MOTION ESTIMATION AND INTERPOLATION IN TIME-VARYING IMAGERY

by

Edward A. Krause

B.Sc.E., University of New Brunswick
(1982)

SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE
DEGREE OF

MASTER OF SCIENCE
IN ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

at the

Massachusetts Institute of Technology
September, 1984
© Massachusetts Institute of Technology 1984

Signature of Author

Department of Electrical Engineering and Computer Science
September, 1984

Certified by

William F. Schreiber
Thesis Supervisor

Accepted by

Arthur C. Smith
Chairman, Departmental Committee on Graduate Students

ARCHIVES

MASSACHUSETTS INSTITUTE
OF TECHNOLOGY

APR 0 1 1985

LIBRARY
MOTION ESTIMATION AND INTERPOLATION IN
TIME-VARYING IMAGERY

by

EDWARD A. KRAUSE

Submitted to the Department of
Electrical Engineering and Computer Science
in September, 1984 in partial fulfillment of the
requirements for the degree of
Master of Science

ABSTRACT

A motion compensated temporal interpolation scheme is presented as a means
of signal enhancement and/or bandwidth reduction in time-varying imagery. Three
distinct operations are performed: estimation of the displacement field, boundary
and parameter estimation, and the interpolation itself. Displacement prediction is
based on a pel recursive gradient-search scheme featuring adaptive apriori predic-
tion, recursive fade compensation, and immediate smoothing of the displacement
vector field. Object boundaries are located by observing both the displacement and
image fields. Object movements are modeled as a combination of linear translation,
rotation, and zooming, and all parameters are computed in a logical manner based
on the detection of simple characteristics evident from the displacement field. The
interpolator constructs complete images by forward and backward interpolation of
the moving objects followed by superposition on a full background extracted from
the known key frames.

Reasonably good results were obtained for each of three test sequences. Almost
all observed imperfections were attributable to errors in the displacement
field.

Thesis Supervisor: Dr. William F. Schreiber
Title: Professor of Electrical Engineering
ACKNOWLEDGEMENTS

I would like to thank Prof. W. F. Schreiber for his ideas, advice, and criticism. I am grateful to Jason Sara who proved to be an invaluable aid in overcoming various system problems. I also wish to thank Jeff Bernstein and Brian Hinman for their assistance in recording one of the image sequences essential to this study.
# TABLE OF CONTENTS

**CHAPTER 1: Introduction** ................................................................. 1

**CHAPTER 2: Review of Processing Techniques** ................................. 3
Perception and Rendition of Motion ......................................................... 3
Interframe Coding ...................................................................................... 5
Conditional Replenishment ........................................................................ 7
Motion Compensation .................................................................................. 8
    Scene Segmentation ................................................................................. 8
    Motion Estimation ................................................................................... 10
    Correlating Regions ............................................................................... 10
    Linear Prediction .................................................................................... 11
    Spatial and Temporal Differences ........................................................ 12
    Motion Estimation in Transform Domains ............................................. 12
    Pel Recursive Estimation ....................................................................... 13
    Example .................................................................................................. 17
    Motion-Compensated Color Coding ..................................................... 20
Noise Reduction by Temporal Filtering ..................................................... 20
Temporal Interpolation ................................................................................. 21

**CHAPTER 3: Motion Prediction** ............................................................ 23
Apriori Prediction of the Displacement Field .......................................... 23
Comparison With Other Prediction Techniques ....................................... 29
Estimating the Spatial and Temporal Gradients ...................................... 33
Luminance Compensation .......................................................................... 39
Performance Evaluation ............................................................................ 40
    Apriori Prediction Using the Previous Frame ...................................... 42
    Apriori Prediction Using the Present Frame ........................................ 44
    Adaptive Apriori Prediction ................................................................. 46

**CHAPTER 4: Image Segmentation and Interpretation** .......................... 49
Image Segmentation ................................................................................... 49
Image Interpretation .................................................................................. 50
# LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Figure 1</td>
<td>Window of Visibility</td>
<td>4</td>
</tr>
<tr>
<td>Figure 2</td>
<td>Smoothing Function</td>
<td>16</td>
</tr>
<tr>
<td>Figure 3</td>
<td>Example of Motion-Compensated Coding</td>
<td>18</td>
</tr>
<tr>
<td>Figure 4</td>
<td>Adaptive Block Structure</td>
<td>19</td>
</tr>
<tr>
<td>Figure 5</td>
<td>Prediction of the Displacement Vector</td>
<td>25</td>
</tr>
<tr>
<td>Figure 6</td>
<td>Computing a Present-Frame Apriori Predictor</td>
<td>27</td>
</tr>
<tr>
<td>Figure 7</td>
<td>Gradient Filter</td>
<td>36</td>
</tr>
<tr>
<td>Figure 8</td>
<td>Examples of Gradient Filtering</td>
<td>38</td>
</tr>
<tr>
<td>Figure 9</td>
<td>Translating Square</td>
<td>41</td>
</tr>
<tr>
<td>Figure 10</td>
<td>Previous Frame Prediction of Displacement</td>
<td>43</td>
</tr>
<tr>
<td>Figure 11</td>
<td>Present Frame Prediction of Displacement</td>
<td>45</td>
</tr>
<tr>
<td>Figure 12</td>
<td>Adaptive Apriori Prediction of Displacement</td>
<td>47</td>
</tr>
<tr>
<td>Figure 13</td>
<td>Computation of Motion Parameters</td>
<td>52</td>
</tr>
<tr>
<td>Figure 14</td>
<td>Blending Edges of Moving Object into Background</td>
<td>56</td>
</tr>
<tr>
<td>Figure 15</td>
<td>Test Sequence 1</td>
<td>57</td>
</tr>
<tr>
<td>Figure 16</td>
<td>Magnitude and Error for Test Sequence 1</td>
<td>59</td>
</tr>
<tr>
<td>Figure 17</td>
<td>Moving and Nonmoving Components of Test Sequence 1</td>
<td>60</td>
</tr>
<tr>
<td>Figure 18</td>
<td>Test Sequence 2</td>
<td>61</td>
</tr>
<tr>
<td>Figure 19</td>
<td>Magnitude and Error for Test Sequence 2</td>
<td>62</td>
</tr>
<tr>
<td>Figure 20</td>
<td>Moving and Nonmoving Components of Test Sequence 2</td>
<td>64</td>
</tr>
<tr>
<td>Figure 21</td>
<td>Test Sequence 3</td>
<td>65</td>
</tr>
<tr>
<td>Figure 22</td>
<td>Magnitude and Error for Test Sequence 3</td>
<td>67</td>
</tr>
<tr>
<td>Figure 23</td>
<td>Moving and Nonmoving Components of Test Sequence 3</td>
<td>68</td>
</tr>
<tr>
<td>Figure 24</td>
<td>Interpolation of Test Sequence 1</td>
<td>74</td>
</tr>
<tr>
<td>Figure 25</td>
<td>Interpolation of Test Sequence 2</td>
<td>75</td>
</tr>
<tr>
<td>Figure 26</td>
<td>Ideal Interpolation of Test Sequence 2</td>
<td>77</td>
</tr>
<tr>
<td>Figure 27</td>
<td>Interpolation of Test Sequence 3</td>
<td>78</td>
</tr>
<tr>
<td>Figure 28</td>
<td>Extrapolation of Test Sequence 3</td>
<td>80</td>
</tr>
</tbody>
</table>
## LIST OF TABLES

Table 1: Motion Parameters for Test Sequence 1 ........................................ 58  
Table 2: Motion Parameters for Test Sequence 2 ........................................ 63
CHAPTER 1

Introduction

It is well known that a series of properly chosen still images can convey the illusion of motion if presented in sequence at a suitable display rate. However, the display rate cannot be chosen carelessly. If we desire freedom from artifacts, such as discontinuous motion caused by aliasing, then we must consider both the velocity of motion and the "sharpness" of the moving object. A rapidly moving object demands a higher frame rate than a slow one, otherwise we may not perceive any motion at all. Of course sharp objects are preferred to blurred ones, but unfortunately this demands a faster display rate if the movement is to appear natural.

When an object moves rapidly across the television screen, there is a noticeable loss of detail. If we sharpen the images and display them again, we obtain clarity but only at the expense of motion continuity. Perhaps the optimum solution is to sharpen the images and boost the display rate. However, there are reasons that we may not wish to do so. The additional bandwidth demanded from communication channels may not be readily available and so we may choose instead to seek powerful coding algorithms to compress the signal. Another possibility is to transmit the signal without change and then construct additional images at the receiver by means of interpolation. In this way a 60 Hz. signal could be interpolated up to 120 Hz., or if the interpolation is indeed successful, a signal of only 30 or perhaps 15 Hz. could be interpolated up to the same rate. Interpolation is potentially valuable, not only as a means of signal enhancement, but also as a form of bandwidth compression.

An interpolator that improves motion rendition must employ motion compensation. In fact, the measurement and interpretation of motion between successive frames is the most critical step affecting performance. The problem can be
approached in two ways. The first does not require any knowledge of object boundaries. It simply interpolates the estimated displacement of each moving pel or each moving block of the image. The second procedure, which has been adopted here for simulation, attempts to locate the boundaries of all significant objects. Although this adds complexity to the problem, it also offers numerous advantages. By interpolating the movement of an entire object it is possible to retain surface detail without distortion or blurring. The area uncovered by the object as it moves can be filled in by accessing the following frame. On the other hand, if the object is large, a single displacement vector may not be sufficient to characterize its motion. An angle of rotation and a magnification factor can also be estimated simply by observing the displacement vector field associated with the image sequence. A reasonably accurate vector field can also be invaluable to the segmentation process which detects object boundaries.

After interpreting the displacement field, the sequence is ready for interpolation. A two-step procedure has been adopted: 1) the present frame is interpolated forward and 2) the next frame is interpolated backwards. A weighted average is then performed where the weighting coefficients are calculated to favor the "closest" of the two key frames. This insures a smooth transition between frames and introduces blurring only when an object undergoes a sudden transformation in shape or does not move as expected.

An advantage of this interpolator is its ability to make decisions. By recognizing the existence of individual objects, it is possible to make crucial interpolation decisions when different objects interact or when new information unexpectedly appears in a scene. It is interesting to note that a conventional motion compensated coder which does not interpolate can achieve high efficiency and yield good results even when the predicted movements are completely wrong! The mistakes may never show up if the results are not used for the purpose of interpolation.
CHAPTER 2

Review of Processing Techniques in Time-Varying Imagery

This chapter is intended to serve as a background for those that follow. A survey of coding and motion prediction methods is provided as an introduction to the estimation scheme to be developed in the next chapter. Segmentation and interpolation are also covered but in less detail.

1. Perception and Rendition of Motion

The spatial and temporal frequency response of the human visual system bears some resemblance to a low pass filter. In the spatial domain, sinusoidal variations above a critical threshold frequency are invisible [1], and in the temporal domain, a similar threshold is known to exist [2]. Koenderink and van Doorn [3] found these two thresholds to be nearly independent and used a window of visibility to approximate the human visual response (Figure 1). The assumption is that an image component will be more or less visible only if both the spatial and temporal frequencies are below their respective thresholds. This theory is a useful one as it allows spatial and temporal experiments to be carried out independently.

If two stimuli have the same spectra within the window of visibility, then we may expect them to appear identical. Assume, for example, that one of these stimuli is a sequence of images (image frames) obtained by sampling the first stimulus in both space and time. Then by the sampling theorem, we would expect the two stimuli to appear identical if the spatial and temporal sampling frequencies are at least twice the maximum spatial and temporal frequency components respectively, of the continuous stimulus. If the maximum frequency components exceed the human thresholds, then the image can be low-pass filtered and then sampled at

---

1 Some experimenters believe that the spatial and temporal threshold frequencies are not independent.
Figure 1: Window of Visibility
a lower rate. For the sampling rates used in conventional television, the low-pass filtering effect of the camera is sufficient.

