1-1-2017

Video Stream Adaptation In Computer Vision Systems

Yousef Sharrab Sharrab
Wayne State University,

Recommended Citation

Sharrab, Yousef Sharrab, "Video Stream Adaptation In Computer Vision Systems" (2017). *Wayne State University Dissertations*. 1742.
https://digitalcommons.wayne.edu/oa_dissertations/1742

VIDEO STREAM ADAPTATION IN COMPUTER VISION SYSTEMS

by

YOUSEF ODEH SHARRAB

DISSERTATION

Submitted to the Graduate School

of Wayne State University,

Detroit, Michigan

in partial fulfillment of the requirements

for the degree of

DOCTOR OF PHILOSOPHY

2017

MAJOR: COMPUTER ENGINEERING

Approved By:

____
Advisor Date

____

____

____

DEDICATION

To my great mother Thuria

To my children, Omar and Jawad

ACKNOWLEDGMENTS

I would like to thank my adviser Dr. Nabil Sarhan. His guidance and directions were of extreme help during my research. I also would like to thank my committee members Dr. Xiaoyan Han, Dr. Loren Schwiebert, and Dr. Song Jiang for their precious feedback. I would like to extend my gratitude to my mother who kept encouraging and supporting me on this track, and to my kids Omar and Jawad who always blame me for being busy and not playing with them. Finally, to my lab colleagues, Musab Al-Hadrusi, Mohammad Alsmirat, Saleh Amareen, and Kamal Nayfah who helped me in programming and scripting.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1 INTRODUCTION

1.1 Overview

Computer Vision (CV) is a science that aims to electronically perceive and understand an image or a sequence of images (i.e. a video) [1]. Popular CV algorithms include object/event detection, recognition, and tracking [2]. CV has been deployed recently in a wide range of applications, including surveillance and automotive industries. Such CV systems include automated video surveillance [3, 4], Wireless Video Sensor Networks (WVSN) [5, 6, 7, 8], mobile surveillance systems [9], Advanced Driving Assistance Systems (ADAS) [10], Vehicle-to-Vehicle/Vehicle-to-Infrastructure (V2V/V2I) video communication [11], traffic monitoring systems, and other Intelligent Transportation Systems (ITS) [12, 13]. According to a recent report from Tractica [14], the market for CV technologies will grow from $5.7 billion in 2014 to $33.3 billion by 2019. Surveillance and automotive industries share over 20% of this market. According to [15], 245 million video surveillance cameras installed globally in 2014. Over 20% are network cameras and around 2% are High Definition (HD) cameras.

This dissertation considers the design of real-time CV systems with live video streaming, especially those over wireless and mobile networks. Such systems include video cameras/sensors and monitoring stations. The cameras should adapt their captured videos based on the events and/or available resources and time requirement. The monitoring station receives video streams from all cameras and run CV algorithms for decisions, warnings, control, and/or other actions. Real-time CV systems have constraints in power, computational, and communicational resources. The metric for the performance of CV systems is the accuracy of the system in perceiving or extracting descriptions of physical objects or events from pictures (i.e. accuracy for detection, recognition, and tracking of object and events). Power consumption has also become a major concern in CV systems, especially those employing battery-operated devices. In such systems, prolonging the battery lifetimes is a primary objective due to its great implications in terms of system cost and availability. In such systems, energy is consumed at the source in each of the three main phases: capturing, encoding, and transmission. Due to the limited amount of energy resources available, power consumption efficiency is one of the most challenging design factors. Since video encoding contributes to most of the overall power consumption at the video stations, the encoding parameter settings used at each station determine the encoding power consumption and bitrate of the video. The bitrate determines the quality of the video and the transmission power consumption of the station.

CV systems are usually real-time. For example, vehicular CV systems and automated video surveillance systems are real-time CV systems. Vehicular CV systems either detect objects and events that may represent safety risks to drivers or detect obstacles to traffic [13]. Automated video surveillance systems alert the security guards of any undesired activity caught on the surveillance cameras. Data and image processing in CV systems are usually intensive and requires large amounts of computational resources and memory. For example, a simple camera with 800×600 resolution can capture more than one megabyte per second without image compression. Image compression algorithms require additional computational resources [13]. In all these systems, video encoding must be in real-time. The performance of encoders in terms of quality, bitrate, and encoding speed is determined by many encoding parameters. In a power/bandwidth/time constrained environment, it is very important to choose the right settings for the parameters that lead to the optimal encoding performance in terms of power consumption, bitrate, encoding speed, and quality [16].

CV systems should adapt the videos according to the dynamic changes in the network and environment [17]. In addition, CV application can adapt to available battery charge to prolong the battery life. Video adaptation has been studied extensively in video streaming in general, but little work has been devoted to computer vision systems. For video streaming, a variety of video adaptation techniques have been studied [18, 19, 20], with the main approaches being transcoding and scaling the video Signal-to-Noise Ratio (SNR), spatial, and/or temporal parameters. SNR, spatial and temporal resolution can be changed simultaneously for efficient use of resources. Most video adaptation techniques considered the video distortion as the primary metric, leading to much literature on rate-distortion characterization and optimization [21] (and references within). In CV systems, however, the main objective is enhancing the event/object detection/recognition/tracking accuracy. The accuracy can essentially be thought of as the quality perceived by machines, as opposed to the human perceptual quality.

The shortage in bandwidth as a result of the popularity of *High Definition* (HD) videos motivate the video encoding community to develop the new encoding standard called *High-Efficiency Video Coding* (HEVC). This standard required half the bandwidth compared to the H.264 encoding standard [22] at the same level of video quality. Unfortunately, HEVC adopts algorithms that have high computational complexity, which makes the video encoding extremely slow and consume tremendous amount of power [22]. Since encoding HEVC videos in real-time is extremely challenging, it attracts researchers to propose new methods that speed up the encoding process by predicting the results of the high computational algorithms faster.

In this dissertation, we propose adaptation techniques to reduce encoding computational complexity while maintaining detection accuracy or video quality. Our proposed techniques tradeoff between the system resources without considerable loss in performance. These techniques require modeling the resources in terms of video parameters that can adapt based on the resources availability.

We model the encoding computation complexity, the power consumption, and the bitrate. In addition, we characterize the detection accuracy and the video quality. To utilize the resources efficiently based on our required performance, we develop a model that provides any desired tradeoff in terms of accuracy, bitrate, and energy consumption. We also study other internal encoding parameters on encoding computation complexity, such as number of reference frames, and search range.

For HEVC, we develop an algorithm that predicts the size of the block currently being processed ($CU_{current}$) without exhaustive RDO calculations. Based on the similarity of recently processed *Coding Units* (CUs) in contents and $CU_{current}$ content, our algorithm substitutes RDO for partitioning the *Largest Coding Unit* (LCU). To prevent error propagation, we introduce other content conditions that must be satisfied. The content conditions are based on Shannon entropy of $CU_{current}$ and its neighbors. Shannon entropy is estimation of the minimum number of bits required to encode a group of symbols, based on the number of occurrence of the symbols in the group [23, 24].

1.2 Main Research Objectives

The main objectives of this dissertation can be summarized as follows:

- Adaptation of live video streams in computer vision systems.
- Modeling and analyzing the power consumption in live video streaming systems.
- Developing a history and entropy based LCU partitioning algorithm for HEVC encoding.

1.3 Detailed Research Plan

This dissertation is organized into the following three main parts.

1.3.1 Adaptation of Live Video Streams in Computer Vision Systems

We analyze video rate adaptation techniques in CV systems, including Automated Video Surveillance (AVS). As in all other video streaming applications, the video streams in CV systems should be adapted to the dynamically changing network conditions. We analyze and compare various video adaptation techniques in terms of both event/object detection accuracy and power consumption. The rate adaptation techniques include spatial (resolution-based), temporal (frame rate-based), and SNR (quantization-based). We analyze the impact of upscaling spatially adapted videos to their original sizes by using super-resolution techniques at the receiver before applying the CV algorithm. In addition, we study the impact of different adaptation combinations. Furthermore, we present an objective function that provides any desired tradeoff in terms of accuracy, bitrate, and energy consumption. By examining the rates of change in each of these metrics, the function favors the setting with a larger subsequent drop in accuracy, a smaller subsequent drop in bitrate, and a smaller subsequent drop in energy.

1.3.2 Modeling of Power Consumption and Bitrate in CV streaming Systems

We develop an aggregate power consumption metric for video streaming systems. We model the video capturing, encoding, and transmission aspects and then provide an overall model of the power consumed by the video cameras and/or sensors. This work has been motivated by Wayne State Multimedia Lab ongoing work on the power-aware design of automated video surveillance systems, which requires accurate, simple, and appropriate power consumption models. The model can help in the dynamic control of various camera/sensor settings, including resolution, frame rate, and quantization to achieve the best overall tradeoff in terms of power consumption, bitrate (and thus bandwidth), and quality. We also analyze the power consumed by the monitoring station, which is due to video reception, potential video upscaling (to the original video resolutions as capture by the sources), and video decoding of all received video streams. For video encoding, we focus primarily on H.264 and show that the model can be generalized to MPEG-4. The performance of encoders in terms of quality, bitrate, and power consumption are determined by many encoding parameters. The proposed model captures the following main parameters: resolution, frame rate, quantization, motion estimation (ME) range, and number of reference frames. In addition, we model the output bitrate of video encoding. The bitrate impacts the medium bandwidth, the video quality, and the transmission power consumption. We validate the models through extensive experiments. We analyze the power consumption models of each phase as well the aggregate power consumption model. The latter is validated using two different cameras. The analysis includes examining individual parameters separately as well examining the impacts of changing more than one parameter at a time.

1.3.3 Fast HEVC Encoding by History and Entropy-Based LCU Partitioning

HEVC is a recent video encoding standard to overcome the bandwidth shortage as a result of the popularity of HD videos. HEVC adopted numerous new tools, such as more flexible data structure representations, which include the *Largest Coding Unit* (LCU) and *Coding Uint* (CU). LCU is a 64×64 block which can be partitioned down into 8×8 CUs [22]. In the partitioning of the LCU into CUs, a high computation algorithms are applied, which makes the encoding too low for real-time systems. This part of the dissertation proposes an algorithm called *History and Entropy-based LCU Partitioning* (HELP) to predict the partition of LCU fast instead of the exhaustive *Rate Distortion Optimization* (RDO) method in HEVC.

HELP algorithm predicts the partition decision for the block currently being processed ($CU_{current}$) based on the weighted average of the termination possibility of all the spatial and temporal neighbors. The partition decision is to split the block to four blocks or to terminate the process of searching for the optimal partition for $CU_{current}$. The termination possibility is defined as the likelihood that $CU_{current}$ will terminate or split based on the decision that has been made for the neighbor block. The neighbor block is each processed block of the same size that is either temporally co-located or spatially share an edge or a corner with $CU_{current}$. To prevent error propagation of predicting partition decision, HELP uses a second condition based on the content of $CU_{current}$. The metric for how much information are in the $CU_{current}$ is the entropy of this block [25].

The prediction is based on the correlation between the entropy of $CU_{current}$, the entropy of its neighbors, and the partition history of the neighbors. We improve the encoding speed of the RDO implemented in HEVC while maintaining the coding efficiency and video quality within acceptable degradation. We demonstrate the effectiveness of the proposed algorithm in comparison with RDO and a published entropy-based algorithm [25] and other existing algorithms through extensive experiments.

CHAPTER 2 BACKGROUND INFORMATION AND RELATED WORK

In this chapter, we provide background information about the three main parts of the dissertation.

2.1 Video Encoding

2.1.1 Overview of Video Encoding

The main video encoders include MPEG-4 Part 2 Standard (or simply MPEG-4) and MPEG-4 Part 10 Standard (or simply H.264). As shown in Figure 2.5, the video encoding process can generally be divided into the following three high-level stages: *Intra and Inter Prediction (Estimation)* Stage, *Transformation, Quantization and Their Inverse* Stage, and *Entropy Coding* Stage. In the estimation stage, both intra-prediction and inter-prediction are used to reduce the spatial and temporal redundancies in the video, respectively. Video data contains spatial and temporal redundancies. Therefore, similarities can be encoded by just considering differences within a frame (spatial), and/or between frames (temporal). The first frame of a sequence or a random-access point is typically intra-coded (i.e., without using information from other frames). Each block of pixels in an intra-frame is predicted using previously-encoded neighboring blocks. For all remaining frames of a sequence or between random access points, inter-coding is usually used, employing block motion compensation to predict blocks from other previously encoded frames. The residuals of the intra-prediction and inter-prediction are then transformed to the frequency domain using Discrete Cosine Transform (DCT) in MPEG-4 or Integer DCT in H.264. (The residual is difference between the original and predicted blocks.) Subsequently, the transform coefficients are quantized, thereby reducing the overall precision of the coefficients and possibly eliminating high frequency coefficients. The quantized transform coefficients are entropy coded and transmitted together with any possible motion vectors (MVs).

2.1.2 Overview of H.264 Standard

As H.264 is the primary focus of this work, let us now discuss it in more detail. H.264 employs many features for more efficient compression and better flexibility with the network environment [26]. Figure 2.6 shows the processing stages of H.264. The processing stages can be described as follows.

One of the main features of H.264 is using multiple reference frames to increase the compression ratio. It allows up to 16 reference frames. In contrast, MPEG-4 allows one reference frame.

Another feature of H.264 is using variable block-size motion compensation, thereby enabling a more accurate segmentation of moving regions and higher compression ratios. The block size ranges from 4×4 pixels to 16×16 pixels. In MPEG-4, the minimum block size is 8×8.

When coding a macroblock, an H.264 encoder can choose from many different intra-modes for I-frames or inter-modes for B- and P-frames. Within each inter mode, the encoder has a wide choice of possible MVs, leading to a huge number of options for coding a macroblock [27]. The *Rate-Distortion Optimization* (RDO) mode selection is an algorithm for choosing the best coding mode for each macroblock, based on the bitrate and distortion cost. It is used for both intra-prediction and inter-prediction. To select the best encoding mode for a macroblock, the algorithm examines all possible combinations of intra- or inter-modes. The bitrate cost r and distortion cost t are combined into a single cost J:

$$J = t + g \times r. \tag{2.1}$$

The RDO mode selection algorithm attempts to find the mode that minimizes the joint cost J. The tradeoff between bitrate and distortion is controlled by the Lagrange multiplier g. An empirical approximation of g as a function of quantization parameter (q) is given by

$$g = 0.852^{(q-12)/3}. \tag{2.2}$$

Further details can be found in [27].

H.264 employs a simplified version of the DCT transform. In particular, it uses a 4×4 or an 8×8 Integer DCT transform, whereas MPEG-4 use an 8×8 DCT.

H.264 employs a quantization design, which includes a Logarithmic step-size control for easier bitrate management by encoders and simplified inverse-quantization scaling. Two methods are available for quantization. The first method uses one of two available quantization matrices to modify the quantization step-size based on the spatial frequency of the coefficient, whereas the second method uses the same quantization step-size for all coefficients. MPEG-4 also allows for non-linear quantization of DC values [28].

H.264 provides two options for entropy coding: *Context Adaptive Binary Arithmetic Coding* (CABAC) and *Context Adaptive Variable Length Coding* (CAVLC). Both perform lossless compression by intelligently coding the syntax elements in the video stream based on their probabilities. CABAC compresses data more efficiently than CAVLC but requires more processing at the decoder.

2.1.3 Overview of High-Efficiency Video Coding (HEVC)

The increasing demands on *High and Ultra High Definition* (HD and UHD) videos increases the need for more bandwidth especially in wireless systems [29]. That demand urged the video encoding community to develop the new encoding standard called *High-Efficiency Video Coding* (HEVC) to improve the encoding efficiency while keeping the quality as in H.264 encoding standard [22, 30]. In comparison to H.264, HEVC offers about double the data compression ratio at the same level of video quality. It supports resolutions up to 8192×4320. HEVC introduces many different techniques in order to improve the coding efficiency, including the introduction of an adaptive quad tree coding [22]. The improved compression performance increases the computational complexity due to the new algorithms such as, adaptive quad tree structure, extra intra-prediction modes, and the comprehensive Rate-Distortion Optimization (RDO) calculations in such a structure.

HEVC main goal is to improve coding efficiency in HD video systems. In order to achieve that, HEVC introduces many techniques including an adaptive quad tree structure [22, 31] and extra intra-prediction modes. Unfortunately, these techniques slow down the encoding process. One way to reduce such computational complexity is by predicting the CU size fast. Although, several ideas were proposed, HEVC encoding speed is still slow for real-time applications.

2.1.4 Adaptive Quad Tree Structure of HEVC

HEVC adopted new features, such as more flexible data structure, which includes the *Coding Unit* (CU), the prediction unit, and the transform unit. In the original HEVC encoder, *Rate Distortion Optimization* RDO algorithm is used for the partitioning of the *Largest Coding Unit* (LCU) into CUs. Unfortunately, the computation complexity of RDO is extremely high for real-time application which opens the door for sub-optimal LCU partitions that reduce the encoding time.

Frames in HEVC are partitioned into LCUs of size 64×64 in the adaptive quad tree structure of HEVC [32]. If 64×64 CU split, it is divided into four CUs of sizes 32×32. In addition, each CU of size 32×32 can be subdivided into four CUs with sizes of 16×16. Furthermore, each CU of size 16×16 can be subdivided into four additional CUs with sizes of 8×8. The standard refers to 64×64, 32×32, 16×16, and 8×8 partition by depth 0, 1, 2, and 3, respectively. Figure 2.1 shows quad-tree CTU structure.

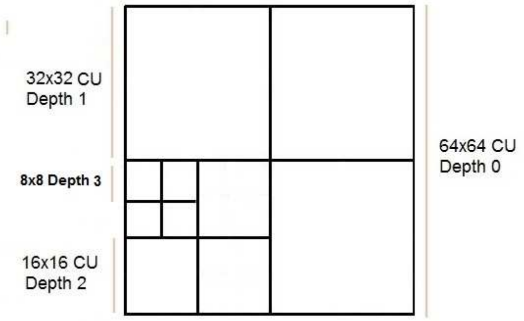


Figure 2.1: Quad-Tree Structure for LCU

Figure 2.2 shows the spatial neighbors for depth 0, 1, and 2. The figure shows that the block currently being processed $CU_{current}$ is surrounded by spatial neighbors with the same size. The encoding of a frame in HEVC is processed in z-order from left to right and top to bottom. Figure 2.3 shows the temporally co-located neighbor of the $CU_{current}$ in the previous frame which has the same size and the same spatial z-order as the $CU_{current}$ in the current frame.

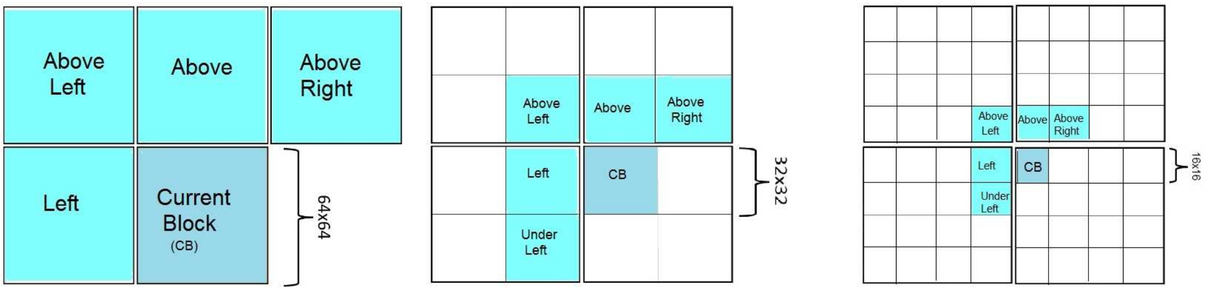


Figure 2.2: Spatial Neighbors of a 64x64, 32x32, and 16x16 CUs [depth 0, 1, and 2]

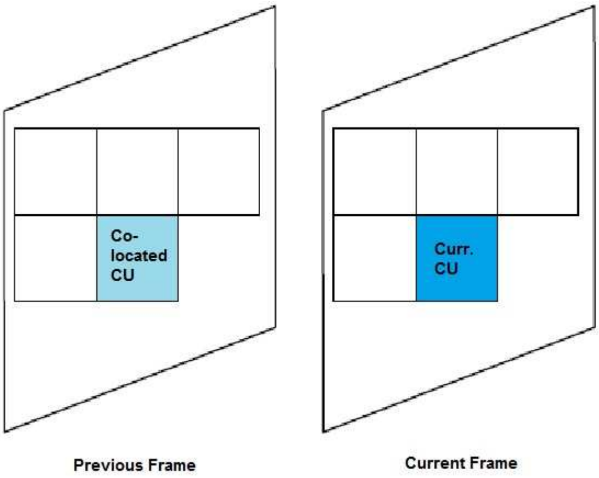


Figure 2.3: Temporal Neighbor (Co-located)

2.1.5 *Related Work on LCU Partitioning*

To increase the encoding speed of HEVC encoder, many techniques have been proposed by different studies. Some of these studies focus only on partitioning process [25, 33, 29, 34, 35, 36, 37]. Study [25] suggested an approach based on how much information is contained within the block, which is measured by a metric called Shannon entropy. For simplicity, we call that algorithm **entropy-based algorithm**, we also call Shannon entropy **entropy** in this dissertation.

