\theta)\,dr\,d\theta\int_{-\pi}^{\pi}\int_{0}^{\infty} r^4\cos\theta L(r,\theta)\,dr\,d\theta\\
&\cdot[\int_{-\pi}^{\pi}\int_{0}^{\infty} r^4\cos\theta L(r,\theta)\,dr\,d\theta]^2-3\int_{-\pi}^{\pi}\int_{0}^{\infty} r^4\cos 3\theta L(r,\theta)\,dr\,d\theta\\
&\cdot\int_{-\pi}^{\pi}\int_{0}^{\infty} r^4\cos\theta L(r,\theta)\,dr\,d\theta[\int_{-\pi}^{\pi}\int_{0}^{\infty} r^4\sin\theta L(r,\theta)\,dr\,d\theta]^2\\
&+3\int_{-\pi}^{\pi}\int_{0}^{\infty} r^4\sin 3\theta L(r,\theta)\,dr\,d\theta\cdot\int_{-\pi}^{\pi}\int_{0}^{\infty} r^4\sin\theta L(r,\theta)\,dr\,d\theta\\
&\cdot[\int_{-\pi}^{\pi}\int_{0}^{\infty} r^4\cos\theta L(r,\theta)\,dr\,d\theta]^2\\
&-\int_{-\pi}^{\pi}\int_{0}^{\infty} r^4\sin 3\theta L(r,\theta)\,dr\,d\theta\int_{-\pi}^{\pi}\int_{0}^{\infty} r^4\sin\theta L(r,\theta)\,dr\,d\theta\\
&\cdot[\int_{-\pi}^{\pi}\int_{0}^{\infty} r^4\sin\theta L(r,\theta)\,dr\,d\theta]^2
\end{aligned}$$

We make the following substitution to simplify the expression.

$$A=\int_{-\pi}^{\pi}\int_{0}^{\infty} r^4\cos 3\theta L(r,\theta)\,dr\,d\theta$$

$$B=\int_{-\pi}^{\pi}\int_{0}^{\infty} r^4\cos\theta L(r,\theta)\,dr\,d\theta$$

$$C=\int_{-\pi}^{\pi}\int_{0}^{\infty} r^4\sin\theta L(r,\theta)\,dr\,d\theta$$

$$D=\int_{-\pi}^{\pi}\int_{0}^{\infty} r^4\sin 3\theta L(r,\theta)\,dr\,d\theta$$

Then,

$$MI5 = AB^3 - 3ABC^2 + 3B^2CD - DC^3$$

$$= \text{Real part of}\{(A+jD)(B-jC)^3\}$$

$$= \text{Real part of}\{[\int_{-\pi}^{\pi}\int_{0}^{\infty} r^4 \cos 3\theta L(r,\theta)\,dr\,d\theta + j\int_{-\pi}^{\pi}\int_{0}^{\infty} r^4 \sin 3\theta L(r,\theta)\,dr\,d\theta] [\int_{-\pi}^{\pi}\int_{0}^{\infty} r^4 \cos\theta L(r,\theta)\,dr\,d\theta - j\int_{-\pi}^{\pi}\int_{0}^{\infty} r^4 \sin\theta L(r,\theta)\,dr\,d\theta]^3\}$$

$$= \text{Real part}\{(\int_{-\pi}^{\pi}\int_{0}^{\infty} r^4 \exp(j3\theta)L(r,\theta)\,dr\,d\theta)(\int_{-\pi}^{\pi}\int_{0}^{\infty} r^4 \exp(-j\theta)L(r,\theta)\,dr\,d\theta)^3\}$$

Thus,

$$MI5 = RP\{\Psi_r(4,\Psi_\theta(L.\exp(j3\theta)).\Psi_r^3(4,\Psi_\theta(L.\exp(-j\theta))\}$$

The rest which are derivable in a manner similar to those adopted for MI1 and MI5 are listed below;

$$MI2 = |\Psi_r(3,\Psi_\theta(L.\exp(j2\theta))|^2$$

$$MI3 = |\Psi_r(4,\Psi_\theta(L.\exp(j3\theta))|^2$$

$$MI4 = |\Psi_r(4,\Psi_\theta(L.\exp(j\theta))|^2$$

$$MI6 = RP\{\Psi_r(3,\Psi_\theta(L.\exp(j2\theta)).\Psi_r^2(4,\Psi_\theta(L.\exp(-j\theta))\}$$

$$MI7 = IP\{\Psi_r(4,\Psi_\theta(L.\exp(j3\theta)).\Psi_r^3(4,\Psi_\theta(L.\exp(-j\theta))$$

RP and IP represent real part and imaginary part of a complex number.

The radial dependence of the moment invariants thus becomes apparent and are listed below;

$MI1 \propto r^3$

$MI2 \propto r^3$

$MI3 \propto r^4$

$MI4 \propto r^4$

$MI5 \propto r^4$

$MI6 \propto r^3$ and $r^4$

$MI7 \propto r^4$

(II) DISCRETE RADIAL AND ANGULAR MOMENT INVARIANTS.

The general radial and angular moment invariant can be expressed as,

$$G = |\Psi_r(k, \Psi_\theta(L.\exp(jm\theta))|^2 \quad \ldots.(A1\text{-}8)$$

If m, the angular factor is chosen to be unity, (A1-8) can be written as,

$$G = |\Psi_r(k, \Psi_\theta(1,0,L)) + j\Psi_r(k, \Psi_\theta(0,1,L))|^2$$
$$= A^2 + B^2$$

where,

$$A = \Psi_r(k, \Psi_\theta(1,0,L))$$
$$B = \Psi_r(k, \Psi_\theta(0,1,L))$$

For our purpose a discrete implementation is needed. A discrete formulation is thus,

$$A = \sum_{r=0}^{R} \sum_{-\pi}^{\pi} r^k L(r,\theta)\cos\theta$$
$$B = \sum_{r=0}^{R} \sum_{-\pi}^{\pi} r^k L(r,\theta)\sin\theta$$

Owing to the rectangular grid sampling and quatization of digital images, the angular and radial moment invariants must be derived through appropriate interpolation schemes. This is especially important when considering values to be assumed by $L(r,\theta)$ for combinations of r and $\theta$ not falling on defined x,y coordinate points in the image.

## APPENDIX A-2.

## LISTING OF COMPUTER PROGRAMS DEVELOPED FOR THE SIMULATION OF ALGORITHMS.

The simulations for this research were done on the VAX11/780 of the Imperial College Centre for Remote Sensing Image Processing Laboratory and several routines were developed to enhance the investigations. The relevant ones form the body of this appendix. Apart from the system routines which are used for reading and writing out images, the program codes have been written in FORTRAN 77.

All the programs were developed by the investigator (P.O.OGUNBONA) during the period of the research. Great care has been exercised in preparing the programs in the form in which they now appear and typographical error should not exist at all. However if there is any such error, the original programs are available on a magnetic tape deposited with the Digital Communications Section of the Imperial College Electrical Engineering Department.

The programs are self explanatory and the documentation in terms of comments should make them meaningful. They have been collated in the order in which they appear in the thesis.