Watson et al. [4] performed experiments with a vertical line moving smoothly across a television screen at fixed velocity. They demonstrated that the sampled image sequence could be made indistinguishable from the continuous one (without low-pass filtering before sampling) even though the latter possessed infinitely high spatial and temporal frequencies. Although this is a special case where the spatial and temporal sampling frequencies cannot be chosen independently, it may be of considerable interest to animators and others dealing with computer-generated imagery. Since these images bypass the prefiltering effect of the camera, very high frequency components are common, and consequently artifacts due to aliasing may arise even at very high sampling rates. In such cases, it may be desirable to simulate the filtering effect of the camera, or any other conceivable technique that achieves suitable motion blur [5].

Television signals cannot be modeled ideally as mere samples of a flawless continuous signal. In the spatial domain, we may wish to account for the dependence among adjacent picture elements of the camera and CRT display. In addition, most image display devices do not display frames in the stroboscopic format associated with conventional sampling. Phosphor decay in a cathode ray tube is exponential while film projectors display images in a sample and hold format. Watson et al. [4] extended their experiments to include the sample and hold model and found that the same conclusions were appropriate after allowing for a few approximations.

2. Interframe Coding

A large percentage of television research has been devoted solely to the purpose of compressing or reducing the amount of information needed to produce television pictures. Many such techniques have proven to be successful and many of these make use of the redundancy that exists between consecutive frames of
most picture sequences.

Typical PCM coders may require a bit rate ranging from 75-86 Mbits/s to achieve the same quality as present day analog television. Intraframe coders may reduce this rate by a factor of two while simple interframe predictive coding methods may typically achieve a bit rate of 20-30 Mbits/s.

If a receiver can predict the appearance of a scene with reasonable success, then only the errors will need to be transmitted, usually at a relatively low bit rate. Most prediction rules are somewhat arbitrary and are often decided by trial and error. Much study has gone into the development of quantizers for transmitting prediction errors [6]. Poor quantizer design may result in slope overload, granular noise, and edge busyness. The quality of an image with visible quantization steps can be subjectively improved by the insertion of pseudorandom noise before and after quantization [7].

Other processing methods involve separating an image into components. For instance, if a two-channel system separates the high-frequency component of an image from the low-frequency component, then the highs can be finely sampled and coarsely quantized, and vice versa for the lows [8, 9]. For interframe coding purposes, it would be advantageous to separate the moving components from a still background. Contour coders or even the high-frequency component of the two channel system can be used with some success.

Picturephone Meeting Service Conference Centers were introduced by the Bell System in July 1982 [10]. Video compression technology is responsible for reducing the bit rate to 1.5 - 3 Mbits/s, including a TDM compressed chrominance signal inserted into the horizontal blanking intervals. Plural channels permit adaptive bit sharing during simultaneous broadcasts. Sabri et al. [11] developed a video coder for Teleconferencing operating at only 50 kb/s. The utilization of conditional replenishment and motion compensation are essential to achieving adequate picture quality at this rate.
3. Conditional Replenishment

If there is little movement between picture frames, then it is wasteful to transmit all of the information associated with each frame. Instead a system may be devised in which only those zones where movement occurs are updated by transmitting the errors and the picture element addresses [38]. Consider, however, what happens when there is not enough bandwidth to update all of the picture elements that have been changed due to some rapid movement. The simplest conditional replenishment system would simply update as much of the picture as possible and leave the rest unchanged. One can easily imagine very unusual noise patterns arising from such a method. More refined algorithms may resort to a lower order quantizer or perhaps update only every second pel that is affected by the movement. Even so it is impossible to guarantee against the occurrence of objectionable noise patterns.

Since replenishment data is usually generated in bursts, a buffer is needed to smooth out irregularities in the transmission rate. Error control may be forward acting or an erroneous line can simply be replaced with the previous one. Buffer overflow can be prevented by adaptive quantization, subsampling, field repeating, or frame freezing. Robbins and Netravali [12] preferred subsampling as a means of buffer control. They described an adaptive interpolation method that produced blurring only in the unpredictable areas of a picture.

Brainard et al. [13] have contrasted conditional replenishment with frame repetition. They found that although 15 new pictures/s could be displayed at 30 Hz with reasonable results, selective replenishment of one quarter of the picture elements per frame gave better motion continuity. However, because of the objectionable noise patterns, they were unable to establish a preference between the two. In fact both methods caused serious impairments during zooming and panning.
4. Motion Compensation

Suppose we are provided with information telling us not only the moving parts of an image, but their velocity and direction as well. Then theoretically we can reconstruct the next image in the sequence without transmitting any information. In other words, we have a motion-compensated system. Of course, the computations needed to provide this motion information are not trivial, and their accuracy may not be perfect, but we can deal with faulty information simply by introducing differential coding. The idea is the same as in conditional replenishment, but now we may assume that smaller corrections are needed and the channel bandwidth can be reduced.

Many investigators have shown that motion compensation can improve the data compression ratio by a factor of 1.5 to 2.0 when compared with conditional replenishment. Ninomiya and Ohtsuka [14] have developed an experimental real-time motion-compensated coder and reported good picture quality at 16 Mb/s for broadcast television signals.

The idea of motion compensation is simple but implementation can be quite complicated. Many processes first require that the image be segmented so that the moving parts can be readily identified. Unless the segmentation is performed at the receiver, the boundary information must be transmitted. In such cases, the channel bandwidth may not need to be increased since motion information for a segmented image can usually be transmitted more efficiently. A similar trade-off concerns computations; time expended for segmentation can often be recovered when computing motion information. A more detailed discussion of both segmentation and motion estimation procedures is justified.

4.1. Scene Segmentation

If we merely wish to classify a pel as moving or nonmoving, we can threshold the frame difference (the difference in intensity of the same pel in consecutive
frames). Limb and Murphy [15] applied the same principle to blocks of 18 pels each and classified the entire block as nonmoving if fewer than 6 pels had frame differences exceeding the threshold.

The idea of using frame differences as a criterion for segmenting images can be improved upon. If a displacement vector is computed in advance for a previous adjacent pel, and the displaced frame difference defined as the intensity difference between pels at both ends of the vector, but in consecutive frames, then a segmentation decision can be made by examining both the frame difference of the present pel and the displaced frame difference of the adjacent one. This approach was used by Netravali and Robbins [16].

A variety of optimal segmentation algorithms have been proposed and all are based on a Markov model [17]. If the possible states of the present pel are either "background" or "moving", then the decision can be made by minimizing a cost function. The cost of transmitting the prediction error will be greater if the pel is moving, and an additional cost will be incurred if it is necessary to inform the receiver that the present pel has a different state from the preceding one. State transition probabilities are predicted based on past experience and the Viterbi algorithm is used to minimize the expected cost. Unfortunately the Viterbi Algorithm is a somewhat lengthy procedure, particularly as the number of states is increased (additional states are required if we allow more than one moving object).

In the vast majority of cases, segmentation is performed to simplify the motion estimation task. However, when the objective is to extract information from an image sequence instead of coding it for the purpose of transmission, it may be worthwhile to use motion information to assist segmentation. For instance Newmann [18] investigated a method for using motion parameters to locate independently moving objects, based on 3D interpretability. Both Hildreth [19] and Nagel [20] contributed valuable information to the treatment of edges and points.
4.2. Motion Estimation

Several methods have been developed to determine the optical flow of an image function, or in other words, the direction and velocity of motion. All methods to be presented here seek to assign displacement vectors (also called motion vectors) to various parts of an image. Perhaps the most important distinction to be made among all of these algorithms concerns the resolution of the resulting displacement vector matrix. In some cases, a displacement vector is computed for every picture element. In others, a motion vector represents an entire block of picture elements, and if the image has been fully segmented, each moving object may be represented by a single vector. Often these displacement vectors are computed at the receiver, but if they are few in number (as may be the case when there is only one vector assigned to each moving object), then it may be efficient to compute the vectors accurately at the transmitter so that they may be transmitted to the receiver.

4.2.1. Correlating Regions

Perhaps the most popular method of displacement estimation is one which uses a correlation function to match regions in consecutive frames. Usually the regions are predefined by dividing the image into blocks so that the displacement vectors are those which link block pairs found to have the highest correlation. In almost all cases, the displacement vector \(d\) is chosen to minimize a correlation function \(r\) of the form:

\[
 r(d) = \sum_{x \in R} f\left(DFD(x, d)\right)
\]  

(1)

where \(x\) is the location of a pel in some region \(R\) and \(DFD\) is the corresponding displaced frame difference. Possibilities for the function \(f\) range from:
\[ f(DFD) = \begin{cases} 1 & \text{if } |DFD| > \epsilon \\ 0 & \text{otherwise} \end{cases} \]

used by Rocca and Zanoletti [21], to

\[ f(DFD) = \log_2 |DFD| + 1 \]

used by Ninomiya and Ohtsuka [14]. Ninomiya and Ohtsuka obtained their displacement vectors by incrementing the vector of the corresponding block in the previous frame by a quantity selected from a prespecified menu. Of course the increment that is selected must be the one which minimizes Eq.(1). Rocca and Zanoletti [21] suggested an interesting but complex searching scheme. A random model of a video signal was constructed based on the distribution of pel intensity levels, the distribution of run lengths (number of adjacent pels with same intensity), and both spatial and temporal correlation. This model is used to generate a random number of lines with random orientation, each moving at its own random but constant velocity. The lines effectively divide the image into polygons which can translate, rotate, change shape, and even appear or disappear. They conclude that the optimum displacement vector minimizing the selected correlation function is the one which successfully matches the corners of polygons in consecutive frames. Thus a unique and unusual search procedure is suggested.

4.2.2. Linear Prediction

Haskell [22] used linear predictive coding (LPC) to estimate pel intensity. The predicted intensity \( \hat{I}(x) \) is obtained as

\[ \hat{I}(x) = \sum_i \alpha_i I(x_i) \]  \hspace{1cm} (2)

where \( \alpha_i \) are the weighting coefficients and \( I(x_i) \) is the intensity of the pel at the
location determined by \( x_i \). Haskell's algorithm was quite flexible. Depending on local conditions, \( x_i \) could be either the address of preceding pels on the same line, the same pel in the preceding frame, or the pel in the preceding frame that corresponds to a predicted displacement vector. The weighting coefficients \( (\alpha_i) \) are derived in the standard LPC manner of minimizing the mean squared prediction error. Haskell proposed using different coefficients for various regions of an image.

4.2.3. Spatial and Temporal Differences

Limb and Murphy [15, 23] tried a very different approach. To determine horizontal velocity within a moving area, the magnitude of the frame difference signal is summed in one accumulator and the horizontal pel difference is summed in a second. If the signs of the two sums are the same, then movement to the left is indicated; if they differ, then movement is to the right. The magnitude of the horizontal velocity component was then obtained by subtracting one accumulator from the other and dividing the result by the sum of the magnitude of the element difference signal. Vertical velocity is computed in an analogous manner. These results were then generalized by Cafforio and Rocca [17] and extended to suit more complex images.

4.2.4. Motion Estimation in Transform Domains

Haskell [24] observed that displacement vectors were often easily detected in the frequency domain. The frame-to-frame delay or advance in time associated with translation produced a detectable linear phase shift in the Fourier Transform.

Stuller and Netravali [25] developed a motion-compensated system using transform coding. The image is divided into blocks, each having an apriori estimate of the displacement vector. A two-dimensional linear transform is computed for each block. Then in each case, the predicted displacement vector is used to locate a corresponding block in the previous frame which must also be transformed. The
differences between the predicted and actual transform coefficients are then coded by taking advantage of the nonuniform frequency distribution and varying importance with respect to reproduction quality. The quantized coefficients are transmitted to the receiver where the displacement vectors are reconstructed and used to reproduce the image. Since we are transmitting just the information needed to improve the receiver’s apriori estimate of the transform coefficients, approximate convergence can be expected if we are dealing with simple translation. For this reason, the process is referred to as recursive transform coding.

