Spring 2015

# Novel use of video and image analysis in a video compression system

John David Stobaugh
*University of Iowa*

# NOVEL USE OF VIDEO AND IMAGE ANALYSIS IN A VIDEO COMPRESSION SYSTEM

by

John David Stobaugh

A thesis submitted in partial fulfillment
of the requirements for the Master of Science
degree in Electrical and Computer Engineering in the
Graduate College of
The University of Iowa

May 2015

Thesis Supervisor: Professor Gary Christensen

Graduate College
The University of Iowa
Iowa City, Iowa

CERTIFICATE OF APPROVAL

_____

MASTER'S THESIS

_____

This is to certify that the Master's thesis of

John David Stobaugh

has been approved by the Examining Committee for
the thesis requirement for the Master of Science degree
in Electrical and Computer Engineering at the May 2015 graduation.

Thesis Committee: _____
Gary Christensen, Thesis Supervisor

_____
Edward Ratner

_____
Milan Sonka

To everyone who will be affected by segmentation and object analysis in the years to come.

## ACKNOWLEDGEMENTS

Without the support of my friends and family, none of this work would have been possible. This research has been made possible through years of building both imaginary and concrete connections between concepts. I thank my wife, Zheng Li Stobaugh, for all her understanding and my research advisor, Ed Ratner, for all his wisdom.

## ABSTRACT

As consumer demand increases for higher quality video at lower bit-rates, so does the need for more sophisticated methods of compressing videos into manageable file sizes. This research attempts to address this demand while still maintaining reasonable encoding times. Modern segmentation and grouping analysis are used with code vectorization techniques and other optimization paradigms to improve quality and performance within the next generation coding standard, High Efficiency Video Coding. This research saw on average a 50% decrease in run-time by the encoder with marginal decreases in perceived quality.

## PUBLIC ABSTRACT

People like to watch videos. Videos need to be made smaller to be transmitted to the viewer. Keeping videos compact without destroying how they look is challenging. To keep videos compact and looking clear requires innovative techniques and strategies. This research looks at some of those strategies and explores new ones as well. Among other things this research explored automating how a video can be broken into its separate parts, improving the speed at which videos can be compressed, and making videos clearer. If done correctly, then the average consumer should have no idea that anything has happened other than their video download speed has improved. For example if the average college student's cell-phone data plan only allows them to watch 120 videos in a month, this research explores ways of allowing that same student to watch 240 videos with the same data plan.

# TABLE OF CONTENTS

# LIST OF TABLES

# LIST OF FIGURES

## Chapter 1 Introduction to Video Coding

Video compression has been an area of growing focus over the past forty years and has been a driving force in the growth of the internet and many electronic viewing devices. To achieve compression, redundant and unnecessary information must be removed in a way that does not damage human visual quality. While the idea of removing redundant information to reduce overall storage/transmission costs is nothing new, the methods of accomplishing these goals are still a subject of research and development.

For video applications, compression comes in three forms, data reduction, recycling, and transformation. Data reduction refers to how information is discarded when unnecessary for the end product. This can be with and without actual loss of information but the transmitted product will be represented with less bits. Data recycling refers to the reuse of information. For example a reference is transmitted rather than the original source information. The axis of the reference can vary and the fullness of data may be complete or partial. Data transformation happens when the same information is moved from one space to another using some relationship. The transformation is generally invertible by either deterministic or predictive methods. In practice there is often a blending of these three ideas between techniques but they represent the core of making more data into less data.

The motivation for video compression is most easily explained by describing the data requirements for a small, uncompressed yuv420p video file. A video that has a frame resolution of 320x240 pixels, a framerate of 30 frames/second, and a duration of 1 minute has a file-size of 320x240x30x60x1.5 = 207,360,000Bytes or 197.75Megabytes. Now if this were the same video but with a resolution of 1920x1080 pixels then the file-size would

be 27 times larger. It becomes clear that compression is necessary for any reasonable transmitting application.

While compression can vary in many ways, it can be measured empirically by signal loss and file-size. Alternative measurements exist that focus more on perceived human quality but these are subjective in nature. What is considered reasonable is largely based on application and transportation limitations [3].

The definition of what is reasonable has continued to evolve over the past forty years. A 320x240 video at 2Mbits/second viewed on a desktop was considered reasonable twenty years ago. Now the system is considered inadequate if a user cannot watch the same video at the same 2Mbits/second but at 1920x1080 on their cellphone. Looking forward, different end uses will continue to evolve to higher resolutions on a broader array of personal mobile devices.

## Chapter 2 Video/Image Basics

### 2.1 Color Space

The RGB is likely the most known color space but for many technical applications there are better options [4]. It can be transformed into other color models that have many positive properties. For video compression, the most common format is YUV which is Y-luminescence and UV- chrominance. YUV is an invertible transform of the RGB color space and is done for both technical and historical reasons. The historic ties lie in backward compatibility to black and white televisions which would only process the luminescence portion of the signal. From a modern technical perspective, the Human Visual System (HVS) is most sensitive to the Y component, therefore more bits are spent there. The separation allows for more emphasis to be given to the Y component of the U and V.

$$\begin{pmatrix} Y \\ U \\ V \end{pmatrix} = \begin{pmatrix} 0.299 & 0.587 & 0.114 \\ -0.147 & -0.289 & 0.436 \\ 0.615 & -0.515 & -0.100 \end{pmatrix} \begin{pmatrix} R \\ G \\ B \end{pmatrix}.$$

*Figure 2-1Matrix representation of an RGB to YUV conversion*

### 2.2 Quality Metrics

Metrics of determining video quality can be broken into two categories: subjective and objective [3]. Subjective assessment generally use human opinion to rate video quality, and while costly, it is often considered to be the most accurate. Objective methods vary in sophistication with some only taking the Peak Signal to Noise Ratio (PSNR) while others use Structural Similarity (SSIM) metrics [2]. Objective metrics having difficulty distinguishing between different types of error. For example a sharp

spike at a single location in a frame may be viewed the same as an error uniformly distributed over the entire frame. To a visual observer the frame with the sharp spike is of lower quality while the frame with the uniform error appears reasonable. Both styles of quality evaluation have their uses and are highly application dependent.

$$PSNR = 10\log_{10}\left(\frac{255^2}{E_{\text{ms}}}\right)$$

*Figure 2-2Formula showing PSNR calculation*

## 2.3 Framerate

An easily overlooked aspect of video compression is the framerate of the video being encoded. This is the rate at which unique images are being recorded and is often measured as frames per second. The significance lies in its relationship to the HVS. Changes to framerate can have an effect on the smoothness, speed, and bit-rate of a video. As the framerate increases, the amount of raw data increases and the difference between frames decreases. This can be beneficial to fast paced motion such as sports. For a slow scene of a conversation between two people, a user likely could not tell between 30 and 60fps.

## 2.4 Progressive and Interlaced

An often poorly understood distinction in the purchase of consumer electronics is the difference between progressive and interlaced video. A progressive video is encoded as it will be perceived by the end user meaning every frame is a whole image. An interlaced video is composed of alternating fields which are spaced at half the framerate [12]. Every frame can be considered to have two fields which are refreshed alternatively (see Figure 2.3). The distinction's importance lies in the analysis of the video as a video

with distinct frames is different than that composed of two fields. There are many techniques to handle this and the chosen technique is application dependent.

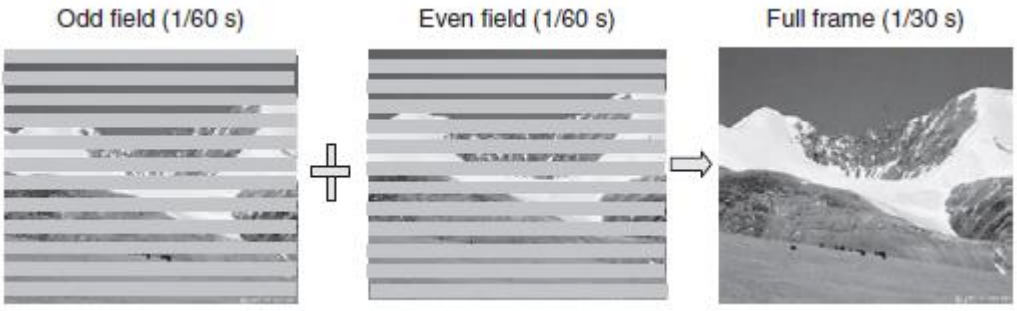



*Figure 2-3 Visual representation of two interlaced fields composing a frame*

## Chapter 3 Lossless/Lossy Common Compression Strategies

Compression in regard to video comes in two flavors: lossless, which primarily uses traditional methods to compress information without losing any information and lossy, which deals with trying to keep only necessary information with varying degrees of acceptable loss [4]. Modern schemas use a combination approach but the majority of the examination in this research dealt with lossy compression.

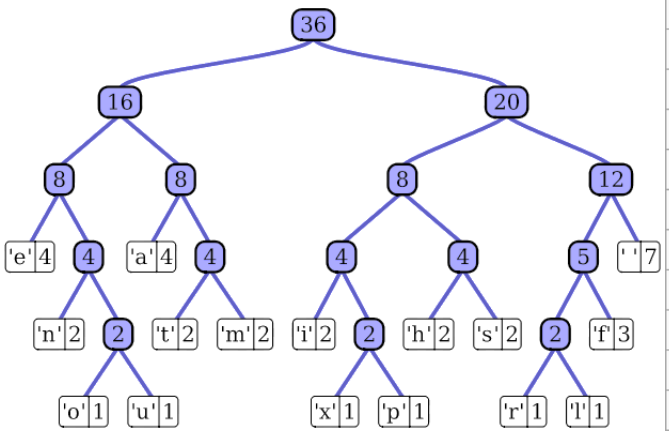
### 3.1 Lossless Encoding

Lossless coding methods are common to many modern applications, but they play an especially important role in image and video compression. They remove statistical redundancies from the final data stream. Information with higher statistical occurrences within a stream is represented using fewer bits. Less frequent occurring information is encoded using more bits. For streams with high amounts of redundancy, this can greatly reduce the information being transmitted.