P.O.OGUNBONA
FEBRUARY 1987

```
C=================================================================
C                        PROGRAM FINCODE
C     PROGRAM IMPLEMENTS  MOTION COMPENSATED INTERFRAME DPCM
C     USING MOTION ESTIMATES OBTAINED VIA THE METHOD OF
C                INVARIANT MOMENTS OF IMAGE BLOCKS.
C-----------------------------------------------------------------
      PARAMETER(NX=256,NY=256)
      INTEGER*2 XSIZE,YSIZE,M1,M2,NEAT
      INTEGER STATUSFLAG,LINENUMBER
      INTEGER*2 M3,M4,DENOM,NDR,NDC,KT1,KT2,KTEMP,K1,K2
      INTEGER*2 IMAGE1(NX,NY),IMAGE2(NX,NY),PIC(32,32)
      INTEGER*2 XD(NX,NY),YD(NX,NY),ICOUNT,CODPIC(NX,NY)
      INTEGER*2 PIC1(32,32),EX,EY,KEMP1,KEMP2,KEMP3,DFD(16,16)
      INTEGER*2 KEMP4,NC1,NC2,NC3,NC4,MOTI,MOTJ,KSUM
      REAL AVFD, FDTH, DENOM1, BTOM, CVAR, DPG, SUM1
      REAL VAR1,VAR2,SUM5,TEMP1,INVM1(2),INVM2(2),AVGS,AVG
      REAL MINDAT,THRESH,SSA,VARPER,PERG,SUMZ,SUMX,VAR,RMS1
      REAL METRIC(-25:25,-25:25),KOUNT1,COUNT1,COUNT2,DPCV
      REAL  SUM2, ADI, SNR1, SNR2, SNR3, UCPEV, AVRG3, AVRG4
      REAL MS1,MS2
C
      CALL VICINIT('FIN2')
      CALL OPENV(STATUSFLAG,2,0,0,0,0)
      CALL OPENV(STATUSFLAG,3,0,0,0,0)
      DO LINENUMBER=1,NY
         CALL READ(STATUSFLAG,2,0,1,0,NX,IMAGE1(1,LINENUMBER),0)
         CALL READ(STATUSFLAG,3,0,1,0,NX,IMAGE2(1,LINENUMBER),0)
      ENDDO
C
      TYPE*,'INPUT THE WINDOW SIZE;XSIZE AND YSIZE'
      EAD(5,*)XSIZE,YSIZE
      TYPE*,'INPUT THE BOUNDS ON THE SEARCH AREA;MOTX AND MOTY'
      READ(5,*)MOTX,MOTY
C
C-----------------------------------------------------------
C
         DENOM=XSIZE*YSIZE
         BTOM=FLOAT(NX*NY)
         KOUNT1=0.0
         COUNT1=0.0
         COUNT2=0.0
         AVRG3=0.0
         AVRG4=0.0
         SUM1=0.0
         TEMP1=0.0
         ADI= 0.0
         DO 5 J=1,NY
```

```
            DO 5 I=1,NX
              COUNT1= FLOAT(IMAGE2(I,J))+COUNT1
              COUNT2= FLOAT(IMAGE2(I,J))**2.0+ COUNT2
              TEMP1=FLOAT(IMAGE2(I,J)-IMAGE1(I,J))
              SUM1= SUM1+TEMP1
              KOUNT1= (TEMP1**2.0)+ KOUNT1
  5       CONTINUE
          AVG= COUNT1/BTOM
          AVGS= COUNT2/BTOM
          DPCV= (KOUNT1/BTOM)-((SUM1/BTOM)**2.0)
          VAR= AVGS- (AVG**2.0)
          TYPE*,'SIGNAL VARIANCE=',VAR
          FDTH= SQRT(KOUNT1/BTOM)
C
          M1=1
          M2=XSIZE
          M3=1
          M4=YSIZE
C
          NDX=NX/XSIZE
          NDY=NY/YSIZE
C
          ICOUNT=0
          NEAT=0
          SUMZ=0.0
          SUMX=0.0
            DO 200 KT2=1,NDY
              DO 100 KT1=1,NDX
               KOUNT1=0.0
             KTEMP=0
             DO 10 J= M3, M4
              DO 10 I= M1, M2
               KTEMP=(IMAGE2(I,J)-IMAGE1(I,J))
                KOUNT1= KOUNT1+ FLOAT(KTEMP)**2.0
  10           CONTINUE
            AVFD=SQRT(KOUNT1/DENOM)
C------SEGMENT INTO MOVING/STATIONARY AREAS-------------
            IF(AVFD .GT. FDTH)THEN
             NEAT= NEAT+ 1
             K2= 1
             DO 30 J= M3, M4
                K1= 1
                DO 20 I= M1, M2
                  PIC(K1,K2)= IMAGE2(I,J)
                  K1= K1+ 1
  20            CONTINUE
                K2= K2+ 1
```

```
  30          CONTINUE
              CALL MOMINV(PIC,XSIZE,YSIZE,INVM1)
C
              MINDAT=999.0
              EY=0
              EX=0
C-----------------MATCHING USING MOMENT INVARIANTS
              DO 40 J=-MOTY,MOTY,1
                   NC3=M3+J
                   NC4=M4+J
                   DO 40 I=-MOTX,MOTX,1
                     NC1=M1+I
                     NC2=M2+I
        IF((NC1.LT.1).OR.(NC2.GT.NX).OR.(NC3.LT.1).OR.(NC4.GT.NY))THE
                  METRIC(I,J)=10000.0
      ELSE
                  K2=1
                  DO 43 J1=NC3,NC4
                    K1=1
                    DO 42 I1=NC1,NC2
                      PIC1(K1,K2)=IMAGE1(I1,J1)
                      K1=K1+1
  42                CONTINUE
                    K2=K2+1
  43              CONTINUE
                  KOUNT1=0.0
                  DO 44 IQ=1,YSIZE
                    DO 44 IZ=1,XSIZE
             KOUNT1=KOUNT1+(FLOAT(PIC1(IZ,IQ)-PIC(IZ,IQ))**2)
  44               CONTINUE
                   SSA=SQRT(KOUNT1/FLOAT(DENOM))
C------SPEED UP ----------------------
                   IF(SSA.LE.FDTH)THEN
                   CALL MOMINV(PIC1,XSIZE,YSIZE,INVM2)
                     SUM5=0.0
                       DO   L= 1, 2
                         TEMP1=INVM1(L)-INVM2(L)
                         SUM5=SUM5+ (TEMP1**2.0)
                       END DO
                       METRIC(I,J)=SUM5/2.0
C
                  IF(METRIC(I,J).LT.MINDAT)THEN
                       MINDAT=METRIC(I,J)
                       EX=I
                       EY=J
                  ENDIF
C
```

```
          ENDIF
          ENDIF
   40      CONTINUE
C
          KEMP1=0
          KEMP2=0
            K2=1
          DO  J= M3, M4
                K1=1
                DO  I= M1, M2
                  I1=I+EX
                  J1=J+EY
                  KEMP1=IMAGE2(I,J)-IMAGE1(I1,J1)
                  SUMX=FLOAT(KEMP1)+SUMX
                  SUMZ=(FLOAT(KEMP1)**2.0)+SUMZ
                  DFD(K1,K2)= KEMP1
                  KEMP1= IMAGE2(I,J)- IMAGE1(I,J)
                  AVRG3= FLOAT(KEMP1) +AVRG3
                  AVRG4= (FLOAT(KEMP1)** 2.0) +AVRG4
                  K1=K1+1
                  END DO
                K2=K2+1
              END DO
            CALL CODE ( DFD, XSIZE, YSIZE)
            K2=1
            DO   J= M3, M4
              K1=1
              DO  I= M1, M2
                ADI= ADI+ FLOAT(DFD(K1,K2))**2.0
                CODPIC(I,J)= IMAGE1(I+EX,J+EY)+ DFD(K1,K2)
                K1= K1+ 1
                END DO
              K2= K2+ 1
              END DO
         ELSE
            K2= 1
            DO  J= M3, M4
                K1= 1
                DO  I= M1, M2
                    KEMP1= IMAGE2(I,J)- IMAGE1(I,J)
                    DFD(K1,K2)= KEMP1
                    K1= K1+ 1
                  END DO
                K2= K2+ 1
              END DO
            CALL CODE ( DFD, XSIZE, YSIZE)
            K2= 1
```

```
            DO  J= M3, M4
              K1= 1
              DO  I= M1, M2
                CODPIC(I,J)= IMAGE1(I,J) + DFD(K1,K2)
                  K1= K1+ 1
                END DO
              K2= K2+ 1
               END DO
        END IF
               M1=M2+1
               M2=M2+XSIZE
100       CONTINUE
           M1=1
           M2=XSIZE
           M3=M4+1
           M4=M4+YSIZE
200    CONTINUE
       DENOM1=FLOAT(NEAT)*FLOAT(DENOM)
       UCPEV= (AVRG4/BTOM)- ((AVRG3/BTOM)**2.0)
       TYPE*,'UNCOMPENSATED PREDICTION ERROR VARI=',UCPEV
       VARPER=(SUMZ/BTOM)-((SUMX/BTOM)**2.0)
       TYPE*,'PREDICTION ERROR VARIANCE=',VARPER
       MS1= AVRG4/DENOM1
       MS2= SUMZ/DENOM1
       TYPE*,'UNCOMPENSATED MSE= ',MS1
       TYPE*,'COMPENSATED MSE= ',MS2
       PERG=10.0*(ALOG10(VAR/VARPER))
       TYPE*,'PREDICTION GAIN=',PERG
       SUM1= 0.0
       SUM2= 0.0
       TEMP1= 0.0
         DO  J= 1, NY
          DO  I= 1, NX
            TEMP1= (FLOAT(CODPIC(I,J))**2.0)
            SUM1= SUM1+ TEMP1
            TEMP1= FLOAT(CODPIC(I,J)- IMAGE2(I,J))**2.0
            SUM2= SUM2+ TEMP1
            END DO
       END DO
         SNR1= 10.0* ALOG10( SUM1/SUMZ)
       TYPE*,'SIGNAL-TO-NOISE RATIO IN M/A=',SNR1
       SNR2= 10.0* ALOG10(SUM1/ SUM2)
       TYPE*,'SIGNAL-TO-NOISE RATIO OVER IMAGE=', SNR2
       SNR3= 10.0* ALOG10(SUM1/ADI)
       TYPE*,'SNR3=',SNR3
C
         CALL OPENV(STATUSFLAG,1,1,0,0,0)
```

```
      DO LINENUMBER=1,NY
         CALL WRITE(STATUSFLAG,1,0,1,0,NX,CODPIC(1,LINENUMBER),0)
      ENDDO
      CALL RELAB2(1,NY,NX)
      END
C=============================================================
C                    SUBROUTINE MOMINV
C-------------------------------------------------------------
            SUBROUTINE MOMINV(IPICT,MW,MH,INVMC)
            PARAMETER MZ=2
            INTEGER*2 COUNT1,COUNT2,MH,MW,IPICT(32,32)
            REAL MMENT(0:MZ,0:MZ),ISUM1,MENT
            REAL XBAR,YBAR,TEMP(12),BUF(6)
            INTEGER*4 I1,J1
            REAL CM(0:MZ,0:MZ),CMN(0:MZ,0:MZ)
            REAL EMP1,EMP2,EMP3,EMP4,INVMC(2)
C
C------------------MODULE TO COMPUTE THE ORDINARY MOMENTS
C
            DO 5 COUNT2=0,MZ
               DO 5 COUNT1=0,MZ
                  ISUM1=0.0
                   DO 10 IY=1,MH
                    DO 10 JX=1,MW
                      I1=(JX)**COUNT1
                      J1=(IY)**COUNT2
                     MENT=0.0
           IF(IPICT(JX,IY).EQ.0)IPICT(JX,IY)=1
              MENT=FLOAT(IPICT(JX,IY))*(FLOAT(I1))*(FLOAT(J1))
                 ISUM1=ISUM1+ABS(MENT)
 10                CONTINUE
                   MMENT(COUNT1,COUNT2)=ISUM1
 5          CONTINUE
C
C---------------COMPUTATION OF CENTRAL MOMENTS NEEDED
C
C
            XBAR=MMENT(1,0)/MMENT(0,0)
            YBAR=MMENT(0,1)/MMENT(0,0)
            CM(0,0)=MMENT(0,0)
            CM(1,0)=0.0
            CM(0,1)=0.0
            CM(2,0)=MMENT(2,0)-XBAR*MMENT(1,0)
            CM(0,2)=MMENT(0,2)-YBAR*MMENT(0,1)
            CM(1,1)=MMENT(1,1)-YBAR*MMENT(1,0)
C
C----------------COMPUTATION OF NORMALISED  CENTRAL MOMENTS
```

```
C
C
              CMN(1,1)=CM(1,1)/(CM(0,0)**2.0)
              CMN(0,2)=CM(0,2)/(CM(0,0)**2.0)
              CMN(2,0)=CM(2,0)/(CM(0,0)**2.0)
C
C-----------------COMPUTATION OF INVARIANT MOMENTS
C
C
              INVMC(1)=CMN(2,0)+CMN(0,2)
              INVMC(2)=((CMN(2,0)-CMN(0,2))**2.0)+4.0*(CMN(1,1)**2.0)
              RETURN
              END
C=============================================================
C---------SUBROUTINE CODE: OBTAINS PREDICTION ERRORS AS INPUT
C-------------OUTPUT IS QUANTIZED PREDICTION ERRORS
C-------------------------------------------------------------
      SUBROUTINE CODE(PERR, XSIZ, YSIZ)
               INTEGER*2 XSIZ, YSIZ, PERR(16,16), IBUF, K1
               REAL  BUF(35), RECLV(25), MINERR
               DATA RECLV/-182,-102,-87,-77,-66,-55,-44,
     +           -35,-26,-17,-10,-4,0,4,10,17,26,35,44,
     +           55,66,77,87,102,182/
               DO J= 1, YSIZ
                   DO I= 1, XSIZ
                       DO K= 1, 25
                              IBUF= RECLV(K)- PERR(I,J)
                              BUF(K)= FLOAT(IBUF) **2.0
                       END DO
                       MINERR= BUF(1)
                       DO K= 1, 25
                            IF( BUF(K) .LE. MINERR)THEN
                              MINERR= BUF(K)
                              K1= K
                            END IF
                       END DO
                       PERR(I,J)= RECLV(K1)
                   END DO
               END DO
               RETURN
               END
```

```
C=======================================================
C          SUBROUTINE MOMINV : COMPUTES THE SEVEN MOMENT
C          INVARIANTS GIVEN A BLOCK OF PIXELS
C-------------------------------------------------------
      SUBROUTINE MOMINV(IPICT,MW,MH,INVMC)
      PARAMETER MZ=3
      INTEGER*2 COUNT1,COUNT2,MH,MW,IPICT(32,32)
      REAL MMENT(0:MZ,0:MZ),ISUM1,MENT
      REAL XBAR,YBAR,TEMP(12),BUF(6)
      INTEGER*4 I1,J1
      REAL CM(0:MZ,0:MZ),CMN(0:MZ,0:MZ)
      REAL EMP1,EMP2,EMP3,EMP4,INVMC(7)
C
C----------------MODULE TO COMPUTE THE ORDINARY MOMENTS
C
      DO 5 COUNT2=0,MZ
            DO 5 COUNT1=0,MZ
                ISUM1=0.0
                DO 10 IY=1,MH
                  DO 10 JX=1,MW
                    I1=(JX)**COUNT1
                    J1=(IY)**COUNT2
                    MENT=0.0
           IF(IPICT(JX,IY).EQ.0)IPICT(JX,IY)=1
           MENT=FLOAT(IPICT(JX,IY))*(FLOAT(I1))*(FLOAT(J1))
                   ISUM1=ISUM1+ABS(MENT)
  10            CONTINUE
                MMENT(COUNT1,COUNT2)=ISUM1
  5         CONTINUE
C
C---------------COMPUTATION OF CENTRAL MOMENTS NEEDED
C
      XBAR=MMENT(1,0)/MMENT(0,0)
      YBAR=MMENT(0,1)/MMENT(0,0)
      CM(0,0)=MMENT(0,0)
      CM(1,0)=0.0
      CM(0,1)=0.0
      CM(2,0)=MMENT(2,0)-XBAR*MMENT(1,0)
      CM(0,2)=MMENT(0,2)-YBAR*MMENT(0,1)
      CM(1,1)=MMENT(1,1)-YBAR*MMENT(1,0)
      TEMP(1)=MMENT(3,0)-(3.0*XBAR*MMENT(2,0))
      CM(3,0)=TEMP(1)+(2.0*MMENT(1,0)*(XBAR**2.0))
      TEMP(2)=MMENT(1,2)-(2.0*YBAR*MMENT(1,1))
      TEMP(3)=(XBAR*MMENT(0,2))-(2.0*(YBAR**2.0)*MMENT(1,0))
      CM(1,2)=TEMP(2)-TEMP(3)
      EMP1=MMENT(2,1)
      EMP2=(2.0*XBAR*(MMENT(1,1)))
      TEMP(4)=EMP1-EMP2
      EMP3=YBAR*MMENT(2,0)
      EMP4=2.0*(XBAR**2.0)*(MMENT(0,1))
      TEMP(5)=EMP3-EMP4
      CM(2,1)=TEMP(4)-TEMP(5)
      TEMP(6)=MMENT(0,3)-(3.0*YBAR*MMENT(0,2))
      CM(0,3)=TEMP(6)+(2.0*(YBAR**2.0)*(MMENT(0,1)))
C
```

```
C------------------COMPUTATION OF NORMALISED  CENTRAL MOMENTS
C
      CMN(1,1)=CM(1,1)/(CM(0,0)**2.0)
      CMN(1,2)=CM(1,2)/(CM(0,0)**2.5)
      CMN(2,1)=CM(2,1)/(CM(0,0)**2.5)
      CMN(3,0)=CM(3,0)/(CM(0,0)**2.5)
      CMN(0,3)=CM(0,3)/(CM(0,0)**2.5)
      CMN(0,2)=CM(0,2)/(CM(0,0)**2.0)
      CMN(2,0)=CM(2,0)/(CM(0,0)**2.0)
C
C-----------------COMPUTATION OF INVARIANT MOMENTS
C
INVMC(1)=CMN(2,0)+CMN(0,2)
INVMC(2)=((CMN(2,0)-CMN(0,2))**2.0)+4.0*(CMN(1,1)**2.0)
BUF(1)=CMN(3,0)-3.0*CMN(1,2)
BUF(2)=3.0*CMN(2,1)-CMN(0,3)
BUF(3)=CMN(3,0)+CMN(1,2)
BUF(4)=CMN(2,1)+CMN(0,3)
BUF(5)=3.0*CMN(1,2)-CMN(0,3)
BUF(6)=3.0*CMN(1,2)-CMN(3,0)
INVMC(3)=(BUF(1)**2.0)+(BUF(2)**2.0)
INVMC(4)=(BUF(3)**2.0)+(BUF(4)**2.0)
TEMP(7)=(BUF(1)*BUF(3))*(BUF(3)**2.0-3.0*(BUF(4)**2.0))
TEMP(8)=(BUF(2)*BUF(4))*(3.0*(BUF(3)**2.0)-BUF(4)**2.0)
INVMC(5)=TEMP(7)+TEMP(8)
TEMP(9)=(CMN(2,0)-CMN(0,2))*((BUF(3)**2.0)-(BUF(4)**2.0))
TEMP(10)=4.0*CMN(1,1)*BUF(3)*BUF(4)
INVMC(6)=TEMP(9)+TEMP(10)
TEMP(11)=(BUF(6)*BUF(3))*((BUF(3)**2.0)-3.0*(BUF(4)**2.0))
TEMP(12)=(BUF(5)*BUF(4))*((3.0*(BUF(3)**2.0))-(BUF(4)**2.0))
INVMC(7)=TEMP(11)+TEMP(12)
RETURN
END
```

```
