Generating Salient Stills Using Block-Based Motion Estimation

by

Joseph H. Kang

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degrees of

Bachelor of Science in Electrical Engineering

and

Master of Engineering in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 1995

Author .....
Department of Electrical Engineering and Computer Science
May 16, 1995

Certified by .....
Walter Bender
Assoc. Director for Information Technology, MIT Media Laboratory
Thesis Supervisor

Accepted by .....
F. R. Morgenthaler
Chairman, Departmental Committee on Graduate Theses

Generating Salient Stills Using Block-Based Motion Estimation

by

Joseph H. Kang

Submitted to the Department of Electrical Engineering and Computer Science
on May 16, 1995, in partial fulfillment of the
requirements for the degrees of
Bachelor of Science in Electrical Engineering
and
Master of Engineering in Electrical Engineering and Computer Science

Abstract

This thesis project takes an in-depth look at a new class of images, called salient stills. Photography represents a discrete moment in time, while video is a temporal medium. Salient stills close the space between the two mediums by reflecting in one image, the aggregate motion of an image sequence, keeping only the salient features. The application of the affine transformation and non-linear temporal processing to a sequence of images can lead to a still image of multi-resolution patches, a larger field of view, and higher overall resolution than any of the frames in the original sequence. Previous work by Teodosio and Bender used optical flow as their method of motion estimation. This thesis looks at applying block-based motion estimation from such applications as MPEG1 and MPEG2 to generate salient stills.

Thesis Supervisor: Walter Bender
Title: Assoc. Director for Information Technology, MIT Media Laboratory

This project was supported in part by IBM and NiF.

To my family:
Dad, Mom, Jean, John, and Susan,
whose love and support made this thesis possible.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Salient Stills

The salient still image is a new representation of moving data which merges the space between photography and video sequences. By transforming a sequence of images onto one still frame, the salient still captures the temporal and spatial information of an image sequence, while preserving the original content and context. This paper will look at new methods of motion estimation, a process critical to the performance of the salient still.

An image sequence consists of zooms, tilts, pans, and other camera movement that changes the field-of-view, vantage point, and perceived resolution. By estimating the motion and then applying the affine transform, we can create a salient still which takes advantage of these temporal changes.

An example of the salient still process can be seen in the images below, which were generated by Teodosio [20]. Figure 1-1 shows four frames taken from a 12 second zoom sequence featuring the cellist Yo-Yo Ma during a performance at Tanglewood. The sequence starts with a close-up view of Mr. Ma and zooms out as a musical assistant walks across the stage. The resultant salient still is shown in Figure 1-2.

Note that the salient still shown in Figure 1-2 contains both the temporal and spatial information of the entire sequence. The close-up frames render Mr. Ma with optimal resolution, while the far shot reveals the context of the sequence. In addition,

Figure 1-1: Four Frames Taken from the Yo-Yo Ma Sequence

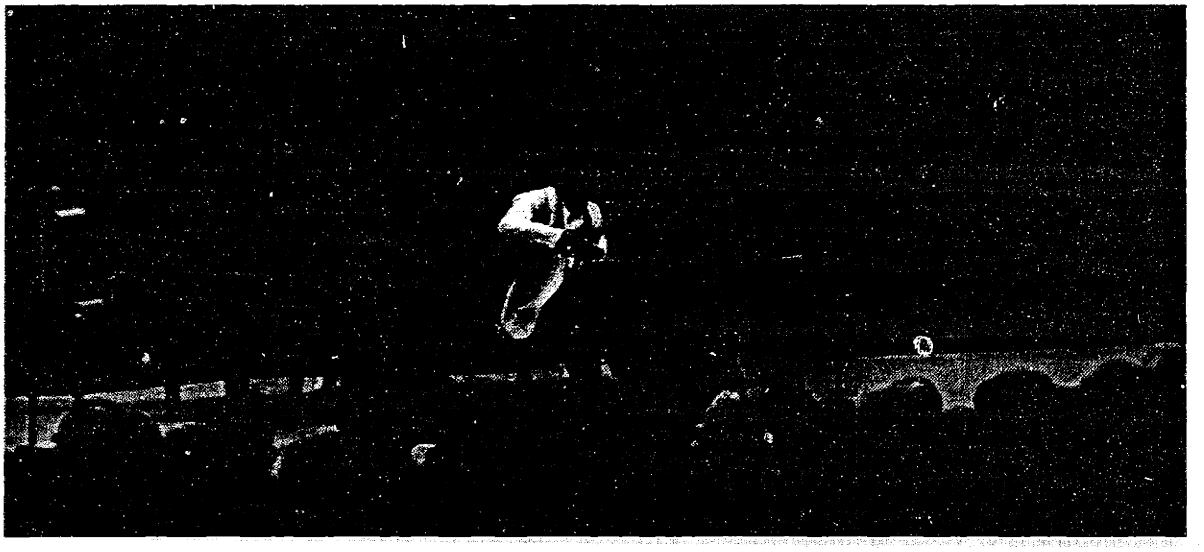


Figure 1-2: The Resultant Salient Still

the middle frames provide enough data redundancy that with appropriately chosen temporal filters, we can either display the motion of the musical assistant as shown in Figure 1-2, or we can remove him altogether. This way, we can keep the audience's attention on the more salient Yo-Yo Ma. This still is shown in Figure 1-3.

Figure 1-3: The Salient Still Without the "Temporal" Musical Assistant

1.2 Motivation

Salient stills can be used for a wide variety of applications. One use is for enhanced reproduction quality. The resolution of individual video frames is less than that of print medium. The salient still increases resolution for zoom sequences, and is useful for removing transient noise (such as salt and pepper noise) found in most video, because a salient still contains many lumination samples for each pel.

An example of this resolution enhancing procedure can be seen in Figure 1-4. Here, two images of Mr. Ma taken from different frames in the sequence have blown up to better show the resolution. The left image was taken from the leftmost (zoom) image in Figure 1-1, while the right image was taken from the rightmost image. As is clearly evident, the zoomed image contains much more high-freqency components, leading to greatened resolution. Thus to achieve maximal resolution, the final still uses the frame with higher resolution to portray Mr. Ma.

Figure 1-4: Yo-Yo Ma taken from Two Different Frames

Next, we deal with the issue of compression. Representing an entire sequence in approximately the space of a single image means less bits. A related use of the salient still is as a scene icon. Video retrieval from an image database usually requires description, searching, and viewing. The salient still would make such a process quicker.

The Yo-Yo Ma image sequence is a good example of this. The entire image sequence could represent a scene. Hence, because the final still contains the spatial and temporal information of the image sequence, it serves the purpose of compressing the data and being used as a scene icon.

Yet another benefit of salient stills is the ease at which photomontages can be created. The use of specialized cameras would no longer be necessary as standard video equipment could capture many scenic frames. From these, the editor could handpick the desired ones and then create the salient still. Figure 1-5 shows a still generated from a 121 frame sequence of the Kennedy procession. Figure 1-6 shows a still created from a 159 frame sequence of a coastline.

Hence, with so many possible applications, it is important to reap the highest quality still possible. Probably most essential to the quality of the salient still is the method at which the motion is estimated. There are many ways to accomplish this with the most prevalent being optical flow and block-based motion estimation. These motion estimates determine how well the sequence of frames are aligned; thus,

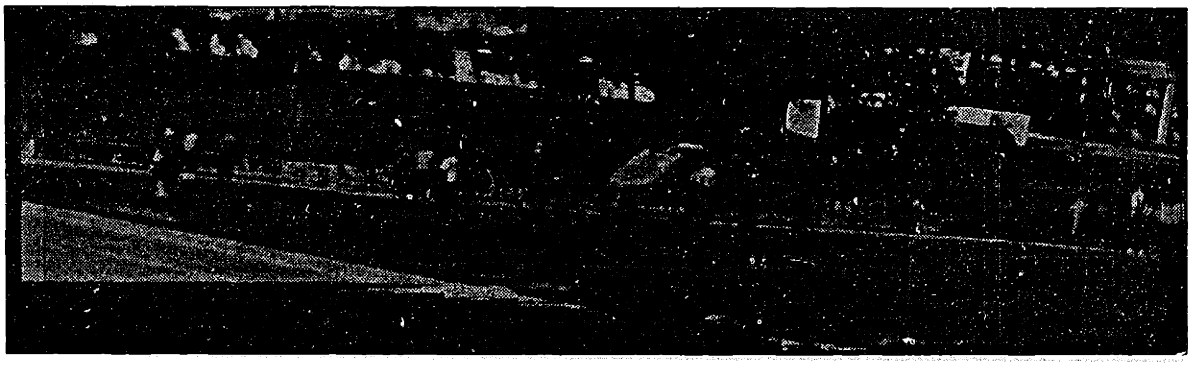


Figure 1-5: Salient Still from the Kennedy Procession

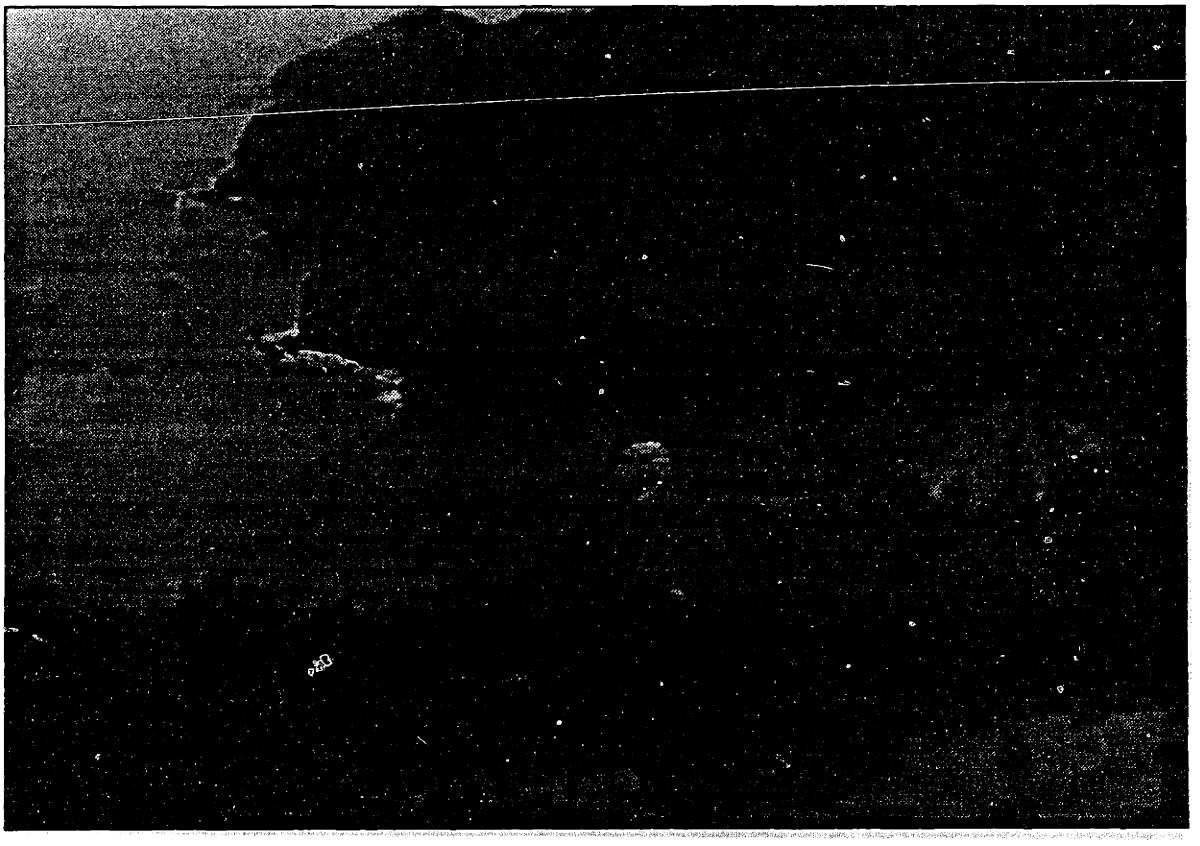


Figure 1-6: Salient Still from a Coastline Sequence

they are directly linked to the quality of the final salient still. Especially with the emergence of block-based video coding standards such as MPEG, the importance of exploring block-based estimators and salient stills can lead to even more applications, while adding minimal additional computation.

