BITSTREAM BASED OVERLAP DETECTION

FOR

MULTI-VIEW DISTRIBUTED VIDEO CODING

BY

IVAN MECIMORE, B.S.

A thesis submitted to the Graduate School

in partial fulfillment of the requirements

for the degree

Master of Science

Major Subject: Electrical and Computer Engineering

New Mexico State University

Las Cruces New Mexico

August 2008
“Bitstream Based Overlap Detection for Multi-View Distributed Video Coding,”
a thesis prepared by Ivan Mecimore in partial fulfillment of the requirements for
the degree, Master of Science, has been approved and accepted by the following:

Dr. Linda Lacey
Dean of the Graduate School

Dr. Charles D. Creusere
Chair of the Examining Committee

Date

Committee in charge:

Dr. Charles D. Creusere, Chair

Dr. Deva K. Borah

Dr. Joe Lakey
DEDICATION

I dedicate this work to those who are special to me, my family and Evan.
ACKNOWLEDGMENTS

I would like to thank my adviser, Dr. Charles D. Creusere, for his encouragement, interest, and patience.
VITA

February 16, 1984 Born at Madrid, Spain

2002-2006 B.S., New Mexico Institute of Mining and Technology, Socorro, New Mexico

2006-2008 M.S., New Mexico State University, Las Cruces, New Mexico

PROFESSIONAL AND HONORARY SOCIETIES

Institute of Electrical and Electronics Engineers, Inc.

PUBLICATIONS


FIELD OF STUDY

Major Field: Digital Signal Processing
ABSTRACT

BITSTREAM BASED OVERLAP DETECTION
FOR
MULTI-VIEW DISTRIBUTED VIDEO CODING
BY
IVAN MECIMORE, B.S.

Master of Science
New Mexico State University
Las Cruces, New Mexico, 2008
Dr. Charles D. Creusere, Chair

In this thesis, we develop and explore a low complexity algorithm for spatial overlap detection and characterization that operates directly in the bitstream of motion-JPEG compressed video. Its low complexity and the fact that it does not require video decoding at the sensor nodes makes it very well suited for wireless sensor networks. We discover that operating in the bitstream domain provides many advantages such as lower complexity and added resilience to noise. In addition to determining the overlapping regions in spatially separated video, the
proposed algorithm can accurately identify and localize non-common foreground objects within the common overlap region of the different cameras field’s of view.
CONTENTS

LIST OF TABLES ................................................. x
LIST OF FIGURES ............................................... xi
1 INTRODUCTION .................................................. 1
2 JPEG COMPRESSION ............................................. 4
  2.1 Baseline Coding System ........................................ 4
  2.2 Modifications to JPEG ......................................... 8
3 OVERLAP DETECTION ........................................... 14
  3.1 Preponderance Domain Feature Extraction ....................... 15
  3.2 Overlap Detection ........................................... 19
4 NON-COMMON ELEMENT REMOVAL ............................... 21
  4.1 Algorithm .................................................. 21
5 RESULTS ....................................................... 24
  5.1 Laboratory Experimentation ...................................... 24
    5.1.1 Laboratory Experiment 1 .................................. 24
    5.1.2 Laboratory Experiment 2 .................................. 25
  5.2 MERL Database Experiments ................................... 28
    5.2.1 MERL Experiment 1 ....................................... 33
    5.2.2 MERL Experiment 2 ....................................... 36
viii
# LIST OF TABLES

2.1 Recommended quantization array. ........................................ 7
2.2 JPEG coefficient coding categories. .................................... 9
2.3 JPEG default DC code. .................................................. 9
2.4 JPEG default AC code. .................................................. 10
5.1 MERL Experiment 1 - Ballroom ....................................... 34
5.2 MERL Experiment 1 - Exit ............................................. 35
5.3 MERL Experiment 1 - Vassar .......................................... 36
5.4 MERL Experiment 2 ..................................................... 37
5.5 MERL Experiment 3 ..................................................... 41
5.6 Quantization Experiment 1 ............................................. 43
5.7 Quantization Experiment 2 ............................................. 44
LIST OF FIGURES

2.1 JPEG Encoding Flowchart .................................. 5
2.2 Zig-zag Sequence ........................................... 6
2.3 A Real and Preponderance Image ............................ 12
3.1 Block diagram for Overlap Detection ......................... 15
3.2 An Example of the Detected Overlap Region ................. 16
3.3 Different stages of the Feature Extraction Process ......... 18
4.1 An Example of Problems occurring from Camera Geometry 22
4.2 Sub-division Process ......................................... 23
5.1 Captures from each camera position used in Experiment 1 26
5.2 Results from Laboratory Experiment 2 ........................ 29
5.3 Frame 1 from each camera of the Ballroom Sequence ....... 30
5.4 Frame 1 from each camera of the Exit Sequence ............. 31
5.5 Frame 1 from each camera of the Vassar Sequence ........... 32
5.6 Example - Non-Common Element Removal .................. 39
1 INTRODUCTION

The distributed source coding of today is based on the information theory work began back in the 1970s by Slepian and Wolf and Wyner and Ziv [10] [14]. In recent years, practical algorithms have been developed to use the parity bits generated by channel codes to correct the differences between an actual sensed signal and an approximation of that signal derived from side information [8]. The introduction of these algorithms combined with the recent interest in distributed sensor networks has led to a great deal of recent research in the area. An off-shoot of this research area has been called distributed video compression (DVC) [9], [3].