C=================================================================
C    SUBROUTINE RADINV: COMPUTES THE RADIAL AND ANGULAR INVARIANT
C                              MOMENTS
C-----------------------------------------------------------------
        SUBROUTINE RADINV (IPIX,XSIZ,YSIZ,INVMC)
        INTEGER*2 XSIZ,YSIZ
        INTEGER*4 XCENT,YCENT,XMAS,YMAS,MASS
        INTEGER*2 KDATA(4),TDATA(4),IPIX(XSIZ,YSIZ)
        REAL DELTAR,DTHETA,SUMA,SUMB,INVMC(4),THETA1
        REAL GEE1,GEE2,A1,C1,S1,G1,BIGA,BIGB,THETA,AR
        REAL SUM1,SUM2,SUM3, L1A, L2A, L1, L2, L3, L4
        INTEGER*2 CPIX(-32:32,-32:32)
        DATA(KDATA(I),I=1,4)/1,-1,2,-2/
        DATA(TDATA(I),I=1,4)/1,1,1,1/
        SUM1= 0.0
        SUM2= 0.0
        SUM3= 0.0
        DO  J= 1, YSIZ
             DO  I= 1, XSIZ
                SUM1= SUM1+ FLOAT(IPIX(I,J))
                SUM2= SUM2+ FLOAT(IPIX(I,J)*I)
                SUM3= SUM3+ FLOAT(IPIX(I,J)*J)
             END DO
        END DO
        MASS= AINT(SUM1)
        XMAS= AINT(SUM2)
        YMAS= AINT(SUM3)
        XCENT= XMAS/MASS
        YCENT= YMAS/MASS
C
C
C=======PERFORM IMAGE AXIS SHIFT TO CENTROID=====
C
        DO  J= 1, YSIZ
              DO  I= 1, XSIZ
                 NI= I- XCENT
                 NJ= J- YCENT
                 CPIX(NI,NJ)= IPIX(I,J)
               END DO
        END DO
        L1=FLOAT(1-XCENT)
        L2=FLOAT(XSIZ-XCENT)
        L3=FLOAT(1-YCENT)
        L4=FLOAT(YSIZ-YCENT)
        L1A=AMAX1(ABS(L1),ABS(L2))
        L2A=AMAX1(ABS(L3),ABS(L4))
        RLIMIT=SQRT((L1A**2)+(L2A**2))
        DELTAR=RLIMIT/4.0
        DTHETA=6.284/(FLOAT(YSIZ+(XSIZ/2)))
C----COMPUTE VALUES FOR INVARIANT MOMENTS
        DO 40 I= 1,4
            INVMC(I)= 0.0
```

```
              SUMA= 0.0
              SUMB= 0.0
              KAY= KDATA(I)
              IFACT= TDATA(I)
              AR= 0.1
 60           CONTINUE
              THETA=-3.142
 50           CONTINUE
              A1= AR**(FLOAT(KAY))
              C1= COS(IFACT*THETA)
              S1= SIN(IFACT*THETA)
              G1= GRTETA(AR,THETA,L1,L2,L3,L4,CPIX)
              BIGA= A1*G1*C1
              BIGB= A1*G1*S1
              SUMA= SUMA+BIGA
              SUMB= SUMB+BIGB
              THETA= THETA+DTHETA
              IF(THETA.LT.3.142)GOTO 50
              AR=AR+DELTAR
              IF(RLIMIT.GT.AR)GOTO 60
              GEE1=SUMA**2.0
              GEE2=SUMB**2.0
              INVMC(I)=SQRT(GEE1+GEE2)
 40           CONTINUE
              RETURN
              END
C=============================================================
C        FUNCTION  GRTETA : OBTAINS THE VALUE OF G(r,0) USING
C                        INTERPOLATION
C-------------------------------------------------------------
              FUNCTION GRTETA(ARF,THETAF,J1,J2,J3,J4,CPIX)
              REAL ARF, THETAF, J1, J2, J3, J4, EX, WY
              INTEGER*2 WY1, EX1, EX2, WY2
              INTEGER*2 CPIX(-32:32,-32:32)
              EX=ARF*COS(THETAF)
              WY=ARF*SIN(THETAF)
        IF((EX.GE.J1).AND.(EX.LE.J2).AND.(WY.GE.J3).
     +     AND.(WY.LE.J4))THEN
C
              EX1= AINT(EX)
              WY1= AINT(WY)
              EX2= ANINT(EX)
              EY2= ANINT(EY)
              ITEMP1= CPIX(EX1,WY1)
              ITEMP2= CPIX(EX2,WY2)
              GRTETA=FLOAT((ITEMP1+ITEMP2)/2)
            ELSE
              GRTETA=0.0
            ENDIF
            RETURN
            END
```

```
C=================================================================
C            PROGRAM FOR ALGORITHM PDDT1
C PROGRAM ESTIMATING MOTION FROM PROJECTIONS OF FRAME,ELEMENT AND
C LINE DIFFERENCES
C-----------------------------------------------------------------
      PARAMETER(NX=256,NY=256)
      INTEGER*2 IMAGE1(NX,NY),IMAGE2(NX,NY),N1,N2,N3,N4
      INTEGER*2 IFD(NX,NY),ISUM,NEAT,I1,J1,DFD(32,32)
      INTEGER*2 CHOICE, XSIZE, YSIZE, KT1, KT2, K1, K2
      INTEGER*2 DENOM, ITEMP, I2, J2, CODPIC(NX,NY)
      INTEGER*2 KTEMP, I3, J3, NDX, NDY
      INTEGER STATUSFLAG,LINENUMBER
      REAL EDY(NY),FDY(NY),EDX(NX),LDX(NX),FDX(NX),LDY(NY)
      REAL NTEMP1,NTEMP2,NTEMP3,FDTH,AXM,AYM,BTOM,PERG
      REAL XLD(NX,NY),YLD(NX,NY),KOUNT1,X1,Y1,DENOM1
      REAL AVFD,VAR1,VAR2,XM(NX,NY),YM(NX,NY),DNOM
      REAL SUM1,SUM2,SUMX,SUMY,SUMW,TEMP,VAR3,VAR4
      REAL TUM1, TUM2, VUC, PR, PL, SUMT1, SUMT2
      CALL VICINIT('MOTION10')
      CALL OPENV(STATUSFLAG,2,0,0,0,0)
      CALL OPENV(STATUSFLAG,3,0,0,0,0)
      DO LINENUMBER=1,NY
        CALL READ(STATUSFLAG,2,0,1,0,NX,IMAGE1(1,LINENUMBER),0)
        CALL READ(STATUSFLAG,3,0,1,0,NX,IMAGE2(1,LINENUMBER),0)
      ENDDO
      TYPE*,'INPUT THE WINDOW SIZE;XSIZE AND YSIZE'
      READ(5,*)XSIZE,YSIZE
      CALL GRDNT2(IMAGE1,XLD,YLD)
      SUMX= 0.0
      SUMY=0.0
      TUM1= 0.0
      TUM2= 0.0
      SUMT1= 0.0
      SUMT2= 0.0
      DNOM= 0.0
       KTEMP= 0
      DENOM= XSIZE*YSIZE
      BTOM= FLOAT(NX*NY)
      KOUNT1=0.0
      DO  J= 1, NY
       DO  I= 1, NX
          IFD(I,J)= IMAGE2(I,J)- IMAGE1(I,J)
          KTEMP= IFD(I,J)
          KOUNT1= (FLOAT(KTEMP)**2.0)+ KOUNT1
          TUM1= TUM1+ FLOAT(IMAGE2(I,J))
          TUM2= TUM2+ (FLOAT(IMAGE2(I,J))**2.0)
         END DO
      END DO
      FDTH= SQRT(KOUNT1/BTOM)
C-------------------------------SIGNAL VARIANCE; VAR3
      VAR3=(TUM2/BTOM)-((TUM1/BTOM)**2.0)
      N1= 1
      N2= XSIZE
      N3= 1
      N4= YSIZE
```

```
      NDX= NX/XSIZE
      NDY= NY/YSIZE
      DO   KT2= 1, NDY
           DO   KT1= 1, NDX
C=============================================================
C                              MODULE 2
C                      THRESHOLDING OF BLOCKS

      KOUNT1=0.0
      KTEMP=0
      DO   J= N3, N4
       DO  I= N1, N2
         KTEMP= IMAGE2(I,J)- IMAGE1(I,J)
         KOUNT1= (FLOAT(KTEMP)**2.0)+ KOUNT1
         END DO
      END DO
      AVFD= SQRT(KOUNT1/DENOM)
      IF(AVFD.GT.FDTH)THEN
       DO J= N3, N4
        DO I= N1, N2
         SUMT1=SUMT1+ FLOAT(IFD(I,J))
         SUMT2=SUMT2+ (FLOAT(IFD(I,J))**2.0)
        END DO
       END DO
C=============================================================
C                           MODULE 3
C                COMPUTATION OF MOTION VECTORS
C-------------------------------------------------------------
          DO  I= N1, N2
              EDX(I)=0.0
              FDX(I)=0.0
              LDX(I)=0.0
              DO  J= N3, N4
                EDX(I)= EDX(I)- XLD(I,J)
                LDX(I)= LDX(I)- YLD(I,J)
                FDX(I)= FDX(I)+ FLOAT(IFD(I,J))
              END DO
        END DO
          DO   J= N3, N4
              EDY(J)=0.0
              LDY(J)=0.0
              FDY(J)=0.0
              DO  I= N1, N2
                EDY(J)= EDY(J)- XLD(I,J)
                LDY(J)= LDY(J)- YLD(I,J)
                FDY(J)= FDY(J)+ FLOAT(IFD(I,J))
              END DO
           END DO
        SUM1=0.0
        SUM2=0.0
          DO  J= N3, N4
              DO  I= N1, N2
              NTEMP1=0.0
              NTEMP2=0.0
              NTEMP3=0.0
```

```
            NTEMP1=(FDY(J)*EDX(I))-(FDX(I)*EDY(J))
            NTEMP2=(EDY(J)*LDX(I))-(EDX(I)*LDY(J))
            IF(NTEMP2.NE.0.0)THEN
              YM(I,J)=NTEMP1/NTEMP2
            ELSE
              YM(I,J)=0.0
            ENDIF
            IF(ABS(YM(I,J)) .LT. 7.0)THEN
              SUM1= SUM1+YM(I,J)
             ENDIF
            NTEMP3= (FDX(I)*LDY(J))-(FDY(J)*LDX(I))
            IF(NTEMP2.NE.0.0)THEN
              XM(I,J)= NTEMP3/NTEMP2
            ELSE
              XM(I,J)= 0.0
            ENDIF
            IF(ABS(XM(I,J)) .LT. 7.0)THEN
              SUM2=SUM2+XM(I,J)
            ENDIF
         END DO
       END DO
       AXM=SUM2/FLOAT(DENOM)
       AYM=SUM1/FLOAT(DENOM)
C
        K2=1
        DO  J= N3, N4
              K1=1
           DO  I= N1, N2
             DNOM= DNOM+ 1.0
       IF((ABS(AXM) .LT. 7.0).AND.(ABS(AYM) .LT. 7.0))THEN
          I1=I-AINT(AXM)
          J1=J-AINT(AYM)
          I2=I-ANINT(AXM)
          J2=J-ANINT(AYM)
       IF((I1.GE.1).AND.(I2.GE.1).AND.(J1.GE.1).AND.(J2.GE.1))THEN
          TEMP=(0.75*IMAGE1(I1,J1))+(0.25*IMAGE1(I2,J2))
          ITEMP=ANINT(TEMP)
          DFD(K1,K2)= IMAGE2(I,J)-ITEMP
       ELSE
          DFD(K1,K2)= IMAGE2(I,J)- IMAGE1(I,J)
       ENDIF
         ELSE
           DFD(K1,K2)= IMAGE2(I,J)- IMAGE1(I,J)
        ENDIF
          SUMX= SUMX+ FLOAT(DFD(K1,K2))
          SUMY= SUMY+ (FLOAT(DFD(K1,K2))**2.0)
          K1=K1+1
          END DO
         K2=K2+1
         END DO
         CALL CODE ( DFD, XSIZE, YSIZE)
C
C================================================ RECONSTRUCTION
          K2=1
          DO   J= N3, N4
```

```
        K1=1
        DO  I= N1, N2
       IF((ABS(AXM).LT.8.0).AND.(ABS(AYM).LT.8.0))THEN
            I1=I-AINT(AXM)
          J1=J-AINT(AYM)
          I2=I-ANINT(AXM)
          J2=J-ANINT(AYM)
  IF((I1.GE.1).AND.(I2.GE.1).AND.(J1.GE.1).AND.(J2.GE.1))THEN
        TEMP= (0.75*IMAGE1(I1,J1))+ (0.25*IMAGE1(I2,J2))
        ITEMP=ANINT(TEMP)
    ELSE
     ITEMP=IMAGE1(I,J)
  ENDIF
      ELSE
         ITEMP=IMAGE1(I,J)
      ENDIF
        CODPIC(I,J)= ITEMP+ DFD(K1,K2)
        K1=K1+1
        END DO
       K2=K2+1
      END DO
   ELSE
 K2= 1
   DO  J= N3, N4
      K1= 1
      DO  I= N1, N2
      DFD(K1,K2)= IMAGE2(I,J)- IMAGE1(I,J)
        K1= K1+ 1
      END DO
    K2= K2+ 1
    END DO
    CALL CODE ( DFD, XSIZE, YSIZE)
    K2= 1
    DO   J= N3, N4
       K1= 1
        DO  I= N1, N2
          CODPIC(I,J)= IMAGE1(I,J)+ DFD(K1,K2)
        K1= K1+ 1
        END DO
        K2= K2+ 1
    END DO
    ENDIF
    N1= N2+ 1
    N2= N2+ XSIZE
  END DO
  N1= 1
  N2= XSIZE
  N3= N4+ 1
  N4= N4+ YSIZE
 END DO
VAR1=(SUMY/BTOM)-((SUMX/BTOM)**2.0)
TYPE*,'COMPENSATED ERROR VARIANCE= ',VAR1
VUC= (SUMT2/BTOM)-((SUMT1/BTOM)**2.0)
TYPE*,'UNCOMPENSATED ERROR VARIANCE=',VUC
PERG=10.0*(ALOG10(VAR3/VAR1))
```

```
      TYPE*,'PREDICTION GAIN=',PERG
C
      MS1= SUMY/DNOM
      TYPE*,'COMPENSATED MSE= ',MS1
      MS2= SUMT2/DNOM
      TYPE*,'UNCOMPENSATED MSE= ',MS2
C
      KOUNT1= 0.0
      SUMW= 0.0
      TEMP= 0
      DO  J= 1, NY
        DO  I= 1, NX
          TEMP=FLOAT(CODPIC(I,J))**2.0
          SUMW= SUMW+ TEMP
          TEMP= FLOAT(CODPIC(I,J)- IMAGE2(I,J))**2.0
          KOUNT1=KOUNT1+ TEMP
          END DO
      END DO
      SNR1= 10.0* ALOG10(SUMW/SUMY)
      TYPE*,'SIGNAL-TO-NOISE RATIO IN M/AREA=',SNR1
      SNR2= 10.0* ALOG10(SUMW/KOUNT1)
      TYPE*,'SIGNAL-TO-NOISE RATIO OVER IMAGE=',SNR2
C
       CALL OPENV(STATUSFLAG,1,1,0,0,0)
       DO LINENUMBER=1,NY
           CALL WRITE(STATUSFLAG,1,0,1,0,NX,CODPIC(1,LINENUMBER),0)
       ENDDO
       CALL RELAB2(1,NY,NX)
       END
```

```
C=============================================================
C              PROGRAM FOR ALGORITHM PDDT2
C PROGRAM ESTIMATING MOTION FROM PROJECTIONS OF FRAME,ELEMENT AND
C LINE DIFFERENCES:LEAST SQUARES ESTIMATION IS USED.
C-------------------------------------------------------------
      PARAMETER(NX=256,NY=256)
      INTEGER*2 IMAGE1(NX,NY),IMAGE2(NX,NY),N1,N2,N3,N4
      INTEGER*2 IFD(NX,NY),ISUM,IPIC(NX,NY),NEAT,I1,I2,J1,J2
      INTEGER*2 CHOICE,XSIZE,YSIZE,DENOM,DFD(32,32)
      INTEGER*2 CODPIC(NX,NY),IPIC2(NX,NY),NDY,NDX
      INTEGER*2 KT1,KT2,K1,K2, KTEMP, ITEMP, K
      INTEGER STATUSFLAG,LINENUMBER
      REAL EDY(NY),FDY(NY),EDX(NX),LDX(NX)
      REAL NTEMP1,NTEMP2,NTEMP3,FDTH,CX,CY
      REAL XLD(NX,NY),YLD(NX,NY),KOUNT1,VAR1,VAR2
      REAL AVFD, FDX(NX), LDY(NY), BTOM, PERG, DNET
      REAL GY(32,2),GX(32,2),FG(32),GYT(2,32),GXT(2,32)
      REAL GTG(2,2),GTE(2,2),GINV(2,2),MINV(2,2),GXG(2,2)
      REAL DV1(2),DV2(2),GXF(2),GTF(2),SUMX,SUMY,SUMW
      REAL TEMP,VAR3,VAR4,DET,VPR, DPG, SUM, SUMT1, SUMT2
      REAL VUC, TUM1, TUM2, TUM3, ADI
      CALL VICINIT('NEWMOT')
      CALL OPENV(STATUSFLAG,2,0,0,0,0)
      CALL OPENV(STATUSFLAG,3,0,0,0,0)
      DO LINENUMBER=1,NY
        CALL READ(STATUSFLAG,2,0,1,0,NX,IMAGE1(1,LINENUMBER),0)
        CALL READ(STATUSFLAG,3,0,1,0,NX,IMAGE2(1,LINENUMBER),0)
      ENDDO
      TYPE*,'INPUT THE WINDOW SIZE;XSIZE AND YSIZE'
      READ(5,*)XSIZE,YSIZE
      CALL GRDNT1(IMAGE1,XLD,YLD)
      DNET= 0.0
      SUMX= 0.0
      SUMY= 0.0
      TUM1= 0.0
      TUM2= 0.0
      SUMT1= 0.0
      SUMT2= 0.0
      ADI= 0.0
      KTEMP=0
      DENOM= XSIZE* YSIZE
      BTOM= FLOAT(NX*NY)
      KOUNT1= 0.0
      DO  J= 1, NY
       DO  I= 1, NX
          IFD(I,J)= IMAGE2(I,J)- IMAGE1(I,J)
          KTEMP= IFD(I,J)
          KOUNT1= (FLOAT(KTEMP)**2.0)+ KOUNT1
          TUM1= TUM1+ FLOAT(IMAGE2(I,J))
          TUM2= TUM2+ (FLOAT(IMAGE2(I,J))**2.0)
          END DO
        END DO
      FDTH= SQRT(KOUNT1/BTOM)
C-------------------------------SIGNAL VARIANCE; VAR3
      VAR3= (TUM2/BTOM)- ((TUM1/BTOM)**2.0)
```

```
          N1= 1
          N2= XSIZE
          N3= 1
          N4= YSIZE
          NDX= NX/XSIZE
          NDY= NY/YSIZE
          DO    KT2= 1, NDY
             DO    KT1= 1, NDX
C=====================================THRESHOLDING OF BLOCKS
C
       KOUNT1= 0.0
       KTEMP= 0
       DO  J= N3, N4
        DO  I= N1, N2
          KTEMP= IMAGE2(I,J)- IMAGE1(I,J)
          KOUNT1= (FLOAT(KTEMP)**2.0)+ KOUNT1
          END DO
       END DO
         AVFD= SQRT(KOUNT1/DENOM)
         IF(AVFD .GT. FDTH)THEN
            DO J= N3, N4
            DO I= N1, N2
              DNET= DNET+ 1.0
              SUMT1= SUMT1+ FLOAT(IFD(I,J))
              SUMT2= SUMT2+ (FLOAT(IFD(I,J))**2.0)
            END DO
          END DO
C=====================================COMPUTATION OF MOTION VECTORS
C
C==========================Y-COMPONENT==========================
         L=1
           DO 45 J=N3,N4
              EDY(J)=0.0
              LDY(J)=0.0
              FDY(J)=0.0
              DO 35 I=N1,N2
                EDY(J)=EDY(J)+XLD(I,J)
                LDY(J)=LDY(J)+YLD(I,J)