4.2.5. Pel Recursive Estimation

At this point a derivation will be presented so that it may be expanded upon in the next chapter. The derivation presented here serves to justify the same result as that observed by Horn and Schunk [26].

Let \( d_x(x, y, t) \) and \( d_y(x, y, t) \) be the horizontal and vertical components of the displacement vector at \((x, y, t)\). Let \( I(x, y, t) \) be the intensity of the pel at \((x, y, t)\), and \( \nabla_x(x, y, t) \) and \( \nabla_y(x, y, t) \), the horizontal and vertical gradients respectively. Then if the magnitude of the displacement vector is small, the following truncated Taylor Series will be valid:

\[
I(x + d_x, y + d_y, t + 1) \approx I(x, y, t + 1) + \begin{bmatrix} \nabla_x & \nabla_y \end{bmatrix} \begin{bmatrix} d_x \\ d_y \end{bmatrix}
\] (3)

Since \((d_x, d_y)\) is the true displacement vector, it can be assumed that:

\[
I(x + d_x, y + d_y, t + 1) = I(x, y, t)
\] (4)

and if we define:

\[
\nabla_t(x, y, t) = I(x, y, t + 1) - I(x, y, t)
\] (5)
then (4) can be substituted into (3) to yield:

\[
\begin{bmatrix}
\nabla_x & \nabla_y
\end{bmatrix}
\begin{bmatrix}
d_x \\
\frac{d_y}{d_y}
\end{bmatrix}
+ \nabla_t = 0
\]

(6)

We must now deal with the problem of finding the displacement vector which satisfies Eq.(6). Unfortunately there are an infinite number of solutions and so we must seek an additional constraint in order to single out the optimum vector. Since we desire that the vector field be smooth, we can define \((\bar{d}_x, \bar{d}_y)\) as the displacement vector previously computed for the preceding pel on the same line, and then select the solution of (6) which minimizes the following expression:

\[
\left( d_x - \bar{d}_x \right)^2 + \left( d_y - \bar{d}_y \right)^2
\]

(7)

Rather than solve for all solutions of (6), we can introduce \(\lambda\) as a Lagrange multiplier and then find the displacement vector which minimizes the following expression:

\[
\left( d_x - \bar{d}_x \right)^2 + \left( d_y - \bar{d}_y \right)^2 + \lambda \left\{ \begin{bmatrix}
\nabla_x & \nabla_y
\end{bmatrix}
\begin{bmatrix}
d_x \\
\frac{d_y}{d_y}
\end{bmatrix} + \nabla_t \right\}^2
\]

(8)

Differentiating with respect to both \(d_x\) and \(d_y\), setting the result to zero, and solving gives:

\[
\begin{bmatrix}
\frac{d_x}{\bar{d}_x} \\
\frac{d_y}{\bar{d}_y}
\end{bmatrix} = \begin{bmatrix}
\frac{\bar{d}_x}{\bar{d}_x} \\
\frac{\bar{d}_y}{\bar{d}_y}
\end{bmatrix} - \frac{\lambda}{1 + \lambda (\nabla_x^2 + \nabla_y^2)} \left\{ \begin{bmatrix}
\nabla_x^2 & \nabla_x \nabla_y \\
\nabla_x \nabla_y & \nabla_y^2
\end{bmatrix}
\begin{bmatrix}
\frac{\bar{d}_x}{\bar{d}_x} \\
\frac{\bar{d}_y}{\bar{d}_y}
\end{bmatrix} + \begin{bmatrix}
\nabla_x \\
\nabla_y
\end{bmatrix} \nabla_t \right\}
\]

(9)

In a sense, this result is a recursive one, since the previous displacement vector esti-
mate is used to predict the next. Perhaps the greatest deficiency lies in the definition of \((\bar{d}_x, \bar{d}_y)\).

In order to discover the effect of \(\lambda\) on Eq. (9), we can define:

\[
\nabla = \nabla_x^2 + \nabla_y^2
\]

and then plot the response of

\[
f(\nabla) = \frac{\lambda}{1 + \lambda \nabla}
\]

for various values of \(\lambda\). From Eq. (9) we see that the displacement vector field is smoother for small values of \(f(\nabla)\) than large ones. Figure 2 shows that when \(\nabla\) is large, \(f(\nabla)\) is very small and is approximately the same for all values of \(\lambda\). So when large spatial gradients arise as a result of edges in the image, the effect of \(\lambda\) is almost negligible and little smoothing occurs, but when the gradient is small in magnitude, \(f(\nabla)\) (and consequently Eq. (9)) is very sensitive to \(\lambda\). We therefore have a simple means of controlling smoothness and reducing noise without seriously affecting the displacement prediction at edges. This observation was not made by Horn and Schunk.

Paquin and Dubois [27] and Netravali and Robbins [28] also developed pel recursive vector estimation schemes. Paquin and Dubois derived their result in a similar manner but did not apply a smoothness constraint. Netravali and Robbins not only omitted the smoothness constraint, but based their derivation on the technique of linear regression. Significant approximations were applied to simplify the result.
Figure 2: Smoothing Function
4.2.6. Example

A simple motion compensated coder was simulated using the block matching scheme discussed in Section 4.2.1. A displacement vector is specified to the receiver by transmitting its address in a predefined vector menu, and differential correction data is subsequently transmitted for each pel. In addition each block can be recursively subdivided into 4 smaller blocks if this succeeds in reducing the bit rate without affecting image quality. Whenever a block is subdivided so that it is representable by 4 displacement vectors instead of one, it may be possible to use a lower order quantizer with the effect of lowering the bit rate. On the other hand, additional overhead is necessary to inform the receiver of the changes, so that further subdivision eventually becomes impractical.

A motion sequence was artificially generated with an intensity function specified as:

\[
I(r, t) = \frac{255 \left(1 + e^{-0.01r}\right) \cos \left(2\pi \frac{r - t}{40}\right)}{2}
\]  

(10)

where \( t \) represents time and \( r \) is the distance from the center of the image. The sequence gives the illusion of circular waves expanding away from the center of the picture as they become attenuated. Figures 3A and 3B show the two frames in the sequence generated for \( t = 5 \) and \( t = 15 \) respectively. Figures 3C and 3D show the same two figures reproduced by motion-compensated coding at an average rate of 0.98 bits per pel instead of the 8 bits per pel used in the original sequence. Figure 4 shows one frame after each sub-block has been quantized to represent its average intensity. It clearly reveals the block sizes resulting from adaptation.
Figure 3: Example of Motion-Compensated Coding
Figure 4: Adaptive Block Structure
4.3. Motion-Compensated Color Coding

Prabhu and Netravali [29, 30] applied earlier work in motion compensation to color coding and found that it was sufficient to estimate the motion parameters using the luminance channel. The same information could then be applied to the two chrominance channels with only a negligible increase in the bit rate. They noted that additional reduction was possible using a separate quantizer for the chrominance components or by transmitting fewer chrominance samples. They also showed that the displacement of objects could be estimated in the composite domain without decomposing the composite signal into its original components. However, they also noted that this estimate was much less reliable.

Iinuma et al. [31] described an interframe coding system capable of transmitting a 4 MHz NTSC color signal. The NTSC signal was multiplexed in the time domain with compressed luminance and chrominance components in sequence and encoded as frame differences. The quality was judged acceptable for teleconferencing and video telephone transmission at 6.3 Mbits/s.

5. Noise Reduction by Temporal Filtering

Noise Reduction is based on the assumption that images are highly correlated from frame to frame, and noise is not. By performing temporal filtering in the nonmoving parts of an image, McMann et al. [32] was able to substantially reduce the noise component without affecting the signal. However, the moving regions remained noisy and this was quite noticeable in slowly moving areas.

Motion-compensated temporal filtering seeks to avoid this problem by filtering in the direction of motion. Displacement can be estimated (preferably for each pel) using the techniques of Section 4.2. Unlike uncompensated filters, it is possible to reduce noise and preserve detail and texture on moving objects, assuming that the displacement vectors are computed accurately and at sufficient resolution.
Huang and Hsu [33] described the use of nonrecursive linear and median temporal filters while Dubois and Sabri [34] demonstrated slightly improved performance with nonlinear recursive filters.

6. Temporal Interpolation

Temporal interpolation refers to the process of generating "inbetween" frames from a series of "key" images. By transmitting just the key frames, and reconstructing the missing ones at the receiver, it is possible to greatly reduce the bandwidth of the communications channel. Although the application may be different, the problem is one which animators have been dealing with for many years [35]. Of course there are distinguishing differences between cartoons and live imagery, and for this reason it may not be wise to share the same solutions.

Computer-generated imagery and most animated sequences are generally much easier to interpolate accurately than live television scenes. For this reason, much research has gone into the development of algorithms which seek and model the movements of objects in three dimensions. Usually this demands that severe constraints be imposed on the moving object. Neumann [36] summarizes some of the most common assumptions:

1. only 1 moving object (or other finite number)
2. object must be rigid
3. objects must be independent
4. objects can translate only
5. objects can rotate only
6. objects can translate and rotate only

None of these assumptions are suited for live imagery. A much preferred procedure is one which can accept all forms of movement and still yield acceptable, but not necessarily perfect, results. If the algorithm must fail, the consequences should
not be disastrous.

Netravali et al. [37] have proposed a simple scheme where each pel is interpolated based on the direction and magnitude of the associated displacement vector. No attempt is made to identify objects and discover the complex details of their movements.
CHAPTER 3

Motion Prediction

In the preceding chapter, a method was presented for predicting displacement vectors based on local spatial and temporal gradients:

\[
\begin{bmatrix}
\tilde{d}_x \\
\tilde{d}_y
\end{bmatrix} = \begin{bmatrix}
\tilde{d}_x \\
\tilde{d}_y
\end{bmatrix} - \frac{\lambda}{1+\lambda(\nabla_x^2 + \nabla_y^2)} \left( \begin{bmatrix}
\nabla_x^2 & \nabla_x \nabla_y \\
\nabla_x \nabla_y & \nabla_y^2
\end{bmatrix} \begin{bmatrix}
\tilde{d}_x \\
\tilde{d}_y
\end{bmatrix} + \begin{bmatrix}
\nabla_x \\
\nabla_y
\end{bmatrix} \nabla_t \right)
\]

(11)

Recall that \( \lambda \) affects the overall smoothness of the displacement vector field by limiting the permissible variation between \((d_x, d_y)\) and the previously computed motion vector \((\tilde{d}_x, \tilde{d}_y)\).

The ability of Eq. (11) to produce satisfactory and reliable results is fully determined by the degree in which \( \nabla_x \) and \( \nabla_y \) approximate the average horizontal and vertical gradients between \((x, y)\) and \((x + d_x, y + d_y)\). In practice, the gradients \( \nabla_x \) and \( \nabla_y \) are usually estimated by observing the behavior of the picture elements adjacent to \((x, y)\). Consequently, significant errors can be expected if the correct displacement vector is large in magnitude or if the image does not vary smoothly between \((x, y)\) and \((x + d_x, y + d_y)\). A modified displacement field estimation technique will be developed and tested to demonstrate improved performance, particularly in the presence of large displacement vectors and sharp discontinuities in the image field.

1. Apriori Prediction of the Displacement Field

Let \( \dot{d}_x(x, y, t) \) and \( \dot{d}_y(x, y, t) \) denote apriori estimates of \( d_x(x, y, t) \) and \( d_y(x, y, t) \) respectively. Then by incorporating \((\dot{d}_x, \dot{d}_y)\) in the motion prediction scheme, it will be shown that the desired spatial gradients \((\nabla_x, \nabla_y)\) are the average
gradients between \( (x + \hat{d}_x, y + \hat{d}_y) \) and \( (x + d_x, y + d_y) \). Then the magnitude of \( (d_x, d_y) \) is no longer relevant; we need only be concerned with the magnitude of \( (d_x - \hat{d}_x, d_y - \hat{d}_y) \).