The entropy value for a CU indicates the amount of information that must be encoded by an image/video compression algorithm [24, 38]. The CU entropy value (E_{cu}) can be calculated by **Shannon Entropy Equation** [24] as follows:

$$E_{cu} = -\sum_{i=0}^{N} \frac{f(i)}{N} \times \log_2 \frac{f(i)}{N}, \tag{2.3}$$

where N and f are the total number of pixels in the CU and the total number of the occurrences of the pixel value i (frequency of i), respectively. This equation was introduced in a *Mathematical Theory of Communication* paper by Claude E. Shannon in 1948 [23]. To increase redundancies among pixels within the same CU, quantization is applied for eliminating the noise before evaluating Equation (2.3). Figure 2.4 shows the pseudocode for the proposed method by [25]. The entropy-based algorithm improved the encoding speed to be faster than all other existing algorithms that intend to do so. The encoding speed is 3.5 faster than the original RDO algorithm with acceptable average bitrate. In addition, the average quality degradation for the proposed method in terms of the PSNR is negligible.

```
Start with the first frame
 I. Start the encoding process of the first LCU in the frame
    II. Calculate and store each entropy value of all possible CUs in LCU ($85$ different CUs)
      Calculate the total average entropy of all possible CUs in the LCU which is the sum of all entropies divided by $85$
      III. Start at depth 0 with CU of the size of $64 \times 64$
      IV. Use the stored entropy value of the CU
        1. If the entropy of CU $> 3.5$, SPLIT to next CU
        2. Else, if the entropy value of CU $< 1.2$, TERMINATE
        3. Otherwise
          a. If the entropy value of CU is within $0.15$ of average entropy, TERMINATE
          b. Else, SPLIT
       V. Go to next CU and repeat steps IV until LCU encoding is done
      Go to next LCU and repeat II through V until frame is done
Go to next frame until video segment is done
```

Figure 2.4: Pseudocode of the Entropy-Based Algorithm

Study [36] proposed CU early-termination algorithm that takes advantage of the correlations between the *Mean Square Error* (MSE) of prediction residuals and the splitting decision in the current CU level. For each CU level, MSE between the prediction block and the origin block is compared with an adaptive threshold. The CU splitting process is terminated early according to that threshold. According to the paper, the proposed algorithm achieves up to 34.83% average encoding time reduction.

Study [33] proposed fast CU size decision based on the depth of the spatial and temporal neighbors. Each of the splitting and termination decisions is based on two main conditions. Splitting decision for $CU_{current}$ is made if either the temporally co-located CU or each of the spatial neighbors has smaller CUs. On the other hand, the termination decision is made if either the temporally co-located CU or 3 or more neighboring CUs does not have any smaller CU. For both splitting and termination, the current processed frame should not be I frame. The authors of [33] claimed a reduction in encoding time of 43% compared to the original HM5.0 encoder for the HD test sequences.

Using *Support Vector Machine* (SVM), study [34] proposed a CU splitting fast termination algorithm. CU splitting is modeled as a binary classification problem, on which SVM is applied. The paper claimed that the proposed algorithm can achieve about 41.9 % time saving under the low delay profile setting compared with the HEVC reference software.

In trying to avoid some of the limitation of the depth-based partition approach [33], study [39] used *Total number of Blocks* (TnB), where it made the decision of splitting/termination based on the total number of blocks in the neighbors CUs. TnB is based on the number of sub CUs that contained within a certain CU. The author claimed that the total number of blocks provides more insight into $CU_{current}$ structure of the neighbors which make the decision of termination/splitting more accurate compared to the RDO method than the depth approach. TnB algorithm made the decision as follows: first, calculate the *Splitting Possibility* (SP) as the total number of sub-blocks in all the neighbors of $CU_{current}$ divided by the maximum possible number of sub-blocks in the same CUs. Second, terminate (consider the current size for $CU_{current}$) if either SP is less than 0.1 for depth 1, or it is less than 0.4 for depth 2. The author claimed a reduction of 44.89% in encoding time compared to RDO that is implemented in the original software.

2.2 Adaptation of Live Video Streams in Computer Vision Systems

Most research on video surveillance focused on developing robust CV algorithms for the detection, tracking, and classification of objects [40, 41] (and reference within) and the detection and classification of unusual events [42] (and reference within). No work, however, considered the bitrate-energy-accuracy tradeoffs in CV systems, including AVS.

As shown in Figure 2.5, CV systems, such as AVS, include multiple video sources (i.e., cameras and/or sensors) which stream videos to a monitoring station. Live video streaming consists of three main phases at the video source side: capturing, encoding, and transmission. Due to the complexities of intra-prediction and inter-prediction, which are used to reduce the spatial and temporal redundancies in the video, respectively, the encoding phase incurs the highest level of power consumption [43]. The main video encoders include MPEG-4 and H.264. At the receiver side of general live video streaming systems, the streams are decoded and then displayed. In AVS systems, the monitoring station decodes the videos, potentially upscales them, runs CV algorithms, and performs appropriate actions, such as generating alerts, adapting the sources, etc. The power consumed by the monitoring station is of lesser importance than that by the video sources because the monitoring station is a full-fledged, outlet-powered system [43]. The bitrate of a video stream is related to the required bandwidth and to its quality. The CV accuracy depends on the quality of the video stream, but CV algorithms are less sensitive to quality than human beings [44].

Adaptation involves selecting the desired capturing and encoding parameters of various video sources

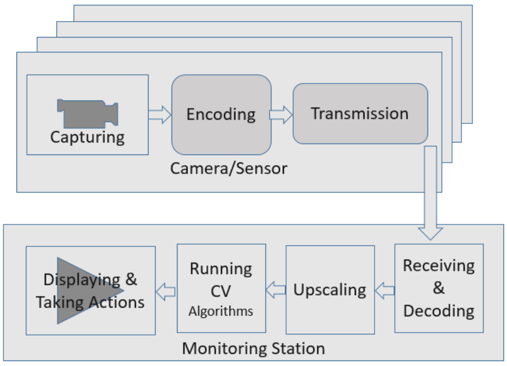


Figure 2.5: An Illustration of Computer Vision Systems

to fit bandwidth and/or energy constraints and/or achieve certain performance. In CV systems, the accuracy of CV algorithm(s) is the most important metric. The main approaches for video streams adaptation include changing the video Signal-to-Noise Ratio (SNR), spatial, and/or temporal parameters. The SNR quality is controlled by changing the quantization parameter, thereby changing the intensity of pixels. Video encoders allow setting the quantization parameter directly or indirectly by setting the target bitrate. The spatial and temporal qualities, however, are controlled by changing the frame size (i.e., resolution or total number of pixels) and frame rate (i.e., number of frames per second), respectively.

For spatially-adapted videos, an upscaling algorithm may be used to restore the videos to their original sizes by the monitoring station before applying the CV algorithm(s). Upscaling uses interpolation to enhance the resolution of an image or video. In this dissertation, we analyze the effectiveness in CV accuracy of popular upscaling (also called super-resolution) algorithms: *Nearest Neighbor*, *Bilinear*, *Bicubic*, *Spline*, and *Lanczos*. The first three algorithms consider the closest pixel, the closest 2×2 pixels, and the closest 4×4 pixels, respectively. Spline and Lanczos consider additional surrounding pixels. New algorithms have been recently proposed in [45, 46].

2.3 *Power Consumption in Live Video Streaming*

Power consumption is a major concern in live video streaming systems in general and in many-to-one video live streaming systems in particular. As shown in Figure 2.5, many-to one streaming systems include multiple video sources (i.e., cameras and/or sensors) which stream videos to a monitoring station. The sources adapt their capturing and encoding parameters based on the current system state, including available resources. The monitoring station receives video streams from all sources, potentially upscales the videos to their original resolutions, decodes the videos, and then runs computer vision algorithms for determining the appropriate actions, such as controlling sources and generating alerts. The process at each source involves three main phases: video capturing, video encoding, and video transmission.

In this dissertation, we develop power consumption models for live video streaming systems in general and analyzes the power consumed by the monitoring station in many-to-one systems. The models capture the impacts of various video capturing and encoding parameters, and thus can help in the dynamic control of various camera/sensor settings to achieve the best overall tradeoff in terms of power consumption, bitrate (and thus bandwidth), and video quality.

Let us discuss next the power consumption in each phase at the source.

2.3.1 Video Capturing Power Consumption

Cameras include image sensors, which are silicon devices that capture images. The most popular sensor type is *Complementary Metal Oxide Semiconductor* (CMOS). It captures light onto an array of light-sensitive diodes, with each diode representing one pixel and converting the light photons into a charge. Each pixel has its own voltage amplifier and can be read directly on an $x - y$ coordinate system. Study [47] characterized the power consumption of a smart sensor, called PARISI (Programmable Analog Retin-like Image Sensor I). The total power consumption for an $N \times N$ sensor with N analog processing units was shown to be given by

$$W_s = c_a\, N^2 + c_b\, N, \tag{2.4}$$

where c_a and c_b are constants. Study [48] developed a power consumption model for CMOS image sensors. The model is similar to [47] but it is more complex and includes systems parameters and not only capturing and encoding parameters.

In this dissertation, we consider the popular CMOS sensors and develops a power consumption model based on extensive experiments. Our developed capturing model is based on that of [47].

2.3.2 Video Encoding Power Consumption

The main video encoders include MPEG-4 Part 2 Standard (or simply MPEG-4) and MPEG-4 Part 10 Standard (or simply H.264). As shown in Figure 2.5, the video encoding process can generally be divided into the following three high-level stages: *Intra- and Inter- Prediction (Estimation)* Stage, *Transformation, Quantization and Their Inverse* Stage, and *Entropy Coding* Stage. In the estimation stage, both intra-prediction and inter-prediction are used to reduce the spatial and temporal redundancies in the video, respectively. The first frame of a sequence or a random access point is typically intra-coded (i.e., without using information from other frames). Each block of pixels in an intra-frame is predicted using previously-encoded neighboring blocks. For all remaining frames of a sequence or between random access points, inter-coding is usually used, employing block motion compensation to predict blocks from other previously encoded frames. The residuals of the intra-prediction and inter-prediction are then transformed to the frequency domain using Discrete Cosine Transform (DCT) in MPEG-4 or Integer DCT in H.264. Subsequently, the transform coefficients are quantized, thereby reducing the overall precision of the coefficients and possibly eliminating high frequency coefficients. The quantized transform coefficients are entropy coded and transmitted together with any possible motion vectors (MVs).

As H.264 is the primary focus of this dissertation, let us now discuss it in more detail. Figure 2.6 shows its processing stages. The main features of H.264 can be summarized as follows. (1) It allows using up to 16 reference frames to achieve high compression ratios, compared with only one in MPEG-4. (2) It uses variable block-size motion compensation, thereby enabling a more accurate segmentation of moving regions and higher compression ratios. The block size ranges from 4×4 pixels to 16×16 pixels, whereas MPEG-4 has a minimum block size of 8×8. (3) It employs a simplified version of the DCT transform. In particular, it uses a 4×4 or an 8×8 Integer DCT transform, whereas MPEG-4 uses an 8×8 DCT. (4) It employs a quantization design, which includes a Logarithmic step-size control for easier bitrate management by encoders and simplified inverse-quantization scaling. (5) It provides two options for entropy coding: *Context Adaptive Binary Arithmetic Coding* (CABAC) and *Context Adaptive Variable Length Coding* (CAVLC). Both perform lossless compression by intelligently coding the syntax elements in the video stream based on their probabilities. CABAC compresses data more efficiently than CAVLC but at the expense of increased processing at the decoder.

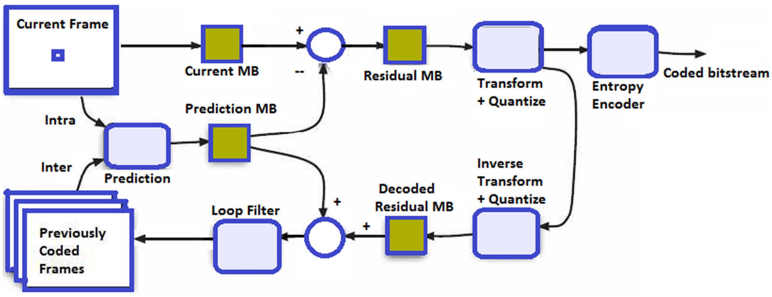


Figure 2.6: Block Diagram of H.264 Encoder

When coding a macroblock, an H.264 encoder can choose from many different intra-modes for I-frames or inter-modes for B- and P-frames. Within each inter-mode, the encoder has a wide choice of possible MVs, leading to a huge number of options for coding a macroblock [27]. The *Rate-Distortion Optimization* (RDO) mode selection is an algorithm for choosing the best coding mode for each macroblock, based on the bitrate and distortion cost. It is used for both intra-prediction and inter-prediction. To select the best encoding mode for a macroblock, the algorithm examines all possible combinations of intra- or inter-modes. The bitrate cost r and distortion cost t are combined into a single cost J by $J = t + g\,r$. The RDO mode selection algorithm attempts to find the mode that minimizes the joint cost J. The tradeoff between bitrate and distortion is controlled by the Lagrange multiplier g. Further details can be found in [27].

Much of the work on H.264 dealt with managing the computation complexity [49, 50]. Study [51] developed a power consumption model in terms of the cycles per instruction and energy per cycle. In [52] studied the power consumption of video streaming from smartphones and mobile devices, but with no model development. Study [53] considered optimization scenarios that determine how to assess the encoding parameters, but the study was limited to Motion Estimation (ME) search algorithms. Specifically, it compared the bitrate, distortion, and cycles per second for each one of these modes. A power-rate-distortion model of a hardware-based encoder was introduced in [54] using the power scalable architecture of H.264, considering only integer and fractional ME search range and I-frame period. Study [55] proposed a joint power-distortion optimization scheme for real-time H.264 video encoding under the power constraint. The encoding modules were divided into basic operation units, such as the sum of absolute differences (SAD) operations. Subsequently, the encoding complexity of basic operation units was determined by summing up the required processor cycles. That study considered only the ME search algorithm and did not study the spatial and temporal effects. Study [5] developed a Power-Rate-Distortion (PRD) framework specifically for a generic video encoder (that applies to H.263). Study [56] developed a model for H.263 by measuring the power consumption of an H.263 encoder running with Full Search and Fast Search ME algorithms as a function of the bitrate, frame rate, and number of macroblocks in the frame.

Dynamic Voltage Scaling (DVS) algorithms reduce energy consumption by changing the processor speed and voltage at runtime depending on the needs of the currently running applications. With DVS technology, the power consumption is a function of processor cycles per second. Therefore, the encoding complexity can be represented as a function of the number of processor cycles per second. As in [57, 5, 56, 58, 59], we consider DVS in the model development.

2.3.3 Video Transmission Power Consumption

Transmission power consumption is affected mainly by the technology or platform, distance, path-loss or environment, and bitrate. The main factors that impact the power consumption in a Wi-Fi platform are the Network Interface Card (NIC) design (including layout, chip design, transmission output power, voltage regulations, and modulation scheme), interactions between NIC and CPU, and software protocol design (such as power management and drivers). Most of the recent work on transmission power consumption focused on video sensor networks. Study [60] developed upper bounds on the lifetime of sensor networks. Study [61] examined the resource utilization behavior of a wireless video sensor and analyzed its performance under resource constraints. Study [62] studied the impact of the transmission power range on energy consumption for wireless sensor networks. Study [51] analyzed the power consumption in video sensor networks, using the model in [61]. Study [63] analyzed the impacts of different transmission power control strategies on wireless sensor networks in general, considering the granularity of power levels.

As discussed earlier, we are interested in developing models in terms of only the video capturing and encoding parameters, and thus we simplify previous transmission models, particularly [61], and adapt them accordingly.

CHAPTER 3 ADAPTATION OF LIVE VIDEO STREAMS IN COMPUTER VISION SYSTEMS

3.1 Introduction

In this chapter, we consider the design of real-time *Computer Vision* (CV) systems in which objects, events, and/or threats are analyzed automatically by running CV algorithms. A primary example of such systems is Automated Video Surveillance. In these systems, multiple video sources (i.e., video cameras and/or sensors) stream videos to a central monitoring systems. These systems have constraints, including those in energy and bandwidth, and thus various video streams should be adapted dynamically based on the available resources and desired performance. The adaptation is achieved by changing the source video capturing and encoding parameters, specifically resolution, bitrate (or quantization parameter), and/or frame rate. Due to their nature, the main performance metric in CV systems is the accuracy of the CV algorithm(s).

Video stream adaptation has been studied extensively in general video streaming systems, but little work has been devoted to CV systems. Most existing work on adaptation considered the video distortion as the primary metric, leading to much literature on rate-distortion characterization and optimization [64, 21, 6, 7] (and references within). As discussed earlier, however, the main objective in CV systems is the CV accuracy. The accuracy can essentially be thought of as the quality perceived by machines, as opposed to the human perceptual quality metrics such as Peak Signal-to-Noise Ratio (PSNR), Mean Squared Error (MSE), and Structural Similarity Index (SSIM) [65]. Only few studies have considered the impact of video streams adaptation on CV accuracy. Study [4] considered the impacts of rate adaptation on accuracy, but only for images and only when SNR adaptation is employed. Study [44] analyzed rate adaptation for only MJPEG videos and did not consider more efficient encoders, such as MPEG-4 and H.264. Moreover, these studies experimented with small datasets and did not consider power consumption. Power consumption is becoming a major concern, especially when the video sources are battery-powered [43] Study [9] utilized CV algorithms to help automatically prioritize video streams to solve bandwidth problem in mobile surveillance systems.

In this chapter, we analyze and compare various video streams adaptation techniques in terms of the detection accuracy, bitrate, and power consumption. The analyzed adaptation techniques include *spatial, temporal*, and *Signal-to-Noise Ratio* (SNR), which adjust the spatial video resolution, frame rate, and quantization parameters, respectively. We also analyze the impact of upscaling spatially-adapted videos to their original sizes by using super-resolution techniques at the receiver before applying the CV algorithm. In addition, the we study the impact of different video streams adaptation combinations.

Furthermore, we present an objective function that captures the overall tradeoff in terms of accuracy, bitrate, and energy consumption. By examining the rates of change in each of these metrics, the objective function also favors the setting with larger subsequent drop in accuracy, smaller subsequent drop in bitrate, and smaller subsequent drop in energy.

We report the results based on over $16,000$ real experiments, with 9 standard video sequences and a newly assembled dataset of 300 actual security and news videos in a wide variety of spatial resolution. These videos have a total of more than 19 hours of recording time. We experiment with two system types. The first includes a computer running FFmpeg encoding with an external camera, whereas the other includes a real surveillance camera with encoding performed by a System-on-Chip (SOC). We study the videos in both H.264 and MPEG-4 encoding standards and assess both the *detection index* and *false positive index*. Finding the probability of false positive for videos is a hard task since it requires human observations of the videos with the imposed markings of detected faces and manual recordings of the results.

The **main contributions** of this part of the dissertation can be summarized as follows: (i) analyzing rate adaptation for H.264 and MPEG-4, (iii) considering both the detection accuracy and power consumption, (ii) analyzing the performance of various upscaling algorithms, (iii) analyzing the impact of the combinations of SNR and spatial adaptations on bitrate, power consumption, and detection accuracy, and (iv) developing an objective function that captures the overall tradeoff in the adaptation process.