Entropy coding is a lossless encoding method that uses statistical information of an entire signal to compress the data. Some form of this statistical information is transmitted with the data stream to allow for decoding. The statistical information can be adaptable and may change as the signal continues to be received.

A common technique in entropy encoding is called Huffman Coding, named after David A. Huffman (see Figure 3.1). This algorithm uses the probability distribution of symbols to create a variable length code to represent data [11]. Items with lower occurrence frequency are given more bits to code while symbols with high occurrence are encoded using fewer bits. Each code is unique and cannot be a prefix for another code. The codes are stored and passed long with the data stream for decoding at the receiving

end. This method is guaranteed to provide optimal symbol code and it is likely the easiest to learn and understand.




| Char | Freq | Code |
|---|---|---|
| space | 7 | 111 |
| a | 4 | 010 |
| e | 4 | 000 |
| f | 3 | 1101 |
| h | 2 | 1010 |
| i | 2 | 1000 |
| m | 2 | 0111 |
| n | 2 | 0010 |
| s | 2 | 1011 |
| t | 2 | 0110 |
| l | 1 | 11001 |
| o | 1 | 00110 |
| p | 1 | 10011 |
| r | 1 | 11000 |
| u | 1 | 00111 |
| x | 1 | 10010 |

*Figure 3-1 Huffman example*

More advanced modified Huffman algorithms exist that take advantage of preexisting knowledge of the symbols or local probability within a signal. These techniques have their uses but, in practice, a Huffman coding can be performed after other encoding techniques have been applied to try and get the most compression possible.

Another common form of entropy encoding is called Arithmetic Coding. It differs from Huffman Coding in that rather than coding each symbol with a unique code, Arithmetic Coding uses a single decimal number range [0.0,1.0) to represent all of the symbols present [16]. The probability table is transmitted with the encoded signal to allow for decoding of the decimal number. Arithmetic Coding also differs in that the

stream cannot be decoded until the whole signal is present or at least until the earliest symbols are guaranteed.

Arithmetic Coding has several key trade-offs with Huffman Coding that make it more useful in certain applications. For example, the decimal code words allow for better compression for small alphabets [7]. The downside is that it is more computationally intensive than Huffman.

Another form of lossless compression is called Run-length Coding. This schema takes advantage of repeat occurrences of symbols within a signal to reduce its size. Sequences are generally coded as the symbol and the number of times it repeats. This can change between implementations of the algorithm but the common thread is to reduce sequences of repeat values into shorter terms. The resulting coding can also be further compressed using other techniques such as Huffman Coding.

### 3.2 Lossy Encoding

Until now, the only methods discussed have been lossless methods of compression. While these forms of compression are useful, in most video applications they do not provide enough reduction in signal size. For example the reduction can often be by a factor of 3 or 4 but in real world applications this is still not nearly enough. The goal then is to modify the original signal to reduce its overall size without affecting the visual clarity of the video. The alternative is to use lossy compression methods which allow for a portion of the data to be sacrificed in order to achieve necessary signal sizes, while retaining visual clarity as perceived by human observers. Signals may be truncated, filtered, or transformed such that recovering the original signal is not possible. The goal is to merely recover enough of the signal so that it is still useful for its application.

The simplest of the lossy compression methods is quantization. This is the transform from one set to a smaller set where there is a many-to-few relationship. A simple example would be the rounding of a fractional number set. Many decimals may point to a single integer and there is no way to distinguish them once the conversion is made. The transform can be uniform or non-uniform depending on the input's distribution and the application.

In video compression, quantization is typically applied to the final transform coefficients. It generally gives greater weight to the lower frequency components while transforming many of the higher frequency components to zero. The quantization matrix is typically standardized but newer encoders allow for custom matrices specified by the user.

### 3.3 Transforms, Wavelets, DCT, Hadamard

While both lossless and lossy methods are important, when run on an unmodified video signal they often do not produce the best possible compression. The strategy should be instead to transform the data in an invertible way such that when compression techniques are used on the signal it will provide the best results. The compressed transform signal then can be transmitted to its destination. Once at the destination the modified signal can be uncompressed and inverted to retrieve the encoded signal. Beyond maximizing compression, many transforms offer beneficial filtering properties that remove data the end user will not notice. There are many transforms and they vary with utility and complexity. The application generally dictates which transform is used.

#### 3.3.1 Discrete Cosine Transform

Discrete cosine transform (DCT) is of particular importance to modern video

compression. It allows for greater compression but can also cause ringing artifacts at sharp transitions within a frame. There are methods of addressing these artifacts to limit quality degradation. While computationally demanding when calculated in a naive fashion, matrix decomposition techniques exist that improve the computational time making this transform viable for more applications. Within an actual encoder, the block matching residual is first transformed to the DCT before later compression steps occur [6].

The DCT equation (Eq. 1) computes the i,j$^{th}$ entry of the DCT of an image.

$$D(i,j) = \frac{1}{\sqrt{2N}} C(i)C(j) \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} p(x,y) \cos\left[ \frac{(2x+1)i\pi}{2N} \right] \cos\left[ \frac{(2y+1)j\pi}{2N} \right]$$

$$C(u) = \begin{cases} \frac{1}{\sqrt{2}} & \text{if } u = 0 \\ 1 & \text{if } u > 0 \end{cases}$$

*Figure 3-2 DCT equation*

### 3.3.2 Deblocking Filter

The Deblocking Filter is a video filter that helps to address the appearance of sharp edges between macroblocks in block based encoding. This filter helps to improve picture quality and is similar to block motion compensation [14].

### 3.3.3 Wavelets

Wavelets are a growing field of interest in video compression. Currently, these transform have only received limited commercial adoption in encoders. JPEG2000, which is an image compression standard, uses them oppose to DCTs, but it has not replaced the more widely adopted JPEG format which uses DCTs. There are many wavelet types that have different advantages [5].

### 3.3.4 Hadamard Transform

The Hadamard Transform is a technique that relies on simple addition and subtraction to create a strictly real matrix output. It has beneficial properties both in computation time and a greater tolerance to channel errors [23]. The transform is applied to the DCT coefficients of intra-coded macroblocks [21].

## Chapter 4 Encoding Standards

A video encoding standard refers to the idea that for any given video encoded according to that standard, that video can be decoded to a watchable format by a device that adheres to the same standard. The encoder is given a great deal of flexibility in how its decisions are made, but the resulting output must adhere to the standard to be decodable. This standardization allows for common use among many devices and users while limiting the hindrance of innovation.

### 4.1 Modern Encoders

Modern video encoders employ a strategy that uses inter/intra frame referencing to be decodable for future viewing. Each frame is broken into a number of smaller blocks. Each smaller block will have individual motion vectors and coding information. The process starts with a key frame which only uses intra referencing so it is decodable without the need of any other frames. Inter and intra referencing is used at each subsequent frame in a sequence.

The technique of breaking a frame into smaller blocks has been common to the H.26x family of video coding standards since H261. While the underlying concept has remained intact, the tools that achieve compression have changed greatly. The latest version, H265, uses blocks of sizes up to 64x64 pixels which have more reference frames and greater motion vector ranges than any of its predecessors. In addition, these blocks can be broken up into even smaller sub-blocks.

The frame sub portions in H265 are referred to as Coding Tree Units (CTUs) ranging from sizes 64x64, 32x32, or 16x16 pixels. These CTUs can also be subdivided

into smaller blocks called Coding Units (CUs). These CUs provide greater flexibility in referencing different frame locations.

A common strategy for selecting a final CTU utilizes a quad tree, recursive structure. First, a CU's motion vectors and cost are calculated. Next, this CU is split into four equal parts where a similar examination happens for each. This subdividing and examining continues until the CUs are of size 4x4. Once the cost of each sub-block for all the viable motion vectors is calculated, they are combined to form a new CU candidate. This new candidate is then compared to its now sister CU and the one with the higher rate-distortion cost is discarded. This process continues until a final CTU is produced for encoding.

### 4.2 Frame Types and Reference Structure

The frame referencing structure can be both static and dynamic with the latter being the most common in more modern encoders. There are three frame types. Each frame type has different restrictions placed on their CTU referencing. Intra predictive frames (I-frames) are unique in that they are entirely self-contained and do not reference any other frame. I-frames are commonly referred to as key-frames as they are an access point into a video. The other two frame types are both inter-referencing frames where they use block information from other frames to be decoded. Predictive frames (P-frames) use block information from past frames. Bi-directional predictive frames (B-frames) use information from both past and future frames to achieve optimal data compression.

The referencing structure of these frames has changed between encoding standards with the more modern being the most flexible. The structure generally starts with an I-frame then proceeds to use a combination of P- and B-frames. Generally P- and B- frames

will only reference P- and I- frames. I-frames are used periodically to refresh the video and allow for skipping through the video by the end user.

4.3 H261/H263

H261 is viewed as being among the first commercially viable video compression standards and set the stage for subsequent standards [14]. It uses static macroblock sizes of 16x16 pixels for luma samples and 8x8 pixels for chroma samples. The DCT transform is used for the motion compensation and Huffman Coding is used for the entropy coding applied to the transform blocks. The standard is limited to only P-frames which limits its compression potential. Despite being outdated, H261 created the foundation that later encoding standards would build upon.