If the spatial gradient vectors are indeed estimated with this in mind, then Figure 5 suggests the following approximation:

\[
I(x + d_x, y + d_y, t + 1) - I(x + \hat{d}_x, y + \hat{d}_y, t + 1) = \begin{bmatrix} \nabla_x & \nabla_y \end{bmatrix} \begin{bmatrix} d_x - \hat{d}_x \\ d_y - \hat{d}_y \end{bmatrix} \tag{12}
\]

Let \( \nabla'_r(x, y, t) \) be defined as the displaced frame difference, varying as a function of space and time, so that:

\[
\nabla'_r(x, y, t) = I \left( x + \hat{d}_x(x, y, t), y + \hat{d}_y(x, y, t), t + 1 \right) - I(x, y, t) \tag{13}
\]

Again it will be convenient to drop the subscripts from the gradient and displacement vectors.

Since \( (d_x, d_y) \) is the true displacement vector, it can be assumed that:

\[
I(x + d_x, y + d_y, t + 1) = I(x, y, t) \tag{14}
\]

In a later section, this assumption will be generalized to allow for moving objects which experience changes in luminance or reflectivity.

Equations (13) and (14) can be substituted into (12) yielding:

\[
\begin{bmatrix} \nabla_x & \nabla_y \end{bmatrix} \begin{bmatrix} d_x - \hat{d}_x \\ d_y - \hat{d}_y \end{bmatrix} + \nabla'_r = 0 \tag{15}
\]
Figure 5: Prediction of the Displacement Vector
As in Section 4.2.5 of Chapter 2, a smoothness constraint will be imposed by limiting the size of:

\[
\left( d_x - \bar{d}_x \right)^2 + \left( d_y - \bar{d}_y \right)^2
\]  \hspace{1cm} (16)

but now \((\bar{d}_x, \bar{d}_y)\) will be obtained by averaging the preceding displacement vectors on both the present line and the preceding line (Figure 6). Note that since the preceding displacement vectors are associated with the same frame, the adjacent locations must be chosen so that \((\bar{d}_x, \bar{d}_y)\) is recursively computable.

As before, the Lagrange multiplier technique is used to find the displacement vector \((d_x, d_y)\) which satisfies (15) and minimizes (16). This is done by minimizing the following expression:

\[
\left( d_x - \bar{d}_x \right)^2 + \left( d_y - \bar{d}_y \right)^2 + \lambda \left\{ \begin{bmatrix} \nabla_x & \nabla_y \end{bmatrix} \begin{bmatrix} d_x - \hat{d}_x \\ d_y - \hat{d}_y \end{bmatrix} + \nabla_i \right\}^2
\]  \hspace{1cm} (17)

Differentiating separately with respect to \(d_x\) and \(d_y\) and setting both results to zero gives:

\[
2 \begin{bmatrix} d_x - \bar{d}_x \\ d_y - \bar{d}_y \end{bmatrix} + 2\lambda \begin{bmatrix} \nabla_x \\ \nabla_y \end{bmatrix} \left\{ \begin{bmatrix} \nabla_x & \nabla_y \end{bmatrix} \begin{bmatrix} d_x - \hat{d}_x \\ d_y - \hat{d}_y \end{bmatrix} + \nabla_i \right\} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}
\]  \hspace{1cm} (18)

or

\[
\begin{bmatrix} 1 + \lambda \nabla_x^2 & \lambda \nabla_x \nabla_y \\ \lambda \nabla_x \nabla_y & 1 + \lambda \nabla_y^2 \end{bmatrix} \begin{bmatrix} d_x \\ d_y \end{bmatrix} = \begin{bmatrix} \bar{d}_x \\ \bar{d}_y \end{bmatrix} + \lambda \begin{bmatrix} \nabla_x^2 & \nabla_x \nabla_y \\ \nabla_x \nabla_y & \nabla_y^2 \end{bmatrix} \begin{bmatrix} \hat{d}_x \\ \hat{d}_y \end{bmatrix} - \lambda \begin{bmatrix} \nabla_x \\ \nabla_y \end{bmatrix} \nabla_i
\]  \hspace{1cm} (19)

If the inverse of the first 2 x 2 matrix is evaluated, then:
\[ \bar{d}(x, y) = \frac{1}{4} \left( d(x - 1, y) + d(x - 1, y + 1) + d(x, y + 1) + d(x + 1, y + 1) \right) \]

Figure 6: Computing a Present-Frame Apriori Predictor
\[
\begin{bmatrix}
\frac{d_x}{d_y}
\end{bmatrix} = \frac{1}{1 + \lambda \left( \nabla_x^2 + \nabla_y^2 \right)} \begin{bmatrix}
1 + \lambda \nabla_y^2 & -\lambda \nabla_x \nabla_y \\
-\lambda \nabla_x \nabla_y & 1 + \lambda \nabla_x^2
\end{bmatrix}
\begin{bmatrix}
\frac{\tilde{d}_x}{\tilde{d}_y}
\end{bmatrix} + \lambda \begin{bmatrix}
\nabla_x^2 & \nabla_x \nabla_y \\
\nabla_x \nabla_y & \nabla_y^2
\end{bmatrix} \begin{bmatrix}
\frac{\tilde{d}_x - d_x}{\tilde{d}_y - d_y}
\end{bmatrix} - \lambda \begin{bmatrix}
\nabla_x \\
\nabla_y
\end{bmatrix} \nabla_i'
\]

This expression conveniently simplifies to give the final result:

\[
\begin{bmatrix}
\frac{d_x}{d_y}
\end{bmatrix} = \begin{bmatrix}
\frac{\tilde{d}_x}{\tilde{d}_y}
\end{bmatrix} - \frac{\lambda}{1 + \lambda \left( \nabla_x^2 + \nabla_y^2 \right)} \begin{bmatrix}
\nabla_x^2 & \nabla_x \nabla_y \\
\nabla_x \nabla_y & \nabla_y^2
\end{bmatrix} \begin{bmatrix}
\frac{\tilde{d}_x - d_x}{\tilde{d}_y - d_y}
\end{bmatrix} + \begin{bmatrix}
\nabla_x \\
\nabla_y
\end{bmatrix} \nabla_i'
\]

Ideally, \( \lambda \) is chosen to insure that the displacement vector satisfies Eq.(15), but generally it is more convenient to adjust this parameter in order to achieve the desired amount of smoothing.

Note that (21) is very similar to (9) except now there are two apriori estimates of \((d_x, d_y)\). Actually, \((\tilde{d}_x, \tilde{d}_y)\) was conceived in order to implement a smoothness constraint, but there is no reason that it cannot serve as an apriori estimate as well. On the other hand, there are instances when \((\tilde{d}_x, \tilde{d}_y)\) would be a preferable substitute for \((d_x, d_y)\) (i.e. spatial smoothness is traded for temporal smoothness). Specifically, consider what would happen if the image becomes very flat so that the spatial gradient vector \((\nabla_x, \nabla_y)\) is zero. Eq.(21) would then reduce to:

\[
\begin{bmatrix}
\frac{d_x}{d_y}
\end{bmatrix} = \begin{bmatrix}
\frac{\tilde{d}_x}{\tilde{d}_y}
\end{bmatrix}
\]

Under normal circumstances, this would be a very desirable simplification since it
assigns useful motion vectors to an object's interior when no motion is indicated by the spatial and temporal gradients. It is also possible that this portion of the image may not be moving at all\(^1\), in which case, the displacement vector errors will be very serious.

An interesting solution has been employed. No distinction is made between \((\vec{d}_x, \vec{d}_y)\) and \((\hat{d}_x, \hat{d}_y)\) and instead, a new predictor \((D_x, D_y)\) is introduced and allowed to adapt to the local image features by switching smoothly between \((\vec{d}_x, \vec{d}_y)\) and \((\hat{d}_x, \hat{d}_y)\) in such a way as to favor the estimate offering the lowest displaced frame difference. As an added benefit, (21) simplifies tremendously since \(\vec{d}\) and \(\hat{d}\) are now the same:

\[
\begin{bmatrix}
  d_x \\
  d_y
\end{bmatrix} = \begin{bmatrix}
  D_x \\
  D_y
\end{bmatrix} - \frac{\lambda}{1+\lambda(\nabla^2_x+\nabla^2_y)} \begin{bmatrix}
  \nabla_x \\
  \nabla_y
\end{bmatrix} \nabla_i
\]

(23)

2. Comparison With Other Gradient Search Motion Prediction Techniques

Before proceeding, it is instructive to compare this result with some of the other pel-recursive algorithms that were presented in Chapter 2. Eq.(11), used by Horn and Schunck, has actually been simplified by accommodating an apriori displacement predictor to improve performance! It must be realized that \((\vec{d}_x, \vec{d}_y)\) as used in (11) does not serve as an apriori predictor at all since \(\nabla_i\) is not a displaced frame difference.

\[
\nabla_i = I(x, y, t+1) - I(x, y, t)
\]

(24)

This can be contrasted with \(\nabla_i\) of Eq.(23):

\(^1\)If the estimation scheme worked predictably, there would be no reason to expect this type of mistake to occur. In a later section, however, an example will show that a common feature in scene imagery will consistently trigger this kind of error.
\[ \nabla'_t = I(x + D_x, y + D_y, t + 1) - I(x, y, t) \]  
(25)

In the ideal case when the image varies uniformly between \((x, y, t + 1)\) and \((x + D_x, y + D_y, t + 1)\), \(\nabla'_t\) can be expressed in terms of \(\nabla_t\):

\[
\nabla'_t = I(x, y, t + 1) + \begin{bmatrix} \nabla_x & \nabla_y \end{bmatrix} \begin{bmatrix} D_x \\ D_y \end{bmatrix} - I(x, y, t) \\
= \nabla_t + \begin{bmatrix} \nabla_x & \nabla_y \end{bmatrix} \begin{bmatrix} D_x \\ D_y \end{bmatrix} 
\]

(26)

This can be substituted in place of \(\nabla'_t\) in (23) to yield:

\[
\begin{bmatrix} d_x \\ d_y \end{bmatrix} = \begin{bmatrix} D_x \\ D_y \end{bmatrix} - \frac{\lambda}{1 + \lambda(\nabla_x^2 + \nabla_y^2)} \begin{bmatrix} \nabla_x \\ \nabla_y \end{bmatrix} \begin{bmatrix} \nabla_t + \begin{bmatrix} \nabla_x & \nabla_y \end{bmatrix} \begin{bmatrix} D_x \\ D_y \end{bmatrix} \end{bmatrix} \\
= \begin{bmatrix} D_x \\ D_y \end{bmatrix} - \frac{\lambda}{1 + \lambda(\nabla_x^2 + \nabla_y^2)} \begin{bmatrix} \nabla_x^2 & \nabla_x \nabla_y \\ \nabla_x \nabla_y & \nabla_y^2 \end{bmatrix} \begin{bmatrix} D_x \\ D_y \end{bmatrix} + \begin{bmatrix} \nabla_x \\ \nabla_y \end{bmatrix} \nabla_t \]

(27)

Now it can be seen that (27) is identical to Horn and Schunck's equation (11) when expressed in this suboptimal form.