The main results can be summarized as follows. Section 3.2 analyzes various video streams adaptation techniques. Section 3.3 discusses the performance evaluation methodology and Section 5.4 presents and analyzes the main results. Finally, conclusions are drawn in the last section.

3.2 Video Stream Adaptation

3.2.1 Analysis of Various Stream Adaptation Techniques

Providing detailed analysis of various video streams adaptation techniques in their rate-accuracy and rate-energy characteristics is required for designing effective computer vision systems, such as AVS. In this chapter, we analyze and compare various video streams adaptation techniques in terms of detection accuracy, required bandwidth, and power consumption. The analyzed streams adaptation techniques include spatial, temporal, and SNR. With SNR, the intensity levels of the frame can be controlled by changing the target bitrate or directly changing the quantization parameter, both of which are studied in this chapter. For spatially-adapted videos, we analyze upscaling the video frames to increase the accuracy at the destination by experimenting with five super-resolution algorithms.

We study the videos in both H.264 and MPEG-4 encoding standards and analyze the results in terms of accuracy, bitrate, and source power consumption. For accuracy, we consider three metrics: *average detection accuracy*, *number of detections* (un-normalized detection accuracy), and *false positive index*. The first metric, also called *detection index*, can be defined as the number of correctly detected faces divided by the total number of faces in all video frames. It is used for the standard sequences, whereas the second metric is used for the dataset of actual videos because the total number of faces in each video is unknown. The false positive index is the probability of false positive.

Table 3.1: An Example for Illustrating the ABE_{OF} Model

First Row of Adaptation Matrix (Accuracy, Bitrate, Energy)	0.99, 32, 2.33	0.98, 30, 2.29	0.30, 26, 2.28	0.05, 11, 2.25
Second Row of Adaptation Matrix (Accuracy, Bitrate, Energy)	0.95, 30, 2.29	0.97, 29, 2.28	0.28, 24, 2.21	0.04, 10, 2.18
First Row of Objective Matrix	0.68	0.96	0.27	0.04

3.2.2 Proposed Accuracy-Bitrate-Energy Objective Function

Since different adaptation techniques exhibit different characteristics in terms of accuracy, bitrate, and power consumption, combining various techniques can be greatly beneficial. Therefore, we develop an objective function called *Accuracy-Bitrate-Energy Objective Function* (ABE_{OF}), which helps in determining the specific adaptation or adaptation combination that can be employed. This objective function considers the accuracy, bitrate, energy consumption, and rate of change of each of them. It takes an $N \times M$ *adaptation matrix*, with rows representing different quantization parameters (or different bitrates) in increasing order and columns representing different resolutions in decreasing order. Each entry is a tuple with accuracy, bitrate, and energy values for the corresponding adaptation. Applying ABE_{OF} on the adaptation matrix produces an *objective matrix*, which includes the overall objective value for each adaptation combination, with larger values being preferred. In addition to favoring higher accuracy, lower bitrate, and lower energy, the objective function examines the rates of change in each of these metrics and favors the setting with larger subsequent drop in accuracy, smaller subsequent drop in bitrate, and smaller subsequent drop in energy. To illustrate the impact of the rate of change in accuracy consideration, if two consecutive rows in the adaptation matrix are as shown in the first two rows of Table 3.1, the second tuple in the first row will have the largest ABE_{OF} value (i.e. 0.96) because of the big drop in accuracy afterwards. The third row in the table shows the first row of the objective matrix.

The overall function can be determined as follows:

$$ABE_{OF} = \frac{(A+1)^m.(|g_A|+1)^n}{(B+1)^p.(|g_B|+1)^q.(E+1)^u.(|g_E|+1)^v}, \tag{3.1}$$

where parameters A, B, E, g_A, g_B, and g_E are the normalized detection accuracy, normalized bitrate, normalized consumed energy, and the rate of change in each of them, respectively. Consequently, these parameters have values in the closed real interval [0, 1]. Constants m, p, u, n, q, and v are the assigned weights, with values between and including 0 and 1, for the normalized accuracy, normalized bitrate, normalized consumed energy, and their rates of change, respectively. Each of these weights is used as exponent (i.e. power) in the equation to obtain a value of 1 (i.e. no effect) for the associated factor (bitrate, accuracy, energy, etc.) when this factor is not considered by the objective function.

We have the choice of ignoring the rate of change in accuracy, bitrate and/or energy consumption in the ABE_{OF} by choosing zero for n, q, and/or v, respectively. In addition, we can ignore the effect of bitrate and energy consumption by setting p and u to zeros. The value of each term in $(A+1)^m$, $(|g_A|+1)^n$, $(B+1)^p$, $(|g_B|+1)^q$, $(E+1)^u$, and $(|g_E|+1)^v$ is within the closed real interval [1, 2]. The added one to each of these terms is to have no effect on ABE_{OF} if the associated factor is zero in the adaptation matrix.

The rates of change can be modeled by the derivative, which can be the second-order derivative, the first-order derivative, or the diagonal difference of a two-dimensional function $f(x,y)$. We use the diagonal difference because it achieves the lowest execution time while producing comparable results. The diagonal difference for function f between two successive points (x,y) and $(x+1,y+1)$ can be expressed as follows:

$$g_f(x,y) = f(x+1,y+1) - f(x,y). \tag{3.2}$$

We discuss next how the weights can be selected and then analyze the time complexity of assessing ABE_{OF}.

Considerations in Weight Assignment

The weights m, p, u, n, q, and v can be preset by system administrators or preferably change dynamically based on the current states of the system and monitored site. The weights for accuracy, bitrate, and energy can be adapted based on the detected events/objects, available bandwidth, and remaining battery level, respectively. As the accuracy in automated video surveillance is of utmost importance, its weight (m) should generally be set to a high value and increase even more when critical objects or events are detected. Similarly, the weight for consumed energy (u) can be based on the source battery level, whereas the weight for the bitrate (p) can be based on the bandwidth utilization of the medium. For example, u can be set to 0 if the source battery is beyond a certain threshold (such as 70%), to 1 if the charge is below another threshold (such as 30%), and to a value inversely proportional to the battery charge otherwise. Likewise, p can be set to 0 if the bandwidth utilization does not exceed a certain threshold (such as 50%), to 1 if the utilization exceeds another threshold (such as 80%), and to a value proportional to the utilization otherwise. The bandwidth utilization can be measured in terms of the *smoothed channel busy ratio*, which is the percentage of the average time the channel is indicated busy during a given instance of time, as specified in standards IEEE 802.11p and SAE J2945.1 [66].

The rate of change in the metrics (i.e., accuracy, energy, and bitrate) and are of secondary importance to the actual metrics, and thus their weights can be set as fractions of the weights of the corresponding metric weights.

Overall Process and Time Complexity

The proposed process for finding the best adaptation can be summarized as follows. During system calibration and potential re-calibration, the system records a short video, including the targets to-be detected or recognized. The system then encodes the recorded video using various adaptations in terms of resolution and quantization parameter or resolution and bitrate. Subsequently, the adaptation matrix is built by finding the accuracy, consumed energy, and bitrate for each adaptation. To avoid manual inspection of the video streams, the accuracy can be determined relative to the adaptation with the highest accuracy. The bitrate can be obtained from the encoded video, whereas the power consumption can be estimated based on the spatial resolution and quantization parameter (or bitrate), using the analytical models in [43]. After the adaptation matrix is generated, the objective function is applied to obtain the objective matrix. Finally, a search for the maximum value in the objective matrix is conducted. This process is repeated only when there are considerable changes in the system (such as used encoder and system parameters) or the monitored site.

The time complexity depends on the number of generated adaptations. Fortunately, the number of different adaptation combinations is practicality limited due to a small number of supported resolutions and bitrates. The quantization parameter can also be changed in certain steps and within a narrow range of practically appropriate values. In most cases, we need to examine only 10 to 25 different adaptations. For each adaptation, we need to encode the short video, find various metrics, run the objective function, and then find the maximum. Assuming, N is the number of adaptations in one parameter/dimension (resolution or bitrate/quantization), the time complexity is $O(N^2)$. Considering the narrow search space, searching for the maximum value in the objective matrix can simply employ the brute-force approach. In addition, the adaptation matrix can be pre-filtered based on our knowledge of the available bandwidth by eliminating those adaptations that exceed the available bandwidth.

Whenever the weights in the objective function change dynamically, the objective matrix should be recomputed and then the adaptation with the largest value will be selected. This process is invoked more frequently than that during calibration and re-calibration, but no encoding is required at various adapta-

Table 3.2: Characteristics of Selected Standard Video Sequences [Frame rate: 30 fps]

Sequence	Duration (s)	Resolution	# Frames
Silent	10	CIF	300
Akiyo	10	CIF	300
Deadline	45.8	CIF	1374
SignIrene	18	CIF	540
vtc1nw	12	4SIF (VGA)	360

Table 3.3: Characteristics of the Collected Video Dataset [Frame rate: 30 fps]

Description	# Videos	Duration (s)	Resolution	# Frames
Security	100	2857	QVGA	85,710
News	200	66096	QVGA	1,982,880
Total	300	68953	QVGA	2,068,590

tion. The same time complexity applies but the actual computational overhead is negligible compared with other tasks performed by the monitoring station, including running the CV algorithms.

3.3 Performance Evaluation Methodology

3.3.1 Used Video Datasets

We use both standard video sequences and 300 real security and news videos of varying quality. Table 4.8 summarizes the characteristics of the selected video sequences. These sequences are selected such that each video frame contains exactly one face, thereby simplifying the computation of the detection and false positive indices. We do not consider sequences with resolutions higher than 4 SIF due to the considered AVS system and since CV algorithms are not as sensitive as humans to resolution (and quality), as demonstrated in the reported results. Lower resolutions and qualities can be used to save power consumption without significantly sacrificing CV accuracy. We collected the 300 real videos from YouTube by searching for keywords, such as security, hidden camera, speeches, and news, and then carefully selecting videos with faces in most of the frames. Table 3.3 summarizes the characteristics of the collected real videos. As discussed later, for certain power consumption experiments, we used a 22-minute video called “Baby Animal Songs by Kidsongs”, also available on YouTube.

3.3.2 Generating Simple Adaptations

We generate various adaptations by applying each adaptation technique individually. For the video sequences, we use the yuv2avi-p2 program to convert the sequences from raw YUV to AVI format while preserving the original quality. We use FFmpeg to convert all videos to different spatial, temporal, and SNR qualities. The adaptations have three dimensions: spatial, temporal, and SNR. With SNR, the intensity levels of the frame can be controlled by changing the target bitrate or directly changing the quantization parameter. We experiment with both options in the experiments.

Each video is evaluated at 48 or 310 different quality levels for single adaptation or combined adaptation, respectively, leading to a total of over $16,000$ experiments. The quality levels are obtained by varying the spatial resolution (i.e., frame size), temporal resolution (i.e., frame rate), and SNR (i.e., quantization parameter or target bitrate). The frame size is varied to lower settings of the original frame sizes and these settings vary based on the video type. The temporal resolution is varied from 1 to 30 fps (frames per second) for all videos. Furthermore, the SNR setting is varied by changing the target bitrate from 1 Kbps to 240 Kbps for all videos. The videos are encoded in a single pass because of the streaming environment in AVS.

3.3.3 Generating Adaptation Combinations

To analyze combining spatial and SNR adaptations, we encode videos to all the combinations of 10 different resolutions and 31 different SNR qualities. We consider 4 CIF sequences: Foreman, Mother-daughter, News, and Silent. We also consider both MPEG-4 and H.264 encoding standards.

3.3.4 Conducting Experiments

We experiment with two system types: *System S* and *System H*. System S includes a computer running FFmpeg encoding, whereas System H includes a real surveillance camera with encoding performed by a System-on-Chip (SOC). The accuracy and bitrate depend primarily on video content, the used encoder, and selected capturing and encoding parameters and thus do not considerably change from one platform to another, whereas the consumed energy depends on the used hardware as well as the video content and capturing and encoding parameters. We consider both H.264 and MPEG-4. For each adaptation or adaptation combination, we measure the required power consumption and determine the bitrate of the encoded video. Subsequently, we decode the video and determine the CV accuracy. For the spatially-adapted videos, we analyze the impact of upscaling the videos to their original sizes before running the CV algorithm at the receiver, using five super-resolution algorithms. We use FFmpeg to upscale the spatially-adapted videos using lossless compression to ensure that no loss in quality happens due to compression. We consider face detection, which is a major CV algorithm, and use the Viola-Jones algorithm [67], as implemented in OpenCV library. The consumed power is measured by "Watts Up? Pro ES AC" Graphic Timer Watt meter.

We use three metrics for the detection accuracy: *average detection accuracy*, *number of detections* (un-normalized detection accuracy), and *false positive index*. These metrics are defined in Subsection 3.2.1. The false positive index (i.e. the probability of false positive) is determined by manually observing in slow motion a small subset of the videos with the imposed markings of detected faces and recording the results.

In System S, which includes a Dell Inspiron 1525 laptop with an Intel Core 2 Duo CPU (Model T5750) running at 2.00 GHz with 3.00 GB memory, we use two experimental setups: *Experimental Setup I* and *Experimental Setup II*. Both setups are used to collect accuracy, bitrate, and power consumption results, but the first provides the encoding power consumption, whereas the second provides the aggregate power consumption due to capturing, encoding, and transmission. The encoding power consumption is generally more than 90% of the aggregate. In both setups, to minimize the effect of other processes while running the experiments, we run the computer with a bare minimum set of processes and drivers. In addition, each experiment is repeated four times, and then the overall results are averaged.

Furthermore, the power consumption due to other system processes running on the computer is measured before each experiment and then subtracted from the total power consumption. In Experimental Setup II, an external Webcam Pro 9000 camera is used to feed a raw video, which is then compressed with FFmpeg using H.264 or MPEG-4. The camera is directed to a computer screen playing the video "Baby Animal Songs" (from the beginning to the end) to ensure that the experiments can be repeated without changes in the video content. The distances between the camera and the monitoring station is within 1 meter. We initially measure the aggregate power. As in [43], we follow the following procedure to separate the power consumption due to each phase. (1) We measure the power consumption of capturing and encoding and then subtract it from the aggregate power consumption to determine the transmission power consumption. (2) We stream the stored video (thereby no capturing is involved) from the computer to the destination, measure the power consumption for this task, and then subtract it from the aggregate power consumption to assess the capturing power consumption. (3) We subtract the capturing and the transmission power consumption from the aggregate power consumption to determine the encoding power consumption.

In System H, we use a CMOS networked surveillance camera and refer to the setup as *Experimental Setup III*. The setup is similar to Experimental Setup II except for the usage of the the Vivotek IP7139 surveillance camera, with built-in 802.11g. The aggregate power consumption results are based on the highly accurate power model developed using the same camera in [43]. Note that the accuracy and bitrate are essentially the same as those in the first system for the same capturing and encoding parameters.

In each experiment involving adaptation combination, the ABE_{OF} constants m, p, u, n, q, and v represent the weights for the normalized detection accuracy, normalized bitrate, normalized consumed energy, and their rates of change, respectively. These weights can be selected based on operator preferences and can be changed dynamically. As a case study, we set the weights as follows: $m = 1$, $n = 0.1$,

Table 3.4: Comparing Upscaling Algorithms in % Detection Accuracy

Kbps	None	Neighbor	Bilinear	Bicubic	Spline	Lanczos
66	0.00	37.185	40.375	40.68	41.475	41.04
220	0.00	86.485	85.27	88.135	88.395	88.07
257	18.125	91.15	90.13	90.16	90.385	90.76
AVG	6.04	71.61	71.93	72.99	73.42	73.29

$p = 0.2$, $q = 0.2$, $u = 0.2$, and $v = 0.2$ for MPEG-4 in the two evaluated systems and $m = 1$, $n = 0.1$, $p = 0.1$, $q = 0.1$, $u = 0.1$, and $v = 0.1$ for H.264.

3.4 Result Presentation and Analysis

For experiments with individual adaptation techniques, the video parameter that is unchanged by the adaptation technique is set to its largest value. Unless otherwise stated, the average results for the sequences in Table 4.8 are reported.

3.4.1 Effectiveness of Upscaling Spatially-Adapted Videos

Let us first discuss the effectiveness of upscaling spatially-adapted videos to their original sizes. Table 3.4 compares various upscaling algorithms in terms of the achieved detection accuracy. Only the results for H.264 are shown since MPEG-4 videos exhibit similar characteristics. The video sequences in Table 4.8 are treated as one long sequence, and the overall detection accuracy is reported. Upscaling algorithms can improve the detection accuracy by a factor of 12 on the average. The best performers are Bicubic, Spline, and Lanczos, with Spline achieving the highest detection accuracy. These three algorithms vary in the detection accuracy by at most 0.60% on the average. Based on the tradeoff between accuracy and time complexity, Bicubic is the best overall performer, and thus it will be assumed from this point on, unless otherwise indicated.

3.4.2 Comparing Video Encoding Adaptation Techniques in Detection Accuracy

Figure 3.1 demonstrate that SNR adaptation and the spatial with upscaling exhibit the best rate-accuracy characteristics. Therefore, changing the bitrate by varying the quantization parameter or the frame resolution has generally the least negative impact on detection accuracy. Temporal adaptation performs worse than spatial adaptation because when the faces in the dropped frames will have no chance to be detected.

Figures 3.1(a) and 3.1(b) compare the rate-accuracy characteristics of the video streams adaptation techniques under two selected groups of video sequences. Since each frame has exactly one face, the detection accuracy changes linearly with the frame rate, but the bitrate is not linear with the frame rate because of the employed compression standards, which exploit temporal correlations among successive frames. For the surveillance, security, news and speech videos, we use the number of detected faces as a metric since the number of faces is unknown. The surveillance and security videos are treated as one long video and then the total number of detected frames is reported. Also, news and speech videos are treated as one long video and then the total number of detected frames is reported.

Figure 3.1(c) compares the four video streams adaptation techniques for news and speech videos. These videos have collectively more than 2 million faces in their frames. The rate-accuracy curves are similar to those of the standard sequences.

Surveillance and security videos are the closest to those that we would expect in AVS systems. Figure 3.1(d) compares the three video streams adaptation techniques for the 100 surveillance and security videos. These videos have collectively more than $60,000$ faces in their frames. Since the quality of these videos is generally lower than news and speech videos, all adaptations have somewhat worse rate-accuracy curves. In addition, the relative performance among different adaptation techniques remains unchanged, but the gap between temporal and spatial adaptations becomes narrower, and the gap between SNR adaptation and spatial with upscaling becomes significantly wider. As explained earlier, the latter gap is somewhat exaggerated. Since the false positives are not considered, this gap is expected to be somewhat smaller as suggested by Figure 3.2.

Figure 3.2 compares the false positive index of various techniques. Spatial with upscaling achieves generally the best in this metric. The variation in the false positive index is due to the reduced quality of the background images.

3.4.3 Comparing Video Encoding Adaptation Techniques in Power Consumption

In AVS applications, power consumption at the video sources is usually a primary concern because these sources may be battery-operated video cameras or sensors. The power consumption in the capturing phase generally depends linearly on the total number of pixels in the video [43], which is equal to the frame rate times the number of pixels in each frame. Thus, spatial and temporal streams adaptations are expected to require lower capturing power consumption than the SNR. In contrast, the power consumption in the transmission phase depends on the achieved video bitrate. Among the three phases, the power consumption in the encoding phase is the most significant. Through actual experiments, we compare the three video streaming adaptation techniques in terms of the aggregate power consumption in capturing, encoding, and transmission stages. Figure 3.3 shows that spatial and temporal adaptations lead to lower overall power consumption as they reduce the power consumption in the encoding and capturing phase. The power consumption results vary with the implementation, but the general behavior will not change as long as the hardware employs dynamic voltage scaling. The results for MPEG-4 exhibits a similar behavior and thus not shown.

3.4.4 Analysis of Combining SNR and Spatial with Upscaling Adaptations

Since spatial adaptation with upscaling and SNR adaptation are the best performers, let us now study how they can be combined. From this point on, the figures show the normalized accuracy (*N. Accuracy*) and the normalized consumed energy index (*N. Consumed Energy*). For the bitrate, however, we show what we call *manipulated normalized bitrate* (*M. N. Bitrate*), which is equal to $\frac{\sqrt{r}}{\sqrt{r_{max}}}$, where r and r_{max} represent the bitrate and maximum bitrate, respectively. This manipulation fits all the greatly varying adaptation matrix bitrates in a smaller range and thus show them clearly.