Teodosio and Bender's original method [21] used optical flow as its method of estimating motion. In an effort to explore other forms of motion estimation and how they affect the resultant salient still, this thesis will look at how well block-based motion estimation can generate salient still images.

1.3 Thesis Work

Work focused on two areas: 1) implementing and improving block-based estimation algorithms to generate salient stills and 2) expanding the application to MPEG

In the first part of my research, salient stills were generated directly from MPEG bitstreams. The basic process is outlined in Figure 1-7.

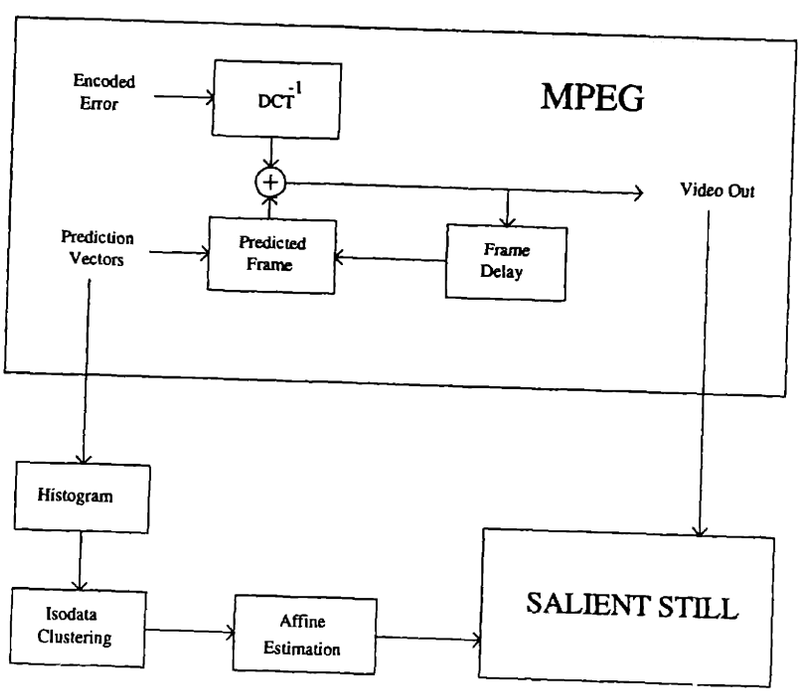


Figure 1-7: Block Diagram: Generating Salient Stills from MPEG Bitstreams

Figure 1-7 gives an overview of the process, but is an oversimplification in that it assumes too narrow of a temporal window. MPEG is a layered coder which does not always send motion estimation information between frames. In most cases, this occurs once for every Group of Pictures (GOP). Salient stills are about still scenes, not still GOPs, so a mechanism is needed to interlink the GOPs that make up a scene.

The second focus involved building a block-based motion estimator and adding improvements to its estimation process. In particular, I used a hierarchical structure to reduce computational complexity and isodata clustering to extract foreground motion. The purpose of segmenting foreground objects is to improve the affine fit of the background.

This thesis is organized as follows:

Chapter 2 provides background information on the original salient stills by Teodosio and Bender, as well as a brief summary of issues concerning MPEG and block-based estimation algorithms.

Chapter 3 discusses the methodology used in the research.

Chapter 4 shows some resultant salient stills that were generated.

Chapter 5 evaluates the relative performance of the work done in this project with the original salient still method. Finally,

Chapter 6 summarizes the work done and proposes work for the future.

Chapter 2

Background

2.1 Salient Still Images by Teodosio and Bender

The basic premise behind the salient still is that one can build a single image which contains the spatial and temporal information of an entire video sequence, while keeping only the salient features.

The original work done by Teodosio and Bender [20] [21] used the following methodology to build these higher resolutional images:

1. Optical flow is calculated between successive frame pairs.
2. Affine coefficients are calculated from the estimated motion parameters.
3. These coefficients are used to translate, scale, and rotate each successive frame pair onto a single high-resolution raster.
4. An assortment of temporal filters (i.e. the weighted median filter) is applied to the image data, resulting in the final image, the salient still.

As a gradient-based method of motion estimation, optical flow is modeled by a continuous variation of image intensity which is a function of position and time:

$$I(x, y, t) = I(x + dx, y + dy, t + dt) \tag{2.1}$$

If the motion field is continuous, we can expand $I(x+dx, y+dy, t+dt)$ using a Taylor series ignoring higher order terms.

$$\frac{dI}{dx}\frac{dx}{dt} + \frac{dI}{dy}\frac{dy}{dt} = -\frac{dI}{dt} \tag{2.2}$$

Note that this model can be solved by assuming that the motion is translational in the x and y directions, but does not model zoom. The affine transformation brings zoom into the model, using an implementation by Bergen [4]. Assuming motion occurs with directional velocities, $p_x(x, y)$ and $p_y(x, y)$, we can derive the affine model to be as follows:

$$p_x(x, y) = a_x + b_x x + c_x y \tag{2.3}$$

$$p_y(x, y) = a_y + b_y x + c_y y \tag{2.4}$$

In this model, a_x and a_y are motion translations; b_x and c_y are scaling factors; and c_x and b_y are rotational factors. Using the results of our motion estimator, we try to find the best affine coefficients to fit this model. Hence, generalizing the equation above, we solve for the parameters by using the least square method and then differentiating the results to find the minimal error.

$$p_x = a_x + b_x x + c_x y \tag{2.5}$$

$$p_y = a_y + b_y x + c_y y \tag{2.6}$$

$$LSE_x = \sum_R (p_x - (a_x + b_x x + c_x y))^2 \tag{2.7}$$

$$LSE_y = \sum_R (p_y - (a_y + b_y x + c_y y))^2 \tag{2.8}$$

Differentiating the error terms by the respective parameters, a, b, and c, we can solve the three linear equations by setting the equation to zero. Here, we solve for a_x, b_x, and c_x.

$$\frac{d}{da_x}LSE = a_x \sum_R 1 + b_x \sum_R x + c_x \sum_R y - \sum_R p_x = 0 \tag{2.9}$$

$$\frac{d}{db_x}LSE = a_x \sum_R x + b_x \sum_R x^2 + c_x \sum_R xy - \sum_R xp_x = 0 \tag{2.10}$$

$$\frac{d}{dc_x}LSE = a_x \sum_R y + b_x \sum_R xy + c_x \sum_R y^2 - \sum_R yp_x = 0 \tag{2.11}$$

Rewriting in matrix form, we solve for the affine coefficients as follows:

$$\begin{bmatrix} \sum_R 1 & \sum_R x & \sum_R y \\ \sum_R x & \sum_R x^2 & \sum_R xy \\ \sum_R y & \sum_R xy & \sum_R y^2 \end{bmatrix} \begin{bmatrix} a_x \\ b_x \\ c_x \end{bmatrix} = \begin{bmatrix} \sum_R p_x \\ \sum_R xp_x \\ \sum_R yp_x \end{bmatrix} \tag{2.12}$$

$$\begin{bmatrix} a_x \\ b_x \\ c_x \end{bmatrix} = \begin{bmatrix} \sum_R 1 & \sum_R x & \sum_R y \\ \sum_R x & \sum_R x^2 & \sum_R xy \\ \sum_R y & \sum_R xy & \sum_R y^2 \end{bmatrix}^{-1} \begin{bmatrix} \sum_R p_x \\ \sum_R xp_x \\ \sum_R yp_x \end{bmatrix} \tag{2.13}$$

Once the matrix equations for the six affine coefficients have been solved, each successive image pair can be warped into a continuous space/time raster from which various temporal filters can be applied. In general, we use the *median* filter to preserve the salient features and exhibit the temporal motion of the original image sequence.

2.2 Block Based Motion Estimation

Motion estimation analyzes the movement of objects in an image sequence and then calculates interframe displacement vectors. To estimate these vectors, several techniques have been proposed. Among these, the major approaches have been gradient based methods [4] [8] [9] [10] and block matching algorithms [7] [11] [18] [19].

In general, the gradient methods are analytically tractable because they make use of iterative solutions. However, because they require the use of derivatives, they are generally limited to short range motion. In practice, derivatives are usually implemented using differences, which not only introduce errors, but amplify high frequency

components. As a result, they are very sensitive to noise. Other problems with optical flow occur when some basic assumptions of optical flow are broken.

One assumption is that the overall lumination of a scene is constant. This, of course, can cause problems when there are changes in the lighting. A second assumption is that the luminance surface is smooth. This assumption fails in areas near motion boundaries.

Unfortunately, there are also many drawbacks to block or token matching. But first, let's discuss the basic methodology. Block-based estimation techniques have been frequently used because they are theoretically straightforward and easy to implement in hardware. In block matching methods, each frame is broken up into small, nonoverlapping blocks. Blocks in the previous frame are matched with corresponding blocks in the current frame via certain criteria such as MAD (mean of absolute differences) and MSD (mean of squared differences). These minimization functions are literally defined as follows:

$$MAD(p_x, p_y) = \frac{\sum_{block_x} \sum_{block_y} |I_{curr}(x, y) - I_{pred}(x + p_x, y + p_y)|}{block_x \times block_y} \tag{2.14}$$

$$MSD(p_x, p_y) = \frac{\sum_{block_x} \sum_{block_y} [I_{curr}(x, y) - I_{pred}(x + p_x, y + p_y)]^2}{block_x \times block_y} \tag{2.15}$$

(p_x,p_y) represents the motion vector which minimizes the function; $I_{curr}(x, y)$ and $I_{pred}(x, y)$ are the original and predicted luminance values.

Note that using the MAD and MSD criteria will generally yield similar results. The difference in theory between the MAD and MSD, however, is that MSD adds an increasing amount of error, the more poorly the blocks match up. As a result, MSD is more sensitive to noise.

A limiting factor in achieving this efficiency is the range at which the motion can be compensated. As the motion of an image sequence is often unknown, the motion estimation program must simply set a search range. Of course, if this search range is too small, the motion estimate will not be accurate, resulting in reduced quality of the reconstructed image and the affine fit of the salient still. Alternatively, if the

search range is too large, there may be a tremendous computational load.

There are other problems which are notable of block-based estimation. One of the key assumptions is that all pixels in each block will undergo the same translation between successive frames. Thus, if there are two or more different kinds of motion within a block (for instance, at the boundary of a moving object), then part of the block will have a wrong motion vector estimate.

A related problem deals with block size. We want the block size to be large enough to get a good correlation with the data. However, the block size must also be small relative to the objects in the image to minimize the possibility of possessing more than one motion per block. The resultant compromise is normally 8x8 or 16x16 sized blocks. To estimate long range motion accurately, 16x16 blocks generally work best because in the larger search space, there is more data to correlate with. This explains why most of my tests were done using 16x16 blocks.

A final problem with block-based estimation occurs when the region of estimation is flat (i.e. all the pels within the region contain similar values). In this case, small amounts of noise will dominate the minimization criterion, causing the blocks to correlate to the noise instead of the actual data.

In short, full search block matching techniques are not only computationally demanding, but also tend to produce noisy motion fields which do not always correspond to true 2-dimensional motion. To ease these problems, hierarchical block matching techniques have been proposed.