                FDY(J)=FDY(J)+FLOAT(IFD(I,J))
  35          CONTINUE
            GY(L,1)=-1.0*EDY(J)
            GY(L,2)=-1.0*LDY(J)
            FG(L)=FDY(J)
            L=L+1
  45       CONTINUE
C MATRIX ----------------------ALGEBRA
         DO 55 I=1,YSIZE
           DO 55 J=1,2
             GYT(J,I)=GY(I,J)
  55       CONTINUE
           DO 85 I=1,2
           DO 85 J=1,2
             SUM=0.0
              DO 65 K=1,YSIZE
                SUM=SUM+GYT(I,K)*GY(K,J)
```

```
 65            CONTINUE
              GTG(I,J)=SUM
 85         CONTINUE
C------MATRIX INVERSION
              DET=(GTG(1,1)*GTG(2,2))-(GTG(1,2)*GTG(2,1))
            IF(DET.NE.0)THEN
           MINV(1,1)=GTG(2,2)/DET
           MINV(1,2)=-GTG(1,2)/DET
           MINV(2,1)=-GTG(2,1)/DET
           MINV(2,2)=GTG(1,1)/DET
          DO 95 I=1,2
           SUM=0.0
           DO 105 J=1,YSIZE
             SUM=SUM+GYT(I,J)*FG(J)
 105         CONTINUE
           GTF(I)=SUM
 95         CONTINUE
          DO 120 I=1,2
            SUM=0.0
            DO 110 J=1,2
              SUM=SUM+MINV(I,J)*GTF(J)
 110          CONTINUE
            DV1(I)=SUM
 120        CONTINUE
          ELSE
           DO I=1,2
             DV1(I)=0
           ENDDO
          ENDIF
C=========================X - COMPONENT====================
                    K=1
              DO 140 I=N1,N2
               EDX(I)=0.0
               LDX(I)=0.0
               FDX(I)=0.0
               DO 130 J=N3,N4
                 EDX(I)=EDX(I)+XLD(I,J)
                 LDX(I)=LDX(I)+YLD(I,J)
                 FDX(I)=FDX(I)+FLOAT(IFD(I,J))
 130           CONTINUE
            GX(K,1)=-1.0*EDX(I)
            GX(K,2)=-1.0*LDX(I)
            FG(K)=FDX(I)
            K=K+1
 140         CONTINUE
           DO 150 I=1,XSIZE
             DO 150 J=1,2
               GXT(J,I)=GX(I,J)
 150         CONTINUE
           DO 180 I=1,2
             DO 180 J=1,2
               SUM=0.0
               DO 170 K=1,XSIZE
                SUM=SUM+GXT(I,K)*GX(K,J)
 170             CONTINUE
```

```
              GXG(I,J)=SUM
 180        CONTINUE
C------------MATRIX  INVERSION-------------
          DET=(GXG(1,1)*GXG(2,2))-(GXG(1,2)*GXG(2,1))
          IF(DET.NE.0)THEN
          GINV(1,1)=GXG(2,2)/DET
          GINV(1,2)=-GXG(1,2)/DET
          GINV(2,1)=-GXG(2,1)/DET
          GINV(2,2)=GXG(1,1)/DET
          DO 205 I=1,2
            SUM=0.0
            DO 190 J=1,XSIZE
             SUM=SUM+GXT(I,J)*FG(J)
 190          CONTINUE
            GXF(I)=SUM
 205        CONTINUE
          DO 220 I=1,2
             SUM=0.0
             DO 210 J=1,2
                SUM=SUM+GINV(I,J)*GXF(J)
 210         CONTINUE
             DV2(I)=SUM
 220      CONTINUE
          ELSE
            DO I=1,2
              DV2(I)=0
            ENDDO
          ENDIF
          CY= DV1(2)
          CX= DV2(1)
          K2= 1
          DO   J= N3, N4
            K1= 1
            DO   I= N1, N2
       IF((ABS(CX) .LT. 7.0) .AND. (ABS(CY) .LT. 7.0))THEN
              I1= I- AINT(CX)
              J1= J- AINT(CY)
              I2= I- ANINT(CX)
              J2= J- ANINT(CY)
        IF((I1.GE.1).AND.(I2.GE.1).AND.(J1.GE.1).AND.(J2.GE.1))THEN
         TEMP=(0.75*IMAGE1(I1,J1))+ (0.25*IMAGE1(I2,J2))
          ITEMP= ANINT(TEMP)
         DFD(K1,K2)= IMAGE2(I,J)- ITEMP
       ELSE
         DFD(K1,K2)= IMAGE2(I,J)- IMAGE1(I,J)
       ENDIF
         ELSE
         DFD(K1,K2)= IMAGE2(I,J)- IMAGE1(I,J)
       ENDIF
              SUMX= SUMX+ FLOAT(DFD(K1,K2))
            SUMY= SUMY+ (FLOAT(DFD(K1,K2))**2.0)
            K1= K1+ 1
          END DO
         K2= K2+ 1
           END DO
```

```
              CALL CODE ( DFD, XSIZE, YSIZE)
C
C================================================ RECONSTRUCTION
            K2= 1
            DO  J= N3, N4
              K1= 1
              DO  I= N1, N2
       IF((ABS(CX) .LT. 7.0) .AND. (ABS(CY) .LT. 7.0))THEN
               I1=I-AINT(CX)
             J1=J-AINT(CY)
             I2=I-ANINT(CX)
             J2=J-ANINT(CY)
         IF((I1.GE.1).AND.(I2.GE.1).AND.(J1.GE.1).AND.(J2.GE.1))THEN
            TEMP= (0.75*IMAGE1(I1,J1))+ (0.25*IMAGE1(I2,J2))
            ITEMP= ANINT(TEMP)
          ELSE
           ITEMP= IMAGE1(I,J)
         ENDIF
        ELSE
          ITEMP= IMAGE1(I,J)
        ENDIF
             CODPIC(I,J)= ITEMP+ DFD(K1,K2)
             ADI= ADI+ (FLOAT(DFD(K1,K2))**2.0)
             K1=K1+1
          END DO
             K2=K2+1
          END DO
         ELSE
        K2= 1
          DO  J= N3, N4
             K1= 1
             DO  I= N1, N2
             DFD(K1,K2)= IMAGE2(I,J)- IMAGE1(I,J)
               K1= K1+ 1
             END DO
          K2= K2+ 1
          END DO
             CALL CODE ( DFD, XSIZE, YSIZE)
             K2= 1
             DO   J= N3, N4
             K1= 1
               DO  I= N1, N2
                CODPIC(I,J)= IMAGE1(I,J)+ DFD(K1,K2)
              K1= K1+ 1
               END DO
             K2= K2+ 1
           END DO
         ENDIF
          N1= N2+ 1
          N2= N2+ XSIZE
        END DO
          N1= 1
          N2= XSIZE
          N3= N4+ 1
          N4= N4+ YSIZE
```

```
        END DO
C
      VAR1=(SUMY/BTOM)-((SUMX/BTOM)**2.0)
      TYPE*,'COMPENSATED ERROR VARIANCE=',VAR1
      VUC= (SUMT2/BTOM)- ((SUMT1/BTOM)**2.0)
      TYPE*,'UNCOMPENSATED ERROR VARIANCE=',VUC
      PERG=10.0*ALOG10(VAR3/VAR1)
      TYPE*,'PREDICTION GAIN=',PERG
      MS1= SUMT2/DNET
      TYPE*,'UNCOMPENSATED MS1= ',MS1
      MS2= SUMY/DNET
      TYPE*,'COMPENSATED MS2=',MS2
C
      KOUNT1= 0.0
      SUMW= 0.0
      TEMP=0.0
       DO   J= 1, NY
         DO  I= 1, NX
           TEMP= FLOAT(CODPIC(I,J))**2.0
           SUMW= SUMW+ TEMP
           TEMP= FLOAT(CODPIC(I,J)- IMAGE2(I,J))**2.0
           KOUNT1= KOUNT1+ TEMP
          END DO
       END DO
        SNR1= 10.0* ALOG10(SUMW/SUMY)
      TYPE*,'SIGNAL-TO-NOISE RATIO IN M/AREA=',SNR1
      SNR2= 10.0* ALOG10(SUMW/KOUNT1)
      TYPE*,'SIGNAL-TO-NOISE RATIO OVER IMAGE=',SNR2
      SNR3= 10.0* ALOG10(SUMW/ADI)
      TYPE*,'SNR WITH CODED DFD=',SNR3
C
       CALL OPENV(STATUSFLAG,1,1,0,0,0)
      DO LINENUMBER= 1, NY
           CALL WRITE(STATUSFLAG,1,0,1,0,NX,CODPIC(1,LINENUMBER),0)
        ENDDO
        CALL RELAB2(1, NY, NX)
C
      END
C=======================SUBROUTINE GRDNT2
        SUBROUTINE GRDNT2(PICT,XGRD,YGRD)
        PARAMETER(NX=256,NY=256)
        INTEGER*2 PICT(NX,NY),BUF1,BUF2
        REAL XGRD(NX,NY),YGRD(NX,NY)
        DO 10 J=1,NY
          DO 10 I=1,NX
           BUF1=0
           BUF2=0
           IF(I.NE.1.AND.I.NE.NX.AND.J.NE.1.AND.J.NE.NY)THEN
             BUF1=2*PICT(I+1,J)+PICT(I+1,J+1)+PICT(I+1,J-1)
             BUF2=2*PICT(I-1,J)+PICT(I-1,J-1)+PICT(I-1,J+1)
            XGRD(I,J)=(FLOAT(BUF1-BUF2))/4.0
           ELSE IF((J.EQ.1).AND.(I.NE.NX))THEN
            XGRD(I,J)=FLOAT(PICT(I,J)-PICT(I+1,J))
```

```
            ELSE IF((J.EQ.NY).AND.(I.NE.NX))THEN
            XGRD(I,J)=FLOAT(PICT(I,J)-PICT(I+1,J))
           ELSE IF(I.EQ.1)THEN
            XGRD(I,J)=FLOAT(PICT(I,J)-PICT(I+1,J))
           ELSE IF(I.EQ.NX)THEN
            XGRD(I,J)=FLOAT(PICT(I,J)-PICT(I-1,J))
           ENDIF
 10     CONTINUE
        DO 20 J=1,NY
          DO 20 I=1,NX
           BUF1=0
           BUF2=0
           IF((I.NE.1).AND.(I.NE.NX).AND.(J.NE.1).AND.(J.NE.NY))THEN
            BUF1=2*PICT(I,J+1)+PICT(I-1,J+1)+PICT(I+1,J+1)
            BUF2=2*PICT(I,J-1)+PICT(I-1,J-1)+PICT(I+1,J-1)
             YGRD(I,J)=(FLOAT(BUF1-BUF2))/4.0
          ELSE IF(J.EQ.1)THEN
             YGRD(I,J)=FLOAT(PICT(I,J)-PICT(I,J+1))
           ELSE IF(J.EQ.NY)THEN
             YGRD(I,J)=FLOAT(PICT(I,J)-PICT(I,J-1))
           ELSE IF((I.EQ.1).AND.(J.NE.NY))THEN
             YGRD(I,J)=FLOAT(PICT(I,J)-PICT(I,J+1))
           ELSE IF((I.EQ.NX).AND.(J.NE.NY))THEN
             YGRD(I,J)=FLOAT(PICT(I,J)-PICT(I,J+1))
           ENDIF
 20     CONTINUE
        RETURN
        END
C--------------SUBROUTINE CODE: OBTAINS PREDICTION ERRORS AS INPUT
C--------------OUTPUT IS QUANTIZED PREDICTION ERRORS
C
      SUBROUTINE CODE(PERR, XSIZ, YSIZ)
              INTEGER*2 XSIZ, YSIZ, PERR(16,16), IBUF, K1
              REAL  BUF(256), RECLV(25), MINERR
              DATA RECLV/-182,-102,-87,-77,-66,-55,-44,
     +           -35,-26,-17,-10,-4,0,4,10,17,26,35,44,
     +           55,66,77,87,102,182/
              DO J= 1, YSIZ
                  DO I= 1, XSIZ
                      DO K= 1, 25
                              IBUF= RECLV(K)- PERR(I,J)
                              BUF(K)= FLOAT(IBUF) **2.0
                      END DO
                      MINERR= BUF(1)
                      DO K= 1, 25
                          IF( BUF(K) .LE. MINERR)THEN
                              MINERR= BUF(K)
                              K1= K
                          END IF
                      END DO
                      PERR(I,J)= RECLV(K1)
                  END DO
              END DO
              RETURN
              END
```

```
C=================================================================
C          PROGRAM  FOR ALGORITHM PDDT3
C PROGRAM ESTIMATING MOTION FROM PROJECTIONS OF FRAME,ELEMENT AND
C LINE DIFFERENCES.
C COMPUTATION IS DONE FOR EVERY MOVING AREA PIXEL BY CENTERING
C A BLOCK ON EACH PIXEL.
C=================================================================C
      PARAMETER(NX=256,NY=256)
      INTEGER*2 IMAGE1(NX,NY), IMAGE2(NX,NY), CODPIC(NX,NY)
      INTEGER*2 IFD(NX,NY), I1, J1, DFD(16,16), NP
      INTEGER*2 XSIZE, YSIZE, KT1, KT2, K1, K2, KTEMP
      INTEGER*2 DENOM, I2, J2, NDX, NDY
      INTEGER STATUSFLAG,LINENUMBER
      REAL FDTH, BTOM, PERG1, DNEAT, DENOM1
      REAL GRX(NX,NY), GRY(NX,NY), KOUNT1, PERG2
      REAL RMS,VAR1, VAR2, VAR3, VAR4, TEMP(16,16)
      REAL SUM1, SUM2, SUM3, SUM4, BUF1, MS1, MS2
      REAL TUM1, TUM2, EX(16,16), EY(16,16), SNR1, SNR2
      COMMON IFD, GRX, GRY, N1, N2, N3, N4
      CALL VICINIT('NEWMOT1')
      CALL OPENV(STATUSFLAG,2,0,0,0,0)
      CALL OPENV(STATUSFLAG,3,0,0,0,0)
      DO LINENUMBER=1,NY
        CALL READ(STATUSFLAG,2,0,1,0,NX,IMAGE1(1,LINENUMBER),0)
        CALL READ(STATUSFLAG,3,0,1,0,NX,IMAGE2(1,LINENUMBER),0)
      ENDDO
      TYPE*,'INPUT THE WINDOW SIZE FOR SEGMENTATION'
      TYPE*,':XSIZE AND YSIZE'
      READ(5,*)XSIZE,YSIZE
      TYPE*,'INPUT THE BLOCK SIZE FOR MOTION COMPUTATION'
      TYPE*,'CHOOSE 1, 2, 3, 4, 5, e.t.c'
      READ(5,*)NP
      CALL GRDNT2(IMAGE1,GRX,GRY)
      TUM1= 0.0
      TUM2= 0.0
      ADI= 0.0
      SUM1= 0.0
      SUM2= 0.0
      SUM3= 0.0
      SUM4= 0.0
      DNEAT= 0.0
      DENOM= XSIZE*YSIZE
      BTOM= FLOAT(NX*NY)
      KOUNT1=0.0
      DO  J= 1, NY
       DO  I= 1, NX
          IFD(I,J)= IMAGE2(I,J)- IMAGE1(I,J)
          KOUNT1= (FLOAT(IFD(I,J))** 2.0)+ KOUNT1
          TUM1= TUM1+ FLOAT(IMAGE2(I,J))
          TUM2= TUM2+ (FLOAT(IMAGE2(I,J))**2.0)
         END DO
      END DO
      FDTH= SQRT(KOUNT1/BTOM)
      VAR3=(TUM2/BTOM)-((TUM1/BTOM)**2.0)
      TYPE*,'SIGNAL VARIANCE= ',VAR3
```

```
      N1= 1
      N2= XSIZE
      N3= 1
      N4= YSIZE
      NDX= NX/XSIZE
      NDY= NY/YSIZE
      DO   KT2= 1, NDY
          DO   KT1= 1, NDX
C=================================================================
C           SEGMENTATION INTO MOVING AREA AND STATIONARY AREAS
C-----------------------------------------------------------------
        KOUNT1=0.0
        DO  J= N3, N4
            DO  I= N1, N2
              KOUNT1= (FLOAT(IFD(I,J))**2.0)+ KOUNT1
            END DO
        END DO
        RMS= SQRT(KOUNT1/DENOM)
        IF(RMS .LT. FDTH)THEN
C
C------------BLOCK IS NOT MOVING AREA.
C
              K2= 1
              DO J= N3, N4
                K1= 1
                DO I= N1, N2
                  DFD(K1,K2)= IFD(I,J)
                  K1= K1+ 1
                END DO
                K2= K2+ 1
              END DO
              CALL CODE(DFD,XSIZE,YSIZE)
              K2= 1
              DO J= N3, N4
                 K1= 1
                 DO I= N1, N2
                   CODPIC(I,J)= IMAGE1(I,J)- DFD(K1,K2)
                   K1= K1+ 1
                 END DO
                 K2= K2+ 1
              END DO
        ELSE
              CALL MOTION(EX, EY, NP)
              DNEAT= DNEAT+ 1.0
              K2= 1
              DO J= N3, N4
                 K1= 1
                 DO I= N1, N2
                   IF(ABS(EX(K1,K2)) .LT. 8.0)THEN
                      T1= FLOAT(I)- EX(K1,K2)
                   ELSE
                      T1= FLOAT(I)
                   END IF
                    IF(ABS(EY(K1,K2)) .LT. 8.0)THEN
                     T2= FLOAT(J)- EY(K1,K2)
                   ELSE
                      T2= FLOAT(J)
```

```
                    END IF
                    I1= AINT(T1)
                    J1= AINT(T2)
                    I2= ANINT(T1)
                    J2= ANINT(T2)
                    TEMP(K1,K2)= IMAGE1(I1,J1)*0.75 +
     +                0.25*IMAGE1(I2,J2)
                    DFD(K1,K2)= IMAGE2(I,J)- AINT(TEMP(K1,K2))
                    SUM1= SUM1+ FLOAT(DFD(K1,K2))
                    SUM2= SUM2+ FLOAT(DFD(K1,K2))**2.0
                    SUM3= SUM3+ FLOAT(IMAGE2(I,J)-IMAGE1(I,J))
                    SUM4= SUM4+ FLOAT(IMAGE2(I,J)-IMAGE1(I,J))**2.0
                    K1= K1+ 1
                    END DO
                  K2= K2+ 1
               END DO
               CALL CODE(DFD,XSIZE,YSIZE)
               K2= 1
               DO J= N3, N4
                  K1= 1
                  DO I= N1, N2
                    CODPIC(I,J)= ANINT(TEMP(K1,K2))+ DFD(K1,K2)
                    ADI= ADI+ FLOAT(DFD(K1,K2))**2.0
                    K1= K1+ 1
                  END DO
                  K2= K2+ 1
               END DO
            END IF
            N1= N2+ 1
            N2= N2+ XSIZE
          END DO
          N1= 1
          N2= XSIZE
          N3= N4+ 1