Figure 3.4 shows the accuracy, bitrate, and energy consumption as a function of both quantization parameter and resolution. In Figure 3.4(a), we notice that the accuracy does not change significantly by varying the quantization parameter, but it changes dramatically by varying the resolution below 50%. On the other hand, in Figure 3.4(b), the bitrate dramatically changes for low quantization parameters. In Figure 3.4(c), the energy consumption is reduced when decreasing the quantization parameter and/or the resolution by up to 50%. Based on these results, intuitively, we should avoid low quantization parameters and resolutions lower than 50% of the original because of the high rate of change in the bitrate and accuracy for low quantization parameters and for low resolution, respectively.

3.4.5 Analysis of Utilizing the Proposed Objective Function

Let us now discuss the application of the developed ABE_{OF} on combining spatial adaptation with upscaling and SNR adaptation. Figure 3.5 shows the ABE matrix produced by ABE_{OF} when varying both the resolution and SNR (i.e., quantization parameter or scaling factor).

Figures 3.6 and 3.7 compare the effectiveness of three SNR selection strategies: *Highest SNR*, *Lowest SNR*, and *Model SNR*. The first two refer to selecting the highest and lowest SNR settings, respectively, with the highest settings corresponding to setting the quantization parameter/scaling factor to their smallest values in the studied range, thereby producing the highest quality and accuracy. Note that the first method produces the best accuracy whereas the second produces the best bitrate and energy consumption. Model SNR, however, is produced by setting the quantization parameter/scaling factor according to ABE_{OF} for the given resolutions. The results demonstrate how the ABE_{OF} achieves an accuracy close to the best accuracy while greatly reducing the bitrate and energy consumption.

Figures 3.8 and 3.9 compare the effectiveness of three resolution selection strategies: *Lowest Resolution*, *Highest Resolution*, and *Model Resolution*. The first two refer to selecting the lowest and highest resolutions in the studied range, respectively. Note that the first method produces the best bitrate and energy consumption, whereas the second produces the best accuracy. Model Resolution, however, is resolution according to ABE_{OF} for the given SNR settings. These results demonstrate that ABE_{OF} achieves an accuracy close to the best accuracy while greatly reducing the bitrate and power consumption.

Figures 3.10 and 3.11 analyze the effectiveness of three combinations of both resolution and SNR: *Lowest Combination*, *Highest Combination*, and *Model Combination*. The first two set both parameters to generate the worst and the best qualities, respectively, within the studied range, whereas Model Combination sets the parameters in accordance with ABE_{OF}. The results include three distinct regions or clusters: one with the highest accuracy values, one with the lowest bitrates and energy consumption, and one with the best tradeoff produced by ABE_{OF}. These results demonstrate the great benefits of efficient adaptation combinations. By combining SNR and spatial adaptation in H.264, a point can be reached where the energy consumption can be reduced by 80% and the bitrate by 98%, while maintaining the accuracy within 10% of the highest possible accuracy. In MPEG-4, however, the power consumption can be reduced by 60% and the bitrate by 99% while maintaining the accuracy within 5% of the highest possible value.

Figure 3.12 analyzes the effectiveness of *Model SNR*, *Model Resolution,* and *Model Combination* in bitrate-accuracy and bitrate-energy characteristics. The figure compares the effectiveness of using ABE_{OF} for SNR adaptation, resolution adaptation, and their combination. These results demonstrate the great benefits of combining adaptation strategies in an efficient manner.

Finally, let us analyze the results with the surveillance camera. Figure 3.13(a) analyzes the effectiveness of *Lowest Resolution*, *Model Resolution*, and *Highest Resolution*. These results a exhibit similar behavior as those in Figure 3.9(a), with the main difference being in the actual values of power consumption. Figure 3.13(a) analyzes the effectiveness of *Lowest Combination*, *Model Combination*, and

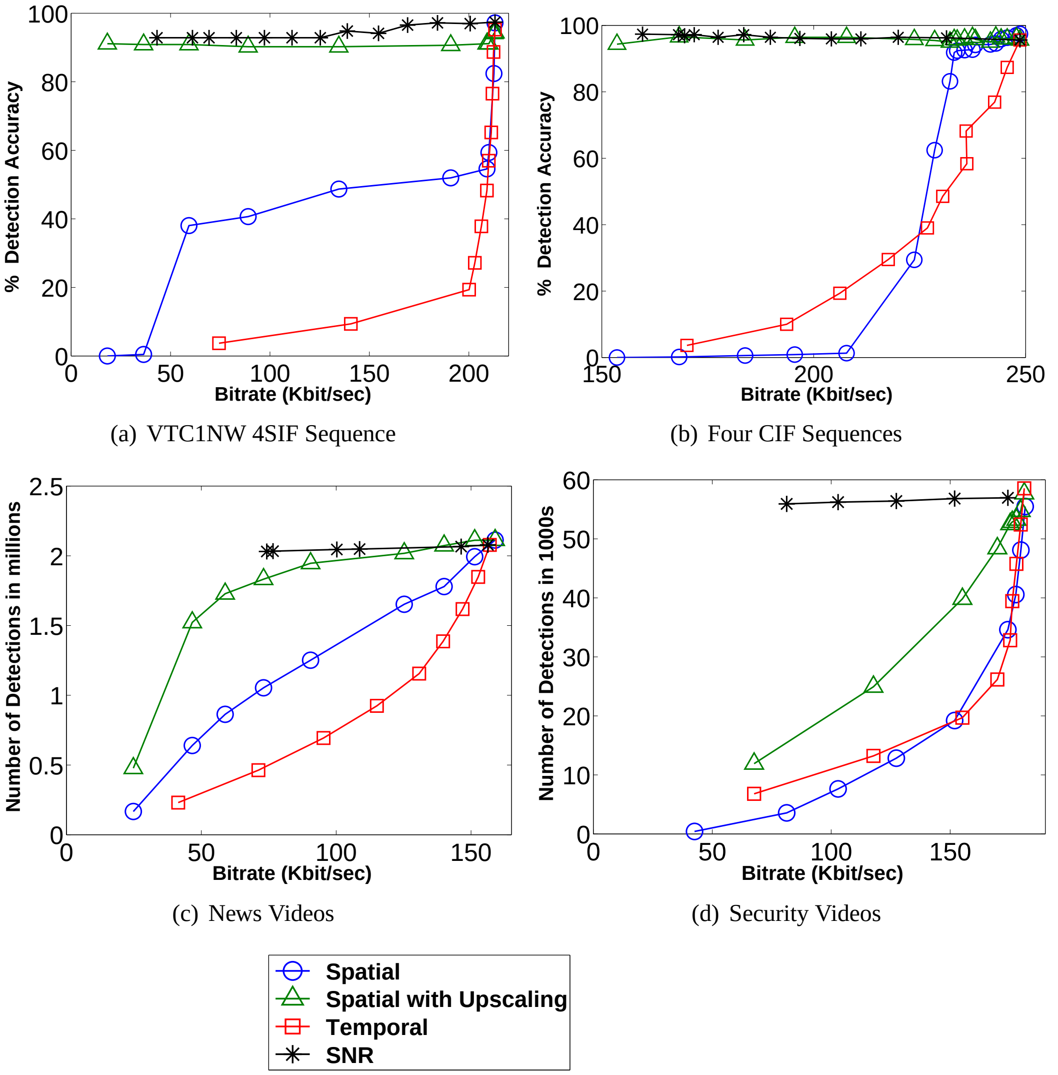
Highest Combination. These results are similar to those in Figure 3.10(b).

3.5 Conclusions

We have analyzed the rate-accuracy characteristics of four video adaptation techniques (spatial, spatial with upscaling, temporal, and SNR) by conducting actual experiments with both H.264 and MPEG-4 encoding standards, considering nine standard sequences and a dataset of 300 real security, and news videos. The results show that SNR adaptation generally achieves the best rate-accuracy characteristics, followed by spatial with upscaling, but the latter performs better in terms of the false positive index. We have compared the performance of five upscaling algorithms. The results show that upscaling provides outstanding improvements in the detection accuracy, but various upscaling algorithms perform close to one other. The Bicubic algorithm provides the best compromise between accuracy and complexity.

We have also analyzed the rate-energy characteristics of spatial, temporal, and SNR video streams adaptations by conducting actual experiments. The results show that SNR adaptation leads to significantly higher power consumption than spatial and temporal adaptations. Therefore, when power consumption at the video sources is a primary concern (such as in AVS systems with battery-operated video cameras and/or sensors), spatial adaptation with later upscaling at the receiver is a reasonable choice as it provides close performance to SNR in terms of detection accuracy but with much lower power consumption. Combining SNR adaptation and spatial adaptation with later upscaling at the receiver, however provides the best overall tradeoff.

Subsequently, we have presented an objective function that captures the overall achieved tradeoff in terms of accuracy, bitrate, and energy consumption. The objective function favors higher accuracy, lower bitrate, and lower energy. By examining the rates of change in each of these metrics, the objective function also favors the setting with larger subsequent drop in accuracy, smaller subsequent drop in bitrate, and smaller subsequent drop in energy. The weights for accuracy, bitrate, and energy consumption can be set based on operator preferences but within a set of acceptable ranges and can also be adapted based on detected events/objects, available bandwidth, and remaining battery level, respectively. The results suggest that we should avoid low quantization parameters and resolutions (specifically resolutions lower than 50% of the original) because of the high rate of change in the bitrate and accuracy, respectively. The objective function can be used in the case of a single adaptation or multiple adaptations. For single adaptation, the results demonstrate that the objective function achieves an accuracy close to the best accuracy while greatly reducing the bitrate and power consumption. For multiple adaptations, the results demonstrate the great benefits of efficient adaptation combinations. By combining SNR and spatial adaptation in H.264, a point can be reached where the energy consumption can be reduced by 80% and the bitrate by 98%, while maintaining the accuracy within 10% of the highest possible accuracy. In MPEG-4, however, the power consumption can be reduced by 60% and the bitrate by 99% while maintaining the accuracy within 5% of the highest possible value.



(a) VTC1NW 4SIF Sequence

(b) Four CIF Sequences

(c) News Videos

(d) Security Videos

(e) Legend

Figure 3.1: Rate Accuracy Curves of Considered Sequences and Actual Security Videos

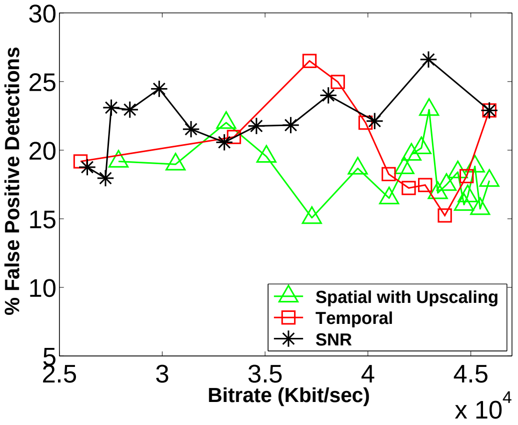


Figure 3.2: False Positive Index

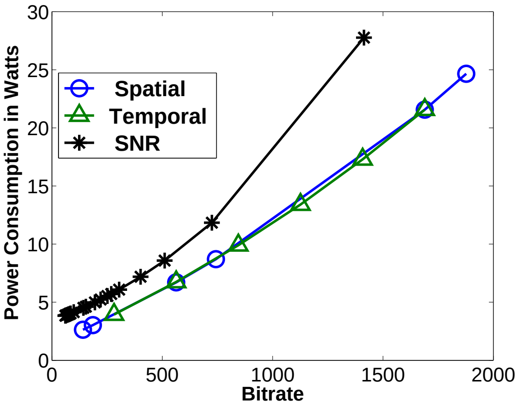
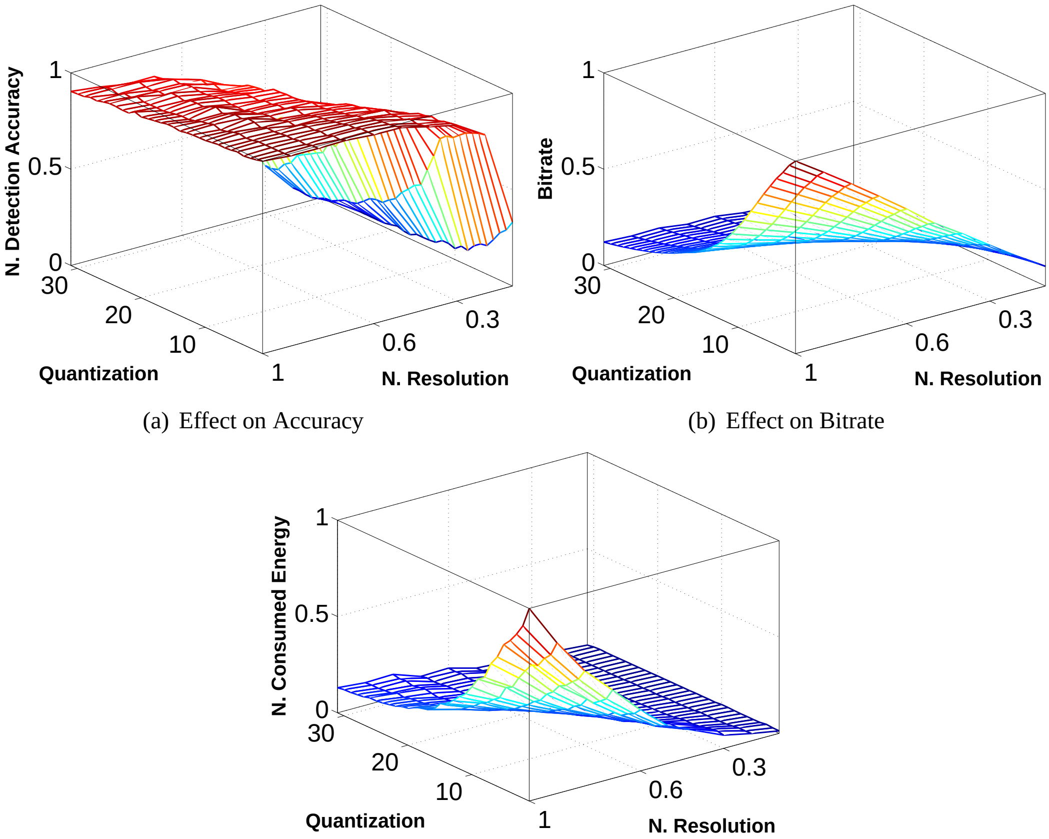


Figure 3.3: Comparing Stream Adaptation Techniques in Aggregate Power Consumption

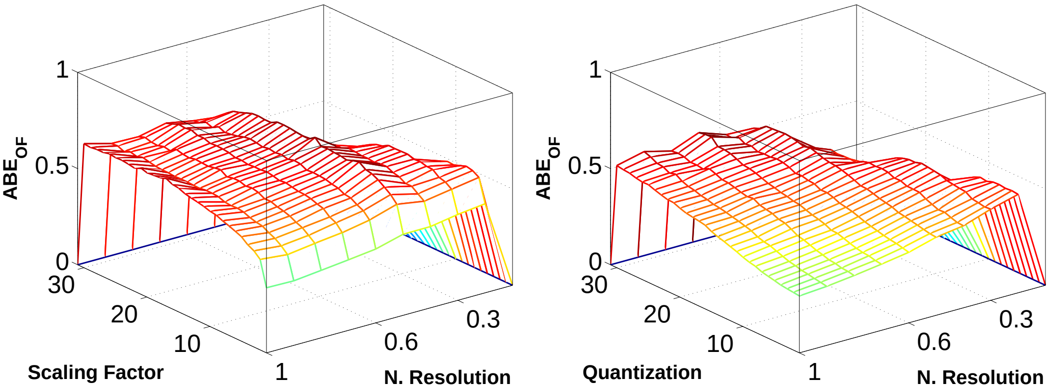


(a) Effect on Accuracy

(b) Effect on Bitrate

(c) Effect on Consumed Energy

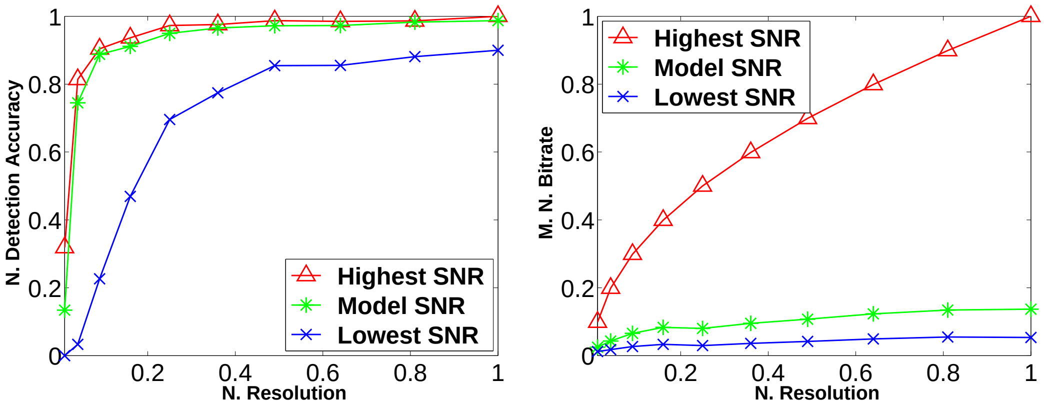
Figure 3.4: Comparing Combinations of Different Resolutions and SNR



(a) MPEG-4

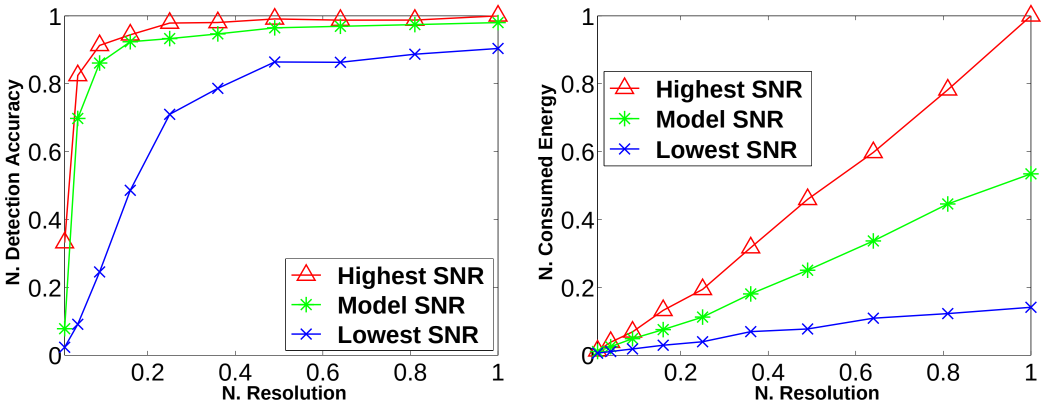
(b) H.264

Figure 3.5: Accuracy-Bitrate-Energy Tradeoff



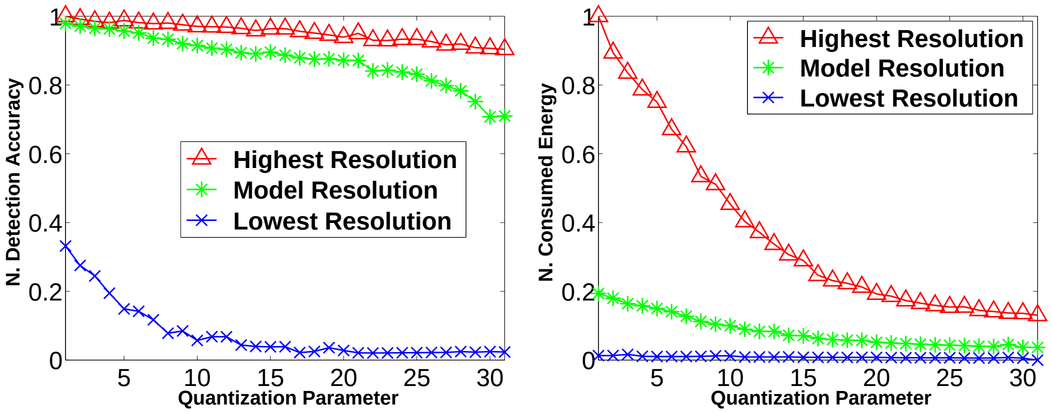
(a) Effect on Accuracy (b) Effect on Bitrate

Figure 3.6: Comparing SNR Selection at Different Resolutions [MPEG-4].