Hierarchical or multiresolution motion estimation techniques [3] [5] [6] [7] take advantage of typical scenes which frequently contain motion at different scales. Essentially, a pyramid structure of two successive input images is generated by low pass filtering and subsampling at multiple levels. A sample pyramid is shown in Figure 2-1 for 3 levels.

Starting at the highest level (the pair of images which have been subsampled the most), motion vectors are computed and refined at each level. Thus, this coarse to fine strategy relies on the fact that the first motion vector is accurate. If the motion vectors at the highest level are inaccurate, the motion estimator will not be able to

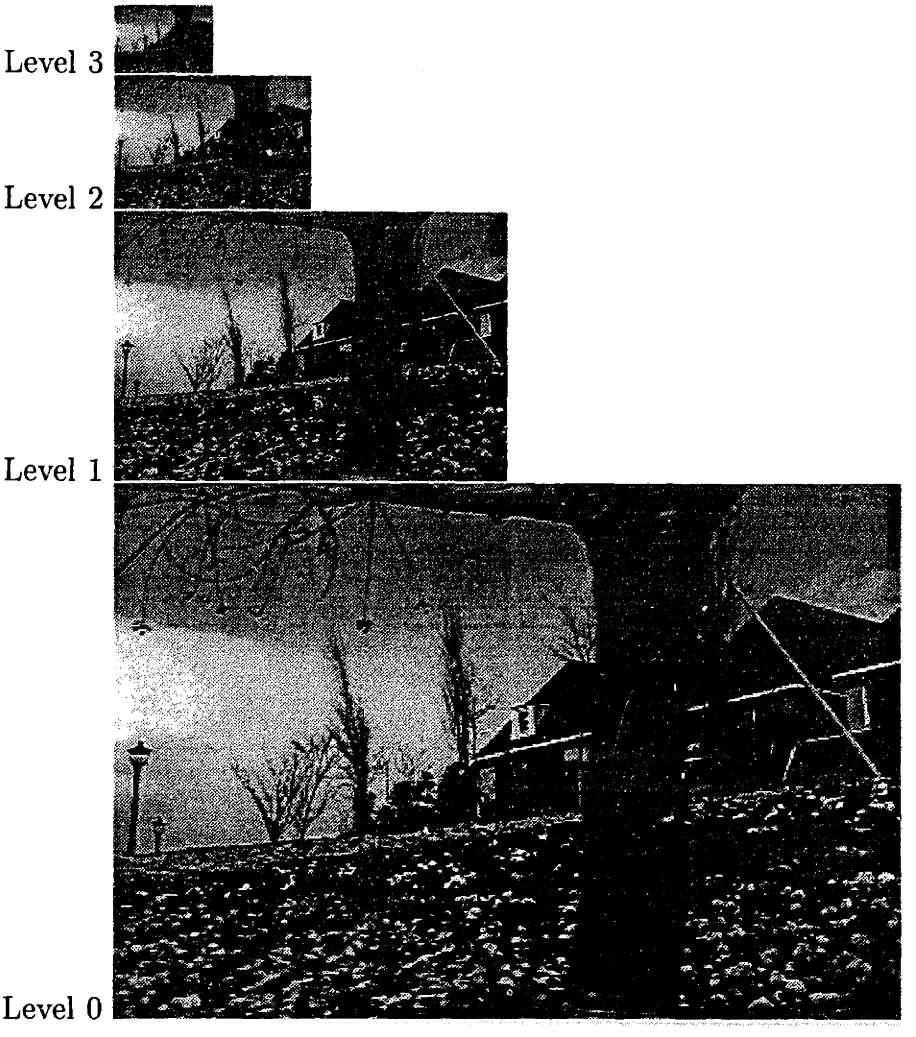


Figure 2-1: Pyramid of Images

converge to the global minima.

Note that along with subsampling the image by a factor of two in the x and y directions, the block sizes are also reduced by a factor of two. Hence, if the motion estimator performs too many levels, the block sizes may be so small that a poor correlation to the data will result.

To summarize, hierarchical block matching techniques give a near optimal solution in minimizing the energy of the prediction error, while easing the computational burden.

2.3 Lossy Video Coders: MPEG1 and MPEG2

MPEG (Motion Pictures Experts Group) [1] [2] is named after the group responsible for standardizing video and associated audio on digital storage media. Whereas MPEG1 has been primarily aimed at storage applications, MPEG2 is being considered for broader applications. There are many differences between MPEG1 and MPEG2, but I'll begin with the similarities.

Both MPEG1 and MPEG2 are layered coders which essentially break the image sequences into smaller sublayers: sequence, group of pictures (GOP), picture (one frame), slice (one row of pixels in a frame), macroblock (16x16 block of pixels), and block (contains the chrominance information). The general encoding process is to use block-based estimation to reduce temporal redundancy, apply the DCT transform to reduce spatial redundancy, quantize the data to achieve a targeted bit rate, and then use run length coding to set the information into a bitstream. A typical diagram of an MPEG encoder and decoder is shown in Figure 2-2 and Figure 2-3.

With regard to motion estimation, MPEG deals with interpolative (noncausal) coding techniques as well as the predictive (causal) techniques. There are three coding picture types: I (intra), P (predictive), and B (bidirectional). The I pictures are coded without any sort of prediction (and hence, holds the greatest quality); P pictures are based on previous I or P frames in the sequence; and B pictures are based on previous and subsequent I and/or P pictures for bidirectional prediction.

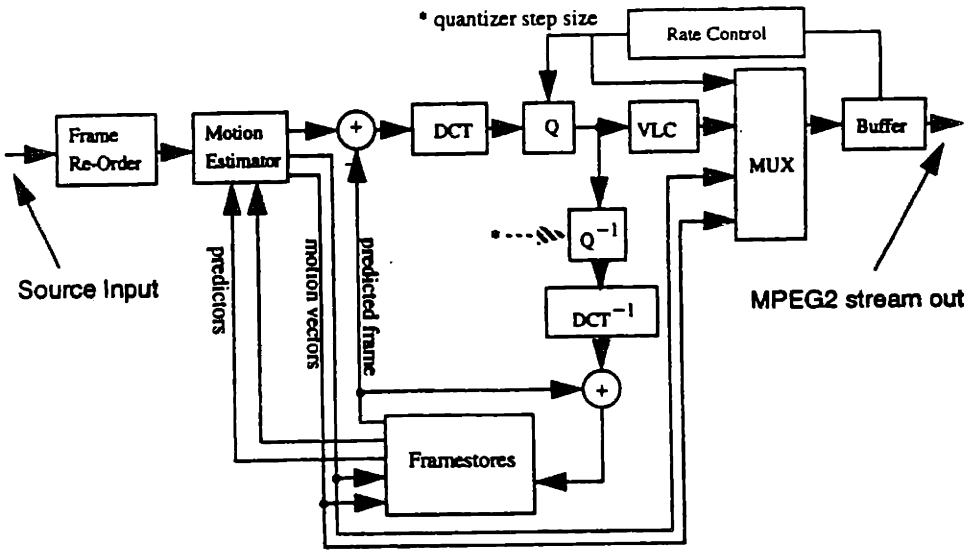


Figure 2-2: MPEG2 Encoder Block Diagram

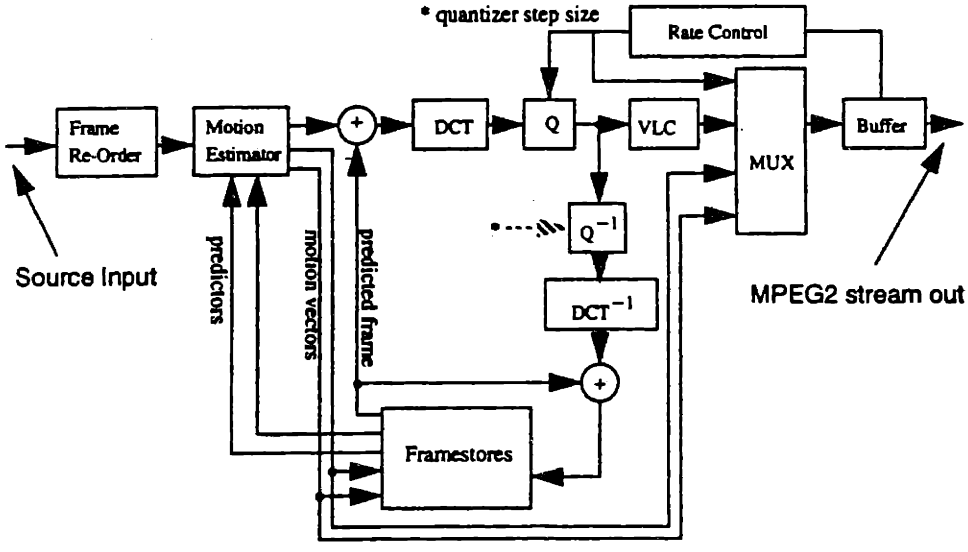


Figure 2-3: MPEG2 Decoder Block Diagram

The most compression is obtained from B pictures, which explains why B pictures also possess the worst quality. Thus, to minimize error propagation throughout the sequence, it makes sense that B pictures are never predicted from previous or future B pictures.

Note that MPEG1 and MPEG2 are both lossy coders. Because they are targeted to be sent at specific bit rates, poorer compression results in quantization levels which are more coarse. The result is data loss and thus, less quality images.

MPEG1 and MPEG2 share the same general approach, but differ primarily in complexity. MPEG2 carries more features than MPEG1, including such items as interlaced video manipulation, scalability, compatibility (with MPEG1), error resilience, and various options for very high resolution video coding . These additional features make the MPEG2 codec more complicated and costly.

Probably the most significant difference between MPEG1 and MPEG2 is the latter's ability to handle interlaced video with either frame or field modes. Whereas MPEG1 can only treat each field or each frame as a separate unit, the MPEG2 coder can exploit the correlation between the two fields in a frame and then select the optimum mode. This is the reason why MPEG2 generally yields higher quality images. It also has an additional type of motion estimation called dual-prime which is essentially field-based prediction that allows the interpolation of two reference fields using one motion vector and a correction vector.

With the basic issues layed out, the rest of this paper will discuss the research I conducted, including the methodology and evaluation of results.

Chapter 3

Methodology

3.1 Applications with MPEG

My first salient still images from block-based motion estimation came from decoding MPEG bitstreams. Calculating the affine coefficients requires decoding the motion vectors and respective addresses for each 16x16 macroblock. Thus, for MPEG1 bitstreams, this process is relatively straightforward.

It is more tricky to decode MPEG2 bitstreams due to its interlaced format. Because field pictures are reduced by a factor of two in the vertical direction, all vertical motion vectors must be scaled by a factor of 2.

Another issue concerns dealing with the bidirectional (B) pictures. With B pictures, there are multiple frame estimates for each frame, which complicates the process. However, with more estimates available, there is more information to achieve a better affine fit. The biggest drawback, though, is that most of the compression is achieved with B pictures, leading to decreased quality of the reconstructed images.

Salient stills are generated from MPEG bitstreams by using the reconstructed picture sequence. Because MPEG coders are lossy, the image will show some coding artifacts, due to both prediction errors and quantization noise. Hence, to optimize resolution in salient still images, I omitted B frames altogether, using only I and P frames. In this manner, the philosophy behind salient stills was upheld.

With this decision made, one last issue needs to be addressed. Because no motion vectors generally exist for I frames (although there are sometimes concealment vectors for MPEG2 sequences), the frequency of I frames would determine the first and last frames used by the salient still. Hence, with the information contained in the MPEG bitstreams alone, it would not be possible to generate a single salient still for the entire sequence. Of course, if I wanted to continuously build a salient still across I frames, all that would be needed is to perform motion estimation on the I frame.

The MPEG decoder that I developed was able to generate salient stills from both MPEG1 and MPEG2 bitstreams. There is an option that allows the user to build one salient still from the entire sequence or to build several stills based on the frequency of I frames. The motion estimator which assigns motion vectors to the I frames performs a full search block-estimation routine, where the user specifies both the search range and the block size.