The focus of the work here is on the problem of multi-view distributed video coding. This area has recently been the target of great research interest [1, 7, 11, 16, 15]. In this application, we will consider multiple video cameras located some small distance apart and oriented in the same general direction. Clearly, there will be correlations in the acquired signals whenever the cameras’ fields of view overlap – exactly how much correlation depends on the amount of overlap and the relative camera geometry since each camera only captures a 2-dimensional projection of the underlying 3-dimensional scene. The main goal of the research here is to determine areas of high correlation enabling the future exploitation for coding purposes of any correlations that might exist between spatially separated cameras.
The major focus of multi-view DVC has been on the problem of synthesizing the side information in the joint decoder [1, 7, 11, 16, 15]. In most cases, it is assumed that the individual encoders are unable to communicate amongst themselves and that the decoder must extract frame correlations without assistance from either encoder. Some of the basic difficulties inherent in these approaches lie in the use of poor side information synthesis at the decoder, an inefficient parity bit generation for the Wyner-Ziv encoder, and the real-time limitations of decoder feedback for rate-control. These limitations of the current approach have led us to explore a new approach: to still achieve a very low complexity video encoding at each node but allow nodes to take advantage of what they may ‘overhear’ from nearby nodes. For example, one node may be able to determine that a portion of its camera’s field of view is currently overlapping that of another node’s camera by passively monitoring and analyzing the communications of nearby nodes.

As discussed, the objective of multi-view distributed coding is to develop a method to effectively reduce hardware and energy costs, namely, to reduce computational complexity, bandwidth requirements and power consumption. The major difficulty with the passive communications approach described above is encoder complexity. This is a result of requiring a sensing node to spend energy and computational processing power to listen to and analyze the signals received from nearby sensors. In general, to listen to a nearby node, we must receive the RF signal, demodulate it, and decode it to reconstruct the video frames. Once
decoded, these frames must be compared to the frames which have been captured locally to determine correlations. Finally, this correlation must be exploited in the encoding process in a way that justifies the extra steps required.

In this thesis, we explore one portion of this problem, namely the efficient estimation of correlation at the sensor nodes. To achieve this goal, we propose to determine the overlap between camera fields of view at one of the cameras by analyzing the frames from nearby nodes in the compressed bitstream domain. While this approach has many advantages over working in the pixel domain, the most significant gains are: (1) the need to decode the passively captured bitstream is eliminated saving both power and computational bandwidth, (2) far less information needs to be processed since overlap detection is performed in the compressed domain, and (3) the final system should be more robust to small shifts in the field of view since the frame overlap calculation and the dependent encoding are both performed in the bitstream domain. Specifically, we propose to use the bitstream domain of motion JPEG encoded frames. Motion JPEG provides a more than adequate method of compression that allows us to analyze each frame independently and with relatively low complexity.
2 JPEG COMPRESSION

The focus of this chapter is on the basics of motion-JPEG compression. While the motion-JPEG used here is not state-of-the-art, it certainly satisfies the desired requirements for a distributed video coding system, namely the video encoder has low-complexity. It should be noted, however, that the proposed approach can be extended to MPEG-1, MPEG-2, and MPEG-4 video encoders by using the I-frame only modes. For the work here, the baseline coding system is used which is based on the Discrete Cosine Transform (DCT) and is specified in the JPEG standard [13]. Since it is possible to view motion-JPEG as a series of independently and sequentially encoded frames, we describe here only how a single frame is encoded.

2.1 Baseline Coding System

In the lossy baseline coding system, the compression itself is performed in three sequential steps: DCT computation, quantization, and variable-length code assignment, as shown in Figure 2.1.

**DCT Computation:** Each image is first subdivided into $8 \times 8$ blocks of pixels. These blocks are processed from left to right and top to bottom. As each sub-block is encountered, the pixels in it are level shifted by $2^{n-1}$ where $n$ is the number of bits per input pixel or 128 in the case of 8-bit pixels. Next, the two-dimensional (2-D) discrete cosine transform is computed for the block. The 2-D DCT of an
Figure 2.1: JPEG Encoding Flowchart

8 × 8 block of input pixels $x(n, m)$ is given by

$$T(u, v) = \sum_{n=0}^{7} \sum_{m=0}^{7} x(n, m) \cos \left(\frac{\pi}{8} (n + \frac{1}{2})u\right) \cos \left(\frac{\pi}{8} (m + \frac{1}{2})v\right).$$  \hspace{1cm} (2.1)

**Quantization:** After the 2-D DCT is performed, a normalization array is used to quantize the transformed array. The baseline coding system recommends the proven quantization array given in Table 2.1 and quantization is then accomplished using

$$\hat{T}(u, v) = \text{round} \left[ q \times \frac{T(u, v)}{Z(u, v)} \right]$$  \hspace{1cm} (2.2)

where $T$ is given by 2.1, $Z(u, v)$ is the quantization array from Table 2.1 and $q$ controls the quality.

The quantization array is specifically designed to perceptually weight the coefficients in the 2-D array. Lastly, the quantized coefficients of the 2-D array are reordered according with to the zig-zag ordering pattern of Figure 2.2 and losslessly encoded in a serial fashion to create the bitstream using the following method.
Figure 2.2: Zig-zag Sequence
Table 2.1: Recommended quantization array.

<p>| | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>16</td>
<td>11</td>
<td>10</td>
<td>16</td>
<td>24</td>
<td>40</td>
<td>51</td>
<td>61</td>
</tr>
<tr>
<td>12</td>
<td>12</td>
<td>14</td>
<td>19</td>
<td>26</td>
<td>58</td>
<td>60</td>
<td>55</td>
</tr>
<tr>
<td>14</td>
<td>13</td>
<td>16</td>
<td>24</td>
<td>40</td>
<td>57</td>
<td>69</td>
<td>56</td>
</tr>
<tr>
<td>14</td>
<td>17</td>
<td>22</td>
<td>29</td>
<td>51</td>
<td>87</td>
<td>80</td>
<td>62</td>
</tr>
<tr>
<td>18</td>
<td>22</td>
<td>37</td>
<td>56</td>
<td>68</td>
<td>109</td>
<td>103</td>
<td>77</td>
</tr>
<tr>
<td>24</td>
<td>35</td>
<td>55</td>
<td>64</td>
<td>81</td>
<td>104</td>
<td>113</td>
<td>92</td>
</tr>
<tr>
<td>49</td>
<td>64</td>
<td>78</td>
<td>87</td>
<td>103</td>
<td>121</td>
<td>120</td>
<td>101</td>
</tr>
<tr>
<td>72</td>
<td>92</td>
<td>95</td>
<td>98</td>
<td>112</td>
<td>100</td>
<td>103</td>
<td>99</td>
</tr>
</tbody>
</table>