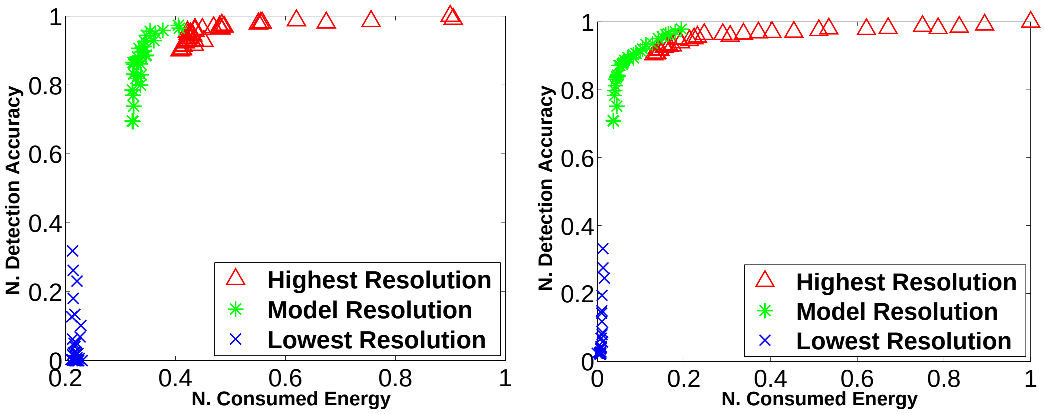
(a) Effect on Accuracy (b) Effect on Consumed Energy

Figure 3.7: Comparing SNR Selection at Different Resolutions



(a) Effect on Accuracy (b) Effect on Consumed Energy

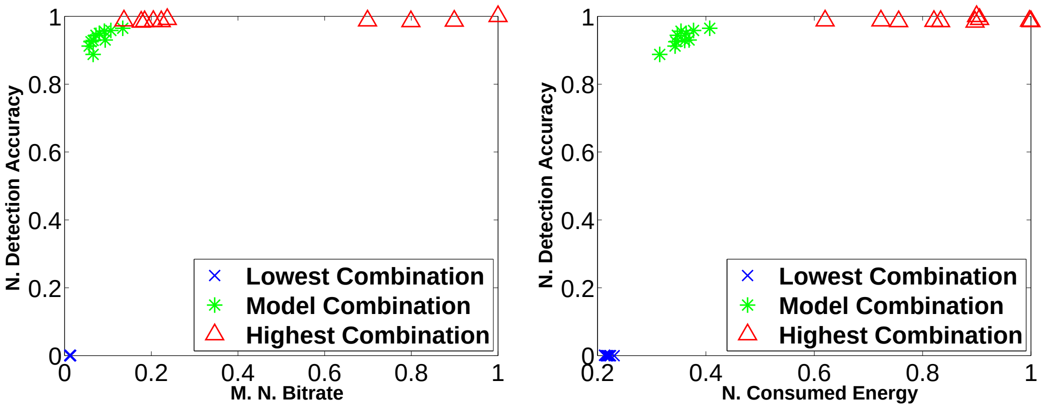
Figure 3.8: Comparing Resolution Selection at Different SNR [H.264, Experimental Setup I]



(a) MPEG-4

(b) H.264

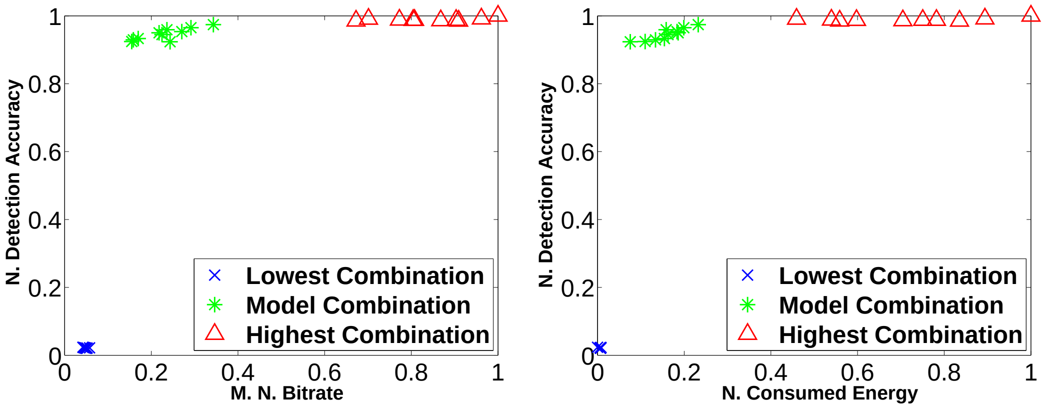
Figure 3.9: Consumed Energy-Accuracy Tradeoff.



(a) Bitrate-Accuracy Tradeoff

(b) Consumed Energy-Accuracy Tradeoff

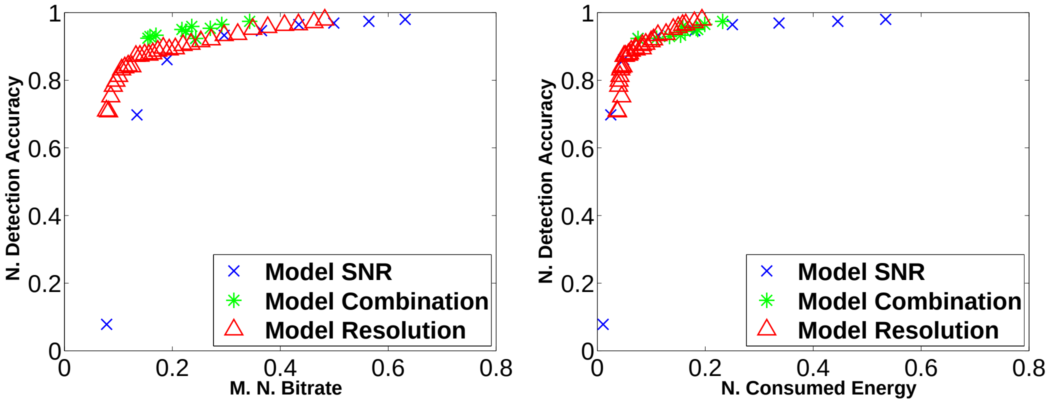
Figure 3.10: Comparing Combinations of Different Resolutions and SNR [MPEG-4]



(a) Bitrate-Accuracy Tradeoff

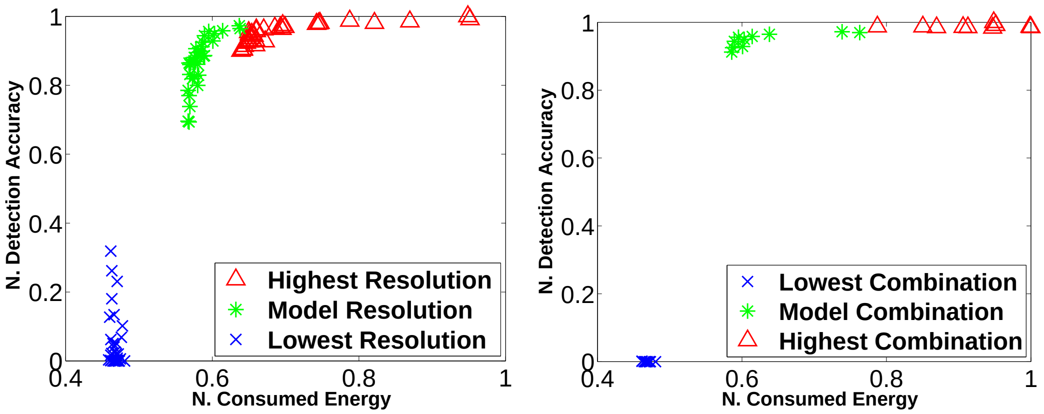
(b) Consumed Energy-Accuracy Tradeoff

Figure 3.11: Comparing Combinations of Different Resolutions and SNR [H.264].



(a) Bitrate-Accuracy Tradeoff (b) Consumed Energy-Accuracy Tradeoff

Figure 3.12: Comparing Model SNR, Model Resolution, and Model Combinations [H.264].



(a) Resolution (b) Combination

Figure 3.13: Comparing Different Adaptations [MPEG-4, Experimental Setup III]

CHAPTER 4 MODELING OF POWER CONSUMPTION IN LIVE VIDEO STREAMING SYSTEMS

4.1 Introduction

Power consumption is a major concern in live video streaming systems in general and in *many-to-one* live video streaming systems in particular. A many-to-one streaming system includes multiple video sources (i.e., cameras and/or sensors) streaming videos to a monitoring station. These systems are typical in video surveillance and wireless video sensor networks [51, 6, 7]. The monitoring station receives video streams from all sources and run computer vision algorithms for determining the appropriate actions, such as controlling sources and generating alerts. The power is consumed by the source in each of the following three phases: capturing, encoding, and transmission. Although power consumption is of utmost importance when the source is battery-powered, reducing power consumption is essential even when the power is available because video sources consume orders of magnitude more resources than scalar sensors [68].

In this chapter, we develop an aggregate power consumption model for live video streaming systems in general and analyze the power consumed by the monitoring station in many-to-one systems. We model the power consumed by the video source in each of the three phases and then provide an overall model of the power consumed by the source. The developed models are based on $1,620$ different experiments, each of which is repeated at least 3 times, totaling more than $4,800$ *actual experiments*. This part of dissertation has been motivated by our ongoing work on the power-aware design of large-scale video surveillance systems [69, 70]. That work requires accurate, simple, and appropriate power consumption models. The developed models can be used to assess the impacts of various video capturing and encoding parameters, and thus can help in the dynamic control of various source settings, including resolution, frame rate, number of reference frames, motion estimation range, and quantization to achieve the best overall tradeoff in terms of power consumption, bitrate (and thus bandwidth), and video quality. (In automated video surveillance, the computer vision accuracy can be considered instead of the quality [69].) Although we experiment with different platforms and video contents for validation purposes, the models do not directly capture the impacts of such factors as well environmental factors and communication strategies; all these factors simply translate to changing constants in the developed models. For video encoding, we develop a power consumption model for both H.264 and MPEG-4, capturing the aforementioned parameters. Since tuning various parameters is often based on power consumption, video quality, and bitrate tradeoffs, we develop a model for the output bitrate of video encoding. The bitrate affects the medium bandwidth, the video quality, and the transmission power consumption. Moreover, we analyze the power consumed by the monitoring station due to the reception, upscaling (to the original video resolutions as captured by the sources), and decoding of the received video streams. Furthermore, we analyze many-to-one systems in terms of bitrate, video quality, and the power consumed by the sources as well as the monitoring station, considering the impacts of multiple parameters simultaneously. Although, we consider the popular H.264 and MPEG-4 encoding, this study can help in deriving models for *High Efficiency Video Encoding* (HEVC) and VP8, which have similar operations.

We validate the developed models through extensive experiments using two types of systems and different video contents. The first includes a regular camera and employs software-based encoding with FFmpeg/x.264. This system allows better flexibility in conducting the experiments. The second includes an actual video surveillance camera with a system-on-chip (SoC) for encoding.

The main unique contributions of this part of dissertation can be summarized as follows. (1) In contrast with prior studies, we model the power consumed in all three phases. (2) We provide the first aggregate power consumption model in terms of various capturing and encoding parameters, including quantization, number of reference frames, search range, and motion estimation algorithms. Up to our knowledge, the impacts on these parameters were not analyzed in prior work. (3) We develop a model for the bitrate achieved by video encoding, considering the aforementioned parameters. (4) We validate and analyze the developed models through extensive experiments, using different types of cameras, systems, and input videos. (5) We provide a detailed analysis of many-to-one systems in terms of bitrate, video quality, and the power consumed by the sources as well as that by the monitoring station, considering the impacts of multiple parameters simultaneously. (5) We show that the overall computation complexity for all phases can approximately be modeled as a linear function of the pixel rate (when fixing the other parameters). The pixel rate is the product of the spatial and temporal resolutions of the raw video.

The rest of the chapter is organized as follows. Section 4.2 develops various models. Section 4.3 discusses the setup of experiments and modeling methodology. Section 4.4 presents the validation results and provides an overall analysis. Finally, conclusions are drawn in the last section.

4.2 Model Development

In this section, we develop the power consumption models for each phase at the source and then develop the aggregate model. we also develop a model of the bitrate. Table 4.1 summarizes the symbols used in this section.

4.2.1 Modeling of the Power Consumed by Video Capturing

To model the power consumed by CMOS sensors, let us first start by generalizing Equation (2.4) to a general mesh of photodiodes and an associated number of A/D processing units. The per-frame power consumption W_s for a video sensor of NxM pixels and K A/D processing units can be given by

$$W_s = c_i \, N \, M + c_b \, K, \tag{4.1}$$

where c_i and c_b are constants. Equation (4.1) shows a direct relationship between the power consumption in video sensors and the spatial resolution. This equation can be extended to capturing a video by

Table 4.1: Descriptions of Used Symbols

Symbol	Description	Symbol	Description
W	Power Consumption (Watt)	X	Computation Complexity (basic operation)
r	Bitrate (Kbit/s)	L	Pixel rate (pixel/s)
F	Frame rate (frame/s)	c	Constant
$N \times M$	Frame dimensions in pixels	$P \times Q$	Dimensions of a macroblock (MB) in pixels
K	# Analog-to-Digital (A/D) Units	Y	# Bits/pixel
R	# Reference frames for ME	d	Distance between sender and receiver (meter)
n	Path-loss index in transmission	S, S'	Displacement in pixels or sub-pixel for ME
q	Quantization parameter	s	Scaling Factor
i	# I frames in Group of Pictures	p	# P frames in Group of Pictures
b	# B frames in Group of Pictures	J	Joint cost (db)
t	Distortion (db)	g	Lagrange multiplier
N	# Operations	v	Voltage (volt)
f	Frequency (Hz)	V	# MVs in a macroblock (MB)
A	Boolean variable that is either 0 or 1		

considering the temporal resolution. Thus, the total capturing power consumption W_C can be expressed as follows: $W_c = F\,W_s = F\,(c_i\,N\,M + c_b\,K)$, where F is the frame rate. The main players in the capturing power consumption are the spatial and temporal resolution. The impacts of a specific sensor type, technology, and/or implementation translate to (changing the values of) constants in the model. Our experiments confirm that the equation applies but with an additional constant:

$$W_c = F\,(c_i\,N\,M + c_b\,K) + c_j, \tag{4.2}$$

where c_j is a constant specifying the power consumed by the sensor when no capturing takes place. The standby power is also consistent with the findings in [48], but this new constant provides a much simpler way to model it.

To simplify the model, we can utilize the direct relationships between N or M and K. The value of K is typically equal to N (but conceptually it might be any fraction of it or M). Furthermore, for a megapixel camera the $N \times M$ term dominates the K term. Therefore, the power consumption can be expressed as follows: $W_c \approx c_i\, F\, N\, M + c_j$. The power consumption can also be expressed in terms of the pixel rate L. The pixel rate is the frame rate multiplied by number of pixels in the frame and thus can be given by $L = F\, N\, M$. Consequently, the power consumption can be given by

$$W_c \approx c_i\, L + c_j, \tag{4.3}$$

where c_i and c_j are constants. The bitrate for the raw video is the frame size (in pixels) times the frame rate (in frames/s) times the number of bits per pixel, and thus it can be expressed in terms of the pixel rate as follows: $r = L\, Y$, where Y is the number of bits per pixel in the raw video. (In our experiments $Y = 12$, since we use the I420 color space). Therefore, the power consumption is also linear with the bitrate.

4.2.2 Modeling of the Power Consumed by Video H.264 Encoding

H.264 has high computational complexity, mainly due to its ME, complex prediction, and RDO [71]. Due to this high complexity, intra-prediction (for I-frames), inter-prediction (for B- and P-frames), RDO, and mode selections have been active areas of research [72]. Since the block size is adaptive, RDO operates on multiple variable block sizes, different intra-prediction modes in I-frames, and different ME vectors in inter-frames. For each macroblock (MB), RDO finds the combination (of block sizes and intra-prediction modes in I-frames and block sizes and ME vectors in inter-frames) with the least RDO cost J (discussed in Subsection 2.3.2), among all possible combinations. For a specific MB and a specific combination, the process proceeds in the following steps: (i) compute the prediction MB, (ii) compute the residual MB, (iii) encode the residual MB (including transformation, quantization, and entropy coding), (iv) decode the MB (including inverse quantization and inverse transformation), (v) reconstruct the MB, (vi) compute distortion, and (vii) compute the cost J. This process is repeated for each combination and then the combination with the minimum cost will be selected for the MB. The whole process is repeated for each MB in the frame.

The power consumption W_e dissipated as a result of encoding a raw video that is captured by a camera is a function of the video encoder computation complexity X_e. As discussed in Subsection 2.3.2, the computation complexity of encoding one frame is primarily the sum of the complexities of (i) inter-prediction and intra-prediction, (ii) transformation, quantization, their inverses, reconstruction, and distortion and cost computations, and (iii) entropy encoding. Consequently, as in [59, 5], the encoding computation complexity X_e for F frames (taking the weighted average of different frames in a second) is given by

$$X_e = X_{inter} + X_{intra} + X_{traquant} + X_{entropy}, \tag{4.4}$$

where X_{inter} is the computation complexity of inter-prediction multiplied by the ratio of inter-frames to the total frames in F frames, X_{intra} is the computation complexity of intra-prediction multiplied by the ratio of intra-frames to the total frames in F frames, $X_{traquant}$ is the transformation, quantization, and their inverses computation complexity for F frames, and $X_{entropy}$ is the entropy encoding computation complexity for F frames.

Inter-Prediction ME Computation Complexity

For inter-prediction RDO, a combination of ME vectors and multiple block sizes are searched for the best cost. A MB can be divided into 16×16, 16×8, 8×16, or 8×8 blocks. Since each 8×8 block can be divided further into 8×4, 4×8, or 4×4 sub-blocks, inter-prediction has 7 size combinations. To select the best combination for one MB in inter-prediction, the encoder performs $16+8+8+4+2+2+1 = 41$ size combinations, leading to 41 RDO operations, in addition to finding the lowest residual in the search range for each of these RDO operations.

We can express X_{inter} as the sum of integer ME complexity ($X_{integer}$) and fractional ME complexity

$(X_{fractional})$:

$$X_{inter} = X_{integer} + X_{fractional}. \tag{4.5}$$

Let us first analyze the integer ME complexity. As discussed earlier, block matching estimation and compensation are used to exploit the temporal locality among successive frames in a video by predicting blocks from previously encoded frames. This process involves partitioning the current video frame into blocks of pixels and then finding the best matching block inside a reference frame for each of these blocks, using a predefined distortion criterion. The best matching block is used for predicting the block in the current frame. Instead of coding the entire block, the encoder includes only the difference between the two blocks (i.e., the residual) and the associated motion vector (MV) specifying the displacement between the two blocks. For additional details, please refer to [73]. One of the commonly used distortion measure is the *Sum of Absolute Differences* (SAD). $SAD(V_x, V_y)$ is defined as the SAD for block A located at (x, y) inside the current frame compared to block B located at a displacement of (V_x,V_y) relative to block A in the reference frame. It can be found by summing the absolute differences between each pixel in block A and the corresponding pixel in block B. In the Full Search (FS) algorithm, if a maximum displacement of S pixels in a frame is allowed, $(2S + 1)^2$ locations have to be searched to find the best match for the current block. For a video with a frame size of $N \times M$ (in pixels) and a frame rate of F and for an encoder that uses a MB size of $P \times Q$ and R reference frames, the integer ME computation complexity $X_{integer}$ can be given by

$$X_{integer} = F \, \frac{N \, M}{P \, Q} \, R \, (2S + 1)^2 \, (2P \, Q - 1 + V \, X_{MV}), \tag{4.6}$$

where $(2P \, Q - 1)$ represents the number of SAD operations for the MB, V is the number of MVs in the MB, and X_{MV} is the number of operations required to calculate the MV. The number of motion vectors is equal to the number of blocks in the MB for a P frame and twice the number of blocks in the MB for a B frames. MVs are coded differentially.