3.2 Building a Block-Based Motion Estimator

3.2.1 Hierarchical Structure

The command line for the motion estimator I built looks as follows:

motest <input> <pred> <needle> <pred_err> <aff> <yvec> <xvec>
-level (number of levels for hierarchical estimation)
-block (size of correlation block)
-search <minx> <maxx> <miny> <maxy> (search range)
-ssd | -sad (error criteria)

Basically, the motion estimator takes an image sequence as input, as well as a host of data for motion estimation such as block size, search range, error criteria, and the number of levels for hierarchical estimation. The estimator outputs the reconstructed image sequence *(pred)*, a needle diagram displaying the magnitude and direction of the motion vectors *(needle)*, an image displaying the residual prediction errors *(pred-*

err), a file containing the resultant affine parameters *(aff)*, and a file containing the x and y vectors *(xvec)* and *(yvec)*.

For this implementation, the hierarchical structure of the motion estimator gave huge computational savings, while yielding very similar results to those of the full search method.

In general, the number of computations required for a k level search is approximately

$$\sum_{l=0}^{k} \frac{N_l \times M_l}{B_l^2} \times B_l^2 \times (2 \times S_l)^2 \tag{3.1}$$

Where for a given level l, the dimensions of the image is N_l x M_l, the block size is B_l, and the search range is from $-S_l$ to $+S_l$ in both the x and y directions.

The first term, $\frac{N_l * M_l}{B_l^2}$, refers to the number of blocks in the image. Meanwhile, $B_l^2 * (2 * S_l)^2$ refers to the computations involved in the search.

Note that $\frac{N_l * M_l}{B_l^2}$ is constant for each level since all parameters are being subsampled by the same factor. Hence, if we assign the constant c to this value, the equation simplifies to

$$c \times \sum_{l=0}^{k} B_l^2 \times (2 \times S_l)^2 \tag{3.2}$$

Plugging $B_l = 16$ and $S_l = 16$, we get the following number of computations for a given number of levels k.

Level	Computations
k=0	8500 * c
k=1	27216 * c
k=2	92480 * c
k=3	278784 * c

Table 3.1: Computational Load for Different Numbers of Hierarchical Levels

Thus, for each additional level, the computational load is reduced by about three times. However, by reducing the computational load, the quality of the motion vectors might have been reduced.

I ran the hierarchical motion estimator for a few different levels, using alternatively both the MAD and MSD criteria. Using a block size of 16 and a search range from -16 to 16 in both the x and y directions, the errors in Table 3.2 are shown for a six frame sequence (hence, there were five predictions in all). Note that error parameters represent the average error per 16x16 block.

Levels	Error	Frame 1	Frame 2	Frame 3	Frame 4	Frame 5
0	MSD	33860.5	30536.9	30887.0	32233.5	40134.9
1	MSD	34726.0	30588.5	30894.2	32572.9	40373.9
2	MSD	35611.8	32818.9	34694.0	32769.8	41377.3
3	MSD	56725.5	55009.4	61952.3	56838.5	65209.8
0	MAD	1563.0	1443.4	1486.3	1474.8	1760.0
1	MAD	1571.3	1444.5	1488.0	1481.3	1762.8
2	MAD	1577.1	1488.4	1543.0	1500.2	1803.6
3	MAD	1953.3	1889.5	1950.1	1881.6	2141.6

Table 3.2: Hierarchical Levels of Estimation Using Different Error Criteria

From Table 3.2, we can see that levels 0 to 2 have similar error results, but level 3 yields a significantly greater amount of error. Note that for three levels, the estimator would first use a 2x2 block size for estimation. Because this is a small amount of data with which to correlate, the initial estimation is probably not accurate, so the resultant vectors have a noticeably less chance of converging to the global optima. For two levels, the motion estimator would initially use a 4x4 block, which is still a little small, but quite a bit better than a 2x2 block which could match with virtually anything.

To get a better idea of how accurate the motion vectors are, it may be helpful to look at the needle diagrams for a translational sequence. These diagrams are shown in Figure 3-1 and Figure 3-2 for all levels using the MSD criteria.

The needle diagrams for Levels 0, 1, and 2 look very similar, while the one for Level 3 has quite a few spurious motion vectors. Thus, to get accurate results while minimizing computational load, I targeted a minimum block size of 4x4 for all future runs. Hence, if I wanted a final block size of 16x16, I would use 2 levels; for a final block size of 8x8, I would use only 1 level.

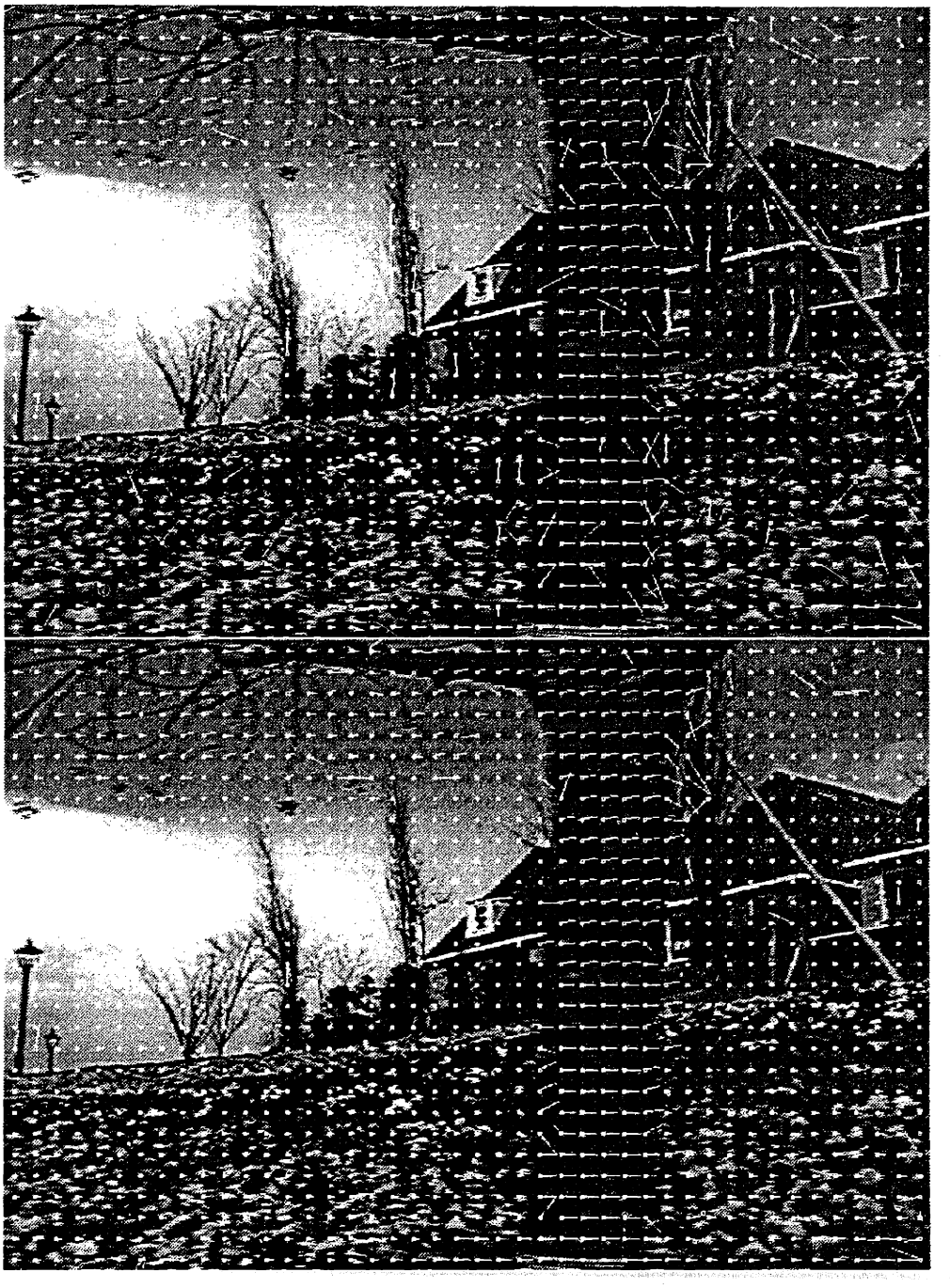


Figure 3-1: Needle Diagrams Using the MSD Criteria

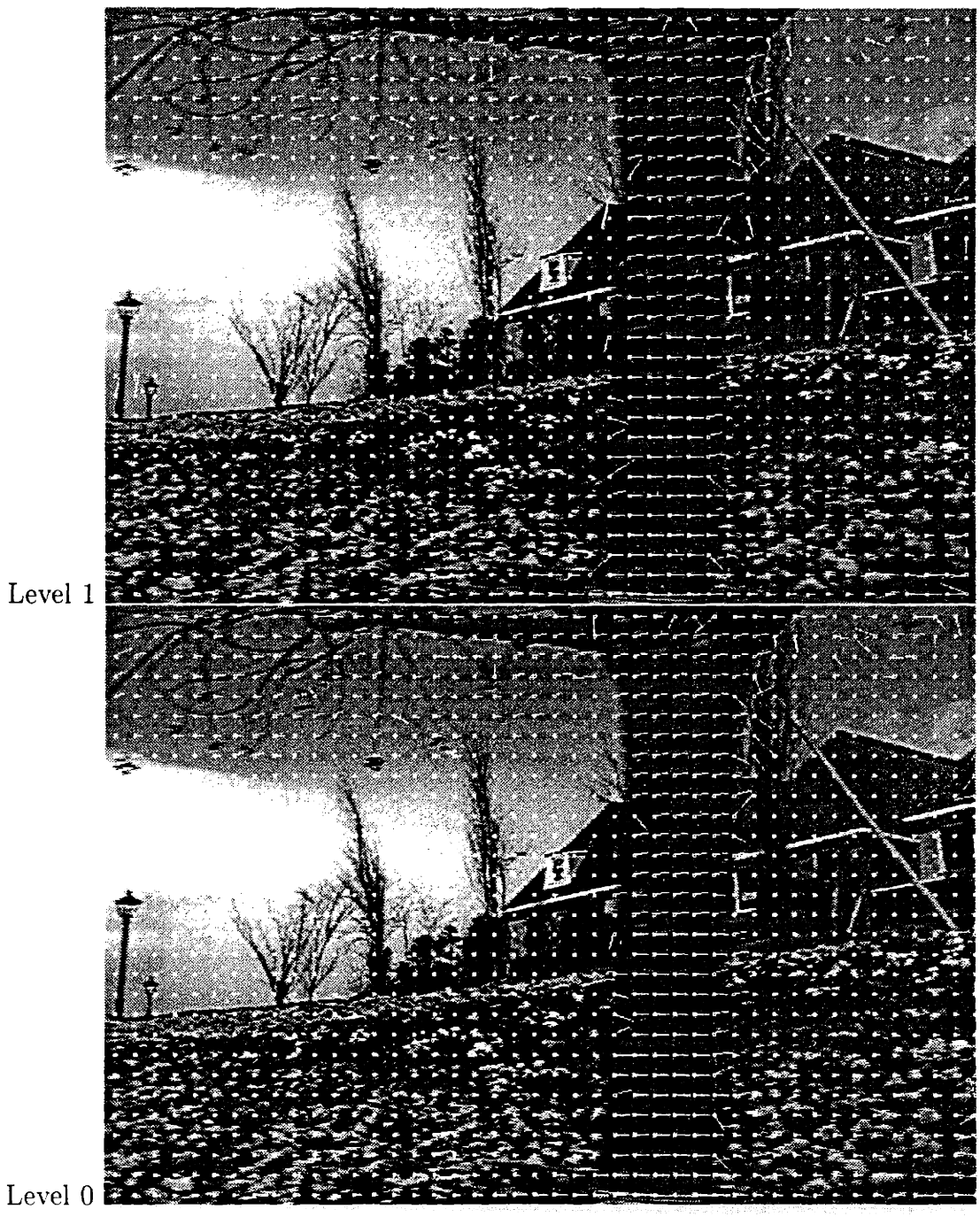


Figure 3-2: Needle Diagrams Using the MSD Criteria (continued)

The full search method (i.e. Level 0 of the hierarchical structure) generated a blurry still. Made up of six frames which were mostly translational, the salient still should have easily aligned the frames. The resultant still is shown in Figure 3-3.