**Variable-length Code Assignment:** The result from the zigzag sequence is a 1-D array of coefficients that are roughly ordered according to increasing spatial frequency. Since natural images generally contain mostly low frequencies, this reordering results in zeros that are often clustered together near the end of this scanning list in the high frequency region of the DCT. The JPEG coding scheme is designed to take advantage of these zeros, resulting in very efficient compression in these areas. Huffman coding is an optimal entropy encoding algorithm used for lossless data compression, and it creates a variable-length code table which is based on the estimated probability of occurrence for each symbol [6]. Since a single Huffman code for all the values of the coefficients would be very large...
and quite unmanageable, JPEG recommends partitioning the possible values into several categories. Each quantized coefficient is represented by a base code and a remainder consisting of its least significant bits (LSBs). The number of possible values in a category is $2^n$ where $n$ is the category index.

The DC coefficient is the first value in the array and is differentially coded, with respect to Tables 2.2 and 2.3, relative to the DC coefficient of the previous sub-block in the raster scan of the image. The nonzero AC coefficients, the remaining values in the block, are coded similarly using Tables 2.2 and 2.4. The code for an AC coefficient, however, relies on a run-length parameter $\text{Run}$ as well as the category classification. The $\text{Run}$ is determined by the number of zero-valued coefficients preceding the nonzero coefficient that is being encoded. After the Base Code has been determined, the table will provide a length value with the number of bits in the codeword being equal to the length value. The first bits indicate the Run/Category value while the remaining bits are the LSBS of the binary number to be encoded.

2.2 Modifications to JPEG

An important part of JPEG in this work is that it encodes each $8 \times 8$ block largely independently. Thus, one can view the information in each block as being represented by the number of bits used to encode it. Consequently, the pattern created by the bit counts of the $8 \times 8$ pixel blocks that form a region of a video
Table 2.2: JPEG coefficient coding categories.

<table>
<thead>
<tr>
<th>Range</th>
<th>DC Difference Category</th>
<th>AC Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>0, -1, 1</td>
<td>0</td>
<td>N/A</td>
</tr>
<tr>
<td>-3, -2, 2, 3</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>-7, ..., -4, 4, ..., 7</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>-15, ..., -8, 8, ..., 15</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>-31, ..., -16, 16, ..., 31</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>-63, ..., -32, 32, ..., 63</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>-127, ..., -64, 64, ..., 127</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>-255, ..., -128, 128, ..., 255</td>
<td>8</td>
<td>8</td>
</tr>
</tbody>
</table>

Table 2.3: JPEG default DC code.

<table>
<thead>
<tr>
<th>Category</th>
<th>Base Code</th>
<th>Length</th>
<th>Category</th>
<th>Base Code</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>010</td>
<td>3</td>
<td>5</td>
<td>110</td>
<td>8</td>
</tr>
<tr>
<td>1</td>
<td>011</td>
<td>4</td>
<td>6</td>
<td>1110</td>
<td>10</td>
</tr>
<tr>
<td>2</td>
<td>100</td>
<td>5</td>
<td>7</td>
<td>11110</td>
<td>12</td>
</tr>
<tr>
<td>3</td>
<td>00</td>
<td>5</td>
<td>8</td>
<td>111110</td>
<td>14</td>
</tr>
<tr>
<td>4</td>
<td>101</td>
<td>7</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Table 2.4: JPEG default AC code.

<table>
<thead>
<tr>
<th>Run/Category</th>
<th>Base Code</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>0/0</td>
<td>1010(=EOB)</td>
<td>4</td>
</tr>
<tr>
<td>0/1</td>
<td>00</td>
<td>3</td>
</tr>
<tr>
<td>0/2</td>
<td>01</td>
<td>4</td>
</tr>
<tr>
<td>0/3</td>
<td>100</td>
<td>6</td>
</tr>
<tr>
<td>0/4</td>
<td>1011</td>
<td>8</td>
</tr>
<tr>
<td>0/5</td>
<td>11010</td>
<td>10</td>
</tr>
<tr>
<td>:</td>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td>F/0</td>
<td>111111110111</td>
<td>12</td>
</tr>
<tr>
<td>F/1</td>
<td>111111111110101</td>
<td>17</td>
</tr>
<tr>
<td>F/2</td>
<td>1111111111110110</td>
<td>18</td>
</tr>
<tr>
<td>F/3</td>
<td>1111111111110111</td>
<td>19</td>
</tr>
<tr>
<td>:</td>
<td>:</td>
<td>:</td>
</tr>
</tbody>
</table>
frame provide us with information about the spatial composition of that region which can, in theory, be used for matching purposes. An example of such an image is shown in Figure 2.3. It is apparent that basic shapes and structures are recognizable between both images by visual inspection. We propose here to use the patterns formed by the bit allocations instead of pixels to perform overlap detection.

Since the number of bits used to encode each $8 \times 8$ block is now the information used for matching, we must have a way to easily extract this information from the compressed bitstream without fully decoding it and consequently losing much of the computational savings desired. As has been discussed above, JPEG uses a Huffman run-length coder for lossless compression as its final step. It is usually necessary to decode the bitstream in order to read the symbols correctly. This means we cannot simply start halfway through the bitstream and begin correctly decoding the sequence.