Equation (4.6) generalizes Equation (3) in [73] to handle multiple reference frames and consider the effect of computing MVs. The complexity for computing a MV (X_{MV}) includes 3 multiplications, 3 additions, 24 shifts, 1 median of 3 MVs, and 2 subtractions. Since the search range (S) is large compared to 1, $(2S+1)^2$ can be simplified to $4S^2$. V and X_{MV} can be regarded as constants (on average) and $P \times Q$ is 16 × 16. Hence, $X_{integer}$ can be given as

$$\begin{aligned} X_{integer} &\approx (4\,F\,N\,M\,R\,S^2)(2 + \frac{V\,X_{MV}}{P\,Q}), \\ &\approx c_{integer}\,L\,R\,S^2, \end{aligned} \tag{4.7}$$

where the pixel rate $L = F\,N\,M$.

Let us now develop an ME complexity model for encoders supporting sub-pixel search. Sub-pixel search considers movements of a non-integer number of pixels from the reference block. The ME process here proceeds in two stages: integer pixel search over a large area and a sub-pixel search around the best selected integer pixel [74]. The complexity depends on the number of operations for interpolating in-between pixels in the block (i.e., pixels at non-integer locations). Figure 4.1 demonstrates the concept of half-pixel and quarter-pixel motion estimation. First, the encoder finds the best integer match. Subsequently, the half-pixel positions immediately next to this best match are searched. Finally, the quarter-pixel positions next to the best half-pixel position are searched [27].

Table 4.2 shows the number of interpolation operations. This complexity depends on the accuracy of the sub-pixel search (half a pixel, quarter a pixel, etc.). The implementation of FS in sub-pixel ME follows a hierarchical way. For quarter-pixel, eight half-pixel pixels around the best integer pixel are examined first, and then eight quarter-pixel pixels around the best half-pixel pixel are checked [75]. Note

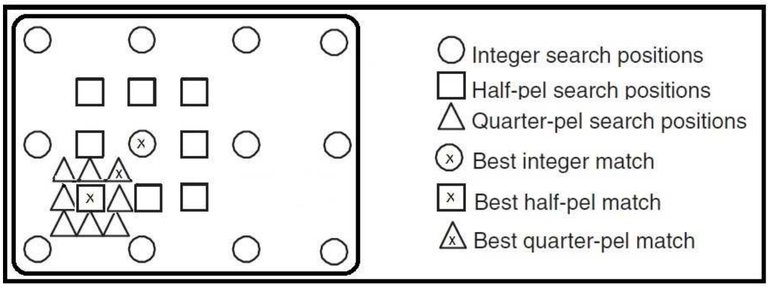


Figure 4.1: Half-Pixel and Quarter-Pixel Motion Estimations

that half-pixel resolution MVs in the Luma component require quarter-pixel resolution vectors in the Chroma components (assuming 4:2:0 sampling). Similarly, quarter-pixel resolution MVs in the Luma component require eighth-pixel resolution vectors in the Chroma components. Assume S' represents the range of the sub-pixel search in pixels and X_{p1} and X_{p2} represent the numbers of pixel interpolation operations for half-pixel accuracy and quarter-pixel accuracy, respectively. Based on Table 4.2, X_{p1} is the number of operations in both the first and second rows (i.e., X_{p1} = $X_{1/2Luma}$ + $X_{1/4Chroma}$) and X_{p2} is the number of operations in the third and fourth rows (i.e., X_{p2} = $X_{1/4Luma}$ + $X_{1/8Chroma}$). The computation complexity $X_{fractional}$ of fractional pixel ME can be given by

$$\begin{aligned} X_{fractional} &= F\,\frac{N\,M}{P\,Q}\,(2S'+1)^2\,(2P\,Q-1+P\,Q\,(X_{p1}+A_{1/4}\,X_{p2})) \\ &\approx L\,(2S'+1)^2\,(2+X_{p1}+A_{1/4}\,X_{p2}), \end{aligned} \tag{4.8}$$

where $A_{1/4}$ is a Boolean variable that is either 0 or 1 for half- and quarter-pixel accuracy, respectively). X_{p1} and X_{p2} are constants as explained above. S' is fixed to 1 in sub-pixel accuracy motion estimation, because the search is only one sub-position in the surrounding eight directions, whether it is half- or quarter-pixel accuracy search. Therefore, $X_{fractional}$ can be expressed as

$$X_{fractional} = c_{fractional}\,L, \tag{4.9}$$

Table 4.2: Per-Pixel Computation Complexity of Interpolation for Fractional Pixel ME

Interpolation	Description	Complexity	# Operations
$1/2$ Pixel Luma	6-tap interpolation: a combination of 6 samples, 3 from each side of a row or a column	$X_{1/2Luma}$	5 add + 4 mul + 1 div
$1/4$ Pixel Chroma	Weighted mean of neighboring pixels and a constant	$X_{1/4Chroma}$	2 mul + 2 add + 1 div
$1/4$ Pixel Luma	Linear interpolation between adjacent samples: combination of 2 samples, 1 from each side of a row or a column	$X_{1/4Luma}$	1 add + 1 div
$1/8$ Pixel Chroma	Linear combination of 4 neighboring integer pixel positions	$X_{1/8Chroma}$	3 add + 4 mul + 1 div

Based on Equations (4.5), (4.7), and (4.9), the total inter-prediction (both integer and fractional) complexity for a full search X_{inter} can be given by

$$X_{inter} \approx c_{integer}\, L\, R\, S^2 + c_{fractional}\, L, \tag{4.10}$$

where $c_{integer}$ and $c_{fractional}$ are constants.

Intra-Prediction Computation Complexity

H.264 exploits the spatial correlation between adjacent blocks in intra-prediction. For the Luma prediction, the prediction block is formed for each 4×4 block or for a 16×16 block. One mode is selected from the supported modes, which are 9 modes for a 4×4 Luma block, 4 modes for a 16×16 Luma block, and 4 modes for each Chroma block. The encoder selects the best mode using RDO. As an illustrating example, in intra-prediction RDO, the number of mode combinations for one MB (16×16 pixels) is $N_8(16N_4 + N_{16})$, where N_8, N_4, and N_{16} represent the number of modes of an 8×8 Chroma block, a 4×4 Luma block, and a 16×16 Luma block, respectively. To select the best mode for one MB in intra-prediction, the encoder performs $4(16 \times 9 + 4) = 592$ RDO calculations [76].

We develop Tables 4.3, 4.4, and 4.5 to assist in computing the complexity of intra-mode selection

X_{intra}. These tables also include a brief description of each intra-mode. The complexity can be found as follows:

$$X_{intra} = F\,N\,M\,(4\,N_c)(\frac{N_{l4}}{4 \times 4} \times 9 + \frac{N_{l16}}{16 \times 16} \times 4), \tag{4.11}$$

where N_{l4}, N_{l16}, and N_c are the average number of operations in each of Tables 4.3, 4.4, and 4.5, respectively. They represent the number of operations to compute a 4×4 Luma prediction block, a 16×16 Luma prediction block, and an 8×8 Chroma prediction block. For example, $N_{l16} \times 4$ is the total number of operations in Table 4.4 or the average of the operations in the table multiplied by 4. In addition, $N_{l4} \times 9 \times 16$ is the total number of operations in Table 4.3 multiplied by 16, where 16 is the number of 4×4 blocks in the MB. Finally, $4N_c$ is the total number of operations in Table 4.5. The total number of operations in each of the three tables above is constant, and thus N_{l4}, N_{l16}, and N_c are constants. Consequently, the intra-mode selection complexity can be simply given by

$$X_{intra} = c_{intra}\,L. \tag{4.12}$$

Quantization, Pixel Rate, and Bitrate Relationships

For homogeneous video contents, we determine by extensive experiments that the bitrate is linearly proportional to the pixel rate L and inversely proportional to the quantization parameter to a certain power, as shown in Figure 4.2. The used experimental setup is discussed in Section 4.3. Hence, the bitrate r can be expressed as

$$r = c_{rate}\,\frac{L}{q^c}, \tag{4.13}$$

where q is the quantization parameter and c_{rate} is a constant. As expected, the quantization parameter has a great impact on the bitrate.

Table 4.3: Number of Operations to Compute a 4×4 Luma Prediction Block

Mode	Description	# Operations
Mode 0	Vertical: the upper row's samples are extrapolated vertically	4 x 4 copy
Mode 1	Horizontal: the left column's samples are extrapolated horizontally	4 x 4 copy
Mode 2	DC: the block is predicted by the mean of upper row's and left column's samples (an average of 8 values for the block)	(8-1) add + 1 div + 4 x 4 copy
Mode 3	Diagonal Down-Left: the samples are interpolated at a 45^o angle between lower-left and upper-right. It rounds the value of three neighboring pixels, each divided by an integer	4 x 4 x (3 mul + 2 add + round)
Mode 4	Diagonal Down-Right: the samples are extrapolated at a 45^o angle down and to the right	same as Mode 3
Mode 5	Vertical-Left: extrapolation at an angle of approximately 26×6^o to the left of vertical, i.e. width/height = 1/2	same as Mode 3
Mode 6	Horizontal-Down: extrapolation at an angle of approximately 26×6^o below horizontal	same as Mode 3
Mode 7	Vertical-Right: extrapolation or interpolation at an angle of approximately 26×6^o to the right of vertical	same as Mode 3
Mode 8	Horizontal-Up: interpolation at an angle of approximately 26×6^o above horizontal	same as Mode 3

Computational Complexities of Transformation, Quantization, Their Inverses, Reconstruction, Distortion, and Cost

Based on [5], the computation complexity $X_{traquant}$ to encode the residual MB (including transformation, quantization, and entropy coding), decode the MB (including inverse quantization and inverse transformation), reconstruct the MB, compute distortion, and compute the cost J can be expressed as

$$X_{traquant} = F\, x_{nzmb}\, m_{nzmb}, \tag{4.14}$$

where F is the frame rate, m_{nzmb} represents the number of nonzero MBs in the video frame, x_{nzmb} is the computation complexities of the transform, quantization, and their inverses for one nonzero MB. Note that a nonzero MB is a MB that has nonzero transform coefficients after quantization. Only nonzero MBs go through the transformation and quantization processes. Also note that x_{nzmb} is constant because it is a systematic algorithm with a specified number of operations (Table 4.6) and $F \times m_{nzmb}$ is directly

Table 4.4: Number of Operations to Compute a 16×16 Luma Prediction Block

Mode	Description	# Operations
Mode 0	Vertical: copy row	16 x 16 copy
Mode 1	Horizontal: copy column	16 x 16 copy
Mode 2	DC: average of 32 values for the block	(32-1) add + 1 div + 16 x 16
Mode 3	Plane: a linear plane function fitted to the upper and left-hand samples H. and V. Clipping ensures $0 < result < 255$	16 x 16 x (5 add + 2 mul + 1 compare + 1 clip)

Table 4.5: Number of Operations to Compute an 8×8 Chroma Prediction Block

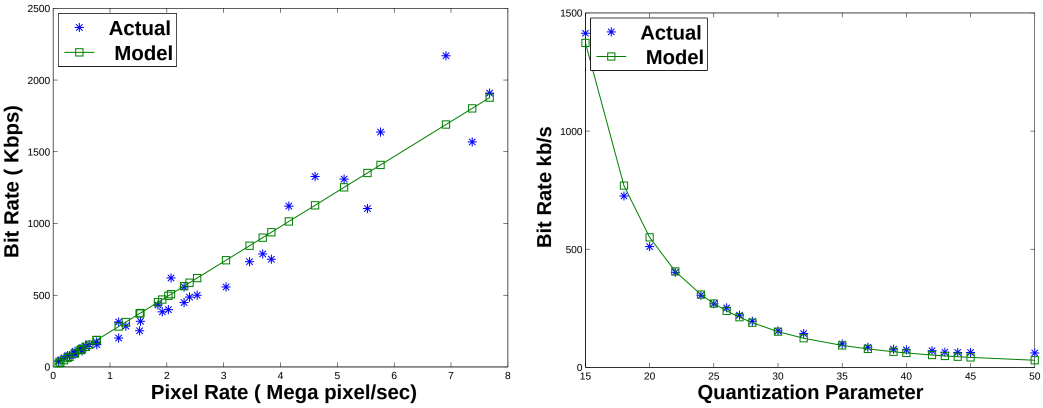
Mode	Description	# Operations
Mode 0	Vertical: copy row	8 x 8 copy
Mode 1	Horizontal: copy column	8 x 8 copy
Mode 2	DC: average of 32 values of macroblock	(16-1) add + 1 div + 8 x 8
Mode 3	Plane: a linear plane function fitted to the upper and left-hand samples H and V.	$8 \times 8 \times$(5 add + 2 mul + 1 compare + 1 clip)

proportional to the bitrate. Therefore,

$$X_{traquant} = c_{traquant}\, r = c_{traquant}\, c_{rate}\, L/q^c = c_{qnt}\, L/q^c, \tag{4.15}$$

where $c_{traquant}$ and c_{rate} are constants and r is the bitrate. The video content coupled with the encoding algorithm and parameters (such as quantization) impact the number of nonzero MBs.

We develop Table 4.6 to determine the complexities of various steps. In the table, the steps involving transform, quantization, inverse quantization, inverse transform, reconstruction, distortion and single cost are repeated either 592 times in case of intra-prediction or 41 times in case of inter-prediction.



(a) Bitrate vs. Pixel rate, NMSE = 0.056

(b) Bitrate vs. QP, NMSE = 0.007

Figure 4.2: Relationships between Bitrate and Pixel Rate and between Bitrate and QP

Entropy Computation Complexity

The entropy complexity $X_{entropy}$ of a frame is linearly proportional to bitrate [5]. Based on Equation (4.13), we can express it as

$$X_{entropy} = c_{entropy}\, r = c_{entropy}\, c_{rate}\, L/q^c = c_{bit}\, L/q^c. \tag{4.16}$$

Overall Power Consumption Model

Based on Equations (4.4), (4.10), (4.12), (4.15), and (4.16), the complexity X_e of encoding F frames can be expressed as

$$\begin{aligned} X_e &= c_{integer}\, L\, R\, S^2 + (c_{fractional} + c_{intra})\, L + (c_{qnt} + c_{bit})\, L/q^c \\ &= c_{integer}\, L\, R\, S^2 + c_L\, L + c_{Lq}\, L/q^c, \end{aligned} \tag{4.17}$$

where $c_{integer}$, $c_{fractional}$, c_{intra}, c_{qnt}, c_{bit}, c_L, and c_{Lq} are constants, L is the pixel rate, R is the number of references, and S is the search range.

Let us now discuss how the overall power consumption can be modeled in terms of encoding com-

Table 4.6: Nonzero MB's Complexity X_{nzmb} of Trans. and Qunt.

Step	Description	# Operations
Transform	# ops to compute transform ($Y = AXA^T$): 1 transpose and 2 4x4 matrix multiplications by blocks in macroblock	(2 x (4 mul + 3 add) x 4 x 4 entries + 16) x 16
Quantization	# ops to compute quantization	(4 mul + 3 add) x 4 x 4 x 16 entries
Inverse Quantization	# ops to compute inverse quantization	(4 mul + 3 add) x 4 x 4 x 16 entries
Inverse Transform	# ops to compute inverse transform $Z = A^TYA$: 1 transpose and 2 4x4 matrix multiplications by blocks in macroblock	(2 x (4 mul + 3 add) x 4 x 4 entries + 16) x 16
Reconstruction	# ops to compute the reconstructed macroblock	(4 x 4 - 1 add) x 4 x 4 x 16
Distortion	# ops to compute Distortion and the Sum of Squared Distortion (SSD) between the original and the reconstructed macroblock	(2 x 4 x 4-1 + 4 x 4) x 16
Single Cost	# ops to compute the single cost for the mode combination: $J = t + gr$	(1 add + 1 mul) x 4 x 4 x 16
Minimum Cost for Intra-Prediction	# ops to find the minimum cost among all mode combinations for the macroblock	(1 initialize + 592 x (1 compare + 1 equal)) x 16
Minimum Cost for Inter-Prediction	# ops to find the minimum cost among all mode combinations for the macroblock	(1 initialize + 41 x (1 compare + 1 equal)) x 16

plexity. As in [5], the power consumption W_e for the encoder can be expressed as $W_e = c_{eff}\, v_{DVS}^2\, f_{CLK}$, where v and f_{CLK} are supply voltage and clock frequency, c_{eff} is the effective switched capacitance of a processor with an energy-scaling feature, such as Dynamic Voltage Scaling (DVS) (discussed in Subsection 2.3.2). However, V is approximately linearly proportional to f_{CLK}. As in [77], the voltage (v_{DVS}) and clock frequency (f_{CLK}) relationship is given by $v_{DVS} = c_1\, f_{CLK} + c_2$, where c_1 and c_2 are constants. Moreover, f_{CLK} is proportional to the computation complexity: $f_{CLK} = c_3\, X_e + c_4$, where c_1 and c_2 are constants. Subsequently, the power consumption can be expressed as

$$\begin{aligned} W_e &= c_{eff}\,(c_a\,X_e + c_b)^2\,(c_d\,X_e + c_e) \\ &= ((c_1\,R\,S^2 + \frac{c_2}{(q+c_q)^c} + c_3)\,L + c_4)^2\,((c_5\,R\,S^2 + \frac{c_6}{(q+c_q)^c} + c_7)\,L + c_8), \end{aligned} \tag{4.18}$$

where c_a, c_b, c_d, c_e, c_1, c_2, c_3, c_4, c_5, c_6, c_7, and c_8 are constants.

From Equation (4.18), we notice that the consumed encoding power depends on the video parameters (spatial and temporal resolutions), video content, performed algorithms (for intra and inter prediction, transform, quantization, etc.), and encoding parameters (such as the fraction of various frame types in the GOP, number of reference frames, quantization, and ME range). The video content coupled with the encoding algorithm and parameters (such as quantization parameter) impact the number of nonzero MBs. The model considers the full search approach and does not directly capture optimization techniques that abort the search early based on some statistics and other algorithms, such as fast intra/inter prediction. The computation complexity of the transform and quantization and their inverses is directly proportional to the number of nonzero MBs in the frame, which is directly proportional to the bitrate and inversely with quantization parameter [78, 58]. The complexity of entropy encoding is directly proportional to the bitrate [5]. Furthermore, the loop filter complexity is a function of number of MBs and frame rate. This leads us to conclude that the H.264 complexity is directly proportional to a weighted sum of the pixel rate and the bitrate.

General Bitrate Model

The bitrate is a function of pixel rate, quantization parameter, number of references, and ME search range. Based on Equation (4.13) and extensive experiments analyzing the impacts of the number of references and the ME range (including those shown in Figures 4.6(b) and 4.7(b)), we can develop a general model for bitrate as a function of pixel rate, quantization parameter, number of references, and ME search range:

$$r = c_n \frac{(c_t - c_s\, R)\,(c_g - c_f\, S)\, L}{(q + c_q)^c}, \tag{4.19}$$

where c_n, c_t, c_s, c_g, c_f, and c_q are constants, L is pixel rate, R is number of references, S is the ME search range, and q is the quantization parameter. The linear relationship with R and S will be evident in the validation results.

4.2.3 Modeling of the Power Consumed by Video Transmission

In the last phase of live video streaming, the video is transmitted to the receiver(s). According to [60, 61], the power W_t consumed in wireless transmission when it is chosen such that the bit error rate (BER) at the receiver side is very low can be expressed as

$$W_t = (c_x + c_z\, d^n)\, r, \tag{4.20}$$

where c_x and c_z are wireless model constants, d is the transmission distance, n is the path-loss index, and r represents the transmission bitrate. Equation (4.20) can be generalized for wired transmission by assuming the path-loss index n is zero. Therefore, the transmission power for wired transmission can be given by

$$W_{wired} = c\, r, \tag{4.21}$$

where c is a wire model constant, and r is the transmission bitrate. Equations (4.20) and (4.21) indicate that the power consumption of transmitting the video is linearly proportional to the transmission bitrate.

In our experiments of wireless video transmission, we confirmed that the model in Equation (4.20) applies but with an additional constant, specifying the power consumption of the wireless circuit when no transmission takes place. For the same technology, platform, distance, path-loss or environment, the model can be simplified as follows:

$$W_t = (c_x\, r + c_y), \tag{4.22}$$

where c_x and c_y are constants.