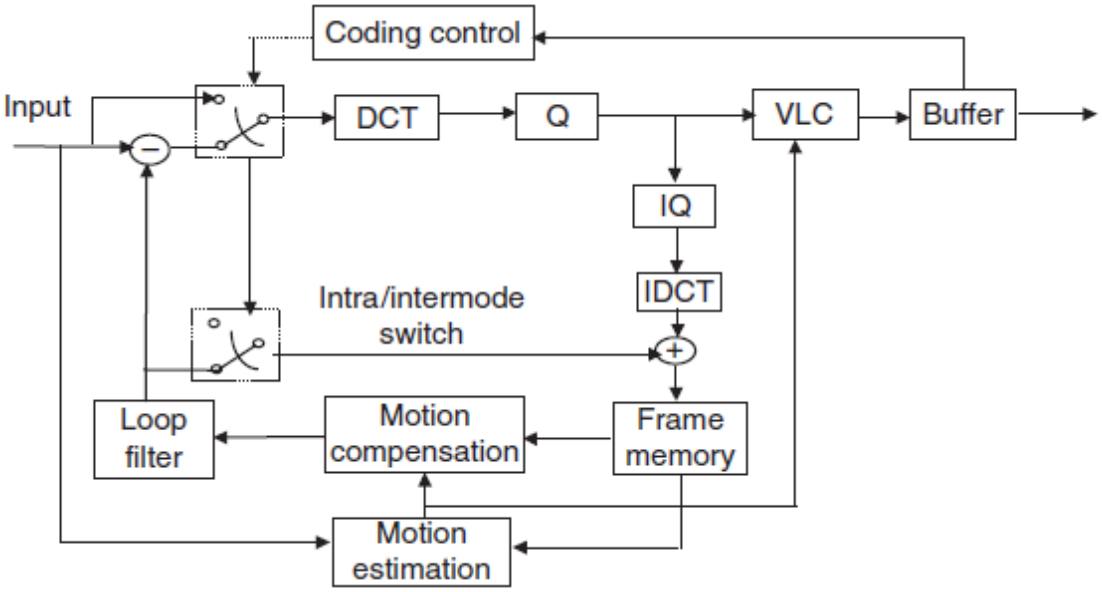



*Figure 4-1 Block diagram of H261 encoder*

H263 began as a low bit rate option for video telecommunication. Based on the design of H261, it extended features and added a greater degree of flexibility for motion

vectors, entropy coding types, and frame types [14]. Half-Pixel Accuracy was also introduced which improved accuracy in motion estimation. The H263 standard also introduced the use of PB-Frame (see Figure 4.2) referencing which increased compression potential allowing for one frame to reference a future and past frame.

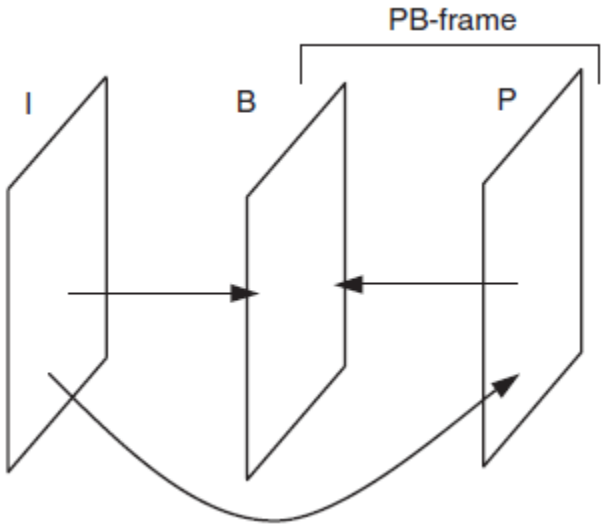



*Figure 4-2 PB-frame visualization*

## 4.4 MPEG2

MPEG2 is still a widely used standard. It is common in broadcast television and DVD's. It was introduced to address certain shortcomings of the MPEG1 standard with improved interlacing support. It has downloadable quantization matrices [13] with better scalability.

## 4.5 H264

The current market leader in video coding standards is H264. It differentiates itself from past encoders in the flexibility of its frame referencing structure, support for smaller macroblocks, the introduction of the network abstraction layer, and many other features that allow it claim a 50% reduction in bit-rates over past encoders [15]. These gains are

accomplished at the cost of greatly increased computational complexity. However, this cost has been mitigated with increasing computing performance.

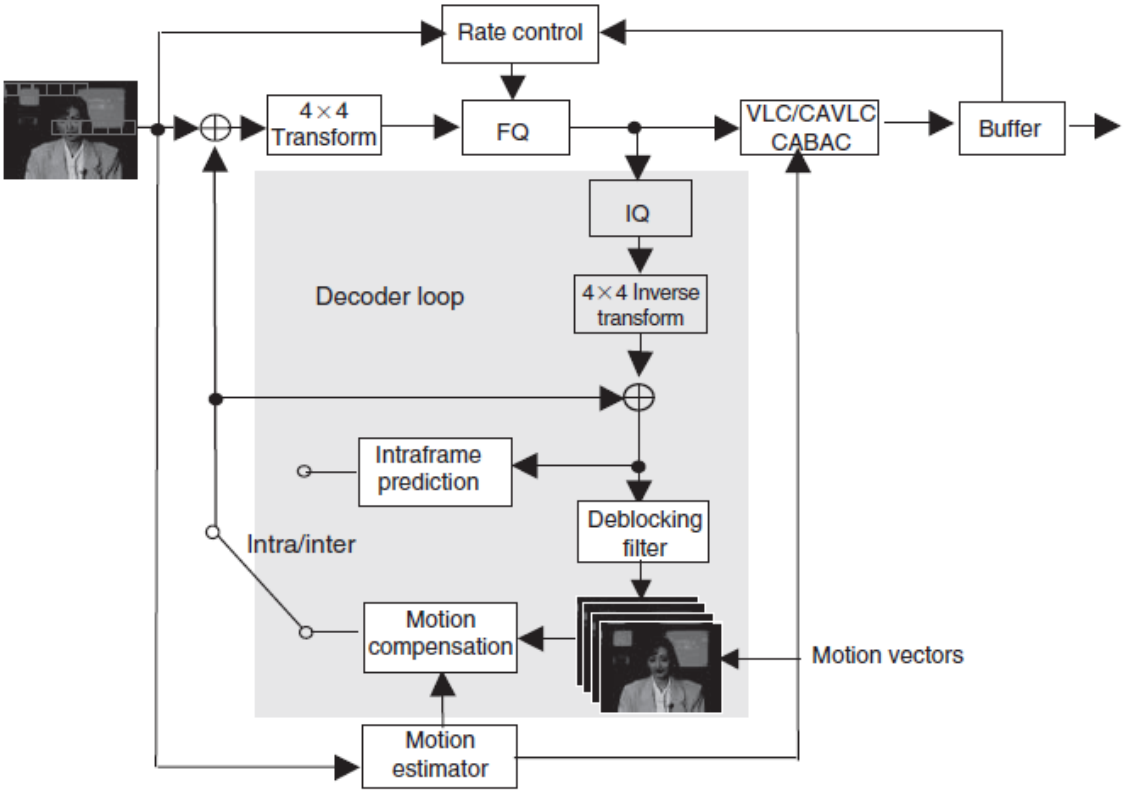



*Figure 4-3 H.264 block diagram*

## 4.6 H265

The most recent encoding standard in the H26* family is High Efficiency Video Coding (HEVC, H265). As with past standards, H265 addresses a number of issues hampering past encoders and better utilizes emerging hardware capabilities. H265 currently claims to maintain the video quality of H264 with half the number of bits being used. It also has higher resolution support and greater bit depths. 3D support has recently been added to H265 with the intention of also handling computer generated graphics.

The standard overall is very similar to its predecessor H264. One of the main contributions of the new standard is its Sample Adaptive Offset (SAO) filter. “SAO is a process that modifies the decoded samples by conditionally adding an offset value to each sample after the application of the deblocking filter, based on values in look-up tables transmitted by the encoder.” [1]

## Chapter 5 Segmentation and Object Analysis

Segmentation is grouping of like pixels using some metric over a window in an attempt to make the information present more meaningful. Pixels are grouped together such that the regions are homogeneous in some way. Methods may vary by process, by the characteristics used to determine homogeneity, and by how success is determined. The goal for any segmentation application is to make the information present more salient. There are many applications which draw upon segmentation and in those applications many approaches may be applied [19].

### 5.1 Thresholding

The simplest thresholding method is to use a single value to separate the pixels into two classes (see Figure 5.1). This method can be expanded upon by dynamically determining this threshold value. Another expansion is to try to vary the thresholding value for different parts of the image. This method can provide reasonable results in some applications but for many it is too simple in its decisions [8].

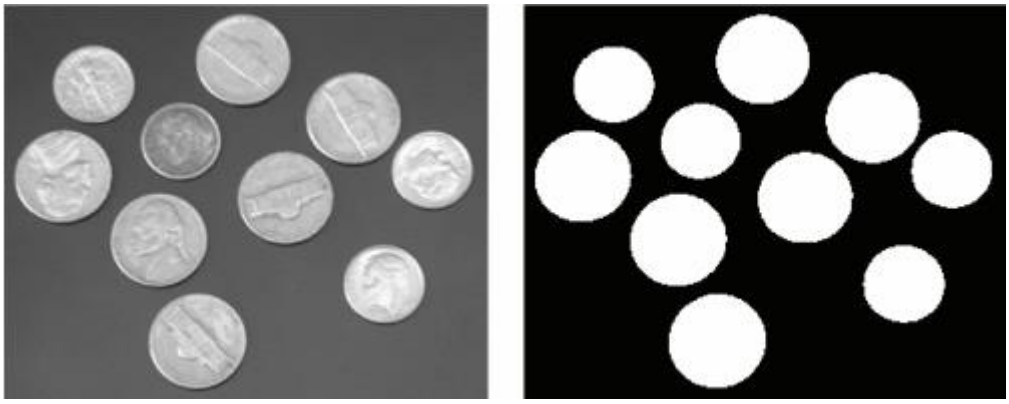


*Figure 5-1 Example of image thresholding*

5.2 K-Means Clustering

Another segmentation technique is k-means clustering which is an iterative technique that tries to separate n observations into k clusters based upon the mean of those clusters [22]. Once an iteration occurs where no observation changes cluster then the algorithm is considered finished.

5.3 Region-based

Region-based segmentation operates under the idea that pixels within a region will share common characteristics and can be grouped together by these commonalities [8]. Region-growing method of segmentation is a bottom-up approach where smaller regions are grouped into larger regions. Pixels start by being their own segment and are grouped depending on some distinguishing criteria or a group of criteria. This continues until no more segments can be grouped and the algorithm terminates. An example of Region-based segmentation being run on an image is shown in figure 5.2.