In order to simplify the process of extracting the bit counts, a slight modification to JPEG is necessary. During the encoding process, we remove the end-of-block symbol from the end of every $8 \times 8$ block. Instead, we include a header that contains the number of bits used to encode each block of pixels. The header is differentially and losslessly encoded to minimize the impact of including this extra information on compression efficiency. This modification has a slight increase in the overall bit rate by about 4% on the image in Figure 2.3, but it is worthwhile
Figure 2.3: An Example of a Real and a Preponderance Image: (a) the real image and (b) the preponderance image with block bit counts. In the preponderance image dark areas are low bit counts while light areas are high bit counts.
since it allows for easy access to the bit counts for each pixel block with very
minimal computational effort. In general, the percentage increase in bit rate will
grow as the overall bit rate decreases and shrink as the bit rate increases. While
we would rather avoid this performance penalty, it is probably worth accepting
since it eliminates the need for video decoding in the sensor node and thus signif-
ically reduces the complexity and yet still allows us direct access to information
that we can use to extract cross-sensor correlations. At the same time, since this
modification is so simple, fast implementations of JPEG can still be used with
only minimal changes.
3 OVERLAP DETECTION

We henceforth assume that the number of bits used to encode each 8 × 8 block of the image is extracted directly from a header included within the modified motion JPEG frame. The number of bits used to encode a block will be called the preponderance number (PN) for the block. Because of the way JPEG is encoded, the PN loosely characterizes the frequency content of the block. If the block contains a lot of high frequency energy, then its PN is high; if it does not, then its PN is low. Using this information, we then can match regions in overlapping images using the patterns formed by their preponderance numbers. This is preponderance of bits detection or PBD for short.

Figure 3.1 shows the block diagram for the proposed overlap detection algorithm. The algorithm can be divided into two subsections. The first step, the preponderance domain feature extraction, processes the information in the preponderance domain and creates a binary map of the frequency structure of the video frame. The second step uses the binary map to compare two video frames for similarity and determine where an overlap might begin. The final result of this process can be seen in Figure 3.2 and should identify the area in two images which share the same field of view.
3.1 Preponderance Domain Feature Extraction

First, gray-level slicing is performed to separate high and low frequency preponderance numbers as follows:

\[
H(x, y) = \begin{cases} 
1 & \text{if } P(x, y) \geq c \\
0 & \text{otherwise}
\end{cases}
\]  

(3.1)

where \( P(x, y) \) is element \((x, y)\) in the preponderance matrix \( P \) and \( H(x, y) \) is element \((x, y)\) in the resulting binary image \( H \).

To find the value of \( c \) around which we slice, we use the \( k \)-means procedure to cluster the high and low frequencies and then select the midpoint of the two means, \( c \). The \( k \)-means algorithm clusters \( n \) objects into \( k \) partitions based on the object’s attributes. To reduce complexity, we have chosen to use the Lloyd algorithm [5] which is an iterative approach. To begin, two clusters are formed, the initial centroid points of these clusters are selected as the minimum and maximum preponderance values. The Lloyd algorithm associates each \( P(x, y) \) with one of
(a) Camera A

(b) Camera B

(c) Combination of Camera A and Camera B

Figure 3.2: The lighter area is the determined overlap region. Since both camera positions are only a slight distance apart the images look similar. The blurring in the image is due to the resolution of a match being restricted to the nearest 8 pixels and a slight change in camera angle.
the two clusters by choosing the closest with respect to the minimum Euclidean distance. The new centroids of the two clusters are then calculated and another round of partitioning using the new cluster centroids is performed. Iteration stops when the centroid of both clusters no longer changes.

After the gray-level slicing is completed, we remove isolated areas of 1s and 0s using the morphological operation of closing [4]. To implement the closing operator, dilation is first performed using a 3x3 rectangular structuring element of ones followed by erosion with the same structuring element. The purpose of the morphological closing operation is to accentuate changes in the neighboring preponderance numbers. Ideally, this procedure will help fine-tune decisions made in the gray-level slicing phase, particularly those preponderance values which fall in the midrange. For example, an area that is classified as low frequency but has many high frequency neighbors could be a classification error, or simply an element which has a moderate amount of frequency content. Values that are in question such as these would ideally be categorized similarly to their neighboring values. Ultimately, this operation divides the video frame into regions of high and low frequency content. All the steps thus far can be implemented entirely using binary and integer operations with the need to only store ones and zeros, allowing for very low computational complexity.
Figure 3.3: Different stages of the feature extraction process. (b) is known at the listening node without decoding the bitstream from (a); from (c), the result of the gray-level slicing procedure, it becomes apparent that this image is very similar to that of an edge detector; (d) is the result of the morphological closing operation with the image now loosely divided into high and low frequency areas.
3.2 Overlap Detection

The detection process applied here will use the feature extracted preponderance map previously discussed to match up the structure of high and low frequency regions in the two frames being evaluated. For overlap detection, to reduce the amount of computation necessary, we assume that a small amount of side information is available describing the relative placement of the cameras. Specifically, we assume that the only information available is which side camera 1 is on relative to camera 2. We believe this assumption is reasonable because most sensing networks have some sort of localization already integrated; as one example, this information is easily extracted from relatively imprecise non-differential commercial GPS units. Using the side information describing the relative position of each camera, we can determine which side of the images might possibly overlap. Consequently, we start our detection process in the top-left corner of the right-most image. A block size, \( s_{qs} \), is specified, good values for which have been determined experimentally to be in the range of 20 to 30. A block \( B \) of size \( s_{qs} \times s_{qs} \) is then selected from the top-left corner of the extracted preponderance map developed from the rightmost camera image. The mean absolute error (MAE) relative to the preponderance map \( C \) from the leftmost camera image is calculated using

\[
MAE(x, y) = \frac{1}{s_{qs}^2} \left( \sum_i \sum_j |B(i, j) - C(x + i, y + j)| \right)
\]  

(3.2)
and the position with the minimum MAE is determined to be the top-left corner of the overlap region.