Table 4.7: Physical Significance of Various Constants

Constants	Reflect the power consumption of
c_{ap}, c_{af}	DVS circuit capacitance, encoding parameters, encoding power consumption per pixel, and video content
c_{bp}, c_{bf}	Slope of linear relationship between frequency and voltage in DVS circuits and DVS capacitance
c_{cp}	Wireless distance, environment, transmission scheme, and capturing power consumption per pixel
c_{dp}	The power consumed by video sensor and transmitter circuits when they are not active

4.2.4 *Modeling the Aggregate Power Consumption*

Equations (4.3), (4.18), and (4.22) can be used to construct the aggregate power consumption model for the video source. The aggregate power consumed W_{agg} as a function of the resolution and frame rate can be found as follows:

$$W_{agg} = (c_{ap}\, L + c_{bp})^2\, (c_{af}\, L + c_{bf}) + c_{c1}\, L + c_{c5} + c_{t1}\, r + c_{t3}.$$

Using Equation (4.13), the model can be simplified to

$$W_{agg} = (c_{ap}\, L + c_{bp})^2\, (c_{af}\, L + c_{bf}) + c_{cp}\, L + c_{dp}, \tag{4.23}$$

where c_{ap}, c_{bp}, c_{af}, c_{bf}, c_{cp}, and c_{dp} are constants. Table 4.7 illustrates the physical significance of various constants in the aggregate power consumption model. Only the main factors are considered.

4.3 *Experimental Setup and Validation Methodology*

We validate the developed models through extensive experiments using two types of systems. The first uses a regular video camera and employs software-based encoding using FFmpeg/x.264. This system allows better flexibility in conducting the experiments. The second includes an actual video surveillance camera with a system-on-chip (SoC) for encoding. We conduct experiments using three experimental setups. *Experimental Setup I* is based on the first system, whereas *Experimental Setup II* and *Experimental Setup III* are based on the latter, but the input videos for these two vary. In all setups, the consumed power is measured by an advanced power meter: Watts Up? Pro ES AC.

Figure 4.3 shows Experimental Setup I. To ensure repeatable measurements, a video rendered on a desktop computer is captured by a Dell Inspiron 1525 laptop computer with an Intel Core 2 Duo CPU (Model T5750) running at 2.00 GHz with 3.00 GB memory, 802.11n Wireless LAN, Ethernet LAN, and an external video camera (Logitech Webcam Pro 9001). The external camera is directed to that desktop computer, which plays a specific movie (from the beginning to the end). The rendered video includes scenes of five children running and playing in a zoo, with much details and fast movements. The camera feeds the captured video in raw format to the laptop computer, which encodes the video with FFmpeg in the case of MPEG-4 and X.264 in the case of H.264. The video is streamed using VideoLan VLC streaming server (Version 1.0.5 Goldeney) running on the computer. For validation, we also include some results using the latest VLC version (VLC 2.2.4) on the same platform. The distance between the server and client is within 1 meter.

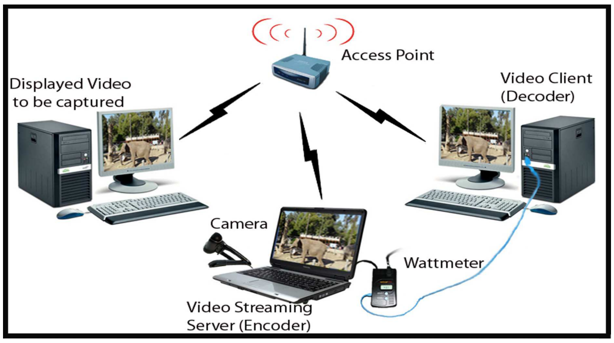


Figure 4.3: Illustration of Experimental Setup I

We measure the power consumed by the streaming server for H.264 and MPEG-4 encoding. For encoding, we vary the spatial (i.e., frame size) and temporal (i.e., frame rate) resolutions generally from 160 × 120 up to 1280 × 720 and from 1 to 30 fps, respectively. For H.264, we also study the effect of varying the quantization parameter, the number of references, and ME range. In each experiment, the video is played for 22 minutes and 41 seconds. The reported power is the average power consumption during the entire video period. Each reported value is the average of 1361 power readings, each of which is obtained during one second of the video. We assume the default encoding parameters in both X.264 and FFmpeg except for those that are under study. Specifically, in validating the model, we assume the following values, if they are not under study: Number of References $(R) = 3$, ME range $(S) = 16$, Quantization Parameter $(q) = 22$, Scaling Factor $(s) = 22$, Frame-Rate $(F) = 30$, Resolution $(N \times M) = 352 \times 288$, and Maximum Pixel-Rate $(L) = 3041280$ pixel/sec.

To minimize the effect of other processes while running the experiments, we run the computer with a bare minimum set of processes and drivers. In addition, each experiment is repeated four times, and then the overall results are averaged. Furthermore, the power consumption due to other system processes running on the laptop computer is measured before each experiment and then subtracted from the total power consumption. We initially measure the aggregate power of the three phases. To separate the power consumption due to each phase, we follow the following procedure. (1) We measure the power consumption of only capturing and encoding and then subtract it from the aggregate power to get the transmission power consumption. (2) We stream the stored video (thereby no capturing is involved) from the laptop computer to the destination, measure the power consumption for this task, and then subtract the amount from the aggregate power consumption to get the capturing power consumption. (3) We subtract the capturing and the transmission power consumption from the aggregate power consumption to get the encoding power consumption.

In Experimental Setup II, we use for further model validation a CMOS networked surveillance camera [5] and [56]. The used camera is Vivotek IP7139, which has a built-in 10/100 Mbps Ethernet and 802.11b/g WLAN. The distances between the camera and the monitoring station is within 1 meter. As

Table 4.8: Characteristics of Used Standard Video Sequences in Experimental Setup III

Sequence	Duration (s)	Resolution	# Frames
Silent	10	CIF	300
Akiyo	10	CIF	300
Deadline	45.8	CIF	1374
SignIrene	18	CIF	540

the differences in power consumption for different temporal and spatial settings can be in a fraction of a watt, we capture a TV channel with the camera for an average of 10 hours in each experiment. The captured video is streamed to a desktop computer using a built-in streaming server that is supplied by the camera's manufacturer. The reported power consumption is the average of $36,000$ power readings during the recording and streaming period. We experiment with both wired and wireless transmission.

In Experimental Setup III, we conduct experiments to further study and validate the impact of changing both the resolution and quantization/bitrate on encoding power consumption. This setup has the same system as Experimental Setup II, but four standard video sequences are used: Silent, Akiyo, Deadline, and SignIrene, as described in Table 4.8. We downscale each video sequence from the original size down to 10% (specifically, we consider 100%, 90%, ..., 10% of the original size). For each of these sizes, we also produce different quality levels by varying the quantization parameter (from 1 to 31). We measure the power consumption while encoding, and then find the bitrate of the encoded video.

With Experimental Setup III, we also analyze the power consumption at the monitoring station of many-to-one video streaming systems due to upscaling and decoding. Additionally, we analyze the quality of the received videos. As discussed earlier, upscaling the video before decoding, greatly improves video quality. We use the decoded frames to measure the quality compared to the original video. As a metric for perceptual video quality, we use *Structural SIMilarity Index* (SSIM) [65] between two images. SSIM improves the popular *Peak Signal-to-Noise Ratio* (PSNR) metric by considering the similarity of the edges between the two compared images, and it is more consistent with human visual perception. Since the human eye is more responsive to brightness than to color, we use only the Luma (Y) components in the YUV color space. SSIM provides a quality reconstruction metric that considers the similarity of the edges between the produced image and the ideal one, whereas other metrics are based on computing the mean squared reconstruction error.

4.4 Model Validation Results and Analysis

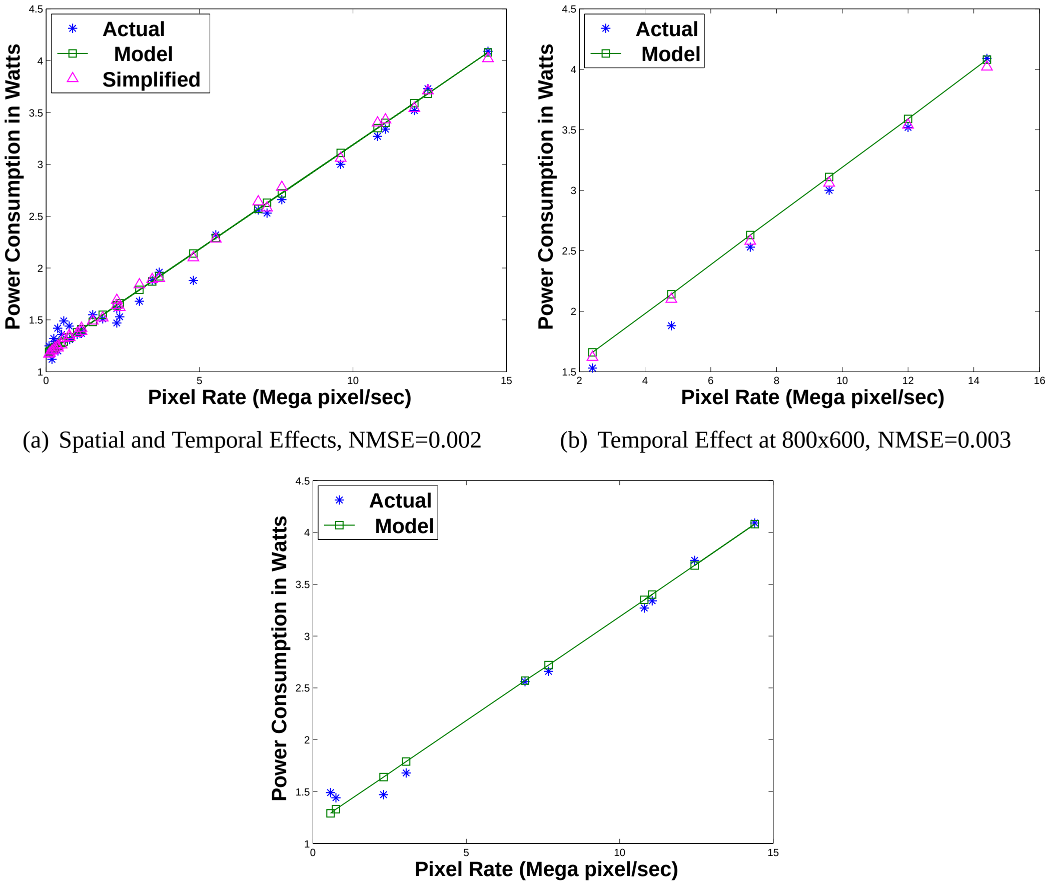
To analyze the goodness of the fit for the developed models, we use and report *Normalized Mean Square Error* (NMSE), where the MSE is divided by the product of the means of the actual and model values. The raw data of main figures can be found at `http://www.ece.eng.wayne.edu/~nabil/power_modeling/power.html`. The results for Experimental Setup I are shown first. For this setup, VLC 1.05 is used unless otherwise indicated. In addition, wireless transmission is assumed unless otherwise indicated.

4.4.1 Validation of the Capturing Model

Figure 4.4(a) validates the developed capturing model (Equation (4.2)) and the simplified capturing model (Equation (4.3)) when both the spatial and temporal resolution are varied. The results show that the model in both forms accurately represents the real behavior. Figures 4.4(b) and 4.4(c) validate the model when only the temporal resolution or spatial resolution is varied, respectively.

4.4.2 Validation of the Power Consumption and Bitrate Models of H.264 Encoding

Figure 4.5 validates the developed power consumption model for encoding (Equation 4.18) for variable frame sizes, frame rates, and quantization parameters. The bitrate in Figure 4.5(c) is varied by changing the quantization parameter. Note that the quantization parameter has a great impact on power consumption and bitrate. Figures 4.6(a) and 4.6(b) validate the power consumption model (Equation 4.18) and the bitrate model (Equation (4.19) as the number of reference frames is varied, respectively. Similarly, Figures 4.7(a) and 4.7(b) validate the models as the ME range is varied, respectively. The



(a) Spatial and Temporal Effects, NMSE=0.002

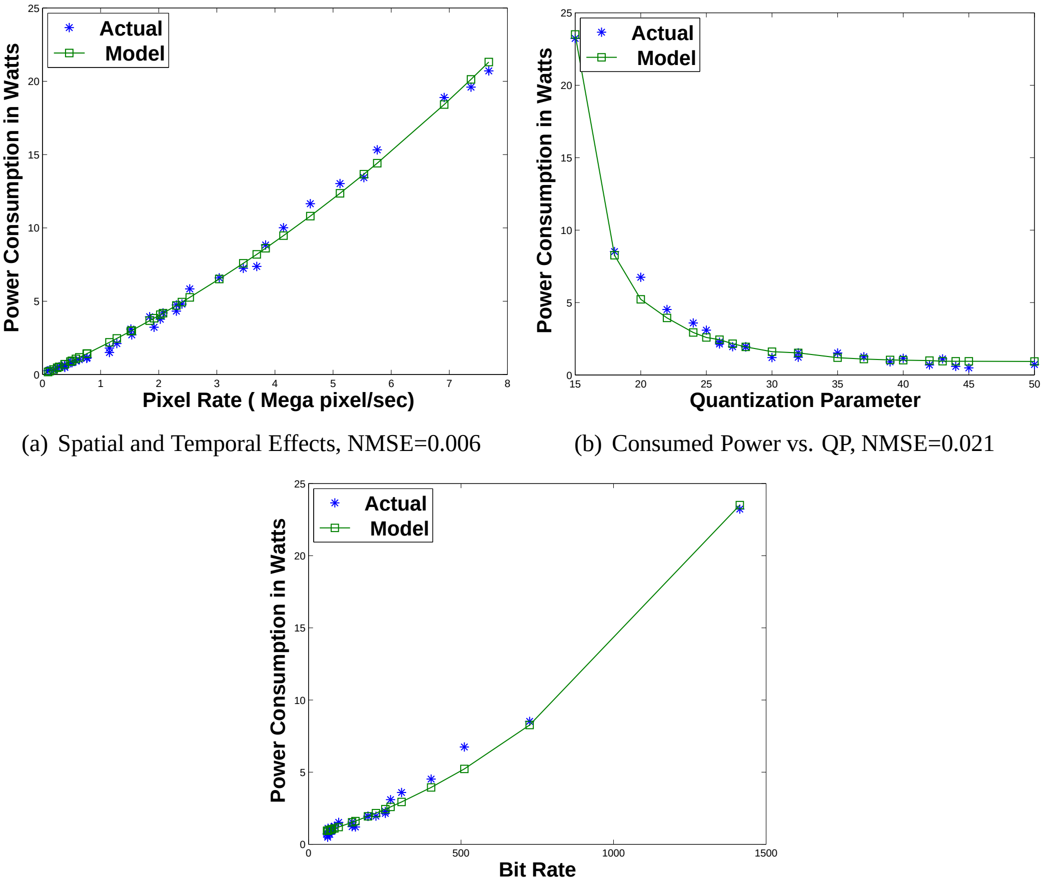
(b) Temporal Effect at 800x600, NMSE=0.003

(c) Spatial Effect at 30 fps, NMSE=0.002

Figure 4.4: Validation of Video Capturing Power Consumption.

inverse linear relationship of the bitrate with the number of reference frames and ME range is clearly evident.

Table 4.9 shows the constants values for the general power consumption model of H.264 (Equation (4.18)) using Experimental Setup I. For the general bitrate model (Equation (4.19)), the constant values on Experimental Setup I are as follows: $c_n = 0.0124$, $c = 3.16$, $c_g = 1249.5$, $c_f = 17.18$, $c_t = 1523.36$, $c_s = 83.03$, and $c_q = 0$.



(a) Spatial and Temporal Effects, NMSE=0.006

(b) Consumed Power vs. QP, NMSE=0.021

(c) Consumed Power vs. Bitrate, NMSE=0.021

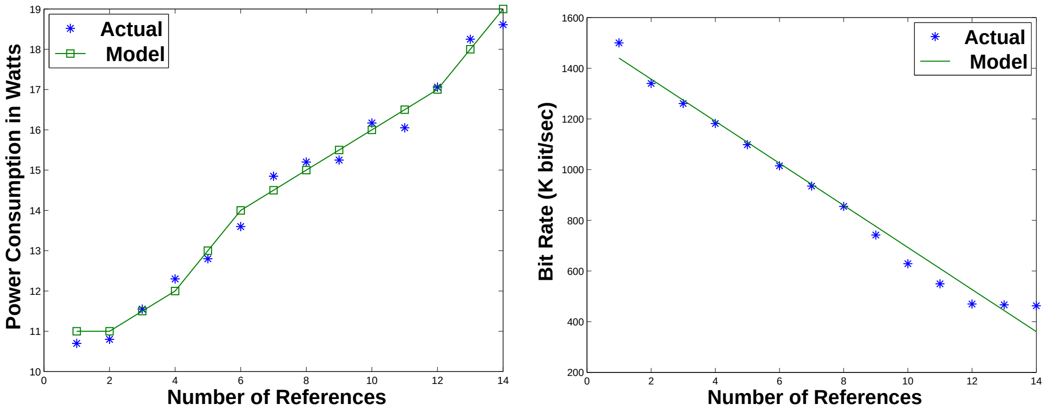
Figure 4.5: Validation of the Impacts of the Spatial, Temp. and QP on Enc. Power Consump.

4.4.3 Validation of the Power Consumption and Bitrate Models of MPEG-4 Encoder

Although the proposed power consumption and bitrate models for encoding (Equations (4.18 and 4.19)) are developed for H.264 due to its popularity and efficiency, they can be generalized for MPEG-4, which share most of the features of H.264. Figure 4.8 shows that both the developed power consumption and bitrate models apply for MPEG-4, but with different constants. Table 4.10 shows the constant values for Experimental Setup I.

4.4.4 Validation of the Transmission Model

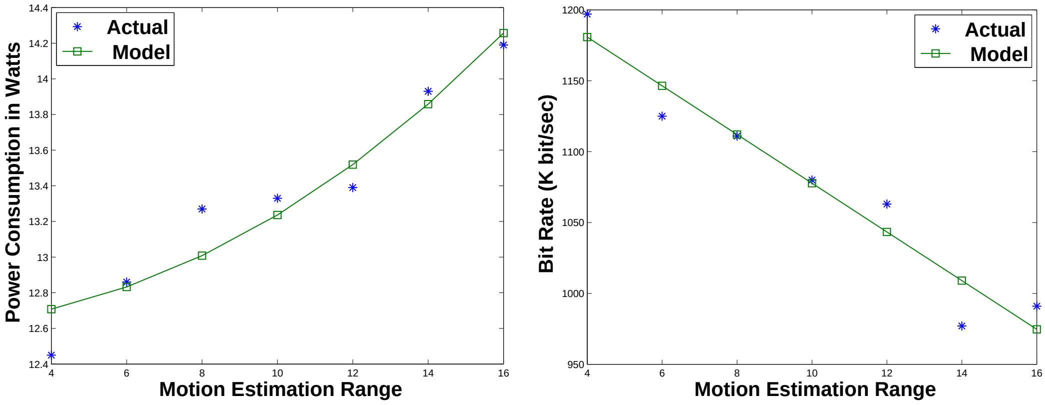
Figure 4.9(a) validates the developed transmission model when both the spatial and temporal resolutions are varied, whereas Figures 4.9(b) and 4.9(c) show the results when varying only the temporal



(a) Consumed Power, NMSE=0.0004

(b) Bitrate, NMSE=0.0026

Figure 4.6: Validation of the Impact of Number of Reference Frames in H.264.



(a) Consumed Power, NMSE=0.00016

(b) Bitrate, NMSE=0.00035

Figure 4.7: Validation of the Impact of ME Range in H.264.

or spatial resolution, respectively. The observed variations from the actual experimental results are primarily due to measurement errors as the power consumed in transmission is low, when compared with other phases.