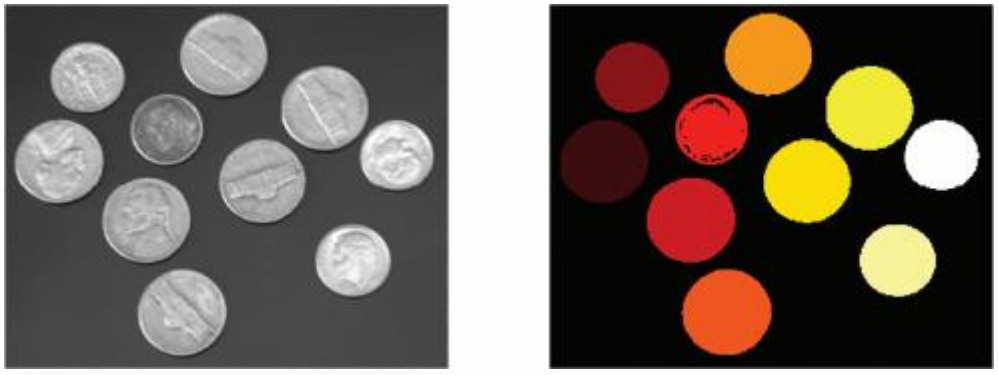


*Figure 5-2 Region-based segmentation example*

Alternatively, a split and merge technique can be used which is a top-down approach. Using a quad-tree structure, an image is split then merged into regions that are given two opportunities to merge in the algorithm (see Figure 5.3).

1. Define a logical uniformity predicate $P(R_i)$.
2. Compute $P(R_i)$ for each region.
3. Split into four disjoint quadrants any region $R_i$ for which $P(R_i) = \text{FALSE}$.
4. Repeat steps 2 and 3 until all resulting regions satisfy the uniformity criterion, that is, $P(R_i) = \text{TRUE}$.
5. Merge any adjacent regions $R_j$ and $R_k$ for which $P(R_j \cup R_k) = \text{TRUE}$.
6. Repeat step 5 until no further merging is possible.

*Figure 5-3 Region-based segmentation algorithm*

Many more techniques for segmenting exist, among them are histograms, graph partitioning, watershed, model based, multi-scales, and more. For all these segmentation techniques there are different trade-offs that must be considered before one is chosen for a particular application.

## 5.4 Segment Base Motion Estimation

Once segments have been generated on a sequence of frames, motion estimation on the segment correspondence between frames can be performed to provide even more information. A number of techniques exist to achieve segment correspondence including L1 norm minimization, block motion estimation, optical flow, motion models, and others [24]. It's worth noting that while the encoder may use some similar strategies in its block motion estimation, the goal is never to replace the encoder nor create a new codec but to simply augment the tools in place.

## 5.5 Research Implementation of Segmentation and Object Tracking

Additional analysis can be used during encoding to aid the encoder's decisions. The added analysis starts with the segmentation of a frame. Segmentation can be generated using a number of techniques which are known in the art. In the preferred embodiment used in this algorithm the edge detection based method is used. A sequence

of frames is analyzed to ascertain the areas of consistent inter frame movements which are labeled as objects for later referencing. A number of object tracking methods are known in the art. These objects are then passed into the encoder to aid in compression (see Figure 5.4-5.5-5.6).

The object tracking process is a multi-layered segmentation based on pixels, edges, time, and persistency. The segmentation across the time-axis offers the potential to have a great impact on the encoding process. Used intelligently, bits can be better spent to achieve improved perceived quality by the end user. On a high level, this could seem like a relative straight forward task but there is challenge in determining exactly when any biasing should take place. This research examined when and where object information could have a positive impact on the encoder. This also leads into another challenge which is determining how a positive impact should be defined.

*Figure 5-4 Unaltered frame as input to analysis engine*

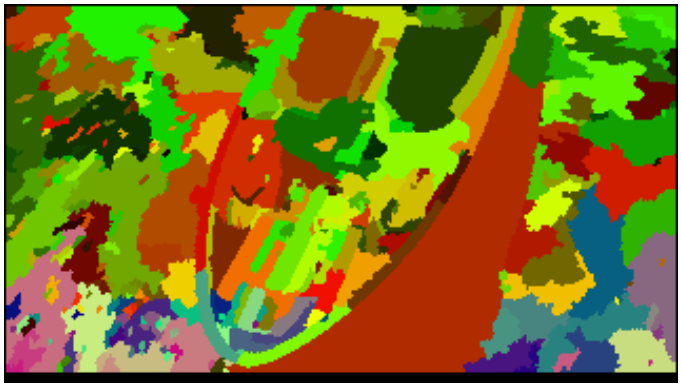


*Figure 5-5 Segmented Frame*

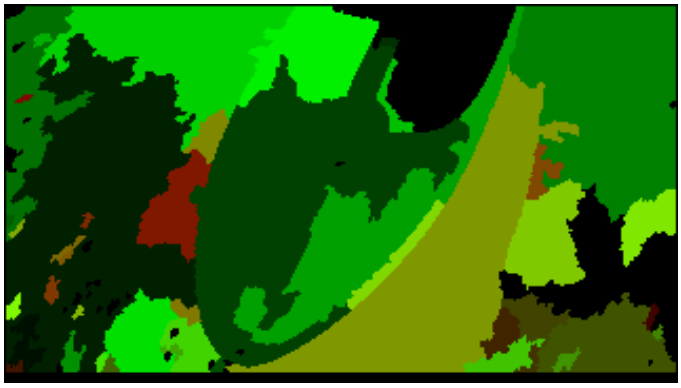


*Figure 5-6 Grouped Segments*

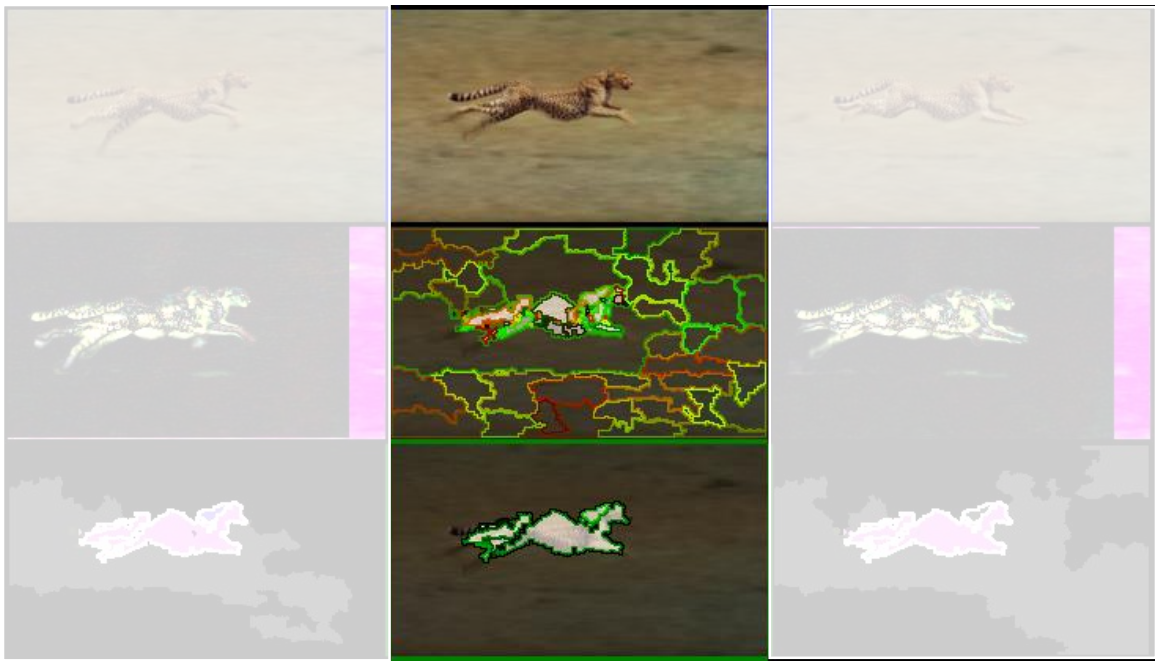


*Figure 5-7 Shows the relationship between an original image (top), its segmentation (middle), and its grouping (bottom)*

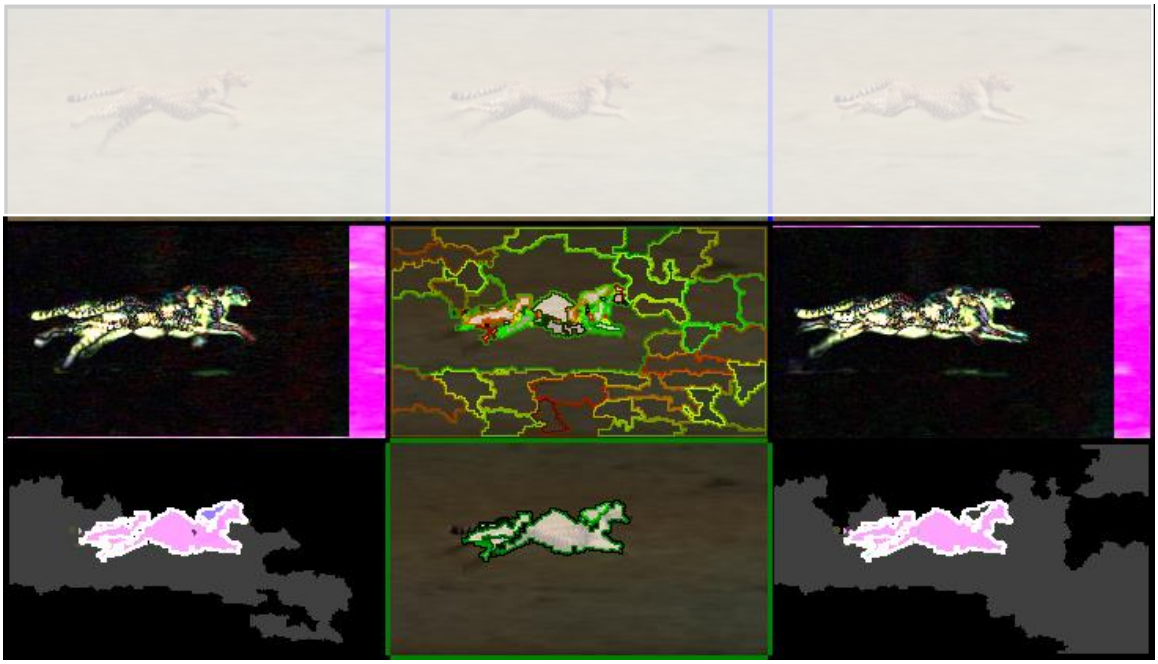


*Figure 5-8 Shows how the segmentation and grouping is examined across frames*

## Chapter 6 Encoding System

### 6.1 Video Macro-traits

During the encoding process, a number of parameters are checked from the video that will be used to maintain consistency. The parameters will include things like frame-rate, video duration, and interlacing. An item like duration is particularly challenging since what a user considers to be one video can easily have three distinct durations. For example the video container, video stream, and audio stream can all have their own reported durations in the header meta-data. Frame-rate can be challenging as well since a video may have a variable frame-rate thus distorting other reported information such as durations. And lastly, interlacing can never be assumed and must checked since incorrectly reported video is common. Without these three checks and others, few things can be considered certain about a video's reported information.