Figure 3-3: Salient Still Generated Using the Full Search Method

There is an obvious amount of blurring going on. Notice that there are mainly two types of motion going on. First is the background which is made up of the house, sky, and flower garden. All these objects have approximately the same velocity from frame to frame. Then, there is the foreground object, the tree, which travels at a much greater velocity because it is closer to the camera. This object is probably obscuring the affine fit. There must be a way to segment the foreground data, so that we can achieve a higher resolution background fit.

3.2.2 Segmentation

Segmentation is an age-old problem for which elegant solutions are still unknown. Segmentation encompasses the general task of identifying some type of information in a scene.

The human visual system is extremely accurate at performing segmentation, mostly through the usage of color, brightness, lighting, texture, motion, and stereo. Algorithms to do this on a computer use similar criteria when performing segmentation but generally with extremely poor results. The reason is that humans use *a priori* knowledge when segmenting a frame like the garden sequence into such objects as a house, tree, garden, and so on. Programming such knowledge into a computer is not a realistic solution.

There have been many attempts to use motion as the basis for segmentation. Some can be found in [15] [17] [22]. However, the type of segmentation that this project needs does not demand tremendous accuracy. While more accurate segmentation will yield better results, any type of segmentation is bound to improve the affine fit. The inherent task, here, is to identify and then extract foreground motion so that the resultant salient still matches up best with the background.

Wang and Adelson [22] used both temporal and spatial information to perform this task. Having completed motion estimation and then calculated some initial affine coefficients, they warped the images onto a high resolution raster and then looked at how well the objects were lined up. Foreground objects were identified as those which matched up poorly. Using an iterative method, several foreground models were formed and then refined.

For this project, it is only necessary to group the foreground objects as one. Hence, a simpler method is proposed.

The affine coefficients are computed by trying to fit six parameters to a model which is a function of block position and estimated motion vector. Thus, if we have the block position and the affine coefficients, we can calculate what the motion vector should be for each block if the affine model were perfect. This can be done by using Equation 2.3 and Equation 2.4.

For most images, the background will occupy considerably more blocks relative to the foreground objects. Thus, it is reasonable to assume that the affine model does a pretty good job in its initial estimation and simply needs a little fine tuning.

If this assumption is correct, we can look at the estimated and actual motion vectors when determining which foreground vectors to segment. If the model is quite accurate, then the estimated and actual vectors should be within a reasonable range of each other. Thus, motion vectors which do not resemble the estimated ones are not used when re-calculating affine coefficients.

Using this segmentation procedure, I generated the needle diagram shown in Figure 3-4. Note that the white and black vectors together represent the original set of motion vectors generated using a two level hierarchy, an original block size of 16, and a search range of -16 to 16 in the x and y directions. The black vectors represent the set of vectors which were extracted.

Figure 3-4: Needle Diagram Showing the Segmented Motion Vectors

From Figure 3-4, we can see that there are essentially three major areas where segmentation occured. First, there is the tree which travels at a greater velocity than the rest of the background. Next, there is the region above the lights (at the left edge of the still) which gets segmented. This region along with the small area in the upper right hand corner of the frame are uniform areas which cause the motion estimator

problems. Because these vectors don't match up well with the affine model, these regions were removed.

Thus, a by-product of this segmentation scheme is not only to extract foreground objects, but also to remove spurious vectors in situations where the motion estimator has problems. Another example of this can be seen on the upper right boundary of the foreground tree. In this area, we can see that the motion boundary is also causing some problems. As a result, vectors in that region are also removed.

Another example is from a zoom sequence called **tennis**. The resultant segmentation can be seen in Figure 3-5.

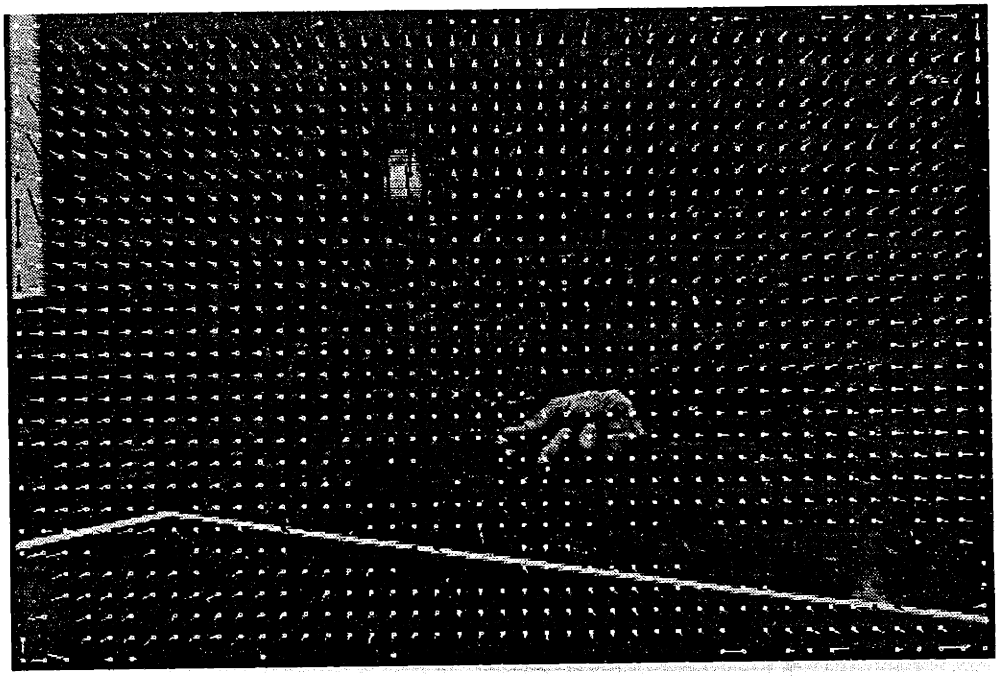


Figure 3-5: Another Needle Diagram Showing the Segmented Motion Vectors

In this example, the only real foreground object is the ping-pong ball. Hence, these motion vectors as well as other spurious ones are extracted. With all such vectors removed, the resultant salient still should match up better with the background, leading to heightened resolution.

The final algorithm is summarized in Figure 3-6.

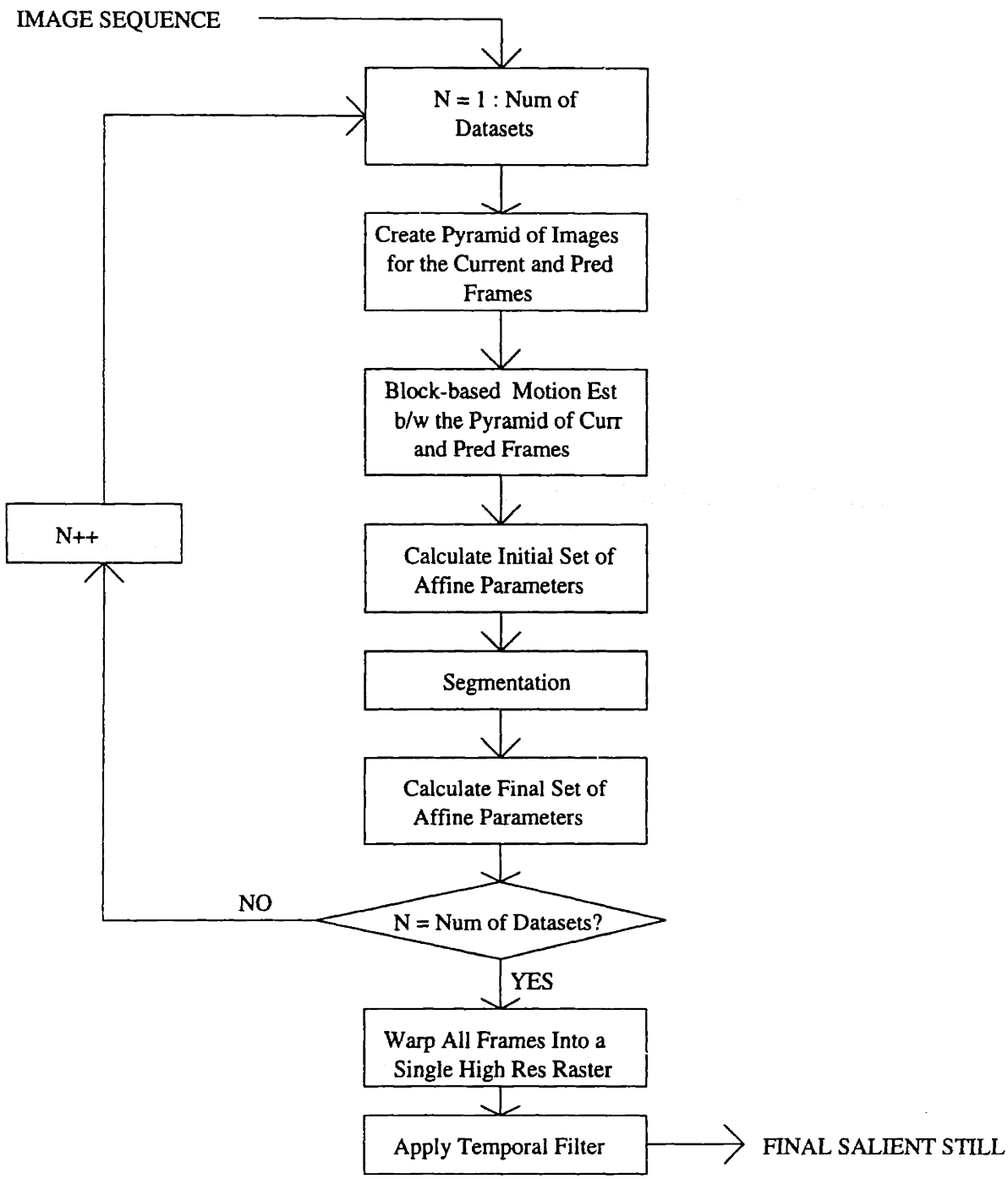


Figure 3-6: Flowchart of the Final Algorithm

Chapter 4

Resultant Salient Stills

4.1 MPEG

I ran the adapted MPEG decoder code on the MPEG-1 bitstream entitled **garden**. This sequence consisted of 73 frames which were 240x352. In all, there were 5 GOPs which consisted of 15 frames each, except for the first GOP which had 13 frames.

The P frames occurred every three frames, so a typical GOP had the display order of IBBPBBPBBPBBPBB. As was mentioned earlier, I omitted B frames to give the salient stills optimal quality. Thus, each GOP used five frames to generate the still. The salient stills generated from each GOP are shown in Figure 4-1 to Figure 4-5.