          N4= N4+ YSIZE
        END DO
        DENOM1= DNEAT* FLOAT(DENOM)
        VAR1= (SUM2/BTOM)- ((SUM1/BTOM)**2.0)
        TYPE*,'COMPENSATED ERROR VARIANCE= ',VAR1
C
        VAR2= (SUM4/BTOM)- ((SUM3/BTOM)**2.0)
        TYPE*,'UNCOMPENSATED ERROR VARIANCE= ',VAR2
C
        PERG1= 10.0* ALOG10(VAR3/VAR1)
        TYPE*,'COMPENSATED PREDICTION GAIN= ',PERG1
C
        PERG2= 10.0* ALOG10(VAR3/VAR2)
        TYPE*,'UNCOMPENSATED PREDICTION GAIN= ',PERG2
C
        MS1= SUM2/DENOM1
        MS2= SUM4/DENOM1
        TYPE*,'COMPENSATED MS ERROR= ',MS1
        TYPE*,'UNCOMPENSATED MS ERROR= ',MS2
        SUM3= 0.0
        SUM4= 0.0
        BUF1= 0.0
        DO J= 1, NY
```

```
          DO I= 1, NX
             BUF1= FLOAT(CODPIC(I,J))**2.0
             SUM3= SUM3+ BUF1
             BUF1= FLOAT(CODPIC(I,J)- IMAGE2(I,J))**2.0
             SUM4= SUM4+ BUF1
             END DO
        END DO
        SNR1= 10.0* ALOG10(SUM3/SUM4)
        TYPE*,'SNR OVER THE IMAGE= ',SNR1
        SNR2= 10.0* ALOG10(SUM3/SUM2)
        TYPE*,'SNR IN THE MOVING AREA= ',SNR2
C
        CALL OPENV(STATUSFLAG,1,1,0,0,0)
        DO LINENUMBER=1,NY
             CALL WRITE(STATUSFLAG,1,0,1,0,NX,
     +       CODPIC(1,LINENUMBER),0)
          ENDDO
        CALL RELAB2(1,NY,NX)
        CALL END
C
        END
C=============================================================
C                    SUBROUTINE MOTION
C-------------------------------------------------------------
        SUBROUTINE MOTION(EX, EY, NP)
        PARAMETER(NX= 256, NY= 256)
        INTEGER*2 IFD(NX,NY), NP, KA, KB, KC, KD
        REAL GRX(NX,NY), GRY(NX,NY), EX(16,16), EY(16,16)
        REAL XPX(16), XPY(16), YPX(16), YPY(16)
        REAL FDX(16), FDY(16), TEMP1, TEMP2, XM, YM
        COMMON IFD, GRX, GRY, M1, M2, M3, M4
C
        K2= 1
        DO J= M3, M4
          K1= 1
          DO I= M1, M2
             KA= I- NP
             KB= I+ NP
             KC= J- NP
             KD= J+ NP
             IF((KA .LT. 1) .OR. (KB .GT. NX)
     +         .OR. (KC .LT. 1) .OR. (KD .GT. NY))THEN
                EX(K1,K2)= 0.0
                EY(K1,K2)= 0.0
              ELSE
               L= 1
               DO J1= KC, KD
                  YPX(L)= 0.0
                  YPY(L)= 0.0
                  FDY(L)= 0.0
                  DO I1= KA, KB
                    YPX(L)= YPX(L)+ GRX(I1,J1)
                    YPY(L)= YPY(L)+ GRY(I1,J1)
                    FDY(L)= FDY(L)- FLOAT(IFD(I1,J1))
                  END DO
                  L= L+ 1
               END DO
```

```
                M= 1
                DO I1= KA, KB
                   XPX(M)= 0.0
                   XPY(M)= 0.0
                   FDX(M)= 0.0
                   DO J1= KC, KD
                     XPX(M)= XPX(M)+ GRX(I1,J1)
                     XPY(M)= XPY(M)+ GRY(I1,J1)
                     FDX(M)= FDX(M)+ FLOAT(IFD(I1,J1))
                   END DO
                   M= M+ 1
                END DO
                TEMP1= 0.0
                TEMP2= 0.0
                M= 1
                DO J1= KC, KD
                  L= 1
                  DO I1= KA, KB
                     BF1= FDX(L)*YPY(M)- FDY(M)*XPY(L)
                     BF2= XPY(L)*YPX(M)- YPY(M)*XPX(L)
                     BF3= FDY(M)*XPX(L)- FDX(L)*YPX(M)
                     IF(BF2 .EQ. 0.0) THEN
                        XM= 0.0
                        YM= 0.0
                       ELSE
                        XM= BF1/BF2
                        YM= BF3/BF2
                     END IF
                     TEMP1= TEMP1+ XM
                     TEMP2= TEMP2+ YM
                     L= L+ 1
                  END DO
                  M= M+ 1
                END DO
                EX(K1,K2)= TEMP1/((2.0*NP+1)**2.0)
                EY(K1,K2)= TEMP2/((2.0*NP+1)**2.0)
              END IF
            K1= K1+ 1
          END DO
          K2= K2+ 1
        END DO
        RETURN
        END
C=============================================================
C                    SUBROUTINE GRDNT2
C-------------------------------------------------------------
        SUBROUTINE GRDNT2(PICT,XGRD,YGRD)
        PARAMETER(NX=256, NY=256)
        INTEGER*2 PICT(NX,NY),BUF1,BUF2
        REAL XGRD(NX,NY),YGRD(NX,NY)
        DO 10 J=1,NY
          DO 10 I=1,NX
           BUF1=0
           BUF2=0
           IF(I.NE.1.AND.I.NE.NX.AND.J.NE.1.AND.J.NE.NY)THEN
             BUF1=2*PICT(I+1,J)+PICT(I+1,J+1)+PICT(I+1,J-1)
             BUF2=2*PICT(I-1,J)+PICT(I-1,J-1)+PICT(I-1,J+1)
```

```
            XGRD(I,J)=(FLOAT(BUF1-BUF2))/4.0
           ELSE IF((J.EQ.1).AND.(I.NE.NX))THEN
            XGRD(I,J)=FLOAT(PICT(I,J)-PICT(I+1,J))
           ELSE IF((J.EQ.NY).AND.(I.NE.NX))THEN
            XGRD(I,J)=FLOAT(PICT(I,J)-PICT(I+1,J))
           ELSE IF(I.EQ.1)THEN
            XGRD(I,J)=FLOAT(PICT(I,J)-PICT(I+1,J))
           ELSE IF(I.EQ.NX)THEN
            XGRD(I,J)=FLOAT(PICT(I,J)-PICT(I-1,J))
           ENDIF
 10     CONTINUE
        DO 20 J=1,NY
          DO 20 I=1,NX
           BUF1=0
           BUF2=0
           IF((I.NE.1).AND.(I.NE.NX).AND.(J.NE.1).AND.(J.NE.NY))THEN
            BUF1=2*PICT(I,J+1)+PICT(I-1,J+1)+PICT(I+1,J+1)
            BUF2=2*PICT(I,J-1)+PICT(I-1,J-1)+PICT(I+1,J-1)
             YGRD(I,J)=(FLOAT(BUF1-BUF2))/4.0
           ELSE IF(J.EQ.1)THEN
             YGRD(I,J)=FLOAT(PICT(I,J)-PICT(I,J+1))
           ELSE IF(J.EQ.NY)THEN
             YGRD(I,J)=FLOAT(PICT(I,J)-PICT(I,J-1))
           ELSE IF((I.EQ.1).AND.(J.NE.NY))THEN
             YGRD(I,J)=FLOAT(PICT(I,J)-PICT(I,J+1))
           ELSE IF((I.EQ.NX).AND.(J.NE.NY))THEN
             YGRD(I,J)=FLOAT(PICT(I,J)-PICT(I,J+1))
           ENDIF
 20     CONTINUE
        RETURN
        END
C--------------SUBROUTINE CODE: OBTAINS PREDICTION ERRORS AS INPUT
C-------------OUTPUT IS QUANTIZED PREDICTION ERRORS
      SUBROUTINE CODE(PERR, XSIZ, YSIZ)
            INTEGER*2 XSIZ, YSIZ, PERR(16,16), IBUF, K1
            REAL  BUF(30), RECLV(25), MINERR
            DATA RECLV/-182,-102,-87,-77,-66,-55,-44,
     +          -35,-26,-17,-10,-4,0,4,10,17,26,35,44,
     +          55,66,77,87,102,182/
            DO J= 1, YSIZ
               DO I= 1, XSIZ
                  DO K= 1, 25
                        IBUF= RECLV(K)- PERR(I,J)
                        BUF(K)= FLOAT(IBUF) **2.0
                  END DO
                  MINERR= BUF(1)
                  DO K= 1, 25
                     IF( BUF(K) .LE. MINERR)THEN
                        MINERR= BUF(K)
                        K1= K
                     END IF
                  END DO
                  PERR(I,J)= RECLV(K1)
               END DO
            END DO
            RETURN
            END
```

```
C=============================================================
C                  PROGRAM FOR ALGORITHM PDDT4
C PROGRAM ESTIMATING MOTION FROM PROJECTIONS OF FRAME,ELEMENT AND
C LINE DIFFERENCES:RECURSIVE LEAST SQUARES ESTIMATION IS USED.
C-------------------------------------------------------------
      PARAMETER(NX= 256, NY= 256)
      INTEGER*2 IMAGE1(NX,NY),IMAGE2(NX,NY)
      INTEGER*2 ISUM,IPIC(NX,NY), ITERAT
      INTEGER*2 CHOICE,DENOM,DFD(32,32),IFD(NX,NY)
      INTEGER*2 CODPIC(NX,NY), IKOL,ICOL, STPC
      INTEGER*2 KT1,KT2,K1,K2,KTEMP,NDX,NDY,NYSIZ,NXSIZ
      INTEGER*2 ITEMP,K,NEAT,I1,I2,J1,J2,N1,N2,N3,N4
      INTEGER STATUSFLAG,LINENUMBER
      REAL XGR(NX,NY), YGR(NX,NY)
      REAL GLBTH,EX, EY, BM1(2), DKN(2), BR1, PKN(2,2)
      REAL KOUNT1, VAR1, VAR2, LCTHR, GKN(8), DO(2)
      REAL DENOM1, BTOM, PERG, FKN, PO(2,2)
      REAL SUMX, SUMY, SUMW, SUMZ, T1, T2, TEST
      REAL TEMP, VAR3, VAR4, DET, VPR, ADI, VUC
      REAL TUM1, TUM2, DPG, SUM, SUM1, SUM2, SUM3
      REAL DNOM, AVRG1, AVRG2, MS1, MS2, TEMPA, TEMPB
      COMMON XGR, YGR, IFD
C
      CALL VICINIT('RECLSE1')
      CALL OPENV(STATUSFLAG,2,0,0,0,0)
      CALL OPENV(STATUSFLAG,3,0,0,0,0)
      DO LINENUMBER=1,NY
       CALL READ(STATUSFLAG,2,0,1,0,NX,IMAGE1(1,LINENUMBER),0)
        CALL READ(STATUSFLAG,3,0,1,0,NX,IMAGE2(1,LINENUMBER),0)
      ENDDO
C
      WRITE(6,*)'INPUT THE WINDOW SIZE,NXSIZ,NYSIZ AND STARTING
     + SIZE;ICOL'
      READ(5,*)NXSIZ,NYSIZ,ICOL
      CALL GRDNT2(IMAGE1,XGR,YGR)
      SUMX= 0.0
      SUMW= 0.0
      SUMY= 0.0
      SUMZ= 0.0
      AVRG1= 0.0
      AVRG2= 0.0
      NEAT= 0
      TUM1= 0.0
      TUM2= 0.0
      ADI= 0.0
      KTEMP=0
      DNOM= 0.0
      DENOM= NXSIZ *NYSIZ
      BTOM= FLOAT(NX*NY)
C=============COMPUTATION OF THRESHOLD=========
      KOUNT1=0.0
      DO  J= 1, NY
         DO I= 1, NX
          IFD(I,J)=IMAGE2(I,J)-IMAGE1(I,J)
          KTEMP=IFD(I,J)
```

```
            KOUNT1= (FLOAT(KTEMP)**2.0)+ KOUNT1
            TUM1= TUM1+ FLOAT(IMAGE2(I,J))
            TUM2= TUM2+ FLOAT(IMAGE2(I,J))**2.0
          ENDDO
        ENDDO
        GLBTH=SQRT(KOUNT1/BTOM)
C
        VAR3=(TUM2/BTOM)-((TUM1/BTOM)**2.0)
        TYPE*,'SIGNAL VARIANCE= ',VAR3
        TUM1=0.0
        TUM2=0.0
        N1= 1
        N2= NXSIZ
        N3= 1
        N4= NYSIZ
        NDX= NX/NXSIZ
        NDY= NY/NYSIZ
        DO   KT2= 1, NDY
          DO   KT1= 1, NDX
C=====================SEGMENTATION OF BLOCKS========
C
        KOUNT1= 0.0
        KTEMP= 0
        DO  J= N3, N4
         DO  I= N1, N2
           KTEMP= IMAGE2(I,J)- IMAGE1(I,J)
           KOUNT1= (FLOAT(KTEMP)**2.0)+ KOUNT1
         END DO
        END DO
        LCTHR= SQRT(KOUNT1/DENOM)
        IF(LCTHR .GT. GLBTH)THEN
        CALL NITIAL( ICOL, DO, PO, N1, N2, N3, N4)
        IKOL= ICOL
        STPC= NXSIZ- IKOL
        ITERAT= 0
 5      CONTINUE
        ITERAT= ITERAT+ 1
        IKOL= IKOL+ 1
        CALL UPDATE( PKN, GKN, FKN, PO, IKOL, N1, N2, N3, N4)
        BR1= GKN(1)* DO(1) + GKN(2)* DO(2)
        BR1= FKN- BR1
        BM1(1)= PKN(1,1)*GKN(1) + PKN(1,2)*GKN(1)
        BM1(2)= PKN(2,1)*GKN(1) + PKN(2,2)*GKN(2)
        BM1(1)= BM1(1)*BR1
        BM1(2)= BM1(2)*BR1
        DKN(1)= DO(1)+ BM1(1)
        DKN(2)= DO(2)+ BM1(2)
        TEMPA= (DKN(1)- DO(1))**2.0
        TEMPB= (DKN(2)- DO(2))**2.0
        TEST= SQRT(TEMPA+ TEMPB)
        IF ((TEST .LE. 4.0) .OR. (ITERAT .EQ. STPC)) THEN
                IF (ABS(DKN(1)) .GT. 8.0) THEN
                        EX= 0.0
                        EY= 0.0
                ELSE
```

```
                CALL MTNY(EY, IKOL,N1, N2, N3, N4)
                EX= DKN(1)
               END IF
C
C-----------MOTION COMPENSATED PREDICTION---------
C
              K2= 1
              DO J= N3, N4
                  K1= 1
                    DO I= N1, N2
                      DNOM= DNOM + 1.0
                      IF(K1 .LE. IKOL)THEN
                         IF((ABS(EX)).LT.8.0)THEN
                            T1= FLOAT(I)- EX
                         ELSE
                            T1= FLOAT(I)
                         END IF
                         IF((ABS(EY)).LT.8.0)THEN
                            T2= FLOAT(J)- EY
                         ELSE
                            T2= FLOAT(J)
                         END IF
                         I1= AINT(T1)
                         J1= AINT(T2)
                         I2= ANINT(T1)
                         J2= ANINT(T2)
                         TEMP= (IMAGE1(I1,J1)*0.75)+
     +                   (0.25*IMAGE1(I2,J2))
                         ITEMP= ANINT(TEMP)
                         DFD(K1,K2)= IMAGE2(I,J)- ITEMP
                      ELSE
                         DFD(K1,K2)= IMAGE2(I,J)- IMAGE1(I,J)
                      END IF
                      SUMX= SUMX+  FLOAT(DFD(K1,K2))
                      SUMY= SUMY+ FLOAT(DFD(K1,K2))**2.0
                      AVRG1= AVRG1+ FLOAT(IFD(I,J))
                      AVRG2= AVRG2+ FLOAT(IFD(I,J))**2.0
                      K1= K1+ 1
                    ENDDO
                    K2= K2+ 1
              ENDDO
              CALL CODE ( DFD, NXSIZ, NYSIZ)
C
C-------------------------------------RECONSTRUCTION
               K2= 1
               DO J= N3, N4
                  K1= 1
                  DO I= N1, N2
                       IF(K1 .LE. IKOL)THEN
                         IF((ABS(EX)).LT.8.0)THEN
                           T1= FLOAT(I)- EX
                         ELSE
                           T1= FLOAT(I)
                         END IF
                         IF((ABS(EY)).LT.8.0)THEN
```

```
                              T2= FLOAT(J)- EY
                         ELSE
                              T2= FLOAT(J)
                         END IF
                              I1= AINT(T1)
                              J1= AINT(T2)
                              I2= ANINT(T1)
                              J2= ANINT(T2)
                         TEMP= (IMAGE1(I1,J1)*0.75)+
     +                          (0.25*IMAGE1(I2,J2))
                          ITEMP= ANINT(TEMP)
                          CODPIC(I,J)=ITEMP+DFD(K1,K2)
                          ADI= ADI+ (FLOAT(DFD(K1,K2))**2.0)
                       ELSE
                    CODPIC(I,J)= IMAGE1(I,J)+ DFD(K1,K2)
                       END IF
                         K1+ 1
                  END DO
                  K2= K2+ 1
               END DO
C
         ELSE
               DO JH= 1, 2
                  DO(JH)= DKN(JH)
               END DO
               GOTO 5
         END IF
      ELSE
         K2= 1
         DO J= N3, N4
             K1= 1
             DO I= N1, N2
                 DFD(K1,K2)= IFD(I,J)
                 K1= K1+ 1
             END DO
             K2= K2+ 1
         END DO
         CALL CODE ( DFD, NXSIZ, NYSIZ)
         K2= 1
         DO J= N3, N4
             K1= 1
             DO I= N1, N2
               CODPIC(I,J)= IMAGE1(I,J)+ DFD(K1,K2)
               K1= K1+ 1
             END DO
             K2= K2+ 1
         END DO

         END IF
         N1= N2+ 1
         N2= N2+ NXSIZ
      END DO
      N1= 1
      N2= NXSIZ