### 6.2 Video Splitting

The goal of this process is to explore possible splitting locations within a video and maintain accounting so there will be no repeated or lost frames between each split video. A video can only be decoded by starting on an I-frame. The locations of these I-frames must be found within a video and exploited as cut locations. Once found, the optimal separation between split portions of the video must be determined. The video is then split at these points and sent for later processing and encoding. The split parts must be verified that they are in fact the correct duration and start-time.

### 6.3 Encoding Process

Once a video has been split into smaller sections, the actual process of encoding the video begins. The videos are first processed in the segmentation/grouping analysis

engine. These results are then passed into the encoder along with the uncompressed YUV. The encoder uses the new information to improve the overall perceived video quality. Once every video sub-portion has been encoded then all parts are concatenated together to produce a finished video.

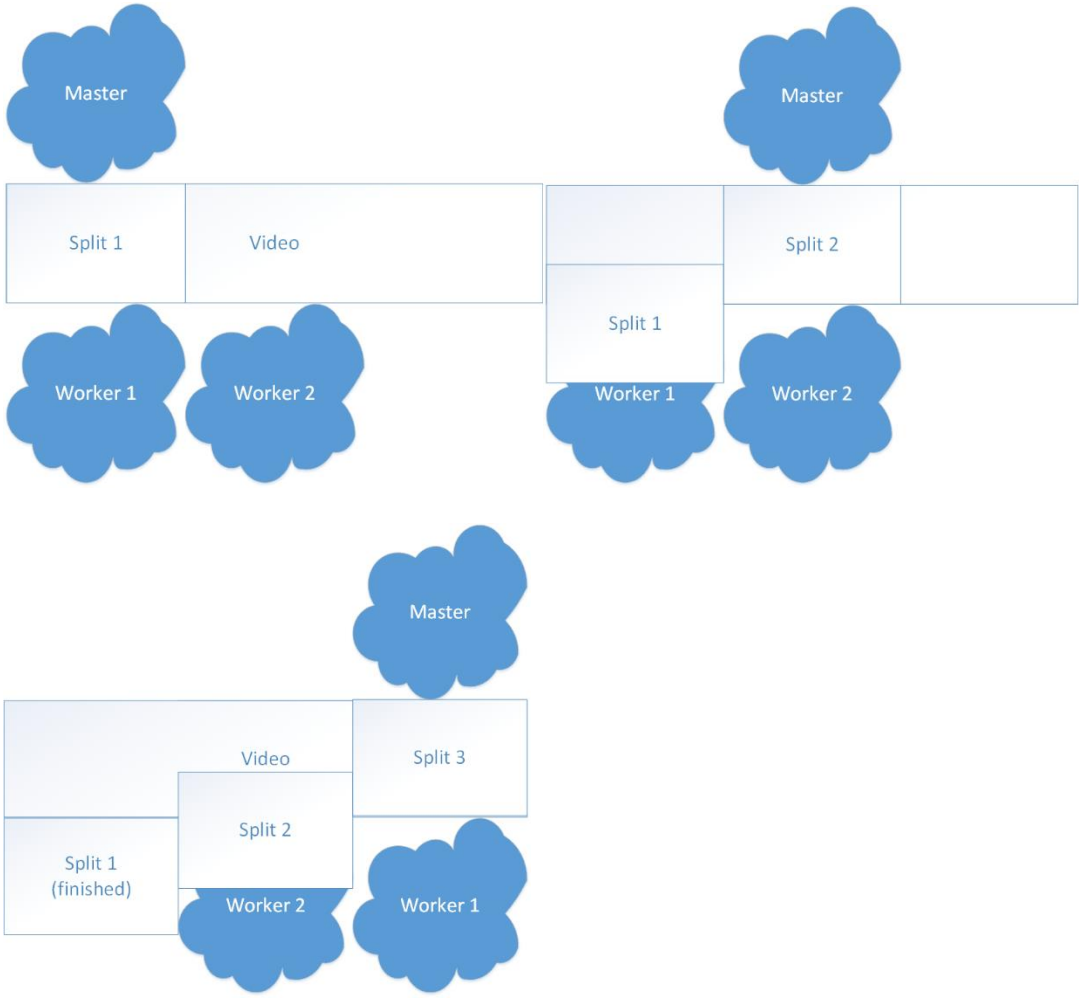



*Figure 6-1 Visualization of the video splitting process as could be done in the cloud. Ordering is top left, top right, bottom left.*

## Chapter 7 Encoding Improvements

The primary tools of this research revolved around the use of different video encoding tools and code-bases. Often, this examination became finding new ways of utilizing these existing tools. The primary video encoder was the HEVC Test Model Encoder. Being a new standard, this encoder was released to provide a guideline for future commercial encoders. It was crucial for this research. Additionally, the video coding tool, Ffmpeg, was used in many areas of video stream examination. Lastly the Lyrical Labs' proprietary object analysis engine [25] was used to generate the objects used in this research.

### 7.1 Video Splitting

This research examined a variety of techniques used in video splitting at a higher level. It also examined the use of different tools to achieve faster video splitting without sacrificing accuracy. The importance of being able to split a video accurately without adding or losing frames (see Figure 7.1) is particularly important as commodity based computing becomes more commonly used and operations continually move to the cloud.

The idea of splitting a video was deceptively simple. A video was examined, cut at the desired frame intervals, and then processed in some manner. The simplest practice was to give a static time interval. As might be guessed, this immediately ran into the issue of frame type since there was no guarantee that an I-frame would be at the desired interval. Of course if all frames were I-frames then the work was done and the video was easy to process. For many cases this did not work as the video had already been encoded. So the video needed to be split at the existing I-frames.

The determination of I-frames was another seemingly easy objective but again there were surprising difficulties. If the referencing structure allowed for B-frames to be forward referencing outside of their sub-portion, then they were no longer decodable. There was no guarantee that this would happen for every sub-portion. If it did happen for half of the 120 pieces to a 2 hour long movie then the resulting 60 frame loss generally would be unacceptable. This was likely to cause audio/video de-synchronization unless the audio was passed with the video and split around the same locations. Passing the audio with the video is undesirable unless it was a part of the use case since it required the transmission of unnecessary data, wasting resources.

There were several ways of determining I-frame locations but each had its own potential pitfalls. The most direct way was to simply decode the video. If a video was only decodable at an I-frame, then allowing the decoder to run at different times within a video would allow for information to be gathered to determine I-frame locations. Alternatively, the header information on each frame can be gathered and analyzed to find the optimal I-frame locations. Other methods existed for determining this information and even self-contained tools can be used with reasonable accuracy. Each method had its own merits depending on the use case and desired efficiency.

This research primarily explored ways of increasing accuracy of sub-divided videos and robustness to error. Accuracy referred to the system's ability to determine where proper split locations were. Robustness meant that if a part of the video was improperly split then it would be detected and accounted for properly.

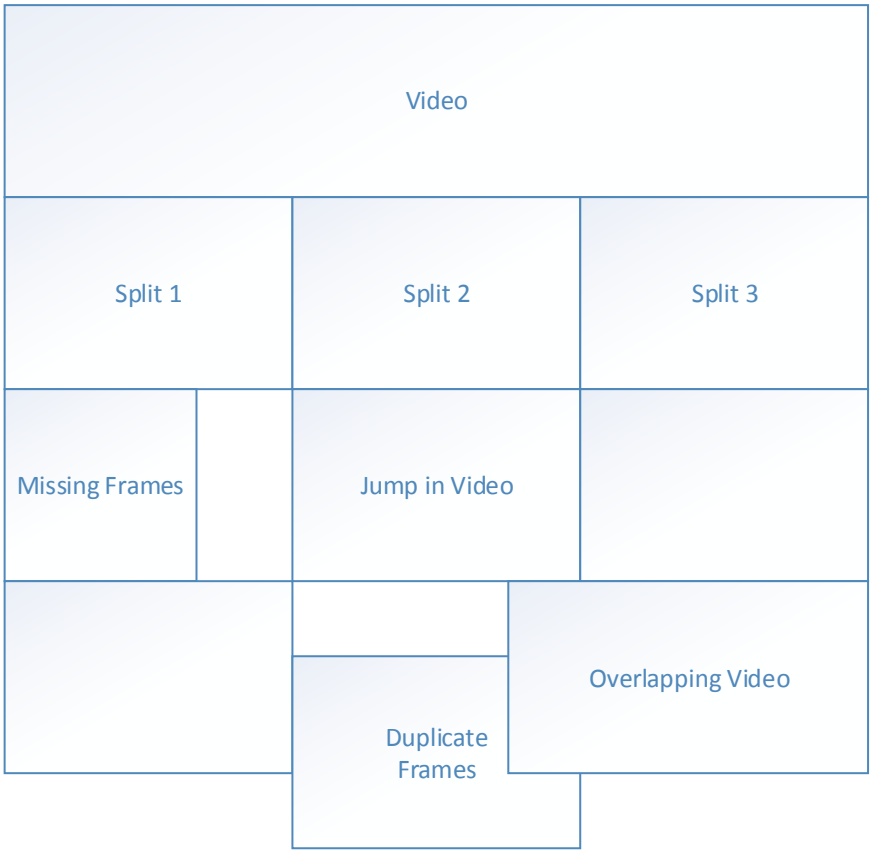



*Figure 7-1 Visualization of Bad Splitting*

## 7.2 Increased Algorithm Computational Efficiency

The HEVC's Test Model was an encoder with many functions that failed to take advantage of many modern vectorization and parallelization techniques. This research examined computationally expensive code areas to find ways of improving speeds using the Intel Compiler and Intel's Integrated Performance Primitives (IPP) functions [20]. Specifically Intel's Amplifier was used on a few small test videos to find these computationally expensive areas. The Amplifier output was used to document time spent at each location. The computationally expensive locations included areas such as rate-distortion calculation, quantization, and interpolation. These were areas which scaled

with the amount of motion estimation happening within the encoder. Different techniques were explored to address the issues discovered.