After we have determined the top-left corner of the overlap, we denote the remaining portion of the image to the right and below the top-left corner point as the greater overlap region. In the next chapter, we will discuss refining the greater overlap region and removing elements which may not be common in both images.
4 NON-COMMON ELEMENT REMOVAL

This chapter will focus on the portion of the overlap detection algorithm which attempts to remove discrepancies between frames. These non-common elements are usually a result of camera angle geometry. Most of the problems arise from objects which are in the camera foreground, and consequently closer than the areas which are used to determine the greater overlap region. Figure 4 gives an illustration of how the geometry and distance to cameras can create an occlusion as well as effect objects that appear in one frame and not another. The objective of this section is to remove elements which are in the greater overlap region but do not appear the same in both frames and thus do not have a high correlation.

4.1 Algorithm

Similarly to the previous steps, the non-common element removal algorithm is applied in the preponderance domain and we do so after having first identified the greater overlap region using the approach in Chapter 3. To identify non-common elements, we divide the overlap region into equally-sized blocks whose dimensions are approximately $sqs/2$. It may not always be possible to divide the overlap region at exactly $sqs/2$ in size, so in such cases we look for the next closest uniform subdivision. Doing this allows the algorithm to achieve finer precision in detecting objects not seen by both cameras. It is important to not divide up the
Figure 4.1: In this example the hexagon would appear differently to each camera. Camera A would see a different side than Camera B, in a more complex object the shape and appearance may be completely different. The star is a foreground object that will reside in the greater overlap region, but is not common to both cameras. These objects should be removed as part of the Non-common Element Removal step.
blocks too small, since blocks whose dimensions are smaller than $8 \times 8$ seem to have too little structure information to allow for an accurate match. We then compare each sub-division in frame A with its corresponding sub-division in frame B. An example of the division process is shown in Figure 4.2. We compare the Mean Absolute Error (MAE) as given by (3.2) to the MAE from the greater overlap region to determine if the sub-divisions match; if not, the correlations are deemed too low and that block is removed from the overlap region. The output of this section gives us the ultimate result from the entire process which is a binary map the size of the first image containing a '1' where the preponderance number leads us to believe there is a significant match and a '0' where we believe they do not.

Figure 4.2: An example of how the sub-division process works. The dark region was determined not to match as part of the overlap detection process. The light region is sub-divided. The sub-divisions in frame A are compared with frame B.
5 RESULTS

This section will demonstrate the performance of the proposed preponderance bit domain scheme experimentally. The performance of the algorithm was measured by processing a collection of test databases and identifying the occurrence of a correct overlap detection. We declare a correct overlap detection occurred when the result is within 8 pixels of the predetermined correct overlap. Each test database was created to test different types of visual data sequences and explore the strengths and weaknesses of the proposed overlap detection algorithm.

5.1 Laboratory Experimentation

The following two experiments were performed to give some initial insight into how the proposed algorithm works on very simple and generally low frequency images. Since the algorithm matches using differences in frequency structure, the sequence of images was fairly simple for these initial tests. The intent of these experiments was to give us a starting point for later experiments and to explore the strengths and weaknesses of the algorithm.

5.1.1 Laboratory Experiment 1

The first experiment is used to evaluate the minimum number of pixels necessary to accurately identify the overlapping portions of two spatially separated frames. The sequences of frames shown in Figure 5.1 was captured from nine camera
positions separated by 13 centimeters apart and with a 6 degree change in the viewing angle. Each individual sequence was captured from the same relative real-world scene, and it is intended to model a true environment by simulating the different viewing angles for multiple cameras. In each image frame, the overlapping region was uncontrolled and varied from nearly the entire image to less than half the image. Using a matching window size that is 21% of the total image size (a $40 \times 40$ window in the preponderance domain) an accurate match was made 72% of the time. By increasing the relative window size to 27% (a $45 \times 45$ window in the preponderance domain), we increase the success rate to 86%.

This experiment is particularly insightful because it allows for a comparison with forthcoming experiments. The frame correlations in this experiment were fairly simple compared to the video frames used in later experiments; the important aspect of these frames, however, is that the amount of structural information in them is generally low. Effectively, this experiment gives us a lower bound on what is to expect for the subsequent experiments.

5.1.2 Laboratory Experiment 2

A second experiment was performed to characterize the number of pixels necessary to properly identify the overlap between an original image and that image with added Gaussian noise. The images used were those from Laboratory Experiment 1. The minimum number of pixels necessary to make an accurate match for differ-
Figure 5.1: Captures from each camera position used in Experiment 1
ent noise levels is shown in Figure 5.2(a). We see from the figure that the proposed
detection scheme is moderately resilient to additive Gaussian noise. In most in-
stances, the preponderance bit detection algorithm requires less than a $25 \times 25$ PN
block or about 8.3% of the 800x600 image. As the variance of the Gaussian noise
increases, at some point the number of blocks required for accurate detection must
also increase. We believe that the number of pixels required to accurately detect
an overlap is related to the useable structure in the preponderance domain. This
hypothesis is supported by Figure 5.2(b) which shows the histogram of the block
dimensions required for correct detection over a $(0, 0.3)$ range of Gaussian noise
variances. By observing the histogram, it is apparent that most detection occurs
in clusters of preponderance block sizes; for example, the first cluster is between
152 and 216, the second between 300 and 408. By comparing this clustering be-
havior to the spatial domain structure of the image, we can see how that structure
affects detection in the bit-domain as the detection window is expanded. As the
two images begin to differ from each other more greatly due to higher levels of
additive noise, more structure and consequently more PN blocks are needed to
accurately detect the overlap. The detection window must then expand until it
gathers enough information to make an accurate decision. Of course, some parts
of the image may not provide enough information to counteract the increase in
variance; these areas usually have little high frequency content and result in pre-
ponderance block sizes where little overlap detection occurs: i.e., the blank spaces
in the histogram of Figure 5.2(b). Interestingly, we have also noticed that the number of blocks used in detection roughly follows the relative entropy for the pixels that map to the regions within the image where the bitstream blocks are being evaluated for overlap. By comparing the histogram to the relative entropy as shown in Figure 5.2(b), we notice that although the magnitudes are very different, the histogram follows the basic shape of the relative entropy. For example, around the 300 and 350 pixel block sizes, the relative entropy has a spike and the histogram seems to follow it. The fact that the histogram follows the relative entropy curve further indicates the utility of bit-domain PBD for extracting information about spatial structure of images.