The algorithm also bears some similarity to that of Netravali and Robbins [28]. After applying several approximations they arrived at the similar result:

\[
\begin{bmatrix} \bar{d}_x \\ \bar{d}_y \end{bmatrix} = \begin{bmatrix} \bar{d}_x \\ \bar{d}_y \end{bmatrix} - \epsilon \begin{bmatrix} \nabla_x \\ \nabla_y \end{bmatrix} \nabla'_t 
\]

(28)
Even though this equation was derived in a very different manner, it bears very close resemblance to (23) except that the somewhat arbitrary constant \( \epsilon \) is used in place of:

\[
\frac{\lambda}{1 + \lambda (\nabla_x^2 + \nabla_y^2)}
\]  

(29)

Of course, \( \lambda \) is also an arbitrary constant, but nevertheless, the term above is greatly affected by the magnitude of the spatial gradient. Since the size of the increment to the previous estimate is almost inversely proportional to the magnitude of \((\nabla_x, \nabla_y)\), smoothness characteristics will differ from those resulting from Netravali's equation.

On the other hand, if computation time is the prime issue, then Netravali’s equation is the clear winner. However, the computation of (29) can be quite satisfactorily approximated with a small look-up table having \( \nabla_x \) and \( \nabla_y \) as inputs. In fact, it is not unreasonable to expect satisfactory performance using only \( 4 \times 4 \) input levels and \( 4 \) output levels.

It should also be emphasized that Netravali obtained his apriori displacement vector estimate \((\bar{d}_x, \bar{d}_y)\) from the previous pel on the same line in the same frame. Thus the second distinguishing difference of Eq.(23) is in the definition of \( \nabla_i' \).

Less similarity exists between (23) and the equation derived by Paquin and Dubois [27]:

\[
\begin{bmatrix}
\frac{d_x}{d_y}
\end{bmatrix} = \begin{bmatrix}
\nabla_x^2 \cdot \nabla_x \nabla_y \\
\nabla_x \nabla_y \quad \nabla_y^2
\end{bmatrix}^{-1} \begin{bmatrix}
\nabla_x \\
\nabla_y
\end{bmatrix} \left( \nabla_i' - \begin{bmatrix}
\nabla_x \\
\nabla_y
\end{bmatrix} \begin{bmatrix}
\bar{d}_x \\
\bar{d}_y
\end{bmatrix} \right)
\]  

(30)

Clearly an inverse for the \( 2 \times 2 \) matrix does not exist (i.e. there is no unique solution) and therefore each individual element in this equation was adjusted by sum-
ming or averaging over a region surrounding the picture element for which a dis-
placement estimate is to be computed. Not only does the summing operation make
the equation solvable but it also smooths the displacement vector field.

Unfortunately, their solution is not ideal. If local detail is lacking in the image,
then the gradients may not change over the entire region of summation, so that:

\[
\begin{bmatrix}
\sum_{R} \nabla_{x}^{2} & \sum_{R} \nabla_{x} \nabla_{y} \\
\sum_{R} \nabla_{x} \nabla_{y} & \sum_{R} \nabla_{y}^{2}
\end{bmatrix}
= \begin{bmatrix}
i \times j \times
\begin{bmatrix}
\nabla_{x}^{2} & \nabla_{x} \nabla_{y} \\
\nabla_{x} \nabla_{y} & \nabla_{y}^{2}
\end{bmatrix}
\end{bmatrix}
\] (31)

where \(i \times j\) is the number of picture elements included in the summation region \(R\). Again, an inverse does not exist. Paquin and Dubois failed to comment on this
possibility and did not provide an alternate formula.

In contrast to the scheme of Netravali and Robbins, Paquin and Dubois used
the computed motion vector of the same picture element in the previous frame as
an apriori displacement vector estimate. At this point, it is necessary to clarify the
definition of \((\hat{d}_{x}, \hat{d}_{y})\) as used in the adaptive method proposed in Section 1.
Although it is obtained from the previous frame, it does not correspond to the pic-
ture element in the same location. Rather, it is defined as follows:

\[
\hat{d} \left( x + d(x, y, t - 1), \ y + d(x, y, t - 1), \ t \right) = d(x, y, t - 1)
\] (32)

This differs from the previous frame predictor of Paquin and Dubois where:

\[
\hat{d}(x, y, t) = d(x, y, t - 1)
\] (33)

If a picture element is located near the edge of a moving object, then the same pic-
ture element in the preceding frame may be associated with a different object mov-
ing in a different direction at different velocity; thus the apriori predictor defined
by (33) may do more harm than good.

Although the predictor obtained through the use of Eq.(32) will certainly be more valuable, the equation is rather difficult to implement. Since \(d(x, y, t-1)\) need not be integral, and because a one-to-one correspondence does not exist between picture elements in consecutive frames, some kind of interpolation is necessary to insure that some pels are not assigned more than one displacement vector estimate while others have none.

A segmentation algorithm (to be discussed in a later section) greatly simplifies this task. This is because the segmentation scheme isolates those objects or regions that can be characterized by a single translation vector. The location of each object in the next frame is then easily estimated by extrapolation.

3. Estimating the Spatial and Temporal Gradients

If we desire the gradient vector \((\nabla_x, \nabla_y, \nabla_t)\) at \((x, y, t)\) then a very good approximation can be obtained by taking simple pel differences:

\[
\nabla_x(x, y, t) \approx \frac{I(x+1, y, t) - I(x-1, y, t)}{2}
\]

(34)

\[
\nabla_y(x, y, t) \approx \frac{I(x, y+1, t) - I(x, y-1, t)}{2}
\]

(35)

\[
\nabla_t(x, y, t) \approx \frac{I(x, y, t+1) - I(x, y, t-1)}{2}
\]

(36)

Since causality is not a strict constraint, there is no reason that a central difference, using the following frame, cannot be used to estimate the temporal gradient. However, the proposed motion estimation algorithm makes use of a displaced temporal gradient defined as:

\[
\nabla'_t(x, y, t) = I(x + D_x, y + D_y, t+1) - I(x, y, t)
\]

(37)
Simple pel differences produce very reasonable estimates of the true gradient vector at selected picture elements. The problem is that the motion estimation algorithm is based on a truncated Taylor Series and often accuracy cannot be achieved by observing the local gradient at a single picture element. On the other hand, accuracy will be guaranteed if we are able to determine the average gradient along the line connecting \((x + D_x, y + D_y, t + 1)\) and \((x + d_x, y + d_y, t + 1)\) where \((d_x, d_y)\) is the unknown displacement vector (Section 1). If we assume that the gradient at the unknown position \((x + d_x, y + d_y, t + 1)\) is the same as the gradient observed at \((x, y, t)\), then it is possible to average the gradient vectors observed at both ends of the line:

\[
\nabla_x (x, y, t) = \frac{1}{2} \left( \frac{I(x + D_x, y + D_y, t + 1) - I(x + D_x - 1, y + D_y, t + 1)}{2} \\
+ \frac{I(x + 1, y, t) - I(x - 1, y, t)}{2} \right) 
\]

\[
\nabla_y (x, y, t) = \frac{1}{2} \left( \frac{I(x + D_x, y + D_y + 1, t + 1) - I(x + D_x, y + D_y - 1, t + 1)}{2} \\
+ \frac{I(x, y + 1, t) - I(x, y - 1, t)}{2} \right) 
\]

Consider what happens if a sharp edge is present in the region where a gradient vector is observed. If the central differences do not encompass the edge at all, then the estimated average gradient will be much too small in magnitude. Similarly, the magnitude will be much too large if the edge is included in the central difference calculation. For this reason, it was decided that the image should be low-pass filtered in order to "smear" the edges before the spatial gradient vectors are estimated. Actually, the image remains unchanged but the gradient algorithm is modified so that the net result is the same. For instance, consider a symmetric
2n+1 point one-dimensional FIR filter with coefficients \((a_n)\) as shown in Figure 7. Then if the horizontal gradient is defined by:

\[
\nabla_x(x, y, t) = I(x+1, y, t) - I(x, y, t)
\]

where the image has been low-pass filtered in the \(x\) direction only so that:

\[
I(x, y, t) = a_0 I(x, y, t)
\]

\[+ a_1 \left( I(x+1, y, t) + I(x-1, y, t) \right) \]

\[+ a_2 \left( I(x+2, y, t) + I(x-2, y, t) \right) \]

\[+ \cdots \]

\[+ a_n \left( I(x+n, y, t) + I(x-n, y, t) \right) \]

and

\[
I(x+1, y, t) = a_0 I(x+1, y, t)
\]

\[+ a_1 \left( I(x+2, y, t) + I(x, y, t) \right) \]

\[+ a_2 \left( I(x+3, y, t) + I(x-1, y, t) \right) \]

\[+ \cdots \]

\[+ a_n \left( I(x+n+1, y, t) + I(x-n+1, y, t) \right) \]
Figure 7: Gradient Filter
then the same result could also be obtained by performing the following computation on the original unfiltered image:

\[
\nabla_x(x, y, t) = (a_0 - a_1) \left( I(x + 1, y, t) - I(x, y, t) \right) \\
+ (a_1 - a_2) \left( I(x + 2, y, t) - I(x - 1, y, t) \right) \\
+ (a_{n-1} - a_n) \left( I(x + n, y, t) - I(x - n + 1, y, t) \right) \\
+ a_n \left( I(x + n + 1, y, t) - I(x - n, y, t) \right)
\]

Similarly, the vertical gradient can be computed by assuming one-dimensional filtering in the y direction. Two-dimensional filtering can provide more accurate gradient estimates when the angle formed by an object boundary is acute, but generally the improvement is not sufficient to justify the added complexity.

A filter which transforms a sharp edge into a triangular edge (Figures 8A and 8B) may seem attractive at first since a gradient vector computed on this new edge would then be identical to the true average gradient along a line having both endpoints on the same edge. In general, the filter which produces the smoother contour of Figure 8C may be preferred. In all cases, the improvement due to filtering is very noticeable when estimating interframe motion for computer-generated synthetic images.

A more straightforward approach is used to compute the displaced temporal gradient \( \nabla_t \). The image intensity in both the present and following frames is simply defined as the average of several picture elements in a small rectangular region sur-
Examples of Gradient Filtering
rounding the relevant spatial coordinates.

This definition of the spatial and temporal gradients was chosen as a similar but preferred alternative to the summing or averaging operation used by Paquin and Dubois when implementing Eq.(31).

4. Luminance Compensation

When an object changes location or position, the quantity of light reflected is likely to change. It may pass into the shadow of some other object, or the lighting conditions may vary. Recalling the derivation of the displacement vector scheme, the algorithm cannot be expected to perform flawlessly under such conditions. An alternate formula is presented to improve performance under such circumstances.

Assume that we have reason to believe that a region of the image is changing in luminance. Then the basic gradient equation of Section 1 can be modified from:

\[ I(x + D_x, y + D_y, t + 1) - I(x, y, t) = - \left[ \begin{array}{c} \nabla_x \\ \nabla_y \end{array} \right] \left[ \begin{array}{c} d_x - \hat{d}_x \\ d_y - \hat{d}_y \end{array} \right] \]

(44)

to:

\[ I(x + D_x, y + D_y, t + 1) - I(x, y, t) + \hat{L}(x, y, t) = - \left[ \begin{array}{c} \nabla_x \\ \nabla_y \end{array} \right] \left[ \begin{array}{c} d_x - \hat{d}_x \\ d_y - \hat{d}_y \end{array} \right] \]

(45)

where \( \hat{L}(x, y, t) \) is the predicted change in the intensity function during movement from \( (x, y, t) \) to \( (x + d_x, y + d_y, t + 1) \). If we define:

\[ \nabla'_{i_x}(x, y, t) = I(x + D_x, y + D_y, t + 1) - I(x, y, t) + \hat{L}(x, y, t) \]

(46)

then the estimation equation based on Eq.(23) can be generalized simply by substituting \( \nabla'_{i_x} \) for \( \nabla_i \).
The method of applying this equation will be more fully understood after reading latter sections. \( \hat{L} \) is selected in a manner similar to the selection of \( \hat{d} \). During the first application of Eq.(47), \( \hat{d} \) and \( \hat{L} \) are both zero. The scene is then segmented (Chapter 4) and the values of \( \hat{d} \) and \( \hat{L} \) in the next frame are estimated simply by observing their values in each section of the segmented image. Thus, it is a recursive process, and the single function of both \( \hat{d} \) and \( \hat{L} \) is to speed convergence. After all, both motion and luminance changes usually persist over several frames, and are rarely instantaneous. Even if this were not the case, the response time of the television camera may be slow enough (depending on the frame rate of course) to permit stable convergence. Actually, if computation time were not an issue, Eq.(47) could be used to perform several iterations on a single frame of the sequence, before proceeding to the next.