The interpolating filter functions had the challenge of implementing convolutions in the HEVC (see Figure 7.2). There were functions present in the IPP to aid implementation but this research found that the initialization time for these functions was too costly and actually caused a slowdown. Instead a localized buffer combined with smaller IPP functions decreased the run-time (see Figure 7.4). This research also experimented with removing the IPP functions and replacing them with an easily vectorizable function that would be optimized by the Intel Compiler (see Figure 7.3). This approach produced equal timings to the localized buffering strategy and it was discarded.

```
for (row = 0; row < height; row++)
{
  for (col = 0; col < width; col++)
  {
    Int sum;

    sum  = src[ col + 0 * cStride] * c[0];
    sum += src[ col + 1 * cStride] * c[1];
    if ( N >= 4 )
    {
      sum += src[ col + 2 * cStride] * c[2];
      sum += src[ col + 3 * cStride] * c[3];
    }
    if ( N >= 6 )
    {
      sum += src[ col + 4 * cStride] * c[4];
      sum += src[ col + 5 * cStride] * c[5];
    }
    if ( N == 8 )
    {
      sum += src[ col + 6 * cStride] * c[6];
      sum += src[ col + 7 * cStride] * c[7];
    }

    Pel val = ( sum + offset ) >> shift;
    if ( isLast )
    {
      val = ( val < 0 ) ? 0 : val;
      val = ( val > maxVal ) ? maxVal : val;
    }
    dst[col] = val;
  }

  src += srcStride;
  dst += dstStride;
}
```

*Figure 7-2 Under-optimized code example from HEVC Test Model Encoder*

```
short * foo = transposeBuffer;
for(row = 0 ; row < loopheight ; row++)
{
    for(col = 0; col < loopwidth; col++)
    {
    Int sum1;

    sum1  = foo[ 0 ] * c[0];
    sum1 += foo[ 1 ] * c[1];
    sum1 += foo[ 2 ] * c[2];
    sum1 += foo[ 3 ] * c[3];
    sum1 += foo[ 4 ] * c[4];
    sum1 += foo[ 5 ] * c[5];
    sum1 += foo[ 6 ] * c[6];
    sum1 += foo[ 7 ] * c[7];

    Short val1 = ( sum1 + offset ) >> shift;
    if ( isLast )
    {
        val1 = ( val1 < 0 ) ? 0 : val1;
        val1 = ( val1 > maxVal ) ? maxVal : val1;
    }
    foo[0] = val1;
    foo++;
    }
    foo += 7;
}
```

*Figure 7-3 Code example of unfurling of for-loop and use of local static buffer*

```
for(row = 0 ; row < loopheight ; row++)
{
    short holder[128];
    for(col = 0; col < loopwidth; col++)
    {

        Int sum1 = 0;
        for(int i = 0 ; i < 8 ; i++){
            sum1 += transposeBuffer[col + i + (row*7) + (row*loopwidth)] * c[i];
        }

        Short val1 = ( sum1 + offset ) >> shift;
        if ( isLast )
        {
            val1 = ( val1 < 0 ) ? 0 : val1;
            val1 = ( val1 > maxVal ) ? maxVal : val1;
        }
        holder[col] = val1;
    }
    for(int i = 0 ; i < loopwidth ; i++){
        transposeBuffer[i + (row*7) + (row*loopwidth)] = holder[i];
    }
}
```

*Figure 7-4 Heap buffer with single pass reassignment to memory buffer*

## 7.3 Block Matching

Within the HEVC test model encoder there were two primary search strategies, full search and diamond search. This research explored the use of different search heuristics to find which had the best block match in reasonable time. This idea was also explored in "Implementation of Motion Estimation Algorithm for H.265/HEVC" [9] and "A fast Three-Step search algorithm with minimum checking points using unimodal error surface assumption" [10].

Specifically four search techniques were used in total. First, the full search was used to find the best match. Next, three different search heuristics were used to determine how close they got to the full search match. The times and the PSNR were compared

across a number of video clips. The three search techniques compared were Diamond algorithms (2), a Hexagonal search algorithm, and a new search technique called Roll search (see Figures 7.5-7.6-7.7-7.8). The Roll search technique attempted to use a hybrid approach of different search techniques in a cyclical fashion. All three approaches produced reasonably similar results for all videos with the purely Hexagonal search being on average the best. Table 7.1 is a sample of the results.

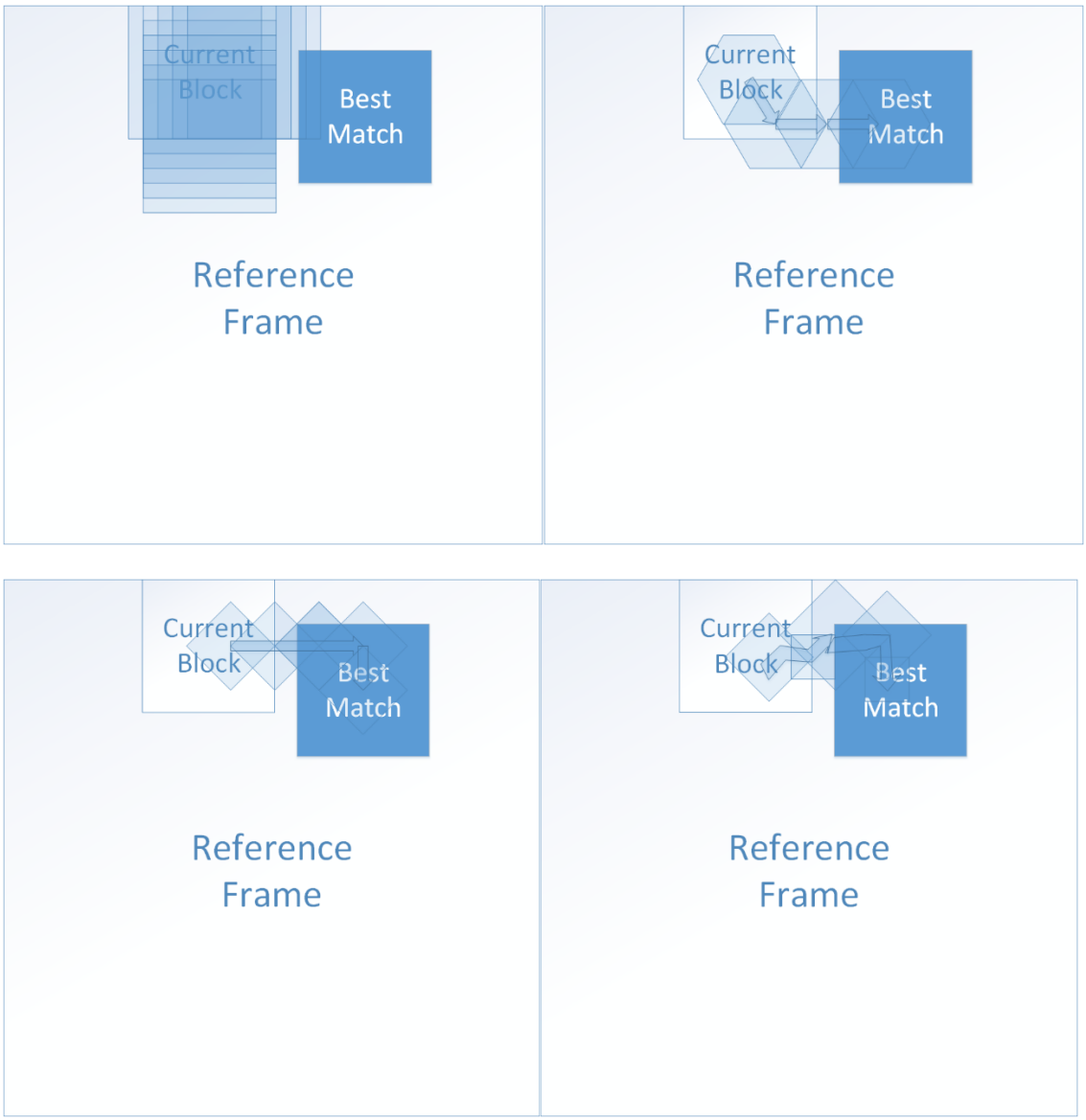



*Figure 7-5 Exhaustive Search (top left), Hexagonal Search (top right), Diamond Search (bottom left), Roll Search (bottom right)*