5.2 MERL Database Experiments

The Mitsubishi Electric Research Laboratories (MERL) provides a database of multi-view video sequences [12]. The following experiments use 3 of the sequences provided which are different real-world environments consisting of 8 different camera views for each different environment shown in Figures 5.3, 5.4, 5.5. The cameras were equally spatially separated, and consequently had similar shifts in the actual video captured. In the following experiments, we compared the first 50 frames of camera 0 to the first 50 frames of camera 1 through 7.
Figure 5.2: Results from Laboratory Experiment 2 showing (a) block dimensions required for match versus noise variance, and (b) comparison of block histogram versus relative entropy.
Figure 5.3: Frame 1 from each camera of the Ballroom Sequence
Figure 5.4: Frame 1 from each camera of the Exit Sequence
Figure 5.5: Frame 1 from each camera of the Vassar Sequence
5.2.1 MERL Experiment 1

The first experiment using the MERL sequences examines how the number of pixels used in determining the overlap affects the accuracy of the match. Specifically, the matching window size was varied in order to demonstrate how the window size affects the accuracy of the greater overlap detection—the non-common element removal will be discussed later. Again, we declare an accurate match has been made whenever the detection scheme calculates the overlap to be within 8 pixels of the true overlap. The three sequences of the MERL database exercise the proposed algorithm over a diverse set of inputs. The amount of pixels in a shift in the three different sequences varies which tests how the algorithm performs over small and large amounts of overlaps.

The results in Tables 5.1, 5.2, 5.3 show the algorithm performs well overall. When using a window size of $20 \times 20$ or greater in the preponderance domain, an accurate detection was achieved 91% of the time. By observing the results in the tables and comparing their respective sequences, a few interesting patterns begin to develop that provide insight into how well the algorithm may perform in general. From Figure 5.3, we can see the frames from the Ballroom sequence generally have higher frequency content than the frames from the other two sequences. Consequently, the performance of the algorithm on the Ballroom sequence peaks somewhere between 20 and 30 frames. We believe this is a result of how the
Table 5.1: Percentage of Correct Detection for the Ballroom Sequence.

<table>
<thead>
<tr>
<th>Ballroom Camera</th>
<th>Matching Window Size</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>10</td>
</tr>
<tr>
<td>1</td>
<td>0.32</td>
</tr>
<tr>
<td>2</td>
<td>0.46</td>
</tr>
<tr>
<td>3</td>
<td>0.56</td>
</tr>
<tr>
<td>4</td>
<td>0.36</td>
</tr>
<tr>
<td>5</td>
<td>0.86</td>
</tr>
<tr>
<td>6</td>
<td>0.44</td>
</tr>
<tr>
<td>7</td>
<td>0.82</td>
</tr>
</tbody>
</table>

higher frequency content is represented in the preponderance domain and more specifically how the structure is represented. As more blocks are added in the matching attempt, the structure which probably begins to appear more random, becoming harder and harder to match. This gives us performance which improves as more blocks are added to the detection process to a certain point, after which adding more blocks begins to add more noise than structural information.

A contrasting example can be seen in the Vassar and Exit sequences. In Figures 5.4 and 5.5, it becomes apparent that the frames from these two sequences have a lot less high frequency content and much more rigid structure than the Ballroom sequence. Consequently, the performance generally improves as the
Table 5.2: Percentage of Correct Detection for the Exit Sequence.

<table>
<thead>
<tr>
<th>Exit Camera</th>
<th>Matching Window Size</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>10</td>
</tr>
<tr>
<td>1</td>
<td>0.02</td>
</tr>
<tr>
<td>2</td>
<td>0.94</td>
</tr>
<tr>
<td>3</td>
<td>0.94</td>
</tr>
<tr>
<td>4</td>
<td>0.84</td>
</tr>
<tr>
<td>5</td>
<td>1.00</td>
</tr>
<tr>
<td>6</td>
<td>0.44</td>
</tr>
<tr>
<td>7</td>
<td>0.48</td>
</tr>
</tbody>
</table>

matching block size becomes larger and larger. A lot of our frequency information is obtained in the edges of the images, as can be seen in Figure 3.3. Since the edges of the Vassar and Exit sequences are well-defined, the structure and frequency information are similarly well-defined. This allows the algorithm to consistently improve its performance as the block size increases, so it is not penalized as is the case with the Ballroom sequence.

In these results, the performance is fairly similar between all the cameras. The results show there is no significant drop-off in performance as the space between camera 0 and the camera of interest increases. This leads to the conclusion that the algorithm is fairly resilient to the problems introduced by differing camera
Table 5.3: Percentage of Correct Detection for the Vassar Sequence.