5. Performance Evaluation

The motion estimation algorithm developed in the preceding sections can be effectively demonstrated with a very simple example. The test sequence consists of a single large square moving from the position shown in Figure 9A to that of Figure 9B. A displacement vector is estimated for each picture element of Figure 9A using each of three different apriori prediction schemes; 1. previous frame prediction (\( \hat{d} \)), 2. present frame prediction (\( \bar{d} \)), and 3. adaptive apriori prediction (\( D \)).

In spite of the apparent simplicity of this example, it poses a special challenge to the motion estimator. This is due to the lack of smooth gradients. The null gradients, observed throughout almost the entire image, are of no practical value, while the extremely large spatial gradients along the four edges of the square may result in severe errors. Gradients of this magnitude are not found in typical televi-
sion transmissions due to the low-pass filtering effect that occurs in camera tubes. On the other hand, sharp edges are typical of animation sequences and large flat (zero-gradient) regions are frequent in almost all time-varying imagery. The displacement vectors associated with the interior picture elements of a moving object lacking detail should be obtained by interpolating from the edges. Otherwise the displacement vectors assigned to blank areas may be random and interpretation of movements becomes difficult.

5.1. Apriori Prediction Using the Previous Frame

Displacement vectors are calculated using:

\[
\begin{bmatrix}
    d_x \\
    d_y
\end{bmatrix}
= \begin{bmatrix}
    \hat{d}_x \\
    \hat{d}_y
\end{bmatrix} - \frac{\lambda}{1 + \lambda (\nabla_x^2 + \nabla_y^2)} \begin{bmatrix}
    \nabla_x \\
    \nabla_y
\end{bmatrix} \nabla'_i
\] (48)

where \( \nabla'_i \) is the displaced frame difference determined by \( \hat{d} \), the apriori displacement vector obtained from the previous frame. As an arbitrary starting condition, \( \hat{d} \) is set to one half of the value of the actual error-free displacement vectors corresponding to each picture element of Figure 9A.

Figure 10A shows the magnitude of the displacement vectors computed using Eq.(48). Except for a luminance scale factor, this figure should ideally be identical to Figure 9A. Figure 10B shows the displaced frame difference (\( \nabla'_i \)) between the two key frames, using the new computed displacement vectors. The neutral or grey regions indicate zero error, while the white and black regions indicate large positive and negative errors respectively.

Observe the presence of large displacement errors at the lower right corner of the square in Figure 10B. Of course, this is due to the large frame difference resulting from the movement of the square, but it is objectionable since it complicates the interpretation of Figure 10A. Also note the lack of smoothness inside the
Figure 10: Previous Frame Prediction of Displacement
boundaries of the square. As suggested by Eq.(48), the computed motion vectors are identical to \( \hat{d} \) in the regions where either \( \nabla_t \) or both \( \nabla_x \) and \( \nabla_y \) become zero.

5.2. Apriori Prediction Using the Present Frame

Displacement vectors are calculated using

\[
\begin{bmatrix}
    d_x \\
    d_y 
\end{bmatrix}
= \begin{bmatrix}
    \bar{d}_x \\
    \bar{d}_y 
\end{bmatrix} - \frac{\lambda}{1 + \lambda (\nabla_x^2 + \nabla_y^2)} \begin{bmatrix}
    \nabla_x \\
    \nabla_y 
\end{bmatrix} \nabla_t'
\]

(49)

where \( \bar{d} \) is the present frame predictor that was illustrated in Figure 6 of Chapter 2. Of course \( \nabla_t' \) is now computed using \( \bar{d} \) instead of \( \hat{d} \). Again Figure 11A shows the magnitude of the updated displacement vectors and Figure 11B shows the difference between the two key frames using the displacement vectors of Figure 11A. Notice that the error is much improved over that of Figure 10B even though something is very wrong with the computed displacement vectors. Evidently, when the spatial gradient became zero, Eq.(49) reduced to:

\[
\begin{bmatrix}
    d_x \\
    d_y 
\end{bmatrix}
= \begin{bmatrix}
    \bar{d}_x \\
    \bar{d}_y 
\end{bmatrix}
\]

(50)

Although this simplification was effective when assigning displacement vectors to the interior of the square, some procedure is definitely needed to prevent these large vectors from propagating outside the object’s boundaries. If Figure 11B had revealed some error outside the lower left boundaries of the square, then this would not have happened, but the risk will always exist when working with a background which, like the object itself, lacks detail.

A final comment concerns the direction of propagation as demonstrated by Figure 11A. This results from the definition of \( \bar{d} \) as the average of displacement
Figure 11: Present Frame Prediction of Displacement
vectors previously computed for the preceding pel on the same line and adjacent pels on the preceding line. Although the propagation outside the boundaries of the square is unfortunate, accuracy within the boundaries requires considerable propagation, preferably in all directions.

A scheme can be devised to achieve uniform propagation in all directions. The sequence in which displacement vectors are computed resembles the scanning pattern of a CRT display, but if logic and simplicity are ignored, then a two-pass scheme can be implemented where the relevant picture elements corresponding to each pass are interleaved in a regular fashion. That is, during the first pass, motion vectors may be assigned to every second pel whether proceeding either horizontally or vertically. Then, during the second pass, \( \tilde{d} \) could be derived from the first-pass displacement vectors of the adjacent pels in any direction. Although the technique is straightforward and would assure propagation in all directions, implementation has not been attempted. After all, Netravali and Robbins [28] simply referred to the single preceding picture element on the same line when resorting to apriori prediction.

5.3. Adaptive Apriori Prediction

The adaptive scheme attempts to take advantage of the present frame predictor's smoothness qualities, while favoring the previous frame predictor at the critical locations where large propagation errors may be triggered. The following formula is used:

\[
\begin{bmatrix}
  d_x \\
  d_y
\end{bmatrix} = \begin{bmatrix}
  D_x \\
  D_y
\end{bmatrix} - \frac{\lambda}{1+\lambda(\nabla_x^2 + \nabla_y^2)} \begin{bmatrix}
  \nabla_x \\
  \nabla_y
\end{bmatrix} \nabla' \tag{51}
\]

where \( D \) is allowed to vary smoothly between the \( \tilde{d} \) and \( \hat{d} \) predictors. It is incremented according to predetermined steps in the direction of the predictor having the lowest displaced frame difference (i.e. lowest \( \nabla' \)). Figures 12A and 12B were
Figure 12: Adaptive Apriori Prediction of Displacement
produced using five transition levels between \( \bar{d} \) and \( \bar{d} \):

1. \( 0.9 \hat{d} \)
2. \( 0.7 \hat{d} + 0.2 \bar{d} \)
3. \( 0.5 \hat{d} + 0.4 \bar{d} \)
4. \( 0.3 \hat{d} + 0.6 \bar{d} \)
5. \( 0.1 \hat{d} + 0.8 \bar{d} \)

Note that for each step, the sum of the two weighting coefficients is 0.9 rather than 1. Thus there is always a tendency to avoid large displacement vector estimates which could otherwise lead to instability as evidenced in Figure 11.

Note that the displacement vectors at the lower right corner of the square (Figure 12A) are much more reasonable than the corresponding vectors of Figure 11A. Also note that the pronounced edges found in the same region using the previous frame predictor (Figure 10A) no longer exist. This has a very significant effect on the performance of the segmentation scheme that is presented in the next chapter.
CHAPTER 4

Image Segmentation and Interpretation

After estimating the displacement vectors, additional nontrivial processing is necessary to extract important motion information. Specifically, the moving objects must be identified, and the time-varying properties of each must be physically characterized. These two processes are defined here as segmentation and interpretation respectively. The segmenter merely attempts to classify an image into subregions that can be individually characterized by a single displacement vector, rotation angle, and zooming (or magnification) value. These three parameters are judged sufficient to model the movements of all objects. The interpreter assigns empirical values for these three parameters and checks for luminance fluctuations.

1. Image Segmentation

All segmentation decisions are based on the information available from three different sources:

1. the image itself
2. the magnitude of the displacement vectors
3. the displaced frame differences

An initial estimate of boundary locations is obtained by locating the edges in the displacement vector magnitude image. In a similar manner, the displaced frame difference image is scanned for additional boundaries as evidenced by prominent thin lines indicating either positive or negative frame difference errors. This feature is prevalent only at object boundaries.

The resulting boundary map is examined for closed contours. Contours that cannot be closed due to large gaps, are dismissed as noise. Those that remain are compared to the actual image. If a single edge is found in the vicinity of a contour,
then the contour location is adjusted to coincide with the edge. The moving objects are then easily identified as the region enclosed by any one contour. In some instances, however, the moving area is the region outside all the closed contours (this is common when the camera is tracking an object in motion). It is the interpreter's task to determine whether the moving region is the contour's interior, exterior, or both.

2. Image Interpretation

At this point we have an image that has been separated into a finite number of components and we now wish to learn what changes are occurring as we progress from the present key frame to the following key frame. If, for each image component, we generate a translation, rotation, and zooming value, then we will have sufficient information to deal with this question.

2.1. Translation, Rotation, and Zooming

The task of assigning a single translation vector to a well defined subregion of an image is trivial. Let \((\hat{d}_x, \hat{d}_y)\) be the translation estimate obtained by averaging \((d_x, d_y)\) throughout the entire region. Then for each picture element, we can define \((\hat{d}_x, \hat{d}_y)\) such that:

\[
(\hat{d}_x, \hat{d}_y) = (d_x - \hat{d}_x, d_y - \hat{d}_y)
\]  

Since it is obtained by subtracting out the translation component, it can be assumed to model only rotation and zooming.

We can now determine the effective center of the region. This location is defined as the source of both rotation and zooming and is identified by finding the picture element with the smallest \((\hat{d}_x, \hat{d}_y)\) magnitude. Recall that smoothness of the \((d_x, d_y)\) vector field was an important factor influencing image segmentation, and
thus it is reasonable to expect the corresponding \((\hat{d}_x, \hat{d}_y)\) vector field to be smooth within regional boundaries. Therefore, if the same minimum is found at \textit{two} or more \textit{distinct} locations, then there may be no point in proceeding. It is sufficient to conclude that either the component is not rotating or changing in size, or else the motion cannot be satisfactorily modeled by these parameters.

Assume that a component is in fact rotating and/or changing in size and that we have found the effective center and denoted it as \((\bar{x}, \bar{y})\). Then consider the picture element at \((x, y)\), located somewhere in the same region and define:

\[
(r_x, r_y) = (x - \bar{x}, y - \bar{y})
\]  

(53)

Thus \((r_x, r_y)\) can be interpreted as a radial vector pointing outwards from the center of the object (Figure 13). The projection of \((\hat{d}_x, \hat{d}_y)\) in the direction of \((r_x, r_y)\) is:

\[
\hat{d}_n = \frac{\hat{d}_x \cdot (r_x, r_y)}{|(r_x, r_y)|}
\]  

(54)

A similar projection in the direction of rotation is:

\[
\hat{d}_\theta = \frac{\hat{d}_x \cdot (r_y, -r_x)}{|(r_x, r_y)|}
\]  

(55)

Note that \(\hat{d}_n\) and \(\hat{d}_\theta\) are scalar quantities which may be either positive or negative.