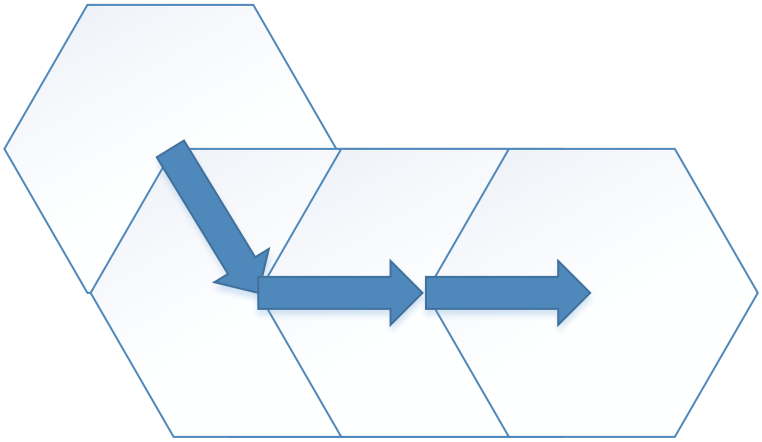


*Figure 7-6 Hexagonal Search*

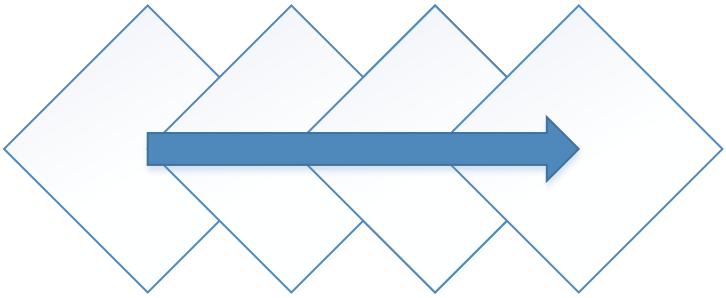


*Figure 7-7 Diamond Search*

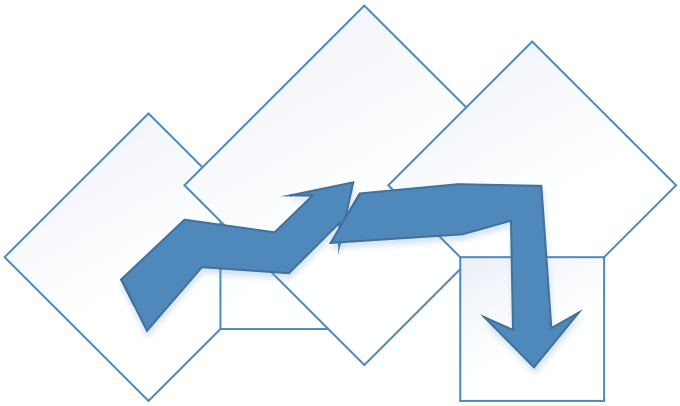


*Figure 7-8 Roll Search*

| Search Type | Frames | Bitrate | Y-PSNR | U-PSNR | V-PSNR | KByte | Time (minutes) |
|---|---|---|---|---|---|---|---|
| HEVC Diamond (More Exhaustive) | 591 | 434 | 41.80 | 45.37 | 45.60 | 1047 | 90.2 |
| Hexagonal | 591 | 442 | 41.80 | 45.36 | 45.57 | 1065 | 84.1 |
| Diamond | 591 | 445 | 41.79 | 45.38 | 45.60 | 1074 | 83.1 |
| Roll | 591 | 440 | 41.79 | 45.38 | 45.63 | 1061 | 82.2 |

*Table 1 Single video sample of different search methods*

7.4 Pruning CUs Using Group and Segment Information

This research sought to use Group and Segment information for a given frame to try and predict areas where more or less examination is needed. The decision to split a motion block was important for finding the best compression size and quality. However, many times in parts of the frame it was unnecessary. This research sought to analyze and prune areas where there was no benefit for splitting (and the associated costs) while still maintaining a low bitrate with reasonable quality.

One method of decision making explored was a counter with a threshold which would determine if pruning was necessary. This counter used simple HEVC flags, object/segment map information, CU depth, and whatever search strategy that found the best CU (Intra, Merge, Inter). This research examined the behavior of the decision process on different clips and different trouble spots to try to find an optimal weights given to each flag.

For each CU, we read in the four corners for each channel of the group/segment maps. Depending on the number of objects present, the counter would be incremented. If there was no information present on the group map then the segment maps were checked. If all four corners were located on the same object then the counter was

incremented. A check of the search type used in final block for a given depth determined whether the count was incremented. This research saw on average a 50% decrease in run-time while generally maintaining perceived quality (see Table 7.2-7.3) over the vector optimized code. Overall, combining the optimizations described in the proceeding section let to an average improvement of 63% compared to the Test Model baseline.

| Version/Video | Aqua | nfl_b | limitless | mustang | Nonsense(mp4) |
|---|---|---|---|---|---|
| Base | 50.7 | 102.9 | 56.6 | 207.2 | 63.1 |
| IntelIPP | 47.3 | 76.8 | 38.9 | 144.2 | 43.7 |
| Pruning | 32.8 | 51.2 | 21.7 | 78.4 | 24.0 |
| PruneAndSplit | 26.9 | 41.4 | 17.6 | 69.6 | 18.2 |

*Table 2 Comparison of run-times (in minutes) for different Motion Estimation search reduction strategies. Shows an average runtime reduction of 50% when compared to the vector-optimized encoder and 63% compared to the baseline.*

| Tag | Base_HEVC | Base_Intel | Split | Prune | Both |
|---|---|---|---|---|---|
| Y-PSNR | 38.23 | 38.23 | 38.19 | 38.16 | 38.15 |
| Kbytes of videos | 4984 | 4984 | 5012 | 4997 | 5035 |
| Encoder Runtime | 220.1 | 161.7 | 111.8 | 90.2 | 82.8 |

*Table 3 Comparison of Modified Early CU (ECU) and Modified Early Split Detection (ESD) over a single video. Shows that PSNR values changed less than 1% despite over a 50% speed increase over the vector-optimized code and 63% over the base encoder.*

## 7.5 Learning Based CTU Selection

In a continued effort to reduce computation time within the encoding process, this research explored the use of a learning based approach in all Coding Tree Unit decisions. There were two main issues being addressed. First, many unnecessary calculations were made at each CTU for both divided and undivided CU candidates. Second, the encoder only examined local information resulting in the possibility of the sequence suffering on a more global scale.

The first issue of unnecessary calculations came in two forms: if the sub-block candidate was not used, then all lower calculations made were unnecessary; if the sub-block candidate was used, then all the calculations made on the undivided CU were unnecessary. Eliminating these calculations would decrease encoding times and not have any impact on quality. At first glance it would seem that the encoder would not know what was best until it has calculated all the costs. However in many cases it was possible for the encoder to make good educated guesses using small motion vectors, low cost, or by prior knowledge. Using motion vectors and cost information was straightforward and was known in the art, but how prior knowledge will be generated and used is a complex topic.

The new method implemented in this research used a classifier to learn about optimal CU examinations. In one embodiment, a neural network classifier was used. The classifier learned using object group analysis, segmentation, localized frame information, and global frame information. First the classifier examined frame information both on a global and local scale. Second the classifier compared the average cost of an entire frame to the local CU cost to gain a sense of consistency. The ratio of the

average frame cost and local CU cost was the third input to the classifier. The fourth input to the classifier was the cost decision history of the local CTUs that had already been processed. This was a count of the number of times a split CU was used in its final CTU. The fifth input was the Early Coding Unit decision as developed in Joint Video Team's on Video Coding HEVC Test Model 12. It utilized information on the motion vector, cost, and type of heuristic search used. This approach was well known in the art. Finally, the level in the quad tree structure was taken as the sixth and final input. The expectation was that the higher the CU was the more likely it was to split. The lower it was, the more likely it was to stop the search early.

Training happened in two steps. The first step was to train on information from a number of video inputs then to create a pre-trained classifier to be used in future encodings. The second step happened during an actual encoding. The classifier was adapted to the new video sequence for which it subsequently influenced the encoder's decisions of whether to bypass unnecessary calculations.

The second issue (the encoder only examined local information) required a more global approach. Currently the most common strategy for planning how a video can be encoded is to encode it twice, called two-pass encoding. Doing twice the work for a result that may not be very different from the first attempt was suboptimal. Also currently in the art only the quantization parameters are normally adjusted for the second pass. This did not address the strong potential for correlations between the different CTU's.

A potentially smarter alternative was to use shorter analyses and a classifier to help guide the CU selection process. The cost decision could be enhanced such that human visual quality could be increased while lowering bit expenditures using a

combination of segmentation, object group analysis, and a classifier. This was done by locating areas of low and high activity then spending more bits on the latter and fewer on the former. In addition, this approach leveraged the correlation information between CTU's to make better global decisions. This gave greater emphasis to areas which were more sensitive to human visual perception. While this did not improve the rating of most objective quality metrics, it could produce a higher perceived quality to end-users. In summary using a combination of information already available to the encoder along with brief analyses, this approach could improve both human visual quality while reducing the number of bits required to encode. An example of the separation between decision types is shown in Figure 7.8.

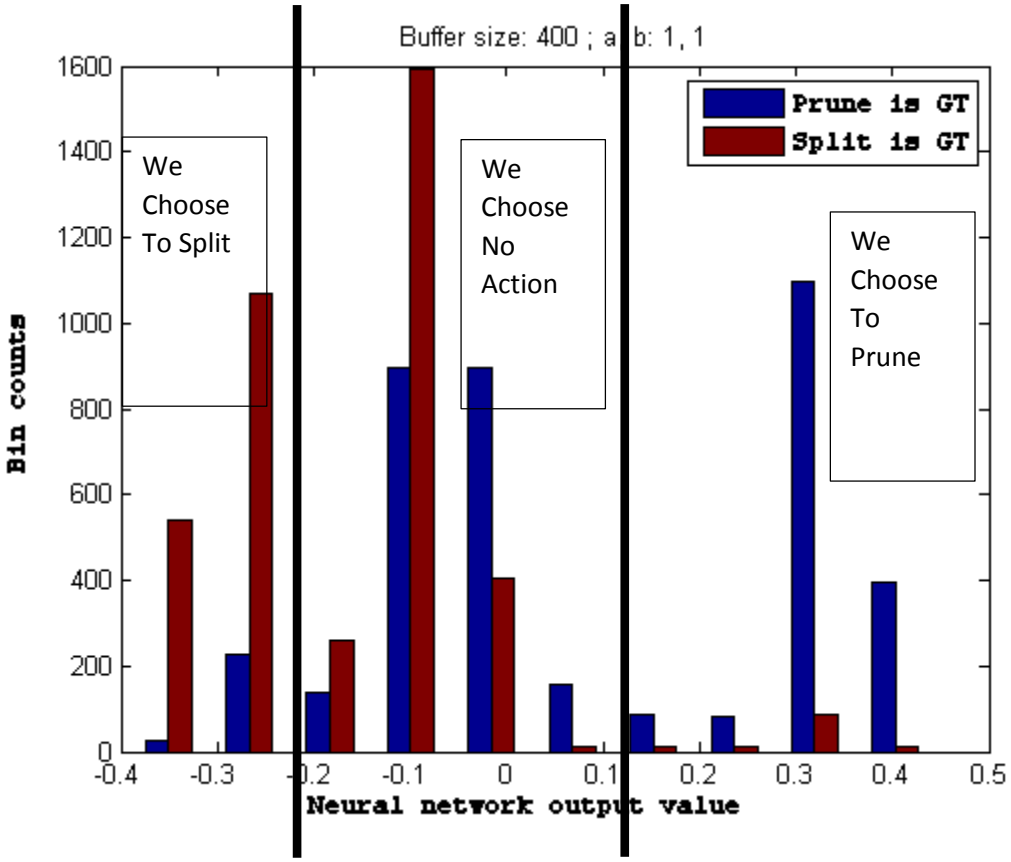



*Figure 7-9 Example separation achieved by the learning based heuristic*

## 7.6 Rate Control Bit-Allocation Decisions

In continued efforts to improve encoding, this research explored the modification of rate-control algorithms to address scene-change responsiveness and better bit allocation. It found that in some cases current rate control algorithms produced subpar results often overspending or underspending bits depending on the situation. Additionally, the use of object analysis in rate control is ongoing and at the point of this submission is unfinished.