GOP1 

Figure 4-1: Salient Stills From MPEG-1 GOPs

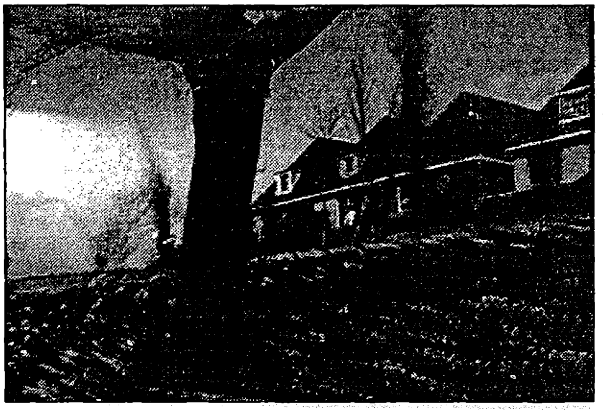
GOP2 

Figure 4-2: Salient Stills From MPEG-1 GOPs (cont)

GOP3 

Figure 4-3: Salient Stills From MPEG-1 GOPs (cont)

GOP4 

Figure 4-4: Salient Stills From MPEG-1 GOPs (cont)

GOP5 

Figure 4-5: Salient Stills From MPEG-1 GOPs (cont)

In these examples, we can see a considerable amount of blurring. This is due to several reasons. First of all, MPEG coding is lossy; hence the reconstructed image will not have as good quality. Second, the fixed block size (16x16) used for motion estimation may be too large relative to the objects in the sequence. As a result, there is a better chance that blocks contain more than one motion, skewing first the motion vectors, and second, the affine fit.

Running the estimator on a similar, but enlarged MPEG2 bitstream, it is easy to see the difference in resolution. Figure 4-6 shows a higher resolution still which was run on just one GOP. The high quality image is a direct result of smaller block to image size ratio and the higher resolution of MPEG2 reconstructed frames.

For completeness, I ran my motion estimator to link I frames with previous GOPs. The resultant salient still from all five GOPs is shown in Figure 4-7.

From Figure 4-7 , it is obvious that the error from fitting the 25 frames into a single still has accumulated. Essentially, inaccurate motion estimation caused a poor affine fit. Salient stills are generated at the decoder end and therefore rely on how good the encoding is. If the encoder does a poor job at either motion estimation or reconstructed image quality, the salient still created from this bitstream is likely to show poor resolution.

It is also important to note that motion estimation to link GOPs was done using the noisy reconstructed frames. The original vectors were calculated using a clean

Figure 4-6: Salient Still From an MPEG-2 GOP

Figure 4-7: Salient Still From Entire MPEG-1 Sequence

image sequence and still were inaccurate. As a result, the vectors which were calculated to link GOPs could only have been worse. Probably the only type of refinement which could be done at the decoder end is foreground extraction.

4.2 Hierarchical Motion Estimator

The motion estimator that I built was tested on two test sequences: the garden and tennis sequence. The garden sequence was used to show the performance of the estimator on translational sequences, while the tennis sequence was designed to test the performance on zoom sequences.

Both sequences had 480x720 resolution. I ran the estimator using a block size of 16, two hierarchical levels, and a search range from -16 to 16 in the x and y directions.

Figure 4-8 shows the salient still generated using the motion estimator on a six frame garden sequence.

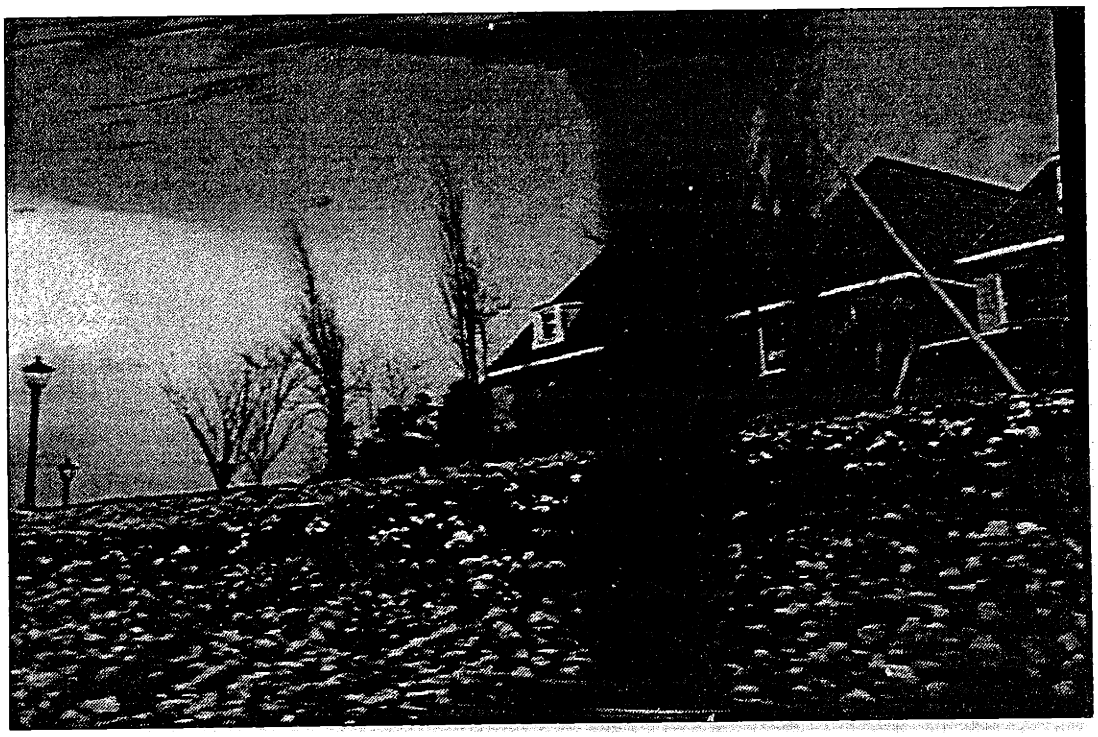


Figure 4-8: Short Garden Still Generated Using Block-Based Estimation

Note how the background matches up very well, with the main blur resulting from the temporal motion of the foreground tree. This image clearly displays even the shingles of the roof, showing the accuracy of the affine fit.

However, there are errors in the frame aligning process, even though it is not as evident in the previous case. In fact, if the motion estimator was applied to a greater number of frames, such errors would become prominent. Figure 4-9 shows the salient still from eleven frames of the garden sequence.

Figure 4-9: Long Garden Still Generated Using Block-Based Estimation

In this example, the resultant still contains more blurs than its predecessor. The match is still very good, but error has accumulated with the addition of each frame. Notice that we can now peer through the tree and see parts of the house which reveal itself later in the sequence. If an even greater number of frames are processed, the tree will become even less visible, as the temporal filter which was used was the median filter.

The still shown in Figure 4-10 was run on an eleven frame **tennis** sequence, using a 16x16 block size, two hierarchical levels, and a -16 to 16 search in the x and y directions.

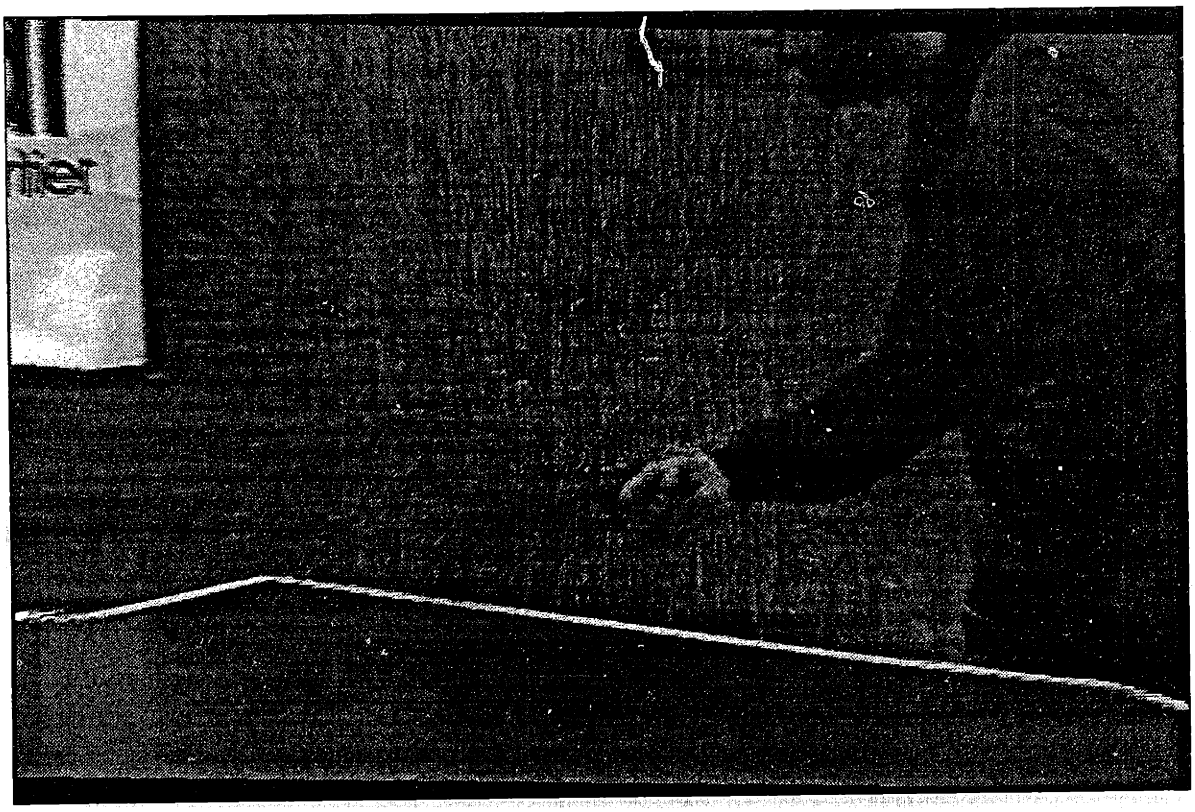


Figure 4-10: Tennis Still Generated Using Block-Based Estimation

Because the affine transformation must model zoom and translation simultaneously, the zoom sequences are generally more difficult to match. However, for this particular sequence, the motion estimator has done a pretty good job with the only noticeable problem at the edges of the ping-pong table. The person's hand and paddle are blurry, because this is where most of the foreground motion took place.

To get a better idea of the block-based estimator's performance, we now look at the salient stills created by Teodosio and Bender.

4.3 Stills by Teodosio and Bender

The original method of generating optical flow used a hierarchical optical flow method of estimating the motion. Figure 4-11, Figure 4-12, and Figure 4-13 display the results from each of the three previous examples.