      N3= N4+ 1
```

```
        N4= N4+ NYSIZ
      END DO
      END


C--------------------SUBROUTINE UPDATE----------
       SUBROUTINE UPDATE(PKN, GKN, FKN, PO,
      +  IKOL, N1, N2, N3, N4)
       PARAMETER(NX= 256, NY= 256)
       INTEGER*2 IKOL, IFDP(NX,NY), N1, N2
       INTEGER*2 NYSZ, N3, N4
       REAL XGRD(NX,NY), YGRD(NX,NY)
       REAL PKN(2,2), GKN(8), FKN, PO(2,2), SUM1, SUM2
       REAL GRDX(16,16), GRDY(16,16), FDP(16,16)
       REAL TP1(2,2), TP2(2,2), TP3(2), BF1
       COMMON XGRD, YGRD, IFDP
                NYSZ= N4- N3+ 1
                K2= 1
                DO J= N3, N4
                   K1= 1
                   DO I= N1, N2
                        GRDX(K1,K2)= XGRD(I,J)
                        GRDY(K1,K2)= YGRD(I,J)
                        FDP(K1,K2)= IFDP(I,J)
                        K1= K1+ 1
                   END DO
                   K2= K2+ 1
                END DO
C--------GKN, FKN
                SUM1= 0.0
                SUM2= 0.0
                SUM3= 0.0
                DO J= 1, NYSZ
                        SUM1= SUM1- GRDX(IKOL,J)
                        SUM2= SUM2- GRDY(IKOL,J)
                        SUM3= SUM3+ FDP(IKOL,J)
                END DO
                GKN(1)= SUM1/NY
                GKN(2)= SUM2/NY
                FKN= SUM3/NY
C--------PKN
C------------PO * GKN * GKNT * PO
                DO I= 1, 2
                  DO J= 1, 2
                        TP1(I,J)= GKN(I)* GKN(J)
                  END DO
                END DO
                DO I= 1, 2
                  DO J= 1, 2
                        SUM1= 0.0
                        DO K= 1, 2
                           SUM1= SUM1+ PO(I,K)* TP1(K,J)
                        END DO
                        TP2(I,J)= SUM1
```

```
                END DO
              END DO
              DO I= 1, 2
                 DO J= 1, 2
                       SUM1= 0.0
                       DO K= 1 ,2
                         SUM1= SUM1+ TP2(I,K)* PO(K,J)
                       END DO
                       TP1(I,J)= SUM1
                 END DO
              END DO
C
C--------GKN * PO * GKNT + 1
              DO I= 1, 2
                 SUM= 0.0
                 DO K= 1, 2
                       SUM1= SUM1+ PO(I,K)+ GKN(K)
                 END DO
                 TP3(I)= SUM1
              END DO
        BF1= 1.0 + (GKN(1)* TP3(1)) + (GKN(2)* TP3(2))
              DO I= 1, 2
                DO J= 1, 2
                  TP2(I,J)= TP1(I,J)/BF1
                END DO
              END DO
              DO I= 1, 2
                 DO J = 1, 2
                       PKN(I,J)= PO(I,J)- TP2(I,J)
                 END DO
              END DO
              RETURN
              END
C------------------SUBROUTINE  NITIAL: COMPUTES INITIAL VALUE
      SUBROUTINE NITIAL( IKL, DON, PON, N1, N2,
     +          N3, N4)
      PARAMETER( NX=256, NY=256)
      INTEGER*2 IKL,IFDN(NX,NY)
      INTEGER*2 N1, N2, N3, N4, NYSZ
      REAL XGRN(NX,NY), YGRN(NX,NY)
      REAL GO(16,16), FO(16), GOT(16,16), GTG(16,16)
      REAL PON(2,2), DON(2), GOF(2), SUM1, SUM2, SUM3
      REAL GRDX(16,16), GRDY(16,16), FDP(16,16), DET
      COMMON XGRN, YGRN, IFDN
      NYSZ= N4- N3+ 1
              K2= 1
              DO J= N3, N4
                K1= 1
                DO I= N1, N2
                       GRDX(K1,K2)= XGRN(I,J)
                       GRDY(K1,K2)= YGRN(I,J)
                       FDP(K1,K2)= IFDN(I,J)
                       K1= K1+ 1
                END DO
                K2= K2+ 1
```

```
                END DO
C
                DO I= 1, IKL
                   SUM1= 0.0
                   SUM2= 0.0
                   SUM3= 0.0
                   DO J= 1, NYSZ
                        SUM1= SUM1- GRDX(I,J)
                        SUM2= SUM2- GRDY(I,J)
                        SUM3= SUM3+ FDP(I,J)
                   END DO
                   GO(I,1)= SUM1/NY
                   GO(I,2)= SUM2/NY
                   FO(I)= SUM3/NY
                END DO
C------GO TRANSPOSE
                DO J= 1, IKL
                   DO I= 1, 2
                        GOT(I,J)= GO(J,I)
                   END DO
                END DO
C-----GOT * GO= GTG
                DO I= 1, 2
                   DO J= 1, 2
                        SUM1= 0.0
                        DO K= 1, IKL
                           SUM1= SUM1+ GOT(I,K)*GO(K,J)
                        END DO
                        GTG(I,J)= SUM1
                   END DO
                END DO
C-----GTG INVERSION = PON
          DET= (GTG(1,1)*GTG(2,2))- (GTG(2,1)*GTG(1,2))
                IF( DET .NE. 0.0)THEN
                        PON(1,1)= GTG(2,2)/DET
                        PON(1,2)= -GTG(1,2)/DET
                        PON(2,1)= -GTG(2,1)/DET
                        PON(2,2)= GTG(1,1)/DET
                ELSE IF(DET .EQ. 0.0)THEN
                        PON(1,1)= 0.0
                        PON(1,2)= 0.0
                        PON(2,1)= 0.0
                        PON(2,2)= 0.0
                END IF
C-----DO= PON * GOT * FO
                DO I= 1, 2
                    SUM1= 0.0
                    DO K= 1, IKL
                        SUM1= SUM1+ GOT(I,K)*FO(K)
                    END DO
                    GOF(I)= SUM1
                END DO
                DO I= 1, 2
                    SUM1= 0.0
                    DO J= 1, 2
```

```
                    SUM1= SUM1+ PON(I,J)*GOF(J)
                  END DO
                  DON(I)= SUM1
               END DO
               RETURN
               END
C------SUBROUTINE MTNY: COMPUTES THE MOTION ALONG Y
               SUBROUTINE MTNY( EY, IKL1, N1, N2, N3, N4)
               PARAMETER(NX=256, NY=256)
               INTEGER*2 IFDM(NX,NY), IKL1, N1, N2
               INTEGER*2 NYSZ, N3, N4
               REAL XGRM(NX,NY), YGRM(NX,NY)
               REAL GY(16,2), FG(16), GYT(2,16), D2(2)
               REAL DET, GINV(2,2), GYF(2), FDP(16,16)
               REAL GYG(2,2), GRDX(16,16), GRDY(16,16)
               REAL SUM1, SUM2, SUM3
               COMMON XGRM, YGRM, IFDM
               NYSZ= N4- N3+ 1
               K2= 1
               DO J= N3, N4
                  K1= 1
                  DO I= N1, N2
                       GRDX(K1,K2)= XGRM(I,J)
                       GRDY(K1,K2)= YGRM(I,J)
                       FDP(K1,K2)= IFDM(I,J)
                       K1= K1+ 1
                  END DO
                  K2= K2+ 1
               END DO
               DO J= 1, NYSZ
                  SUM1= 0.0
                  SUM2= 0.0
                  SUM3= 0.0
                  DO  I= 1, IKL1
                       SUM1= SUM1- GRDX(I,J)
                       SUM2= SUM2- GRDY(I,J)
                             SUM3= SUM3+ FDP(I,J)
                  END DO
                  GY(J,1)= SUM1/NX
                  GY(J,2)= SUM2/NX
                  FG(J)= SUM3/NX
               END DO
C------GY TRANSPOSE
               DO I= 1, NYSZ
                  DO J= 1, 2
                       GYT(J,I)= GY(I,J)
                  ENDDO
               ENDDO
C------GYT * GY
               DO I= 1, 2
                  DO J= 1, 2
                    SUM1= 0.0
                    DO K= 1, NYSZ
                        SUM1= SUM1+ GYT(I,K)*GY(K,J)
                    END DO
```

```
                    GYG(I,J)= SUM1
                  END DO
                END DO
C----GYG INVERSE
                DET=(GYG(1,1)*GYG(2,2))-(GYG(1,2)*GYG(2,1))
                IF (DET .NE. 0.0) THEN
                  GINV(1,1)= GYG(2,2)/DET
                  GINV(1,2)= -GYG(1,2)/DET
                  GINV(2,1)= -GYG(2,1)/DET
                  GINV(2,2)= GYG(1,1)/DET
C-----GYT * FG
                  DO I= 1, 2
                    SUM1= 0.0
                      DO J= 1, NYSZ
                         SUM1= SUM1+ GYT(I,J)*FG(J)
                      END DO
                      GYF(I)= SUM1
                  END DO
C-----MOTION ESTIMATE EY
                  DO I= 1, 2
                   SUM1= 0.0
                     DO J= 1, 2
                        SUM1= SUM1+ GINV(I,J)*GYF(J)
                     END DO
                     D2(I)=SUM1
                  END DO
                  EY= D2(2)
                ELSE
                  EY=0.0
                END IF
                RETURN
                END
```

```
C=============================================================
C                        PROGRAM: INTPOL
C     PROGRAM IMPLEMENTS MOTION COMPENSATED FRAME INTERPOLATION
C     USING MOTION ESTIMATES OBTAINED VIA THE METHOD OF
C                INVARIANT MOMENTS OF IMAGE BLOCKS.
C                 TWO FEATURES EXTRACTED.
C-------------------------------------------------------------
       PARAMETER(NX= 350, NY= 240)
       INTEGER*2 XSIZE,YSIZE,M1,M2
       INTEGER STATUSFLAG,LINENUMBER
       INTEGER*2 M3, M4, DENOM, NDX, NDY, KT1, KT2, K1, K2
       INTEGER*2 IMAGE1(NX,NY), IMAGE2(NX,NY), INTPIC(NX,NY)
       INTEGER*2 LINTPOL(NX,NY), CHOICE
       INTEGER*2 PIC1(32,32), PIC2(32,32)
       INTEGER*2 EX, EY, MOTX, MOTY, NI, NI1, NJ, NJ1
       REAL INVM1(2), INVM2(2), TH1, TEMP1, TEMP5
       REAL MINDAT, RMS1, RMS2, COUNT1, TEMP2, TEMP
       REAL METRIC(-25:25,-25:25), BTOM, TEMP3, TEMP4
       REAL C1, C2, STD1, STD2, CAVG1, CAVG2, SUM
       REAL AVG1, AVG2, CVAR1, CVAR2, A0, A1
       CALL VICINIT('INTPOL')
       CALL OPENV(STATUSFLAG,2,0,0,0,0)
       CALL OPENV(STATUSFLAG,3,0,0,0,0)
       DO LINENUMBER= 1, NY
           CALL READ(STATUSFLAG,2,0,1,0,NX,IMAGE1(1,LINENUMBER),0)
           CALL READ(STATUSFLAG,3,0,1,0,NX,IMAGE2(1,LINENUMBER),0)
       ENDDO
C
       TYPE*,'INPUT THE WINDOW SIZE;XSIZE AND YSIZE'
       READ(5,*)XSIZE,YSIZE
       TYPE*,'INPUT THE BOUNDS ON THE SEARCH AREA;MOTX AND MOTY'
       READ(5,*)MOTX,MOTY
       TYPE*,' CHOOSE THE FILTER COEFFICIENT FOR INTERPOLATION'
       TYPE*,' 1 ------FOR----- 0.5/0.5 IMAGE'
       TYPE*,'  '
       TYPE*,' 2 ------FOR----- .75/.25 IMAGE'
       TYPE*,'  '
       TYPE*,' 3 ------FOR----- .25/.75 IMAGE'
       TYPE*,'  '
       TYPE*,' 4-------FOR------.66/.33 IMAGE'
       TYPE*,'  '
       TYPE*,' 5-------FOR------.33/.66 IMAGE'
       READ(5,*)CHOICE
       IF( CHOICE .EQ. 1) THEN
                A0= 0.5
                A1= 0.5
       ELSE IF( CHOICE .EQ. 2) THEN
                A0= 0.75
                A1= 0.25
       ELSE IF( CHOICE .EQ. 3) THEN
                A0= 0.25
                A1= 0.75
       ELSE IF( CHOICE .EQ. 4) THEN
                A0= 0.66
```

```
                A1= 0.33
        ELSE IF( CHOICE .EQ. 5) THEN
                A0= 0.33
                A1= 0.66
        END IF
C     -----------------THRESHOLD FOR SEGMENTATION-------
C
        DENOM= XSIZE* YSIZE
        BTOM= FLOAT(NX*NY)
        TEMP1= 0.0
        COUNT1= 0.0
        TEMP2= 0.0
        DO J= 1, NY
          DO I= 1, NX
             TEMP1= FLOAT(IMAGE2(I,J)- IMAGE1(I,J))
             COUNT1= CUONT1+ (TEMP1** 2.0)
             TEMP2= A0*IMAGE1(I,J)+ A1*IMAGE2(I,J)
             LINTPOL(I,J)= ANINT(TEMP2)
          END DO
        END DO
        TH1= SQRT( COUNT1/ BTOM)
C
C       --------------SEGMENTATION AND -----------------
C       ------------INTERPOLATING FILTER----------------
C
        M1= 1
        M2= XSIZE
        M3= 1
        M4= YSIZE
        NDX= NX/XSIZE
        NDY= NY/YSIZE
        STD1= 0.0
        AVG1= 0.0
        DO  KT2= 1, NDY
          DO  KT1= 1, NDX
             COUNT1= 0.0
             TEMP1= 0.0
             TEMP2= 0.0
             TEMP3= 0.0
             TEMP4= 0.0
             TEMP5= 0.0
             DO  J= M3, M4
                DO I= M1, M2
                 TEMP1= FLOAT( IMAGE2(I,J)- IMAGE1(I,J))
                 COUNT1= COUNT1+ (TEMP1** 2.0)
                 TEMP2= TEMP2+ FLOAT(IMAGE1(I,J))
                 TEMP4= TEMP4+ FLOAT(IMAGE2(I,J))
                 TEMP3= TEMP3+ FLOAT(IMAGE1(I,J))**2.0
                 TEMP5= TEMP5+ FLOAT(IMAGE2(I,J))**2.0
                END DO
             END DO
             RMS1= SQRT( COUNT1/DENOM )
             IF( RMS1 .GT. TH1 ) THEN
                CAVG1= TEMP2/DENOM
                CAVG2= TEMP4/DENOM
```

```
                CVAR1= (TEMP3/DENOM)- (CAVG1**2.0)
                CVAR2= (TEMP5/DENOM)- (CAVG2**2.0)
                C1= ABS(CAVG1- AVG1)
                C2= ABS(CAVG2- AVG1)
        IF((C2 .LE. STD1) .AND. (C1 .GT. STD1))THEN
                    DO J= M3, M4
                       DO I= M1, M2
                            INTPIC(I,J)= IMAGE2(I,J)
                       END DO
                    END DO
        ELSE IF( (C1 .LE. STD1) .AND. (C2 .GT. STD1))THEN
                       DO J= M3, M4
                          DO I= M1, M2
                            INTPIC(I,J)= IMAGE1(I,J)
                          END DO
                        END DO
        ELSE
            K2= 1
            DO  J= M3, M4
                K1= 1
                DO  I= M1, M2
                    PIC1(K1,K2)= IMAGE2(I,J)
                    K1= K1+ 1
                END DO
                K2= K2+ 1
            END DO
            CALL MOMINV( PIC1, XSIZE, YSIZE, INVM1)
            MINDAT= 9998.0
C           EX= 0
C           EY= 0
            DO  J= -MOTY, MOTY, 1
              NC= M3+ J
              ND= M4+ J
              DO I= -MOTX, MOTX, 1
                   NA= M1+ I
                   NB= M2+ I
       IF((NA .LT. 1) .OR. (NB .GT. NX) .OR. (NC .LT. 1)
     +              .OR. (ND .GT. NY)) THEN
                   METRIC(I,J)= 100000.0
C                  EX= 0
C                  EY= 0
                ELSE
                   K2= 1
                   DO  J1= NC, ND
                      K1= 1
                      DO  I1= NA, NB
                         PIC2(K1,K2)= IMAGE1(I1,J1)
                         K1= K1+ 1
                      END DO
                      K2= K2+ 1
                   END DO