Many video encoders use a system of controlling bit spending to maintain either a certain quality standard or proper bit limit. Proper bit spending means that if a connection is limited to 2Mbits a second then the encoder can create a video stream that adheres to that limit. In contrast, quality control means that the error within the video will be maintained at a set level while ignoring the cost of however many bits are needed. As can be observed these two options are mutually-exclusive and often the bit-rate is used for consumer applications. The constant bit-rate choice can lead to a video with intermittent poor quality in the eyes of the consumer. On the other hand a streaming video that is constantly buffering to maintain a quality standard is equally bad, if not worse, than a video with intermittent poor quality. This was inferred from the behavior of every media streaming service this research encountered anecdotally.

The rate control algorithms use recent history from a video to determine its current spending habits attempting to predict the number of bits required in a given moment. These methods generally do a good job but there are times when things go wrong. This research observed a common troubled area was at the start of videos. A problem occurs if a user knows that a particularly low bit-rate is acceptable for a video

yet there is no way to communicate that information to the encoder. As a result, the encoder will produce poor quality results until it builds enough history to determine that it is able to be more generous in its outputted quality (see Figure 7.9). If it were a single occurrence then this would be a small issue. However, for a system that attempts to split a video into many parts before encoding, this can be a very serious issue.

*Figure 7-10 Visual difference between distorted (top) and undistorted (bottom) frame. Notice the blockiness around the fish's mouth in the top frame.*

The goal then is to make the encoder produce reasonable results in a dynamic fashion that does not completely break any bit-stream restrictions. This research explored a combination approach in reaching this goal. First, the research explored native components within the encoder for the possibility of solving the problem. With some success, the encoder was able to be tweaked to allow for faster changes in bit rate spending and to changing quality circumstances. This produced some positive results, but the bit spending problem still persisted. This research found that if the encoder were given a slightly increased bit spending cap at the start of a video, in combination with increased adaptability, then quality was maintained throughout the video with a minimal increase to bit-rate.

This research observed a second problem that was similar but distinct from the first. Videos with areas of very low complexity transitioning to high activity were unable to adapt fast enough and vastly overspent their bit budget thus reducing video quality from that point on. This issue was again related to the lack of speed in adaptability, but coming from the opposite direction, the rate control was too generous with bit spending rather than being too frugal. To correct the problem, this research explored the use of triggering a reset within the encoder rather than merely trying to change course. This research originally found ways of exploiting sudden changes in bit-predicted spending to flag those areas for a fresh start. While simple, this proved to be effective in many cases. The research later explored the use of scene transition information generated from the object analysis engine, but currently this approach has not achieved any improvement in quality and adaptability.

Lastly, this research is currently exploring the use of the object analysis engine in influencing bit-spending on a per-frame and macroblock basis. The exploration of better budgeting bits in areas of high activity is of continued interest to this research and the industry.

## Conclusion

This research, while varied, highlights the room for improvement in modern encoding solutions. There will be continued gains in video compression and clarity as work transitions to the cloud, object analysis improves, and encoders leverage more sophisticated machine learning strategies. Videos will continue to become smaller in file size as technology strives to meet the demands of consumers.

We were able to make significant improvement in various parts of the encoding system. The improved I-frame examination method has proven robust in application on thousands of video clips including full-feature length content. The rate-control improvements have addressed the number of pathological cases that had been seen before with the older rate control. We were able to achieve an average 63% speed improvement of the H.265 encoder over the Test Model baseline with negligible impact to the video quality. Finally, we explored a learning-based framework for saving computational time in the H.265 encoder that shows very promising early results.

As was previously mentioned, the potential for improved bit spending and quality assessment still exists in modern encoders. The allocation of these bits remains of high interest as 4k and 8k content begin to enter the market. These higher resolutions require more intelligent bit spending that will leverage techniques explored in this thesis. In addition to bit allocation, other parts of the research can be explored further. The video splitting method in the cloud can be potentially modified to accommodate traditional server units. This could decrease encoding times and improve frame accuracy for content. A hybrid approach of the splitting could be implemented that should maintain accuracy while improving speeds for both the cloud and traditional servers.

The learning based pruning and splitting heuristic shows high potential for further improving encoding times. It is in a prototype stage and must be implemented within an encoder to demonstrate its viability. However, the separation achieved by this experimentation highlights the potency of this technique.

As a whole this research examined various topics within a modern video compression system and consistently showed improvement in run times while maintaining visual clarity and frame accuracy.

**References**

[1] G.J. Sullivan; J.-R. Ohm; W.-J. Han; T. Wiegand (2012-05-25)."Overview of the High Efficiency Video Coding (HEVC) Standard" (PDF). IEEE Transactions on Circuits and Systems for Video Technology

[2] Zhou Wang*;* Alan Conrad Bovik*;* Hamid Rahim Sheikh*;* Eero P. Simoncelli, "Image Quality Assessment: From Error Visibility to Structural Similarity" IEEE TRANSACTIONS ON IMAGE PROCESSING, VOL. 13, NO. 4, APRIL 2004

[3] Milan Sonka; Vaclav Hlavac; and Roger Boyle, Image Processing, Analysis, and Machine Vision. Thomson Learning, 2008 Chapters 2: The image, its representations and properties

[4] Yun Q. Shi; Huifang Sun Image and Video Compression for Multimedia Engineering, 2008 Chapter 1: Introduction

[5] L. Escalin Tresa; Dr. M. Sundararajan "Comparative Analysis of Different Wavelets in DWT for Video Compression" 2014 International Conference on Circuit, Power and Computing Technologies [ICCPCT]

[6] Li Zhang; Xiaolin Wu; Ning Zhang; Wen Gao; Qiang Wang; and Debin Zhao "Context-based Arithmetic Coding Reexamined for DCT Video Compression" Supported by National Natural Science Foundation Research Program of China (No. 60672088), the Research Fund for the Doctoral Program of Higher Education (No.20060213014) and Special Foundation of President of The Chinese Academy of Sciences (No. 20064020 and No. 20066120)

[7] Amir Said "Introduction to Arithmetic Coding - Theory and Practice" HP Laboratories Palo Alto

[8] Marques, O. "Practical Image and Video Processing Using MATLAB" Wiley-IEEE

[9] Davis P, Sangeetha Marikkannan "Implementation of Motion Estimation Algorithm for H.265/HEVC" in International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering Vol. 3, Special Issue 3, April 2014

[10] Jong-Nam Kim, Tae-Sun Choi, "A fast Three-Step search algorithm with minimum checking points using unimodal error surface assumption," in IEEE Transaction on Consumer Electronics, vol. 44, no. 3, pp. 638-648, Aug. 1998.

[11] D.A. Huffman, "A Method for the Construction of Minimum-Redundancy Codes", Proceedings of the I.R.E., September 1952, pp 1098–1102.

[12] Yun Q. Shi; Huifang Sun Image and Video Compression for Multimedia Engineering, 2008 Chapter 15: Fundamentals of Digital Video Coding

[13] Yun Q. Shi; Huifang Sun Image and Video Compression for Multimedia Engineering, 2008 Chapter 16: Digital Video Coding Standards: MPEG1/2 Video

[14] Yun Q. Shi; Huifang Sun Image and Video Compression for Multimedia Engineering, 2008 Chapter 19: ITU-T Video Coding Standards H.261 and H.263

[15] Yun Q. Shi; Huifang Sun Image and Video Compression for Multimedia Engineering, 2008 Chapter 20: A New Video oding Standard: H.264/AVC

[16] Yun Q. Shi; Huifang Sun Image and Video Compression for Multimedia Engineering, 2008 Chapter 5: Variable-Length Coding: Information Theory Results

[17] Ken Cabeen; Peter Gent "Image Compression and the Discrete Cosine Transform" College of the Redwoods

[18] Meteficha, "this is an example of a huffman tree" Public Domain: Huffman Wikipedia

[19] Marques, O. “Practical Image and Video Processing Using MATLAB” Wiley-IEEE Press Chapter 15
[20] Intel® Integrated Performance Primitives Reference Manual, Volume 2: Image and Video Processing
[21] Yun Q. Shi; Huifang Sun Image and Video Compression for Multimedia Engineering, 2008 Chapter 4: Transform Coding
[22] Suman Tatiraju; Avi Mehta “Image Segmentation using k-means clustering, EM and Normalized Cuts” Donald Bren School of Information and Computer Sciences, UCI
[23] Pratt, W.K.; Kane, J.; Andrews, H.C.; , "Hadamard transform image coding," Proceedings of the IEEE , vol.57, no.1, pp. 58- 68, Jan. 1969 doi: 10.1109/PROC.1969.6869
[24] Milan Sonka; Vaclav Hlavac; and Roger Boyle, Image Processing, Analysis, and Machine Vision. Thomson Learning, 2008 Chapters 16: Motion analysis
[25] Ed Ratner; Iuri Frosio; Daniel Raburn; Shahab Mirghorbani, Lyrical Labs Proprietary Object Analysis Engine, Lyrical Labs LLC