<table>
<thead>
<tr>
<th>Vassar Camera</th>
<th>Matching Window Size</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>10</td>
</tr>
<tr>
<td>1</td>
<td>0.64</td>
</tr>
<tr>
<td>2</td>
<td>0.34</td>
</tr>
<tr>
<td>3</td>
<td>0.06</td>
</tr>
<tr>
<td>4</td>
<td>0.90</td>
</tr>
<tr>
<td>5</td>
<td>0.64</td>
</tr>
<tr>
<td>6</td>
<td>0.50</td>
</tr>
<tr>
<td>7</td>
<td>0.82</td>
</tr>
</tbody>
</table>

angles, such as translation, pose and occlusion. For all of the MERL sequences, a window size of between 20 and 30 appears to result in the most reliable detection.

5.2.2 MERL Experiment 2

A second experiment has also been performed using the MERL images, the results of which are summarized in Table 5.4. Comparing images from cameras 1 through 7 with images from camera 0 of the Ballroom sequence, we can measure how well the algorithm removes objects which appear in one cameras field of view and are absent in the other cameras field of view. Specifically, using a window size of 25, we look at how the mean squared error of the estimated overlap region is reduced.
Table 5.4: MSE Improvements for the Ballroom sequence.

<table>
<thead>
<tr>
<th>Camera</th>
<th>Ratio of MSEs</th>
<th>Percentage Kept</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.18</td>
<td>0.72</td>
</tr>
<tr>
<td>2</td>
<td>0.28</td>
<td>0.63</td>
</tr>
<tr>
<td>3</td>
<td>0.41</td>
<td>0.53</td>
</tr>
<tr>
<td>4</td>
<td>0.45</td>
<td>0.46</td>
</tr>
<tr>
<td>5</td>
<td>0.36</td>
<td>0.49</td>
</tr>
<tr>
<td>6</td>
<td>0.40</td>
<td>0.45</td>
</tr>
<tr>
<td>7</td>
<td>0.39</td>
<td>0.46</td>
</tr>
</tbody>
</table>

as the non-common elements are removed. The ratio of MSEs is the new MSE divided by the old MSE. This gives us a comparison of how much the removal of non-common elements reduced the error between the two images. “Percentage Kept” as labeled in Table 5.4 is an indicator of how much area of the images have been removed for being relatively uncorrelated to each other.

The results in Table 5.4 illustrate what can be observed and is rather intuitive. As the distance between the cameras increases, the correlations between the two images begins to lessen. The algorithm responds by removing more and more, lowering the ratio of MSEs and consequently lowering the percentage kept. Since removing the entire image would produce the lowest MSE, it is important to eval-
uate the MSE alongside the Percentage Kept. Ideally, the algorithm should keep as much as possible of the two images and remove only what is necessary to reduce the MSE. However, since the non-common element removal works in blocks, there will be blocks which contain both common and non-common elements. The decision to remove these blocks can be adjusted by changing the amount of similarity necessary for a match to be confirmed. Overall, these results are impressive as we are able to keep at least half the image and reduce the overall MSE quite a bit. Figure 5.6 is a demonstration of the power of the non-common element removal. The foreground objects which do not overlap consistently are almost completely removed and the similar background objects are kept.
(a) Camera 0  (b) Common Elements  (c) Non-common Elements

(d) Camera 4  (e) Common Elements  (f) Non-common Elements

Figure 5.6: An example of the performance of the non-common element removal algorithm
5.2.3 MERL Experiment 3

The final experiment using the MERL database of test sequences is an exercise to explore the impact of using different approaches for the overlap detection phase. The portion of the algorithm that we focus on here is the method used for choosing a value for gray-level slicing. As discussed above, the purpose of the gray-level slicing is to determine which preponderance values relate to low frequencies and which relate to high frequencies. Since some images will consist of more high frequency and some will have a lot more lower frequencies we wish for this value to be adaptive to the image. To measure the effectiveness of the algorithm we compared results of the proposed method which uses a $k$-means algorithm to two different approaches; one using mean of the preponderance values and another using the median bootstrap method described in [2]. This should give us a good idea of performance since the proposed algorithm will be compared to both a simpler and a more complex choosing method.

This experiment uses the first 50 frames from the 3 sequences in the MERL database of test sequences. Again, a correct detection is determined to be when the algorithm detects the overlap to be within 8 pixels of a correct overlap. The detection window size was selected to be $25 \times 25$ block in the preponderance domain.

By observing the results in Table 5.5, we can see that both the $k$-means and
Table 5.5: Comparison of results using the mean, $k$-mean and bootstrap methods.

<table>
<thead>
<tr>
<th>Camera</th>
<th>Ballroom</th>
<th></th>
<th>Exit</th>
<th></th>
<th>Vassar</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Mean</td>
<td>$k$-Mean</td>
<td>Bootstrap</td>
<td>Mean</td>
<td>$k$-Mean</td>
<td>Bootstrap</td>
</tr>
<tr>
<td>1</td>
<td>0.86</td>
<td>0.94</td>
<td>0.98</td>
<td>0.26</td>
<td>0.94</td>
<td>0.36</td>
</tr>
<tr>
<td>2</td>
<td>0.92</td>
<td>0.96</td>
<td>0.88</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
</tr>
<tr>
<td>3</td>
<td>0.98</td>
<td>1.00</td>
<td>0.98</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
</tr>
<tr>
<td>4</td>
<td>0.96</td>
<td>0.96</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
</tr>
<tr>
<td>5</td>
<td>0.86</td>
<td>0.96</td>
<td>0.90</td>
<td>1.00</td>
<td>0.84</td>
<td>1.00</td>
</tr>
<tr>
<td>6</td>
<td>0.80</td>
<td>0.98</td>
<td>0.98</td>
<td>0.96</td>
<td>0.58</td>
<td>0.90</td>
</tr>
<tr>
<td>7</td>
<td>0.96</td>
<td>1.00</td>
<td>1.00</td>
<td>0.24</td>
<td>0.20</td>
<td>0.16</td>
</tr>
</tbody>
</table>