Since the effective center should be unaffected by rotation and zooming, it can be assumed that the new location of \((x, y)\) after experiencing rotation and zooming, is:

\[
(\bar{x} + r_x + \hat{d}_x, \bar{y} + r_y + \hat{d}_y)
\]  

(56)
Figure 13: Computation of Motion Parameters for Object Rotating Clockwise and Increasing in Size
Then, as shown in Figure 13, the angle of rotation with respect to \((\bar{x}, \bar{y})\) is:

\[
\theta = \tan^{-1}\left( \frac{\check{d}_\theta}{|\check{r}_x, \check{r}_y| + \check{d}_n} \right)
\]

(57)

and the magnification factor is:

\[
\hat{K} = \frac{|(\check{r}_x + \check{d}_x, \check{r}_y + \check{d}_y)|}{|(\check{r}_x, \check{r}_y)|}
\]

(58)

If we restrict ourselves to small rotations only, then (57) simplifies to:

\[
\theta \approx \frac{\check{d}_\theta}{|\check{r}_x, \check{r}_y| + \check{d}_n}
\]

(59)

and (58) can be approximated by:

\[
\hat{K} \approx \frac{|(\check{r}_x, \check{r}_y)| + \check{d}_n}{|(\check{r}_x, \check{r}_y)|}
\]

(60)

Quite accurate rotation and zoom estimates can be achieved by averaging \(\theta\) and \(\hat{K}\) over an entire region, particularly if it is large in area or image resolution is high. A weighted average, using weighting coefficients that are proportional to the length of \((\check{r}_x, \check{r}_y)\), would be even more effective. It is also possible to evaluate the reliability of an estimate by comparing the mean with the standard deviation. A large standard deviation, relative to the mean, indicates poor modeling of motion by the predicted parameter. In such cases, it may be wise to discard the estimate and assume a value of zero for interpolation.

In many instances, an object will experience a transformation in shape so that motion cannot be modeled merely by translation, rotation, and zooming. Even in
such circumstances, the above equations can still be expected to approximate the true displacement vector, rotation angle, and zooming factor. In fact when errors do occur, they usually can be attributed to inaccurate displacement vectors \((d_x, d_y)\) and not to deficiencies in the interpretation equations.

Finally, the average change in luminance as an object moves from one location to the next is observed. That is, the quantity:

\[
I(x + d_x, y + d_y, t + 1) - I(x, y, t)
\]  

(61)

is averaged over all picture elements of a moving area. The statistic not only aids interpolation (Chapter 5) but also serves as an apriori estimate of the luminance change when estimating the next set of displacement vectors.

2.2. Extracting the Nonmoving Component

It is not sufficient to simply quantify the movements of each image component. In most instances, we will also need to know what is behind the moving regions. The underlying reasons should be apparent. When an object moves, it usually uncovers some background. It is, of course, impossible to interpolate correctly without knowing what is uncovered.

The solution is simple. First classify the different components as either moving or nonmoving by applying a suitable threshold to the previously computed translation, rotation, and zooming estimates. A complete nonmoving image component is then constructed by forming the union set of the nonmoving components and the components which are believed to exist behind the moving regions. The latter image components are all extracted from the following key frame. In other words, to determine what is behind a moving component, we simply refer to the corresponding region in the following key frame. Of course, these regions may also be moving, but in almost all cases, this does not matter. Since the interpolated or "inbetween"
images are constructed by superimposing the moving components on top of the nonmoving component, *it will not matter if the nonmoving component is inaccurate, provided the erroneous region remains covered at all times.*

When a portion of one image is substituted into the corresponding region of a second image, disturbing boundary discontinuities may arise. The extent of these artifacts depends on the validity of the substitution, which, in most cases, is quite reasonable. However, if a significant segmentation error occurs, traces of a moving component may appear in the non-moving image. It would then appear that a portion of the moving object was left behind after movement. To reduce the visibility of this phenomenon, the second key frame is allowed to blend into the portion of the nonmoving image component that is just outside the object boundaries (as determined by the segmenter). This is illustrated in Figure 14.

3. Test Sequence 1

Test Sequence 1 consists of a square that becomes deformed along its edges as it translates, rotates, and increases in size (Figure 15). Displacement vectors were computed using the procedure developed in Chapter 4. In order to produce useful results, the previous-frame predictor of the motion prediction algorithm had to be artificially set to at least 45% of the correct value. In other words, motion compensation is unsuccessful when the motion parameters estimated during the previous frame are less than 45% of the correct parameter values associated with the present frame. Therefore since we are dealing with only two frames, we will simply assume that the square had been observed moving during the previous frame interval, resulting in previous frame predictors that are 50% of the ideal parameters corresponding to the present frame. Note that this assumption allows us to resolve the ambiguity concerning the direction of rotation of the object. We may now assume that this direction will be the same as it was in the previous frame.
Figure 15: Test Sequence 1
The magnitude of the estimated displacement vectors is shown in Figure 16A and the corresponding displaced frame difference image is shown in Figure 16B. This information supplements the information already presented by the first key frame (Figure 15A), and allows the segmenter to easily extract the moving square from the nonmoving background. Figure 17A shows the moving image component and Figure 17B reveals the black background as the nonmoving component.

For this example, the challenge is not in segmentation, but in the modeling of motion. Notice that the intensity of the square in Figure 16A is not uniform. Since it is both expanding and rotating clockwise, the displacement at the lower right corner must exceed that of the upper left corner. It is this variation in the magnitude and direction of the displacement vectors which allows estimation of rotation and magnification parameters. The estimated and correct motion parameters are displayed in Table 1. Note the improvement achieved by performing a second iteration with the displacement vector equation.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Actual Value</th>
<th>Predicted Value</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>First Iteration</td>
<td>Second Iteration</td>
<td></td>
</tr>
<tr>
<td>Translation (x pels, y pels)</td>
<td>(64, 64)</td>
<td>(51, 50)</td>
<td>(64, 63)</td>
<td></td>
</tr>
<tr>
<td>Rotation Angle (deg.)</td>
<td>45</td>
<td>31</td>
<td>43</td>
<td></td>
</tr>
<tr>
<td>Magnification Factor</td>
<td>1.5</td>
<td>1.31</td>
<td>1.52</td>
<td></td>
</tr>
</tbody>
</table>

4. Test Sequence 2

Consider a bouncing ball sequence (Figure 18) which consists of six key frames. Displacement vectors are computed for each frame interval using the adaptive a priori prediction scheme. Consider the third key frame, occurring just before the ball strikes the ground. Displacement vector magnitudes and displaced frame differences are shown in Figures 19A and 19B respectively. These images are used to estimate the approximate boundaries of the moving object. When this estimate is
Figure 16: Displacement Magnitude and Displacement Error for Test Sequence 1
Figure 17: Moving and Nonmoving Components of Test Sequence 1
Figure 18: Test Sequence 2
Figure 19A: Displacement Magnitude

Figure 19B: Displacement Error
refined by observing the key frame itself, it becomes flawless. The resulting moving and nonmoving components are shown in Figures 20A and 20B respectively.

Motion parameters for the moving component are displayed in Table 2. Notice that the translation vector is very large in relation to the size of the image (the image has been subsampled down to 128 by 128 pels). Nevertheless, quite accurate results were obtained, and this is, to a large extent, attributable to apriori estimation of motion. Unfortunately, when we apply the same procedure to the next key frame, we are not so fortunate. The sudden impact and direction reversal as the ball bounces was, of course, unanticipated and the algorithm was thus unable to track the ball's movement from the forth to the fifth key frame. The effect of this error on interframe interpolation is discussed in the next chapter. At present, it is sufficient to state that the error may not have occurred if more key frames were available. After all, if we estimate the height of the building from which the ball is dropped, we can conclude that we are dealing with fewer than two frames per second! This can be contrasted with the 60 Hz frame rate of conventional television.

<table>
<thead>
<tr>
<th>TABLE 2</th>
<th>Motion Parameters for Fourth Frame of Test Sequence 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parameter</td>
<td>Actual Value</td>
</tr>
<tr>
<td>Translation (x pels, y pels)</td>
<td>(6.5, 35.2)</td>
</tr>
<tr>
<td>Rotation Angle (deg.)</td>
<td>0</td>
</tr>
<tr>
<td>Magnification Factor</td>
<td>1.05</td>
</tr>
</tbody>
</table>

5. Test Sequence 3

This sequence depicts a walking mechanical toy called a "zoid" (Figure 21), filmed with a television camera operating at a low frame rate. Unlike the previous sequences, the somewhat complex manner of movement makes it more difficult to segment. Again the displacement vectors are essential to carrying out this task.
Figure 20A: Moving Component

Figure 20B: Nonmoving Component
Figure 22A shows the magnitude of these vectors when computed for the third key frame of the sequence (Figure 21C), and Figure 22B shows the corresponding displaced frame differences. The moving components remaining after segmentation are displayed in Figure 23A. The target has actually been subdivided into 6 sub-components, each distinguishable by a different grey level in this image. Actually the zoid consists of fewer than 6 moving pieces but this is not apparent from the photographed images. Figure 23B is the nonmoving component obtained by combining selected parts of both the present and following key frames. Although pieces of the zoid are missing in this figure, much of it still remains. This is because the image was derived from only two key frames, and thus there was insufficient information to fully determine what was hidden behind. It remains to be seen whether this will affect the interpolation process to be discussed in the next Chapter. Recall that in most cases, the background components that cannot be determined are not needed for interpolation.

Since this sequence was not artificially generated, there is no easy means for evaluating the accuracy of the computed motion parameters. Perhaps the best solution, is to delay this task until after interpolation. The interpolated or "inbetween" images can then be compared to the key frames.
Figure 22A: Displacement Magnitude

Figure 22B: Displacement Error
Figure 23A: Moving Components

Figure 23B: Nonmoving Component
CHAPTER 5

Interpolation

Motion prediction and image segmentation both involve an element of uncertainty. Both operations are subject to errors which can occasionally be quite serious. One such example arose in the previous chapter when the resulting bounce after a ball struck the ground was not anticipated. Interpolation, on the other hand, is a systematic and straightforward operation. Although motion prediction and segmentation errors may be made more conspicuous, there is no danger of introducing additional errors, excluding those which are attributable to approximation and rounding.

1. Method

Any two key frames of an image sequence can be interpolated in two different ways. The first approach is to consider, in turn, each picture element of the inbetween frame, and determine the corresponding picture element in the first key frame. In order to locate this corresponding pel, it is necessary to observe the displacement vector of every picture element in the key frame. Implementation of such an algorithm is at best time-consuming and inefficient.

The alternative approach is to examine the first key frame to determine the new location of each picture element so that the inbetween frame can be constructed. For instance, consider an object in the first key frame with an effective center specified by \((\bar{x}, \bar{y})\). If \((\dot{d}_x, \dot{d}_y)\) represents the translation of the object relevant to the two key frames, then the new location of the object center in the inbetween frame, \((\bar{x}_i, \bar{y}_i)\), can be computed using

\[
\begin{bmatrix}
\bar{x}_i \\
\bar{y}_i
\end{bmatrix} = \begin{bmatrix}
\bar{x} \\
\bar{y}
\end{bmatrix} + f \begin{bmatrix}
\dot{d}_x \\
\dot{d}_y
\end{bmatrix}
\]  

(62)
where \( f \) is a constant ranging from 0 to 1 depending on the time occurrence of the inbetween frame. If \( f \) is 0.5 then the inbetween frame will be spaced in time halfway between the two key frames while values of 0 and 1 correspond to the time of the first and second key frames respectively.