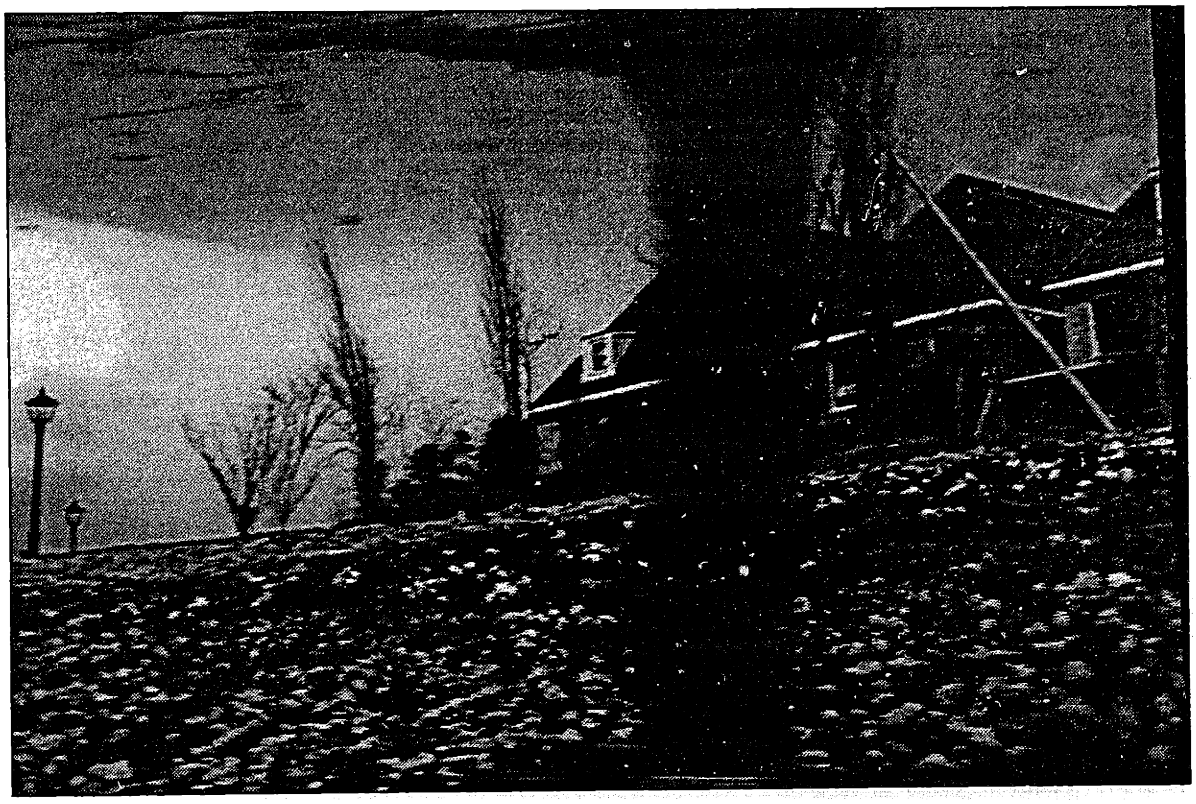


Figure 4-11: Short Garden Still Generated Using Optical Flow

For the short garden sequence, the original method has done a similar job as the block-based motion estimator. The background has excellent resolution, except for the foreground tree.

The long garden sequence, also, shows comparable results with some noticeable blur being introduced by the greater number of frames which need to be processed.

Looking at the tennis salient still, I discovered two major differences of note. First, notice how well the ping-pong table is lined up. This process has done a much better job at lining this table up. The bottom edge of the sign which is located on the left edge of the still signals another improvement.

Figure 4-12: Long Garden Still Generated Using Optical Flow

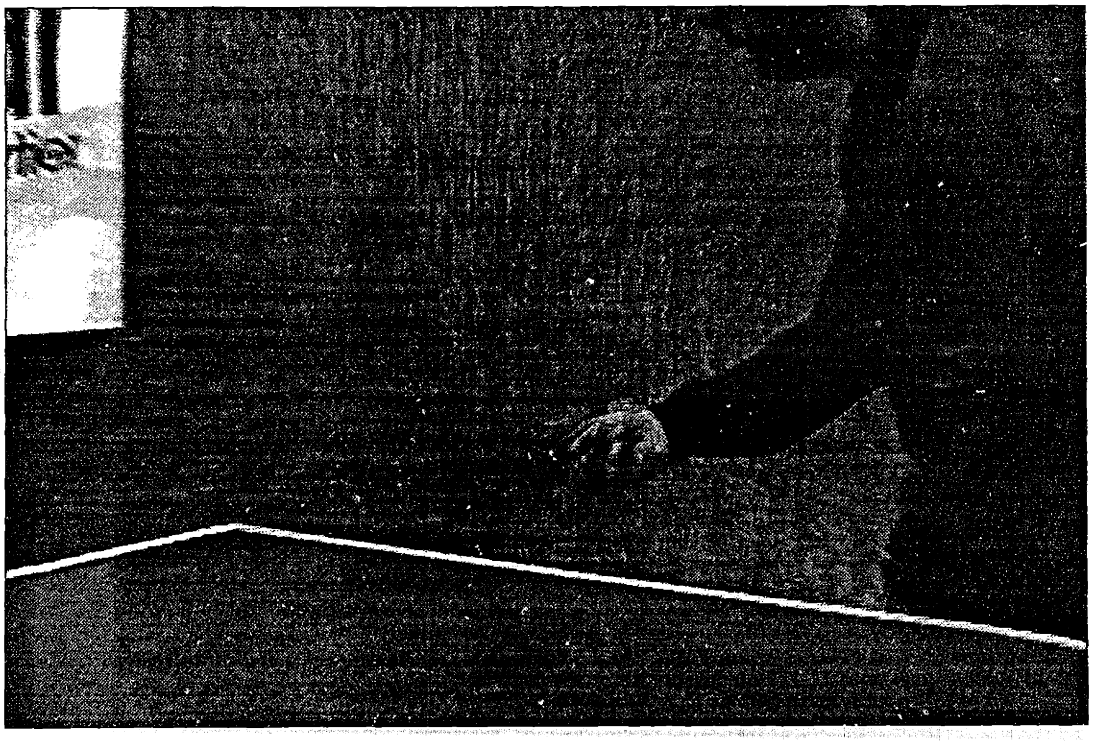


Figure 4-13: Tennis Still Generated Using Optical Flow

These improvements do not come without costs. First of all, the smaller letters on the sign have not lined up as well, making them more difficult to read. In addition, the player's head has been distorted a little and there is greater blur on the wall.

Chapter 5

Evaluation

The block-based estimator yields similar results with the original optical flow based process on translational sequences. For the more difficult zoom sequences, however, both methods possess both advantages and disadvantages. One conclusion can be garnered from the results: the process needs further refinement to better model camera motion.

It is difficult to evaluate the relative performances of both processes. Common criteria such as signal-to-noise ratios cannot be calculated because there is no reference still. In addition, the *real* affine parameters are unknown.

Hence, in order to better evaluate the performances, I created a couple sequences with known affine parameters.

5.1 Test Sequences

5.1.1 Translational Sequence

The first sequence was developed using a single frame from the garden sequence. The original sequence contained frames of dimensions 480x704. The sequence I created had dimensions of 480x640, where the first frame was lined up at the left edge and each succeeding frame was shifted to the right by 8 pels.

With nine frames in all, the correct still should have a final dimension of 480x704, and its affine coefficients should only reflect the rightward shift of 8 pels. Thus, the *true* affine coefficients for each pair of successive frames are as shown below.

$$A_x = 8/640 = .0125, \quad B_x = 1.0, \quad C_x = 0.0$$
$$A_y = 0.0, \quad B_y = 0.0, \quad C_y = 1.0$$

Using the optical flow method of motion estimation, the resultant still had the final dimension of 483x707 with the estimated affine parameters as shown in Table 5.1

Frame	A_x	B_x	C_x	A_y	B_y	C_y
1,2	.01214	.99881	.00113	.00006	.00010	.99978
2,3	.01216	.99869	.00108	.00005	.00008	.99981
3,4	.01214	.99866	.00111	.00004	.00006	.99982
4,5	.01212	.99850	.00109	.00004	.00008	.99984
5,6	.01210	.99851	.00118	.00004	.00004	.99984
6,7	.01219	.99899	.00109	.00003	.00000	.99985
7,8	.01221	.99900	.00102	.00003	.00001	.99986
8,9	.01225	.99901	.00074	.00003	.00004	.99986

Table 5.1: Affine Parameters Calculated Using Optical Flow

The salient still generated by using block-based motion estimation was 480x703. The calculated affine parameters are shown in Table 5.2.

Frame	A_x	B_x	C_x	A_y	B_y	C_y
1,2	.01234	.99958	.00014	.00000	.00000	1.00000
2,3	.01235	.99972	.00004	.00000	.00000	1.00000
3,4	.01233	.99957	.00005	.00000	.00000	1.00000
4,5	.01236	.99985	.00006	.00000	.00000	1.00000
5,6	.01235	.99966	.00002	.00000	.00000	1.00000
6,7	.01236	.99979	-.00001	.00000	.00000	1.00000
7,8	.01235	.99978	-.00002	.00000	.00000	1.00000
8,9	.01235	.99967	-.00005	.00000	.00000	1.00000

Table 5.2: Affine Parameters Calculated Using Block-Based Motion Estimation

Based on the raw numbers, the block-based method does a better job in estimating the affine coefficients and accordingly, yields a final still which has more similar dimensions. However, these differences are rather minor and cannot be seen in the salient still images shown in Figure 5-1, Figure 5-2, and Figure 5-3.

Figure 5-1: The Original Salient Still

Figure 5-2: Salient Still Generated Using Optical Flow

Figure 5-3: Salient Still Generated Using Block-Based Estimation

Judging from the previous still images, both of the motion estimators have done a fine job. Neither of the images can be distinguished from one another.

5.1.2 Zoom Sequence

The zoom sequence was created from the table tennis sequence. Using the first frame, each succeeding frame zoomed in eight pixels in the x and y directions, and then scaled to achieve the dimensions of the first frame. In all, nine frames were generated with the dimensions 480x704. Given this setup, the final dimension of the still is calculated as shown below.

$$xdim: \quad 704 * \frac{704}{704-16*8} = 860$$
$$ydim: \quad 480 * \frac{480}{480-16*8} = 654$$

Where $704-16\times8$ and $480-16\times8$ represent the x and y dimensions, respectively, of the smallest zoomed image.

The *real* affine coefficients are shown in Table 5.3.

Frame	A_x	B_x	C_x	A_y	B_y	C_y
1,2	.00000	.97727	.00000	.00000	.00000	.96667
2,3	.00000	.97674	.00000	.00000	.00000	.96552
3,4	.00000	.97619	.00000	.00000	.00000	.96429
4,5	.00000	.97561	.00000	.00000	.00000	.96296
5,6	.00000	.97500	.00000	.00000	.00000	.96154
6,7	.00000	.97436	.00000	.00000	.00000	.96000
7,8	.00000	.97368	.00000	.00000	.00000	.95833
8,9	.00000	.97297	.00000	.00000	.00000	.95652

Table 5.3: Real Affine Parameters

Using the original optical flow method of motion estimation, the resultant still had the final dimension of 653x859 with the estimated affine parameters shown in Table 5.4

The salient still generated by using block-based motion estimation had the dimension 653x856. The resulting affine parameters are shown in Table 5.5.

Frame	A_x	B_x	C_x	A_y	B_y	C_y
1,2	.00005	.97691	-.00009	.00013	.00018	.96661
2,3	.00002	.97699	.00004	.00004	-.00011	.96519
3,4	-.00002	.97601	.00001	.00005	.00001	.96449
4,5	.00004	.97557	.00008	.00000	.00011	.96268
5,6	.00002	.97506	-.00001	.00007	.00003	.96193
6,7	-.00001	.97442	-.00007	.00010	-.00009	.95929
7,8	.00002	.97360	.00011	.00007	.00011	.95838
8,9	.00004	.97286	.00001	.00006	-.00010	.95635

Table 5.4: Affine Parameters Calculated Using Optical Flow

Frame	A_x	B_x	C_x	A_y	B_y	C_y
1,2	.00004	.97734	-.00004	.00002	-.00001	.96652
2,3	.00004	.97645	.00002	.00002	.00049	.96562
3,4	.00010	.97664	.00016	.00003	.00060	.96418
4,5	-.00005	.97634	-.00024	.00010	.00018	.96267
5,6	-.00001	.97593	.00011	.00006	.00049	.96237
6,7	.00000	.97471	-.00002	.00004	.00053	.95968
7,8	.00003	.97409	.00019	.00006	-.00004	.95844
8,9	.00003	.97315	-.00006	.00000	-.00026	.95651

Table 5.5: Affine Parameters Calculated Using Block-Based Motion Estimation

According to the previous figures, both estimators have again done a very similar job, with the optical flow method doing a slightly better job. The differences in the stills of Figure 5-4, Figure 5-5, and Figure 5-6 are not visible.