Bootstrap estimation are much better than using the mean. The $k$-means method has the best results in the Vassar scene and roughly similar performance to the bootstrap method in Ballroom and then falls behind for the Exit scene. However, the bootstrap method requires many more calculations since the number of iterations required to get a good estimate of the median is over 100. Since the data is one-dimensional, the $k$-means approach requires many less computations. Overall, the performance improvement by the bootstrap method does not warrant the increased computational complexity. Nonetheless, the experiment is important because it shows the proposed algorithm is gives very similar results to more computationally complex approaches.
5.3 Quantization

The final experiment is a test of the algorithm’s robustness and a measure of how quantization affects performance. To measure the impact of quantization on the performance, the quantization was increased by a factor of 2 and decreased by a factor of $\frac{1}{2}$ and the results measured. The level of quantization used by the baseline compression scheme is determined in (2.2). To adjust the level of quantization we can add a scaling value $q$ to this equation as follows:

$$\hat{T}(u, v) = \text{round} \left[ q \times \frac{T(u, v)}{Z(u, v)} \right]. \quad (5.1)$$

This experiment uses the Ballroom sequence from the MERL database. As in the previous experiments, the first 50 frames of the sequence from camera 0 were compared to the first 50 frames from cameras 1 through 7. A correct match is determined to occur when the overlap estimate is within 8 pixels of the correct overlap.

By observing the results in Tables 5.6 and 5.7, it would appear that the quantization has a negligible effect on the overall performance. By comparing these results to those of the previous MERL experiments in Table 5.1, they appear to be consistent with the non-scaled quantization. The mean performance for the different scale factors: $q = \frac{1}{2}$, $q = 1$, and $q = 2$, respectively are 0.83, 0.84, and 0.81. These numbers support the theoretical premise the algorithm was based upon. As explained in Chapter 3, the overlap algorithm divides the preponderance domain
Table 5.6: Percentage of correct detection with increased quantization. \( (q = \frac{1}{2}) \)

<table>
<thead>
<tr>
<th>Ballroom Camera</th>
<th>Matching Window Size</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>10</td>
</tr>
<tr>
<td>1</td>
<td>0.58</td>
</tr>
<tr>
<td>2</td>
<td>0.60</td>
</tr>
<tr>
<td>3</td>
<td>0.46</td>
</tr>
<tr>
<td>4</td>
<td>0.66</td>
</tr>
<tr>
<td>5</td>
<td>0.44</td>
</tr>
<tr>
<td>6</td>
<td>0.34</td>
</tr>
<tr>
<td>7</td>
<td>0.68</td>
</tr>
</tbody>
</table>

into low and high numbers of bits. The number of bits produced by an \( 8 \times 8 \) block is loosely related to the amount of high frequency. The change in preponderance values across the entire image should be proportional to the amount of quantization applied, leaving us with similar results as long as the quantization is not so extreme that it eliminates too much frequency information.
Table 5.7: Percentage of correct detection with decreased quantization. \((q = 2)\)

<table>
<thead>
<tr>
<th>Ballroom Camera</th>
<th>Matching Window Size</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>10</td>
</tr>
<tr>
<td>1</td>
<td>0.48</td>
</tr>
<tr>
<td>2</td>
<td>0.32</td>
</tr>
<tr>
<td>3</td>
<td>0.44</td>
</tr>
<tr>
<td>4</td>
<td>0.64</td>
</tr>
<tr>
<td>5</td>
<td>0.60</td>
</tr>
<tr>
<td>6</td>
<td>0.26</td>
</tr>
<tr>
<td>7</td>
<td>0.76</td>
</tr>
</tbody>
</table>

6 Conclusions and Future Work

In this thesis, we have introduced a novel approach for detecting both spatial frame overlap and non-common foreground objects directly from the compressed motion-JPEG bitstreams of multi-view distributed sensor networks. The proposed algorithm for detection relies on the preponderance of bits used to encode each \(8 \times 8\) block of pixels in the baseline JPEG compression framework. The preponderance scheme introduced here to perform these tasks is intended to support low power, low complexity distributed video coding and is designed to be simple enough to operate in a remote sensing video node, using information captured passively from
nearby sensors in the network.

There are several avenues of research to pursue in future work. The focus of this project has been to develop an algorithm to identify correlations between spatially separated cameras in a multi-view distributed sensor network. The next obvious step from here is using the extracted overlap information to encode frames from one sensor node conditionally with respect to the correlated information sent from the other node. Wyner-Ziv encoders have been widely researched as a solution to exploiting correlations between frames. One of the fundamental difficulties of these encoders is the poor side information at the decoder. The information extracted from the bitstream can be potentially used to address this difficulty. At the same time, since we have introduced a method for extracting useful information from the compressed bitstream it would be very beneficial to be able to extend some of the algorithm’s operations to continue using the bitstream in order to exploit other redundancies.

Another useful line of inquiry to pursue is the possibility of performing at least a portion of image processing in the compressed bitstream domain. This research has looked into some of the possibilities of working in the bitstream and many new advantages can be explored. By observing Figure 3.3(c) it becomes apparent that the output from gray-level slicing step in the overlap detection algorithm is very similar to that of an edge detector. The implementation of edge detectors in the compressed bitstream domain to reduce computational complexity in both the
amount of data processed and removing the necessity of decoding the bitstream is only a very small step in the wider realm of possibilities to explore.

Thus far, from our experimental results the overlap detection and non-common foreground object removal in this work has appeared to operate effectively, providing results which often exceed 90% in accuracy. With a few small adjustments the encoder can be adapted to identify and exploit correlations between consecutive frames, significantly improving accuracy. By using frame-to-frame correlations along with sensor-to-sensor correlations, we can, in theory, achieve a lower bitrate and maintain a higher quality in distributed coding systems.
REFERENCES