The new location of the object center must be known before considering the effects of rotation and zooming. In the latter case, \( K \) will represent the magnification ratio and will be less than 1 if an object is decreasing in size, and greater than one if it is becoming larger. Then the following relationship describes the mapping of \((x, y)\), in the key frame, to \((x_i, y_i)\) in the inbetween frame:

\[
\begin{bmatrix}
  x_i \\
  y_i
\end{bmatrix}
= \begin{bmatrix}
  \bar{x}_i \\
  \bar{y}_i
\end{bmatrix} + \begin{bmatrix}
  x - \bar{x} \\
  y - \bar{y}
\end{bmatrix} \left[ 1 + f (K - 1) \right]
\]

If the object rotates by an angle \( \theta \), then a simple rotation matrix suffices to yield the final result:

\[
\begin{bmatrix}
  x_i \\
  y_i
\end{bmatrix}
= \begin{bmatrix}
  \bar{x}_i \\
  \bar{y}_i
\end{bmatrix} + \begin{bmatrix}
  \cos(f \theta) & \sin(f \theta) \\
  -\sin(f \theta) & \cos(f \theta)
\end{bmatrix} \begin{bmatrix}
  x - \bar{x} \\
  y - \bar{y}
\end{bmatrix} \left[ 1 + f (K - 1) \right]
\]

Eq.(64) was derived for forward interpolation of the first key frame. A similar equation can be derived for backward interpolation of the second frame but before this can be done, the second image must be segmented. It would be unwise to assume that the boundaries are identical to those of the first key frame after allowing for mere translation, rotation, and zooming. In other words, a corresponding object in the second frame will be assigned the same motion parameters, but the boundaries may differ slightly. In this way, the inbetween frame is generated using both forward and backward interpolation, and the two resulting images are averaged. The weighting coefficients are \(1 - f\) and \(f\) for the forward and backward
interpolated images respectively. This insures that the transition from one key frame to the next will be smooth and free of discontinuities.

Interpolation is performed by using Eq.(64) and the corresponding backward interpolation equation to determine the new location and orientation of each object. These objects are then superimposed on the nonmoving image component generated using the procedure of Section 2.2 of Chapter 4. However, the algorithm is not perfect. In many instances, more than one pel in the key frame will map to the same pel in the inbetween frame while some pels in this frame will emerge with no corresponding point in the key frame. There are two reasons for this phenomenon: 1. If an object is changing in size, then a one-to-one mapping function cannot be achieved, 2. translation, rotation, and zooming parameters are floating-point quantities and thus integer truncation is ultimately required when implementing the mapping function with Eq.(64). Again, this may violate the one-to-one mapping constraint, especially if the object is rotating or changing in size. A simple scheme has been implemented to insure that the interpolated object does not emerge with missing pels. If the object is increasing in size, then each pel is mapped to more than one pel (depending on the value of $K$) in the interpolated image. If the object is becoming smaller than some points in the key frame may be discarded before mapping. Of course averaging is the preferred operation but this requires more computations. In all other cases, the mapping is from 1 to 4 pels. Although this latter measure counteracts the effect of integer truncation, it would appear to reduce the resolution of moving objects by a factor of 2. Actually it is only the object edges which are affected due to the method of implementation. When a small object is interpolated to become a larger one, all the object detail is preserved but, nevertheless, the edges may become jagged. On the other hand, more information can usually be obtained by interpolating the movement of the corresponding larger object in the other key frame. It is then very easy to adjust the weighting constant in order to emphasize the estimate derived from this larger
object.

Good edge resolution is a desirable quality. However, if we realize that the eye is often incapable of discerning detail along the edges of a rapidly moving object, then the interpolated images become more acceptable. In fact, a filtering operation has been implemented in order to blend the edges of a moving object into the nonmoving background. The amount of filtering is directly related to the velocity of motion so that small segmentation and motion estimation errors (often present when the velocity is high) are made less visible. Of course, a velocity vector is directional, and it is therefore desirable to perform maximum filtering in the direction of the translation vector. More sophisticated mapping techniques based on spatial interpolation are certainly conceivable, but efficient implementation would require a little more thought.

2. Test Sequence 1

We now consider the interpolation of the square as it undergoes translation, rotation, and magnification. Figures 24A and 24B are the first and second key frames respectively. Figures 24C and 24D are the two inbetween frames generated using Eq.(64) with \( f \) equal to \( \frac{1}{3} \) and \( \frac{2}{3} \) respectively. Note the blurring along the edges which not only masks the reduced resolution, but also smooths the evolution of the straight edges in the first frame to the jagged edges in the second.

3. Test Sequence 2

Since the background of the bouncing ball sequence does not change, the moving components of the inbetween and key frames are conveniently displayed as a single image (Figure 25). To minimize confusion, only one interpolated version is shown between key frames. The two are easily differentiated due to the edge filtering present in the inbetween frames. In practice, the key frames would also tend to
Figure 24: Interpolation of Test Sequence 1
Figure 25: Interpolation of Test Sequence 2
be blurred depending on the properties of the camera. If this were not the case, as in animated sequences, it would be necessary to blur the image artificially. For these reasons, it would be necessary to blur all the images and add more inbetween frames before the bouncing ball sequence could be displayed satisfactorily.

Notice that the first and last inbetween frames (Figure 25) were obtained by interpolating between a full circle and a half circle. Unfortunately the forward and backward interpolations yield very different results as can be deduced from the averaged images in this figure. Notice, however, that the problem could have been prevented by using only the backward interpolation when the object enters the screen, and the forward interpolation when it exits. The required modifications are minor but it is not clear if the situation would be improved. Since the half circle bears little resemblance to a full circle, the estimated motion parameters tend to be rather inaccurate and if we were to interpolate in only one direction, a sharp discontinuity would arise as we near the unused key frame.

Recall that motion parameters could not be estimated successfully immediately after the ball struck the ground. Consequently, the ball was not reproduced at all between the fourth and fifth key frames (Figure 25).

Figure 26 shows the ideal sequence desired after interpolation (without filtering). If the locations of the circles in this image are compared to those of Figure 25, one may deduce that simple linear interpolation is not sufficient. An infinite number of higher order interpolation functions could be conceived but perhaps it is sufficient to state that a potential for improvement does exist and no implementation difficulties are foreseen.

4. Test Sequence 3

Interpolation of the zoid sequence is perhaps the most interesting since several components are involved. The two key frames are depicted in Figures 27A and 27B respectively. Figures 27C and 27D are the inbetween frames constructed using
Figure 26: Ideal Interpolation of Test Sequence 2
Figure 27: Interpolation of Test Sequence 3
Eq. (64) with \( f \) equal to \( \frac{1}{3} \) and \( \frac{2}{3} \) respectively. Edge rendition is perhaps objectionable, particularly where a component experiences rotation, but again this is due to the interpolation method, and to some extent, the segmentation process where some computations are performed on low resolution images. In other respects, however, the reproduction is quite good and all components fit together well. Also note that the erroneous regions of the nonmoving image component (Figure 23B) have been satisfactorily covered in Figures 27C and 27D.

It is interesting to extrapolate Figure 27A in an attempt to predict in advance the appearance of Figure 27B. This is easily done by setting \( f \) equal to 1 and performing a forward interpolation. Figures 28A and 28B are the ideal and extrapolated images respectively. All differences can be attributed to various imperfections in either the motion estimation, segmentation, or interpolation processes. On the other hand, a similar statement can be made concerning the similarities between the two figures.
Extrapolation of Test Sequence 3
CHAPTER 6

Conclusions

Consider the motion estimation, segmentation, interpretation, and interpolation processes that have described independently in the preceding chapters. The combined process is an efficient one, for in many cases the features of one sub-process are ideally suited to those of another. For instance, the displacement vector field is smoothed considerably at all locations except those where boundaries are likely to exist. Consequently it does not complicate segmentation. As another example, the full resolution of the displacement field is utilized to predict translation, rotation, and magnification values. Finally, a segmented image allows easy computation of apriori predictors for the next image frame. The alternative of extrapolating the location of each displacement vector would be difficult as a one-to-one mapping function does not exist.

Three examples were used to demonstrate the performance of the interpolation process. The square that was translating, rotating, and growing all at the same time required a reasonable apriori motion estimate before its movement could be captured, but this certainly is not surprising considering the amount of change between the two key frames. With this in mind, the interpolation was very successful and inspired considerable confidence in the procedure.

The bouncing ball experiment would also have been very successful if not for the temporary confusion caused by the sudden bounce and subsequent change of velocity. Again this may not have occurred except for the unusually low frame rate, but nevertheless, the incident draws forth an important observation. It is quite possible that the mistake would not have happened if the location of the ball had not been predicted in advance. Needless to say, it would be unwise to ban apriori prediction altogether as this would have caused the process to fail after only two frames.
On the other hand, the algorithm can be made more intelligent. After all, a human observer certainly would have expected the ball to bounce even after his eyes had adjusted to the downward motion. Obviously the interpolator cannot be given human intelligence, but perhaps a simple observation, such as a deformation in the shape of the ball, could have been used to predict a change in velocity. Of course this would require that the interpolator be provided with at least one key frame showing the ball as it strikes the ground.

Unlike the others, the zoid sequence was not artificially generated. The interpolation demonstrated that several subcomponents could be processed individually yet still appear as a single unit. During movement, almost all object features were preserved without blurring, and except for the rendition of edges, the interpolated images were quite accurate.

A number of modifications were applied to a well established method for displacement field prediction. An adaptive apriori prediction scheme was tested and found to deliver superior results when compared to both previous and present frame prediction. It was successful in assigning the correct displacement vectors to the interior of a moving object lacking detail, without triggering errors which could otherwise propagate past the boundaries and across a background that also lacked detail. Luminance compensation was also introduced in response to the frequent fluctuations in the intensity of image components. A very effective improvement concerns the evaluation of local intensity gradients. A strong effort was made at developing a function that best approximates the average gradient required for reliable solution of the displacement vector equation.

The segmentation process makes use of standard edge detection algorithms to locate object boundaries in both the displacement vector field and image field. Translation, rotation, and magnification values are computed in quite a simple manner and accuracy is again dependent on that of the displacement field.
Both forward and backward interpolation were found necessary to prevent movement discontinuities that could result from interpolating with imperfect data. Moreover, the same discontinuities would have been unavoidable if a moving object experienced some kind of deformation, and was interpolated in only one direction. In such cases where the parameters are faulty or movement is exceedingly complex, the image formed by averaging the forward and backward estimates may not be clearly visible. Certainly the effect is quite different from blurring but then it is difficult to conjecture as to which effect is more objectionable to the observer. If the purpose of temporal interpolation is to improve the rendition of moving images, then it is difficult to judge the degree of success attained without viewing the resulting sequence in real time.

It should be mentioned that one of the reasons for isolating moving objects is so that they may be interpolated without loss of detail. In all but the most complex imagery, this is a very realistic expectation. If each pel is interpolated independently of its neighbors, then the potential exists for severe distortion. On the other hand, the movement of a large object can be deduced after observing the displacement vectors of a large number of pels. For this reason, we can expect noisy or faulty data to have little effect on the end result.

Suppose we have isolated the same moving object in consecutive frames, but we observe that is has been deformed so that its movement cannot be ideally modeled simply by translation, rotation, and zooming. By recognizing the boundaries of the object, it may be feasible to deduce a mapping function for each pel, so that detail can be preserved as the deformation is performed gradually. Of course the displacement vectors may serve as an invaluable aid, and in the end, the corrected vectors could be verified or further improved through their use as apriori estimates prior to a second iteration with the motion equation. Unfortunately, the process may be too complex for immediate implementation in television receivers, but applications may exist in the fields of computer graphics and animation.
For television imagery, a system which isolates simple moving objects, and interpolates them by translation, rotation, and zooming, can deliver excellent performance in many situations. It can often deduce what is behind moving objects, and if necessary, make important decisions when moving objects cross the same path or interact in some other manner. Ultimately, it will fail when the subjects become too complex in either form or movement, and in such cases an alternative procedure is essential. Fortunately it would be a simple matter to interpolate independently all moving pels that escape the segmentation process. Clarity may be sacrificed to some extent, but this may be quite acceptable if the movements are indeed complex.
BIBLIOGRAPHY