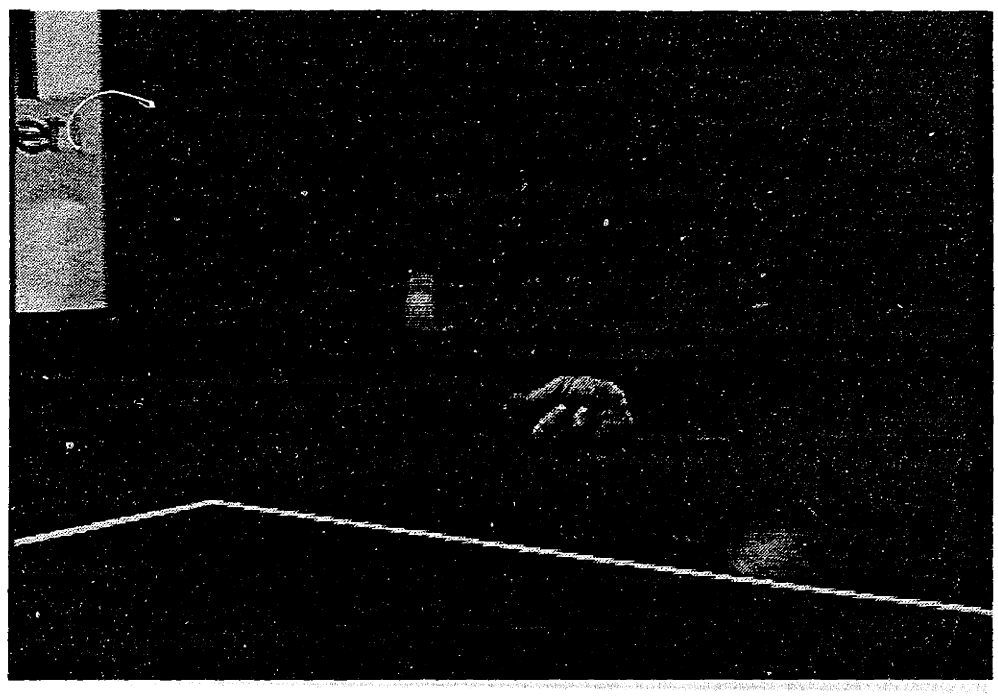


Figure 5-4: The Original Salient Still

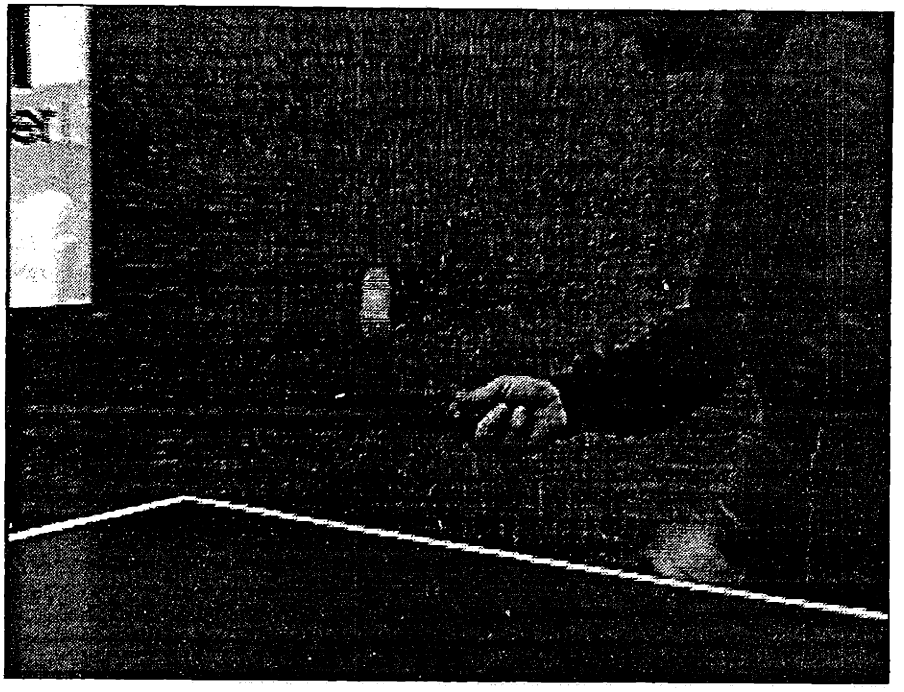


Figure 5-5: Salient Still Generated Using Optical Flow

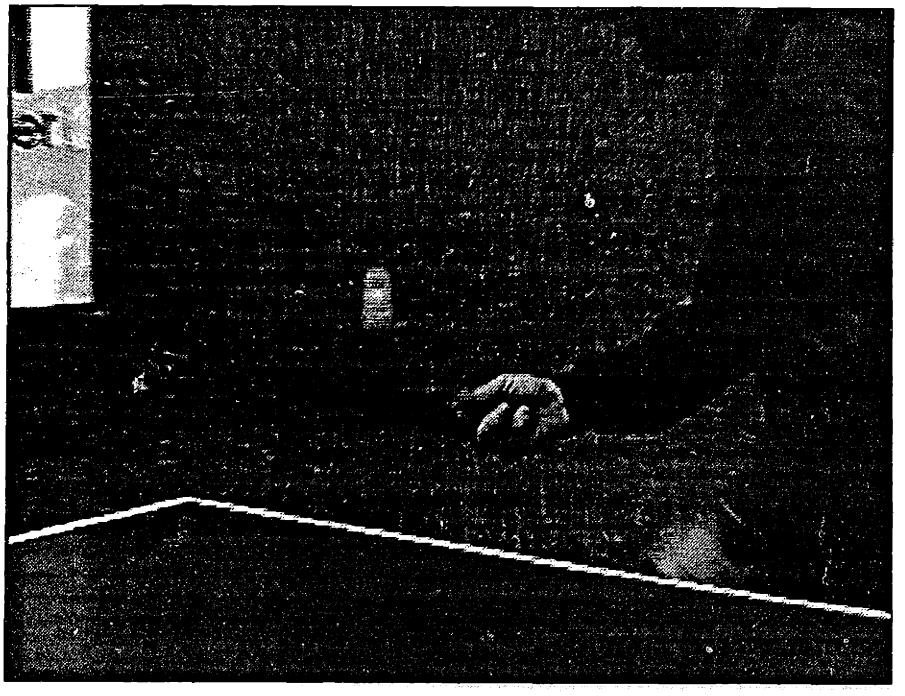


Figure 5-6: Salient Still Generated Using Block-Based Estimation

Chapter 6

Conclusion

The block-based and optical flow motion estimators yield very similar results. Based on the calculated affine parameters as well as the rendered stills, the block-based method appeared to do a slightly better job on the translational sequences while the optical flow method performed better on the zoom sequences.

It is significant that the block-based motion estimator can yield similar stills to those of the original, optical flow based method. Block-based methods are often used in video coding, because motion information is only sent for a block a data, rather than for every pixel.

As today's society becomes an information society, video coding methods will become more predominant. Hence, to extend the application of salient stills while preserving a high degree of resolution, it was significant that the MPEG2 generated stills possessed good quality.

Although the results are promising, there are still some improvements which are needed. One limitation of the salient still generating process is that the affine model is not able to handle three dimensional motion. An improved model is the perspective model. Adding two parameters to the affine model, the perspective model is designed to compensate for perspective distortion. However, in a recent implementation, the eight coefficients did not always converge to the expected values. An in-depth look at this process is definitely needed.

In general, the results are encouraging. The salient stills created in this paper

have shown a remarkable ability to capture the spatial and temporal information of an image sequence, while maximizing the resolution. Even without the affine model's ability to model three-dimensional motion, salient stills have completed its purpose.

Bibliography

[1] ISO/IEC 11172, "Information Technology - Coding of moving picture and associated audio for digital storage media as up to about 1.5 Mbits/s", Committee Draft, 1993.

[2] ISO/IEC CD 13818-2, Recommendation H.262, "Generic Coding of Moving Pictures and Associated Audio", Committee Draft, Seoul, November 1993.

[3] E.H. Adelson, C.H. Anderson, J.R. Bergen, P.J. Burt, and J. M. Ogden, "Pyramid Methods in Image Processing", RCA Engineer, Nov/Dec 1984.

[4] J.R. Bergen, P.J. Burt, R. Hingorami, and S. Pelag, "Computing Two Motions From 3 Frames", David Sarnoff Research.

[5] M. Bierling, "Displacement Estimation by Hierarchical Blockmatching", Proceedings of the SPIE, Visual Communications and Image Processing '88, Vol. 1001, p. 942-951.

[6] P.J. Burt and E.H. Adelson, "The Laplacian Pyramid as a Compact Image Code", IEEE TransCom, COM-31(4), 1983.

[7] Frederic Dufaux and Murat Kunt, "Multigrid Block Matching Motion Estimation with an Adaptive Local Mesh Refinement", Proceedings of the SPIE, Visual Communications and Image Processing, vol. 1818, p. 97-102.

[8] C.S. Fuh and P. Maragos, "Region-Based Optical Flow Estimation'," Proceeding of the IEEE Conference on Computer Vision and Pattern Recognition, p. 130-135, June 1989.

[9] D.J. Heeger, "Optical Flow Using Spatiotemporal Filters," International Journal of Computer Vision, p. 279-302, 1988.

[10] B.K.P. Horn and B.G. Schunck. "Determining Optical Flow", *Artificial Intelligence*, vol. 17, p. 185-203, August 1981.

[11] Caspar Horne, "Improving Block Based Motion Estimation by the Use of Global Motion", Procedings of the SPIE, Visual and Image Processing, vol. 2094, p. 576-580.

[12] Roger Kermode, "Building the Big Picture: Enhanced Resolution from Coding", Master's Thesis, Massachusetts Institute of Technology, June 1994.

[13] Didier Le Gall, "MPEG: A video compression standard for multimedia applications", Communications of the ACM, April 1991/Vol 34, No. 4

[14] J. S. Lim, *Two-Dimensional Signal and Image Processing*, Prentice Hall, 1993.

[15] Bede Liu, King-Wai Chow, and Andre Zaccarin, "A Simple Method to Segment Motion Field for Video Coding", Proceedings of the SPIE, Visual Communications and Image Processing, vol. 1818, p. 542-550.

[16] Steve Mann, "Compositing Multiple Pictures of the Same Scene", Proceedings IS and T Annual Meeting, May 1993.

[17] Michael T. Orchard, "Predictive Motion Field Segmentation for Image Sequence Coding", ICASSP, vol. 4, p. 1977-1980.

[18] Brian Schwartz and Ken Sauer, "Integral Projection Methods for Block Motion Estimation", Proceedings of the SPIE, Visual and Image Processing, vol. 2094, p. 553-557.

[19] Vassilis Seferidis and Mohammad Chanbari, "Generalized Block Matching Motion Estimation", Proceedings of the SPIE, Visual Communications and Image Processing, vol. 1818, p. 110-119.

[20] Laura Teodosio, "Salient Stills", Master's Thesis, Massachusetts Institute of Technology, September 1992.

[21] Laura Teodosio and Walter Bender, "Salient Stills: Content and Context Preserved", Proceedings of the ACM Multimedia Conference, Anaheim, August, 1993.

[22] John Y.A. Wang and Edward Adelson, "Spatio-Temporal Segmentation of Video Data", Proceedings of the SPIE, Image and Visual Processing II, vol. 2182, San Jose, February 1994.

Acknowledgments

There are many people I'd like to thank:

Walter Bender, for being a patient and informative advisor.

Edward Adelson, for help with the hierarchical and segmentation process.

Henry Holtzman, for answering all my MPEG questions.

Costa Sapuntzakis, for help with the affine warping process.

Roger "Woja" Kermode, for the great diagrams and general comic relief.

Shawn Becker, for taking the time to answer all my image processing questions, as well as starting me out with the motion estimator.

Daniel Gruhl, for being the all-time LaTex god.

Ray, Frank, and Hoony, for keeping me sane throughout the years.

And of course, Susan, who has always been my source of inspiration.