                   TEMP= 0.0
                   SUM= 0.0
                   DO J2= 1, YSIZE
                      DO I2= 1, XSIZE
```

```
                        TEMP= FLOAT( PIC1(I2,J2)- PIC2(I2,J2))
                        SUM= SUM+ (TEMP** 2.0)
                     END DO
                  END DO
                  RMS= SQRT( SUM/DENOM)
                  IF( RMS .LT. TH1) THEN
                     CALL MOMINV( PIC2, XSIZE, YSIZE, INVM2)
                     SUM= 0.0
                     TEMP= 0.0
                     DO L=1, 2
                         TEMP= INVM1(L)- INVM2(L)
                         SUM= SUM+ (TEMP** 2.0)
                     END DO
                     METRIC(I,J)= SUM/2.0
                     IF( METRIC(I,J) .LT. MINDAT) THEN
                       MINDAT= METRIC(I,J)
                       EX= I
                       EY= J
                     END IF
                  END IF
               END IF
             END DO
           END DO
           NI= AINT(EX*A0)
           NJ= AINT(EY*A0)
           NI1= AINT(EX*A1)
           NJ1= AINT(EY*A1)
           DO  J= M3, M4
             DO  I= M1, M2
               TEMP1= A0* FLOAT( IMAGE1(I+NI, J+NJ))
               TEMP2= A1* FLOAT( IMAGE2(I-NI1,J-NJ1))
               INTPIC(I,J)= ANINT(TEMP1+ TEMP2)
             END DO
           END DO
         END IF
      ELSE
       TEMP2= 0.0
       TEMP3= 0.0
       DO  J= M3, M4
        DO  I= M1, M2
         TEMP2= TEMP2+ FLOAT( IMAGE1(I,J)+ IMAGE2(I,J))
         TEMP3= TEMP3+ (FLOAT(IMAGE1(I,J))**2.0)
     +    +(FLOAT(IMAGE2(I,J))**2.0)
         TEMP= FLOAT(IMAGE1(I,J))* A0+
     +    FLOAT(IMAGE2(I,J))* A1
         INTPIC(I,J) = ANINT(TEMP)
        END DO
       END DO
       AVG1= TEMP2/(DENOM*2.0)
       STD1= (TEMP3/(DENOM*2.0))- (AVG1**2.0)
      END IF
      M1= M2+ 1
      M2= M2+ XSIZE
   END DO
   M1= 1
```

```
      M2= XSIZE
      M3= M4+1
      M4= M4+ YSIZE
END DO
CALL OPENV(STATUSFLAG,1,1,0,0,0)
CALL OPENV(STATUSFLAG,12,1,0,0,0)
CALL ADJUST(1,NY,NX)
CALL ADJUST(12,NY,NX)
DO LINENUMBER=1,NY
  CALL WRITE(STATUSFLAG,1,0,1,0,NX,INTPIC(1,LINENUMBER),0)
  CALL WRITE(STATUSFLAG,12,0,1,0,NX,LINTPOL(1,LINENUMBER),0)
END DO
CALL RELAB2(1,NY,NX)
CALL RELAB2(12,NY,NX)
END
C=====================================================
C                  SUBROUTINE MOMINV
C-----------------------------------------------------
              SUBROUTINE MOMINV(IPICT,MW,MH,INVMC)
              PARAMETER MZ=2
              INTEGER*2 COUNT1,COUNT2,MH,MW,IPICT(32,32)
              REAL MMENT(0:MZ,0:MZ),ISUM1,MENT
              REAL XBAR,YBAR,TEMP(12),BUF(6)
              INTEGER*4 I1,J1
              REAL CM(0:MZ,0:MZ),CMN(0:MZ,0:MZ)
              REAL EMP1,EMP2,EMP3,EMP4,INVMC(2)
  C
  C===========MODULE TO COMPUTE THE ORDINARY MOMENTS
  C
              DO 5 COUNT2=0,MZ
                 DO 5 COUNT1=0,MZ
                    ISUM1=0.0
                     DO 10 IY=1,MH
                      DO 10 JX=1,MW
                        I1=(JX)**COUNT1
                        J1=(IY)**COUNT2
                       MENT=0.0
                      IF(IPICT(JX,IY).EQ.0)IPICT(JX,IY)=1
            MENT=FLOAT(IPICT(JX,IY))*(FLOAT(I1))*(FLOAT(J1))
                      ISUM1=ISUM1+ABS(MENT)
    10               CONTINUE
                      MMENT(COUNT1,COUNT2)=ISUM1
    5         CONTINUE
  C
  C=========COMPUTATION OF CENTRAL MOMENTS NEEDED
  C
            XBAR=MMENT(1,0)/MMENT(0,0)
              YBAR=MMENT(0,1)/MMENT(0,0)
              CM(0,0)=MMENT(0,0)
              CM(1,0)=0.0
              CM(0,1)=0.0
              CM(2,0)=MMENT(2,0)-XBAR*MMENT(1,0)
              CM(0,2)=MMENT(0,2)-YBAR*MMENT(0,1)
              CM(1,1)=MMENT(1,1)-YBAR*MMENT(1,0)
  C
```

```
C========COMPUTATION OF NORMALISED  CENTRAL MOMENTS
C
              CMN(1,1)=CM(1,1)/(CM(0,0)**2.0)
              CMN(0,2)=CM(0,2)/(CM(0,0)**2.0)
              CMN(2,0)=CM(2,0)/(CM(0,0)**2.0)
C
C=============COMPUTATION OF INVARIANT MOMENTS
C
  INVMC(1)=CMN(2,0)+CMN(0,2)
  INVMC(2)=((CMN(2,0)-CMN(0,2))**2.0)+4.0*(CMN(1,1)**2.0)
              RETURN
              END
```

```
C$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
C                                                                   $
C                        PROGRAM FILTER                             $
C    PROGRAM IMPLEMENTS MOTION COMPENSATED TEMPORAL FILTERING       $
C    USING MOTION ESTIMATES OBTAINED VIA THE METHOD OF              $
C               INVARIANT MOMENTS OF IMAGE BLOCKS.                  $
C$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
        PARAMETER(NX= 350, NY= 240)
        INTEGER STATUSFLAG,LINENUMBER
        INTEGER*2 DENOM, NDX, NDY, KT1, KT2, K1, K2
        INTEGER*2 IMAGE1(NX,NY), IMAGE2(NX,NY)
        INTEGER*2 IMAGE3(NX,NY), FTPIC(NX,NY)
        INTEGER*2 NFTPIC(NX,NY), PIC1(32,32)
        INTEGER*2 PIC2(32,32), PIC3(32,32), NP, FLT
        REAL TEMP1, TEMP2, TEMP3, TEMP4, TEMP5, KMED
        REAL MINDAT,RMS1, RMS2, COUNT1, COUNT2, THR
        REAL BTOM, AX, AY, TH1, TH2, KMIN, KMAX
        COMMON IMAGE1, IMAGE2, IMAGE3, M1, M2, M3, M4
        COMMON PIC1, PIC2, PIC3
        COMMON MTX, MTY, NXSIZ, NYSIZ
        CALL VICINIT('FLT1')
        CALL OPENV(STATUSFLAG,2,0,0,0,0)
        CALL OPENV(STATUSFLAG,3,0,0,0,0)
        CALL OPENV(STATUSFLAG,4,0,0,0,0)
        DO LINENUMBER=1,NY
      CALL READ(STATUSFLAG,2,0,1,0,NX,IMAGE1(1,LINENUMBER),0)
      CALL READ(STATUSFLAG,3,0,1,0,NX,IMAGE2(1,LINENUMBER),0)
      CALL READ(STATUSFLAG,4,0,1,0,NX,IMAGE3(1,LINENUMBER),0)
        ENDDO
C
        TYPE*,'     '
        TYPE*,' MAKE A CHOICE OF FILTERING SCHEME'
        TYPE*,'------------------------------------------'
        TYPE*,' NON RECURSIVE MEAN FILTER------- 1         '
        TYPE*,'  '
        TYPE*,' NON RECURSIVE MEDIAN FILTER----- 2         '
        TYPE*,'   '
        READ(5,*) FLT
        TYPE*,'        THIS FILTER  USES THE      '
        TYPE*,'FEATURE BASED APPROACH TO MOTION ESTIMATION'
        TYPE*,' THE TECHNIQUE SEARCHES OVER AN AREA.'
        TYPE*,' INPUT THE BOUNDS ON THE SEARCH AREA:'
        TYPE*,' MTX AND MTY'
        READ(5,*)MTX,MTY
        TYPE*,'ALSO INPUT THE WINDOW SIZE; NXSIZ AND NYSIZ'
        READ(5,*)NXSIZ,NYSIZ
C
C-----THRESHOLD FOR SEGMENTATION INTO STATIONARY AND MOVING AREAS
C
        DENOM= NXSIZ* NYSIZ
        BTOM= FLOAT(NX*NY)
        TEMP1= 0.0
        COUNT1= 0.0
        COUNT2= 0.0
        TEMP2= 0.0
```

```
          DO J= 1, NY
            DO I= 1, NX
              TEMP1= FLOAT(IMAGE2(I,J)- IMAGE1(I,J))
              TEMP2= FLOAT(IMAGE2(I,J)- IMAGE3(I,J))
              COUNT1= COUNT1+ (TEMP1** 2.0)
              COUNT2= COUNT2+ (TEMP2** 2.0)
             END DO
          END DO
          TH1= SQRT( COUNT1/ BTOM)
          TH2= SQRT( COUNT2/ BTOM)
          M1= 1
          M2= NXSIZ
          M3= 1
          M4= NYSIZ
          NDX= NX/NXSIZ
          NDY= NY/NYSIZ
          DO  KT2= 1, NDY
              DO  KT1= 1, NDX
C
C--------CLASSIFICATION INTO STATIONARY AND MOVING AREAS---
C
                COUNT1= 0.0
                COUNT2= 0.0
                K2= 1
                DO  J= M3, M4
                   K1= 1
                   DO I= M1, M2
                     PIC2(K1,K2)= IMAGE2(I,J)
                     TEMP1= FLOAT( IMAGE2(I,J)- IMAGE1(I,J))
                     TEMP2= FLOAT( IMAGE2(I,J)- IMAGE3(I,J))
                     COUNT1= COUNT1+ (TEMP1** 2.0)
                     COUNT2= COUNT2+ (TEMP2** 2.0)
                     K1= K1+ 1
                   END DO
                   K2= K2+ 1
                END DO
                RMS1= SQRT( COUNT1/DENOM )
                RMS2= SQRT( COUNT2/DENOM )
C
C---------DERIVATION OF MOTION ESTIMATES FOR
C----------MOTION COMPENSSATION
C
                IF( RMS1 .GT. TH1) THEN
                      NP= 1
                      THR= TH1
                      CALL MOVE2( NP, THR)
                ELSE
                      K2= 1
                      DO J= M3, M4
                        K1= 1
                        DO I= M1, M2
                          PIC1(K1,K2)= IMAGE1(I,J)
                          K1= K1+ 1
                        END DO
                        K2= K2+ 1
```

```
                END DO
            END IF
            IF (RMS2 .GT. TH2) THEN
                NP= 3
                THR= TH2
                CALL MOVE2( NP, THR)
            ELSE
                K2= 1
                DO J= M3, M4
                  K1= 1
                  DO I= M1, M2
                    PIC3(K1,K2)= IMAGE3(I,J)
                    K1= K1+ 1
                  END DO
                  K2= K2+ 1
                END DO
            END IF
C
C-------TEMPORAL FILTERING OF IMAGE-----------
C
C--------MEAN FILTERING-----------------
C
            IF (FLT .EQ. 1) THEN
                K2= 1
                DO J= M3, M4
                  K1= 1
                  DO I= M1, M2
                   TEMP1= FLOAT( PIC1(K1,K2)+ PIC2(K1,K2)+
     +                PIC3(K1,K2))
                    TEMP1= TEMP1/3.0
                    FTPIC(I,J)= ANINT(TEMP1)
                    K1= K1+ 1
                   END DO
                   K2= K2+ 1
                END DO
C
C---------MEDIAN FILTERING----
C
            ELSE IF (FLT .EQ. 2) THEN
                K2= 1
                DO J= M3, M4
                  K1= 1
                  DO I= M1, M2
                      TEMP1= FLOAT( PIC1(K1,K2))
                    TEMP2= FLOAT( PIC2(K1,K2))
                    TEMP3= FLOAT( PIC3(K1,K2))
                    KMIN= AMIN1(TEMP1, TEMP2)
                    KMAX= AMAX1(TEMP1, TEMP2)
                    IF (KMAX .LE. TEMP3) KMED= KMAX
                    IF (KMIN .GT. TEMP3) THEN
                        KMED= AMAX1( KMIN, TEMP3)
                    END IF
                    FTPIC(I,J)= ANINT(KMED)
                    K1= K1+ 1
                  END DO
```

```
                    K2= K2+ 1
                  END DO
               END IF
               M1= M2+ 1
               M2= M2+ NXSIZ
            END DO
             M1= 1
             M2= NXSIZ
             M3= M4+1
             M4= M4+ NYSIZ
            END DO
C
C--------UNCOMPENSATED FILTERING--------------
C
         IF (FLT .EQ. 1)THEN
             DO J= 1, NY
             DO I= 1, NX
        TEMP1= FLOAT( IMAGE1(I,J)+ IMAGE2(I,J)+ IMAGE3(I,J))
               TEMP1= TEMP1/3.0
               NFTPIC(I,J)= ANINT(TEMP1)
             END DO
           END DO
          ELSE IF (FLT .EQ. 2) THEN
            DO J= 1, NY
              DO I= 1, NX
                  TEMP1= FLOAT( IMAGE1(I,J))
                TEMP2= FLOAT( IMAGE2(I,J))
                TEMP3= FLOAT( IMAGE3(I,J))
                KMIN= AMIN1(TEMP1, TEMP2)
                KMAX= AMAX1(TEMP1, TEMP2)
                IF (KMAX .LE. TEMP3) KMED= KMAX
                IF (KMIN .GT. TEMP3) THEN
                   KMED= AMAX1(KMIN, TEMP3)
                END IF
                NFTPIC(I,J)= ANINT(KMED)
              END DO
                    END DO
          END IF
            END
C=============================================================
C FEATURE BASED TECHNIQUE OF MOTION ESTIMATION               =
C-------------------------------------------------------------
       SUBROUTINE MOVE2( NP, THR)
             PARAMETER( NX=350, NY=240)
             INTEGER*2 K1, K2, NP,NA, NB, NC, ND, JX, JY
             INTEGER*2 PAC1(32,32), PAC2(32,32), PAC3(32,32)
             INTEGER*2 PIX1(NX,NY), PIX2(NX,NY), PIX3(NX,NY)
             INTEGER*2 WKSP(32,32), I1, J1
             REAL METRIC(-15:15,-15:15), TEMP, SUM, MINDAT
             REAL AINV1(2), AINV2(2), THR, TTHR
             COMMON PIX1, PIX2, PIX3, MA, MB, MC, MD
             COMMON PAC1, PAC2, PAC3
             COMMON MX, MY, NXSZ, NYSZ
             MINDAT= 9998.0
             JX= 0
```

```
      JY= 0
      CALL MOMINV( PAC2, NXSZ, NYSZ, AINV1)
      DO  J= -MY, MY, 1
         NC= MC+ J
         ND= MD+ J
         DO I= -MX, MX, 1
           NA= MA+ I
           NB= MB+ I
           IF((NA .LT. 1) .OR. (NB .GT. NX) .OR. (NC .LT. 1)
     +       .OR. (ND .GT. NY)) THEN
              METRIC(I,J)= 100000.0
           ELSE
              K2= 1
              DO  J1= NC, ND
                 K1= 1
                 DO  I1= NA, NB
                   IF ( NP .EQ. 1)THEN
                     WKSP(K1,K2)= PIX1(I1,J1)
                   ELSE IF( NP .EQ. 3)THEN
                     WKSP(K1,K2)= PIX3(I1,J1)
                   END IF
                   K1= K1+ 1
                 END DO
                 K2= K2+ 1
              END DO
              SUM= 0.0
              DO M= 1, NYSZ
                 DO L= 1, NXSZ
                  TEMP= FLOAT(PAC2(L,M)- WKSP(L,M))
                  SUM= SUM+ TEMP**2.0
                 END DO
              END DO
              TTHR= SQRT(SUM)
              IF (TTHR .LT. THR) THEN
               CALL MOMINV( WKSP, NXSZ, NYSZ, AINV2)
               SUM= 0.0
               TEMP= 0.0
               DO L=1, 2
                 TEMP= AINV2(L)- AINV1(L)
                 SUM= SUM+ (TEMP** 2.0)
               END DO
               METRIC(I,J)= SUM/2.0
               IF( METRIC(I,J) .LT. MINDAT) THEN
                   MINDAT= METRIC(I,J)
                   JX= I
                   JY= J
               END IF
              END IF
            END IF
         END DO
      END DO
       IF (NP .EQ. 1) THEN
          K2= 1
          DO J= MC, MD
            K1= 1
```

```
C========COMPUTATION OF NORMALISED  CENTRAL MOMENTS
C
              CMN(1,1)=CM(1,1)/(CM(0,0)**2.0)
              CMN(0,2)=CM(0,2)/(CM(0,0)**2.0)
              CMN(2,0)=CM(2,0)/(CM(0,0)**2.0)
C
C============COMPUTATION OF INVARIANT MOMENTS
C
  INVMC(1)=CMN(2,0)+CMN(0,2)
  INVMC(2)=((CMN(2,0)-CMN(0,2))**2.0)+4.0*(CMN(1,1)**2.0)
              RETURN
              END
```

REFERENCES


1. **ABU-MOSTAFA, Y.S** and Psaltis, D. (1984) Recognitive Aspects of Moment Invariants. IEEE Trans. on Pattern Analysis and Machine Intelligence, Vol. PAMI-6, No.6, pp698-706.

2. **AGGARWAL, J.K.** (1983) Dynamic Scene Analysis. Image Processing and Dynamic Scene Analysis. Ed. by T.S.Huang. Springer-Verlag Berlin Heidelberg. pp40-73.

3. **ANDERSON, B.D** and Moore, J.B. (1979) Optimal Filtering. Prentice-Hall, Engelwood Cliffs N.J.

4. **ASADA, M,** Yachida, M and Tsuji, S. (1984) Analysis of Three-dimensional Motions in Blocks World. Pattern Recognition, Vol.17, No.1, pp57-71.

5. **BARNEA, D.I** and Silvermann, H.F. (1972) A Class of Algorithms for Fast Digital Image Registration. IEEE Trans. on Computers, Vol. C-21, No.2, pp179-186.

6. **BERGMANN, H.C.** (1983) Analysis of Different Displacement Estimation Algorithms for Digital Television Signals. Image Sequence Processing and Dynamic Scene Analysis. Ed. by T.S.Huang, Springer-Verlag Berlin, Heidelberg 1983 pp215-234.

7. **BERGMANN, H.C.** (1984) Motoion Adaptive Frame Interpolation. International Zurich Seminar on Digital Communications. pp57-61.

8. **BEYER, S.** (1983) Two Motion Adaptive Interframe Coding Techniques for Air to Ground Video Signals. Image Sequence Processing and Dynamic Scene Analysis. Ed. by T.S.Huang, Springer-Verlag Berlin Heidelberg 1983 pp275-291.

9. **BORGHESI, P,** Cappellini, V, et. al. (1984) Two Special Digital Processing Systems for Recognition of Moving Objects. ICASSP ´84, pp12A.9.1-12A.9.4.

10. **BOWLING, C.D** and Jones, R.A. (1985) Motion Compensated Image Coding with a Combined Maximum a posteriori and Regression Algorithm. IEEE Trans. Comms. Vol. Com-33, No.8 pp884-857.

11. **CAFFORIO, C** and Rocca, F. (1976) Methods for Measuring Small Displacements of Television Images. IEEE Trans. Info. Theory, Vol. IT-22, No.5, pp573-579.

12. **CAFFORIO, C** and Rocca, F. (1979) Tracking Moving Objects in Television Images. Signal Processing, 1, pp133-140.

13. **CAFFORIO, C.** (1982) Remarks on the Differential Method for the Estimation of Movement in Television Images. Signal Processing, 4, pp45-52.

14. **CAFFORIO, C** and Rocca, F. (1983) The Differential Method for Image Motion Estimation. Image Sequence processing and Dynamic Scene Analysis. Ed. by T.S.Huang. Springer-verlag Berlin Heidelberg 1983, ppl04-124.

15. **CANDY, J.V.** (1986) Sgnal Processing, the model-based aproach. Mc. Graw-Hill Book Company.

16. **CARLSON, S** and Reillo, C. (1986) Contour Based Representation of the Displacement Field for Motion Compensated Image Coding. ICASSP '86, Tokyo, Japan, ppl61-164.

17. **CONNOR, D.J** and Limb, J.O. (1974) Properties of Frame Difference Signals Generated by Moving Images. IEEE Trans. Comms. Vol. Com-22, No.10, ppl564-1575.

18. **DUBOIS, E,** Prasada, B and Sabri, M.S. (1981) Image Sequence Coding. Image sequence Analysis. Ed. by T.S.Huang. Springer-Verlag, Berlin Heidelberg 1981, pp229-285.

19. **DUBOIS, E** and Sabri, S. (1984) Noise Reduction in Image Sequences using Motion Compensated Temporal Filtering. IEEE Trans. Comms. Vol. Com-32. No.7, pp826-831.

20. **DUBOIS, E.** (1985) The Sampling and Reconstruction of Time-varying Imagery with Application in Video Systems. Proceedings of IEEE, Vol. 73 No.4, pp502-522.

21. **ELNAHAS, S.E** and Tzou, K.H. (1986) Hybrid Interframe Coding of Video Signals with Backward-Acting Motion Detection. ICASSP '86, Tokyo, Japan, ppl65-167.

22. **ERICSSON, S.** (1985) Fixed and Adaptive Predictors for Hybrid Predictive/Transform Coding. IEEE Trans. Comms. Vol. Com-33, No.12, ppl291-1302.

23. **FANG, J-Q** and Huang, T.S. (1984) Some Experiments on Estimating the 3-D Motion Parameters of a Rigid Body from two Consecutive Image Frames. IEEE Trans. Pattern Recognition and Machine Intelligence, Vol. PAMI-6, No.5, pp545-554.

24. **FISCHLER, M.A** and Elschlager, R.A. (1973) The representation and Matching of Pictorial Structures. IEEE. Trans. Computers, Vol. C-27, No.4, pp359-366.

25. **FURNER, R.R,** Christiansen, R.W, et. al. (1986) Motion Compensated Vector Quantization. ICASSP '86, Tokyo, Japan,

pp989-992.

26. **FURUKAWA, A,** Koga, T and Iinuma, K. (1984) Motion-adaptive Interpolation for Video-conference Pictures. IEEE International Conference on Communications, Amsterdam, Links for the future, IEEE/Elsevier Science Publisher B.V (North-Holland), pp707-710.

27. **GONZALEZ, R.C** and Wintz, P. (1977) Digital Image Processing, Addison-Wesley Publishing Co. Inc.

28. **GOSHTASBY, A,** Gage, S.H and Bartholic, J.F. (1984) A Two-stage Cross Correlation Approach to Template Matching. IEEE Trans. Pattern Analysis and Machine Intelligence, Vol. PAMI-6, No.3, pp374-378.

29. **GROVES, M.H,** Rajala, S.A and Synder, W.E. (1984) Calculation of Displacement Fields by Means of the Motion Detection Transform. ICASSP '84, pp23.6.1-23.6.3.

30. **HABIBI, A.** (1977) Survey of Adaptive Image Coding Techniques. IEEE Trans. Comms. Vol. Com-25, No. 11, pp1275-1284.

31. **HASKELL, B.G** and Schmidt, R.L. (1975) A Low-bit-rate Interframe Coder for Videotelephone. BSTJ Vol.54, No.8, pp1475-1495.

32. **HASKELL, B.G.** (1975) Entropy Measurements for non-adaptive and adaptive, frame-to-frame, linear predictive coding of videotelephone signals. BSTJ Vol.54, No.6, pp1155-1174.

33. **HASKELL, B.G,** Gordon, P.L et. al. (1977) Interframe Coding of 525-line, Monochrome Television at 1.5Mbits/s. IEEE Trans. Comms. Vol. Com-25, No.11, pp1339-1348.

34. **HOUKES, Z.** (1983) Motion Parameter estimation in TV-pictures. Image Sequence Processing and Dynamic Scene Analysis. Ed. by T.S.Huang, Springer-Verlag Berlin Heidelberg, 1983, pp249-263.

35. **HU, M-K.** (1962) Visual Pattern Recognition by Moment Invariants. IRE Trans. Info. Theory. IT-8, pp179-187.

36. **HUANG, T.S** (ed) (1979) Picture Processing and Digital Filtering. Springer-Verlag, Berlin, Heidelberg.

37. **HUANG, T.S** and Hsu, Y.P. (1981) Image Sequence Enhancement. Image Sequence Analysis. Ed. by T.S.Huang. Springer-Verlag Berlin Heidelberg, 1981, pp289-309.

38. **HUANG, T.S** and Tsai, R.Y. (1981) Image Sequence Analysis: Motion Estimation. Image Sequence Analysis. Ed. by

T.S.Huang. Springer-Verlag, Berlin, Heidelberg, 1981, ppl-18.

39. **JAIN, J.R** and Jain, A.K. (1981) Displacement Measurement and Its Application in Interframe Image Coding. IEEE Trans. Comms. Vol. Com-29, No.12, ppl799-1808.

40. **JONES, R.A** and Bowling, C.D. (1983) An Adaptive Gradient Approach to Displacement Estimation. Image Sequence Processing and Dynamic Scene Analysis. Ed. by T.s.Huang, Springer-Verlag, Berlin, Heidelberg, 1983, pp235-248.

41. **KANEKO, H** and Ishiguro, T. (1980) Digital Television Transmission using Bandwidth Compression Techniques. IEEE Communicatios Soc. Magazine, Vol. 18, No.4, ppl4-22.

42. **KAPPAGANTIULA, S** and Rao, K.R. (1985) Motion Adaptive Interframe Image Coding. IEEE Trans. Comms. Vol. Com-33, No.9, ppl011-1014.

43. **KRETZMER, E.R.** (1952) Statistics of Television Signals. BSTJ Vol.31, pp751-763.

44. **LEESE, J.A,** Novak, C.S and Taylor, V.R. (1970) The Determination of Cloud Pattern Motions from Geosynchronous Satellite Image Data. Pattern Recognition Vol.2, pp279-292.

45. **LEESE, J.A,** and Novak, C.S and Clarke, B.B (1970) An Automated Technique for Obtaining Cloud Motion from Geosynchronous Satellite Data Using Cross Correlation. Journal of Applied Mteorology Vol.10.

46. **LENZ, R** and Gerhard, A. (1983) Image Sequence Coding using Scene Analysis and Spatio-temporal Interpolation. Image Sequence Processing and Dynamic Scene Analysis. Ed. by T.S.Huang. Springer-Verlag, Berlin, Heidelberg. pp264-274.

47. **LIMB, J.O** and Murphy, J.A. (1975) Estimating the Velocity of Moving Images in Television Signals. Computer Graphics and Image Processing 4, pp311-327.

48. **LIPPMANN, R.** (1980) Video transmission of Ariel Scenes at Reduced Frame Rate Using motion Compensation. IEEE International Conference on Communications, Seatle, Wash., June 8-12, 1980.

49. **LIPPMANN, R.** (1982) Continuous Movement Regeneration in Low-frame-rate Ariel Images. IEE Conference on Electronic Image Processing, U.K, July 1982, ppl94-198.

50. **LIPPMANN, R.** (1983) Low-frame-rate Air-to-ground TV Using Displacement Estimation. 27th. Annual International

Technical Symposium of the society of Photo-optical Instrumentation Engineers (SPIE), San Diego, Calif, Aug. 1983, pp71-74.

51. **MATSUDA, K,** Tsuda, T et. al. (1984) A New Motion Compensation Coding Scheme for Video Conferences. IEEE International Conference on Communications. Amsterdam. Links for the future. IEEE/Elsevier Science Publisher B.V (North-Holland), pp234-237.

52. **MAX, J.** (1960) Quantizing for Minimum Distortion. IRE Trans. Info. Theory, Vol. IT-6, pp7-12.

53. **MUSMANN, H.G,** Pirsch, P and Gallert, H.J. (1985) Advances in picture Coding. Proceedings of IEEE, Vol.73, No.4, pp523-548.

54. **NAGEL, H.H.** (1981) Image Sequence Analysis: What can we learn from applications ? Image Sequence Analysis. Ed. by T.S.Huang, Springer-Verlag, Berlin, Heidelberg 1981, pp229-285.

55. **NAGEL, H.H.** (1982) On Change Detection and Displacement Vector Estimation in Image Sequences. Pattern Recognition Letters 1, pp55-59.

56. **NAGEL, H.H.** (1983) Overview on Image Sequence Analysis. Image sequence Processing and Dynamic Scene Analysis.Ed. by T.S.Huang, Springer-Verlag, Berlin, Heidelberg. 1983, pp2-39.

57. **NAGEL, H.H.** (1984) Recent Advances in Image Sequence Analysis. Premiere Colloque Image Traitment, Synthese, Technologie et Application, pp545-557.

58. **NAHI, N.E.** (1969) Estimation Theory and Applications. John Wiley and Sons, Inc. New York.

59. **NETRAVALI, A.N** and Robbins, J.D. (1979) Motion-compensated Television Coding: Part 1, BSTJ Vol.58, No.3, pp631-670.

60. **NETRAVALI, A.N** and Stuller, J.A. (1979) Motion-compensated transform Coding. BSTJ Vol.58, No.7, ppl703-1718.

61. **NINOMIYA, Y** and Ostsuka, Y. (1984) a Motion-ccompensated Interframe Coding Scheme for NTSC Colour Television Signals. IEEE Trans. Comms. Vol. Com-32, No.3, pp328-334.

62. **O´ NEAL, J.B.** (1966) Predictive Quantizing Systems (Differential Pulse Code Modulation for the Transmission of Television Signals), BSTJ Vol.45, pp689-721.

63. **PEASE, R.F.W** and Limb, J.O. (1971) Exchange of Spatio-temporal Resolution in Television Coding. BSTJ Vol.ŁŁ, No.ŁŁ, ppl91-199.

64. **PLOMPEN, R.H.J.M,** Ggroeveld, J.G.P et. al. (1986) The Performance of a Hybrid Videoconferencing Coder Using Displacement Estimation in the Transform Doamain. ICASSP ´86, Tokyo, Japan, ppl57-160.

65. **RABINER, L.R** and Gold, B. (1975) Theory and Application of Digital Sigal Processing. Prentice-Hall Inc. Englewood Cliffs. New Jersey, U.S.A.

66. **REDDI, S.S.** (1981) Radial and Angular Moment Invariants for Image Identification. IEEE Trans. on Pattern Recognition and Machine Intelligence, Vol. PAMI-3, No.2, pp240-242.

67. **RICHARD, C,** Benveniste, A and Kretz, F. (1984) Recursive Estimation of Local Characteristics of Edges in TV Pictures as applied to ADPCM Coding, IEEE Trans. Comms. Vol. Com-32, No.6, pp718-728.

68. **RIDDLE, A.N** and Rajala, S.A. (1981) A Method for Simultaneous Velocity Detection and Object Identification Using Fourier Transformation. ICASSP ´81, pp202-206.

69. **ROACH, J.W** and Aggarwal, J.K. (1980) Determining the Movement of Objects from a Sequence of Images. IEEE Trans. on Pattern Analysis and Machine Intelligence, Vol. PAMI-2, No.6, pp554-562.

70. **ROBBINS, J.D** and Netravali, A.N. (1983) Recursive Motion Compensation: A Review. Image Sequence Processing and Dynamic Scene Analysis. Ed. by T.S.Huang, Springer-Verlag, Berlin, Heidelberg, 1983, pp75-103.

71. **ROCCA, F** and Zanoletti, S. (1972) Bandwidth Reduction via movement Compensation on a model of the random video process. IEEE Trans. Comms. Vol. Com- No. , pp960-965.

72. **ROSENFELD, A.** (1984) Image Analysis: Problems, Progress and prospects. Pattern Recognition. Vol.17, No.1, pp3-12.

73. **SRINIVASAN, R** and Rao, K.R. (1985). Predictive Coding Based on Efficient Motion Estimation. IEEE Trans. Comms. Vol. Com-33, No.8, pp888-895.

74. **STULLER, J.A** and Netravali, A.N. (1979) Transform Domain motion Estimation. BSTJ Vol.58, No.7, ppl673-1702.

75. **TEAGUE, M.R.** (1980) Image Analysis via The General Theory of Moments. Journal of the Optical Society of America,

Vol.70, No.8, pp920-930.

76. **TITMARSH, E.C.** (1939) The Theory of Functions. Oxford University Press. 2nd. Ed. pp38-39.

77. **TSAI, R.Y** and Huang, T.S. (1981) Estimating Three-dimensional Motion Parameters of a Rigid Planar Patch. IEEE Trans. on Acoustic, Speech and Signal Processing, Vol. ASSP-29, No.6, ppl147-1152.

78. **TSAI, R.Y.** (1983) Estimating 3-D Motion Parameters and Object Surface Structures from the Image Motion of Conic Arcs, I: Thoeretical Basis. ICASSP '83. Boston. ppl22-125.

79. **TSAI, R.Y** and Huang, T.S. (1984) Uniqueness and Estimation of Three-dimensional Motion Parameters of Rigid Objects with Curved surfaces. IEEE Trans on Pattern Analysis and Machine Intelligence, Vol. PAMI-6, No.1, ppl3-26.

80. **VANDERBURG, G.J** and Rosenfeld, A. (1977) Two-stage Template Matching. IEEE Trans. on Computers, Vol. C-26, No.4, pp384-393.

81. **WALKER, D.R** and Rao, K.R. (1984) New Techniques in Pel-recursive Motion Compensation. IEEE International Conference on Communications. Amsterdam. Links for the future, IEEE/Elsevier Publisher B.V (North-Holland).

82. **WALKER, D.R** and Rao, K.R. (1984) Improved Pel-recursive Motion Compensation. IEEE Trans. Comms. Vol. Com-32, No.10, ppl128-1134.

83. **WELLS, S.C.** (1985) Efficient motion Estimation for Interframe Video Coding Applications. Electronics Letters, Vol.21, No.7, pp289-290.

84. **WONG, R.Y** and Ernest, L.H. (1978) Scene Matching with Invariant Moments, Computer Graphics and Image Processing 8, ppl6-24.

85. **WONG, R.Y** and Ernest, L.H. (1978) Sequential Hierarchical Scene Matching. IEEE Trans. on Computers, Vol. C-27, No.4, pp359-366.

86. **YEN, B.L** and Huang, T.S. (1983) Determining 3-D Motion and Structure of a Rigid Body Using Straight Line Correspondences. ICASSP '83, Boston, ppl18-124.

87. **ZSCHUNKE, W.** (1977) DPCM Picture Coding with Adaptive Prediction. IEEE Trans. Comms. Vol. Com-25, No.11, ppl295-1302.