4.4.5 Validation of the Aggregate Power Consumption Model

Figure 4.10 validates the aggregate power consumption model using Experimental Setup I (with Webcam Pro 9000 and software-based encoding) for both H.264 and MPEG-4. The results for H.264 are shown for both VLC streaming server 1.0.5 and 2.2.4. These results demonstrate the accuracy of the model and that it applies for H.264, MPEG-4, and different versions of H.264 encoders, but with

Table 4.9: Constant Values for H.264 Power Consumption Model [Experimental Setup I]

Constant	Value	Constant	Value	Constant	Value
c_1	$7.437.10^{-9}$	c_2	$4.8.10^{-5}$	c_3	$7.96.10^{-11}$
c_4	$2.3392.10^{-4}$	c_5	$1.58.10^{-6}$	c_6	$1.46.10^{-3}$
c_7	$1.71.10^{-8}$	c_8	0	c	3.16
c_q	0				

Table 4.10: Constants Values for MPEG-4 Power Consumption and Bitrate Models

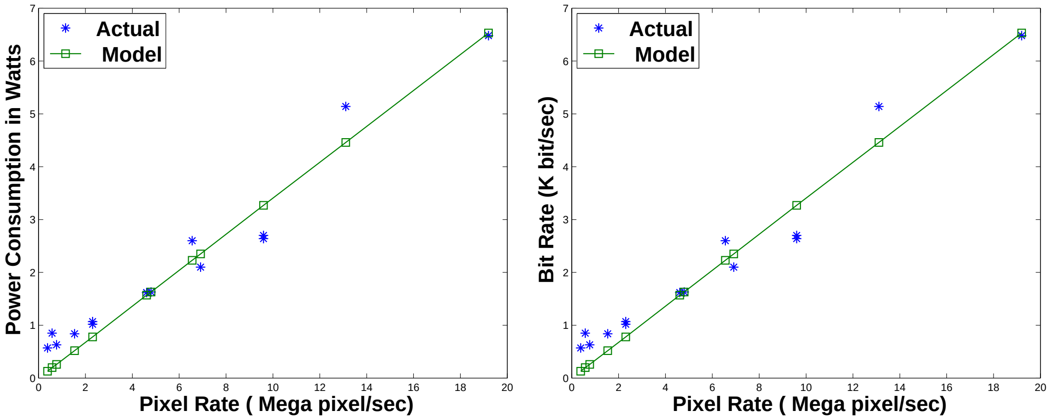
Constant	Value	Constant	Value	Constant	Value
c_1	$16.77.10^{-9}$	c_2	$9.86.10^{-7}$	c_3	0
c_4	10.0	c_5	$2.16.10^{-8}$	c_6	$9.86.10^{-7}$
c_7	0	c_8	0	c	0.5
c_q	3.0	c_n	$2.7.10^{-3}$	c_g	1249.5
c_f	17.18	c_t	1523.36	c_s	1522.36

different constant values.

To further validate the developed aggregate power consumption model, we use Experimental Setup II (with Vivotek IP7139 surveillance camera). Figure 4.11 shows the validation results for both wired and wireless transmission. As expected, wireless transmission consumes more power than wired.

4.4.6 Further Validation and Analysis

Since the spatial resolution and quantization parameter are major contributors to encoding complexity, power consumption, and bitrate, let us analyze the overall behavior and validate the developed models when varying both parameters at the same time. Figures 4.12 and 4.13 illustrate the overall impacts of spatial resolution and quantization parameter on the encoding power consumption and achieved bitrate, respectively, and further validate the developed models. Similarly, Figure 4.14 shows the SSIM video quality results by comparing the decoded and the original videos. These results demonstrate that downscaling the spatial resolution before transmission and then upscaling to the original resolution by the monitoring station can significantly reduce power consumption and bitrate without having a considerable impact on video quality for a wide range of downscaling levels. By combining quantization and



(a) Consumed Power, NMSE=0.043

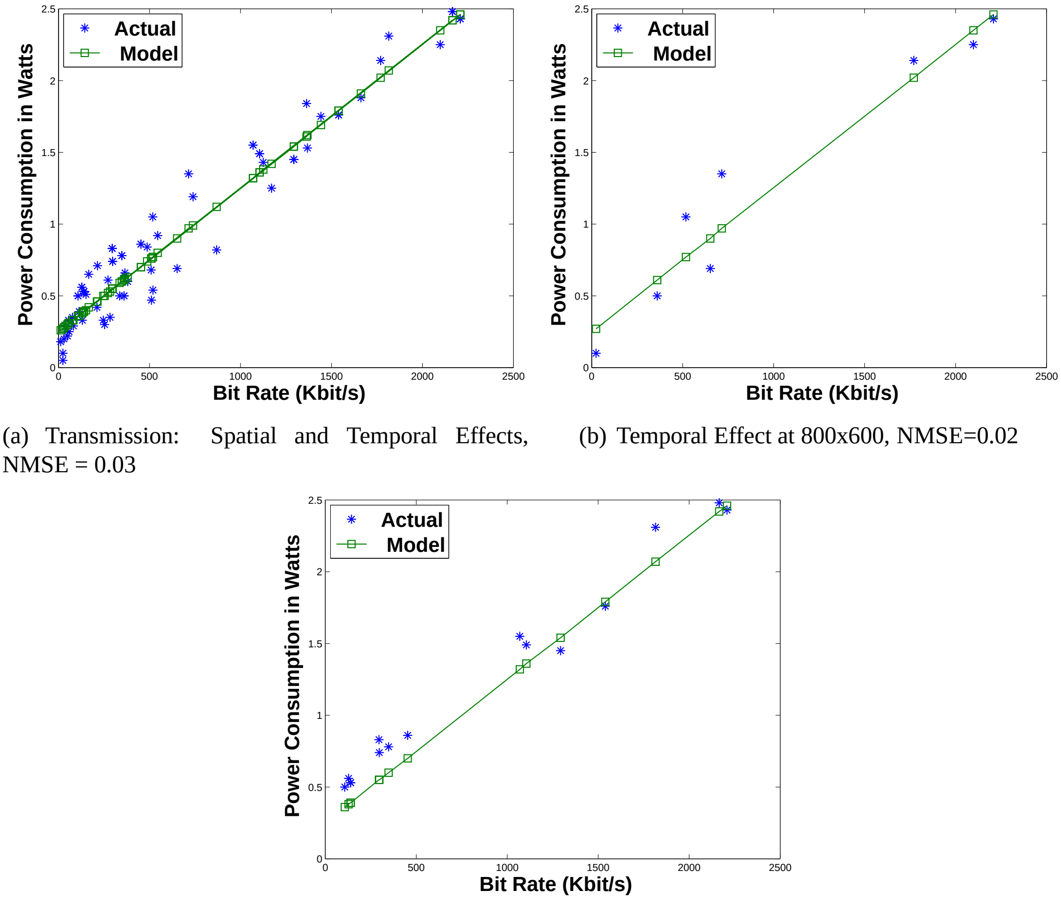
(b) Bitrate, NMSE=0.018

Figure 4.8: Validation of the Spatial and Temp. Effects on MPEG-4 Enc. Power Consumption.

spatial resolution adaptations in H.264 encoding, the bitrate is reduced to 1% of the original bitrate and the consumed power is reduced to 4% of the original, while reducing the quality to only 88% of the original. For MPEG-4, the bitrate is reduced to 1% and the power is reduced to 45%, while keeping the quality higher than 78% of the original.

4.4.7 Analysis of Power Consumption by the Monitoring Station

Let us now analyze the power consumed by the monitoring station for receiving, upscaling, and decoding the received video streams. Figure 4.15(a) shows the consumed power, whereas Figure 4.15(b) shows the percentage of power consumption for handling one stream by the monitoring station to the encoding power consumed by the source. Note that the power consumed by receiving, upscaling, and decoding one stream is smaller than 0.5 Watt and the percentage of the consumed power relative to the encoding power consumption is smaller than 2% for the spatial resolution of half the original and quantization parameter smaller than 20. The overall power consumed by the monitoring station is expected to be proportional to the number of received streams, but sublinearly as the power consumed in receiving n streams is less than n times the power consumed by each due to the nature of operation of the receiver.



(a) Transmission: Spatial and Temporal Effects, NMSE = 0.03

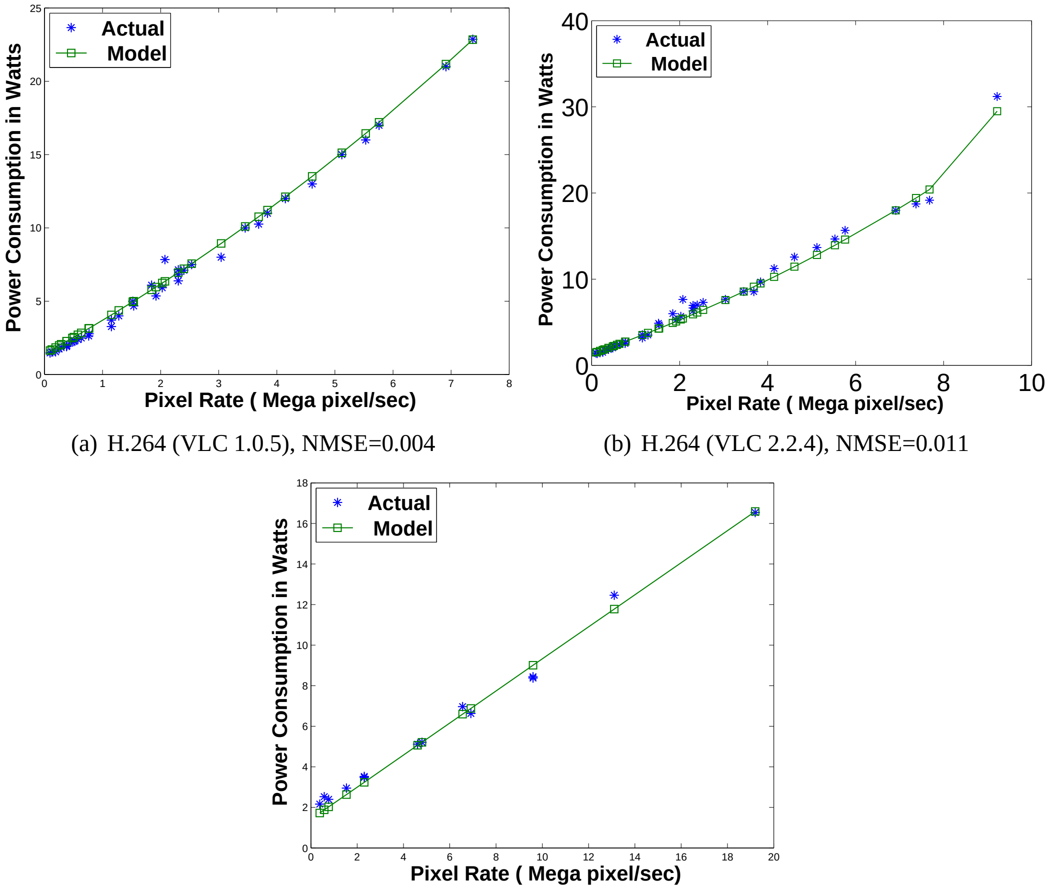
(b) Temporal Effect at 800x600, NMSE=0.02

(c) Spatial Effect at 30 fps, NMSE=0.02

Figure 4.9: Validation of the Transmission Power Consumption.

4.5 Conclusions and Future Work

We have developed an aggregate power consumption model for live video streaming systems. The model can help in the dynamic control of various camera/sensor settings, including resolution, frame rate, and quantization, to achieve the best overall tradeoff in terms of power consumption, bitrate, and quality. Specifically, we have modeled the video capturing, encoding, and transmission aspects and then have provided an overall model of the power consumed by the video sources. We have also analyzed the power consumed by the monitoring station in many-to-one systems due to the reception, upscaling, and decoding of the received video streams. In addition, we have analyzed the perceived quality at the monitoring station. Moreover, we have modeled the output bitrate of video encoding. Furthermore,



(a) H.264 (VLC 1.0.5), NMSE=0.004

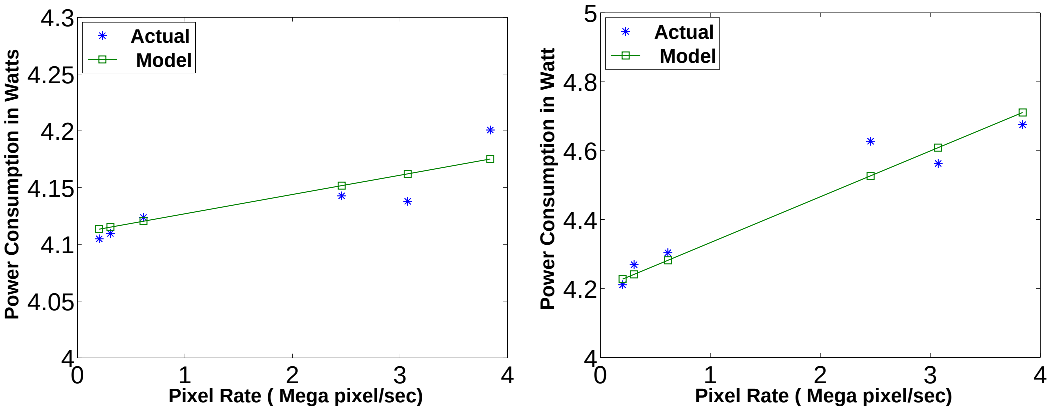
(b) H.264 (VLC 2.2.4), NMSE=0.011

(c) MPEG-4, NMSE=0.003

Figure 4.10: Validation of the Aggregate Power Consumption Model.

we have validated the developed models through extensive experiments using two different systems and different video contents.

The main conclusions can be summarized as follows. (1) The overall computation complexity for all phases can approximately be modeled as a linear function of the pixel rate when varying only the frame rate and frame size. (2) For high spatial and/or temporal resolution, the video encoding consumes more than 90% of the power, while capturing consumes less than 6% and transmission less than 4%. (3) H.264 generally consumes more than three times the power consumed by MPEG-4. (5) The quantization parameter affects power consumption in an exponential fashion. (6) Other encoding parameters, such as the number of references and the ME search range, vary the power consumption by up to 10%. (7)

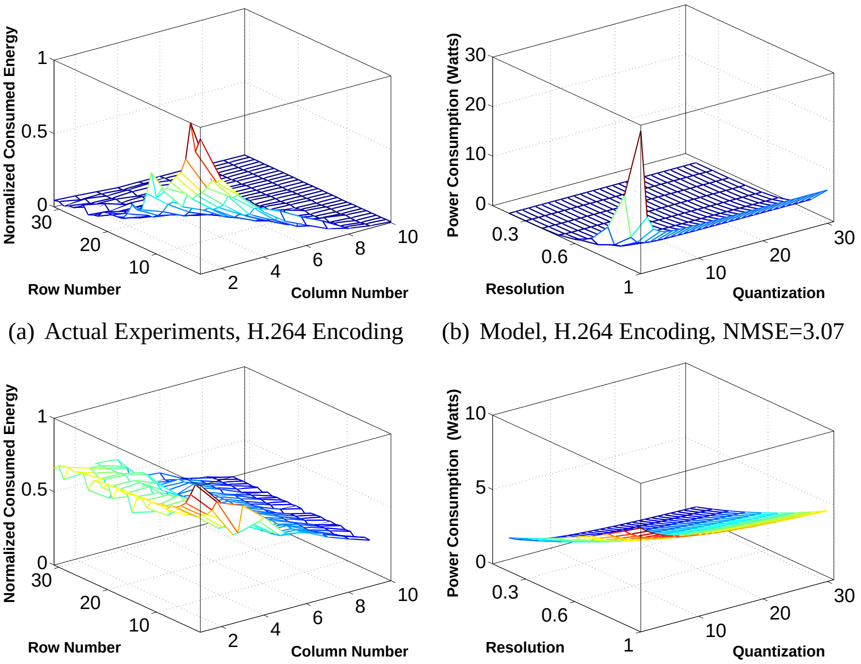


(a) Wired Tx, MPEG-4, NMSE=0.00002 (b) Wireless Tx, MPEG-4, NMSE=0.00015

Figure 4.11: Further Validation of the Aggregate Power Consumption Model.

Tuning of parameters must be done based on power consumption, video quality and bitrate tradeoffs. (8) The complexities of inter-prediction, intra-prediction, RDO mode selection, and sub-pixel search can be expressed as a linear function of the pixel rate. Similarly, the aggregate power consumption is a linear function of the pixel rate. (9) By combining quantization and spatial resolution adaptations in H.264 encoding, the bitrate is reduced to 1% of the original bitrate and the consumed power is reduced to 4% of the original, while reducing the quality to only 88% of the original. For MPEG-4, the bitrate is reduced to 1%, the power is reduced to 45%, while reducing the quality to only 78% of the original. (10) The power consumed by upscaling and decoding one stream by the monitoring station is smaller than 0.5 Watt per stream in the considered system. The percentage of this power relative to the encoding power consumption is smaller than 2% for a spatial resolution of half the original and a quantization parameter smaller than 20.

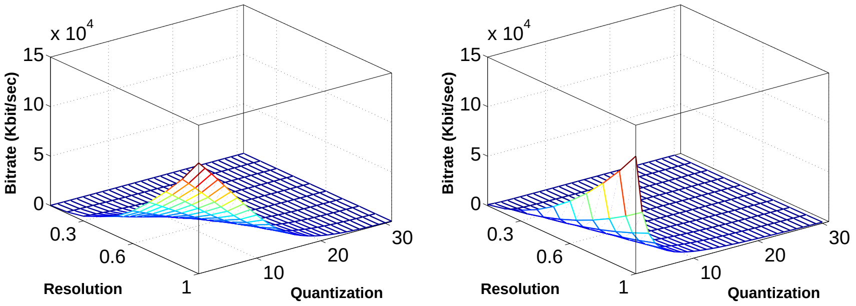
In future work, we will adapt the encoding model to other encoders, including High Efficiency Video Coding (HEVC) and VP9.



(a) Actual Experiments, H.264 Encoding (b) Model, H.264 Encoding, NMSE=3.07

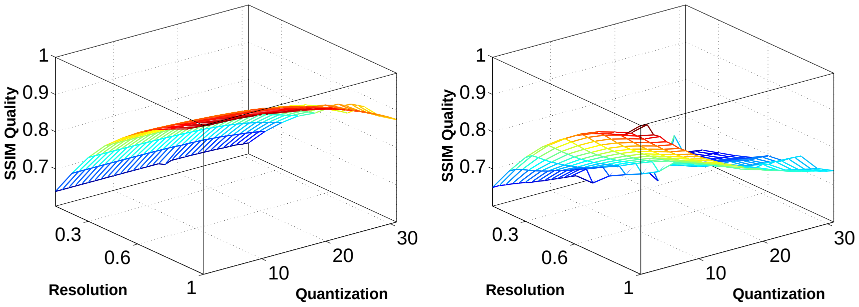
(c) Actual Experiments, MPEG-4 Encoding (d) Model, MPEG-4 Encoding, NMSE=0.041

Figure 4.12: Effect on Power Consumption by Varying Quantization and Resolution.



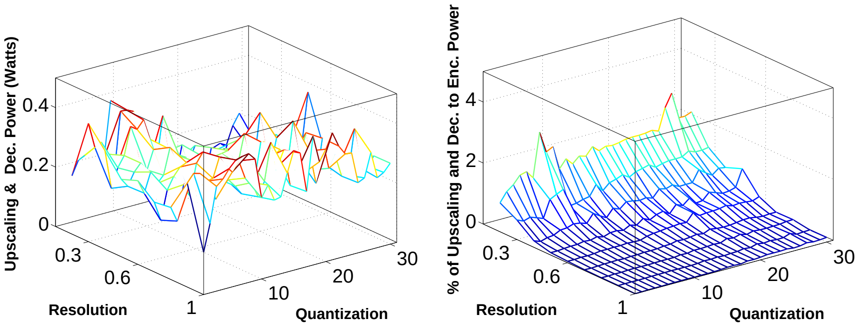
(a) Actual Experiments, H.264 Encoding (b) H.264 Encoding Model, NMSE=0.0041

Figure 4.13: Effect on Bitrate by Varying Quantization and Resolution.



(a) H.264 Encoding (b) MPEG-4 Encoding

Figure 4.14: Effect on SSIM Quality by Varying Quantization and Resolution.



(a) Receiving, Upscaling, and Decoding (b) %Consumed Energy Relative to Encoding

Figure 4.15: Power Consumption by the Monitoring Station [Experimental Setup III]

CHAPTER 5 FAST HEVC ENCODING BY HISTORY AND ENTROPY-BASED LCU PARTITIONING

5.1 Introduction

The popularity of *High and Ultra High Definition* (HD and UHD) videos increases the demand for real-time applications with such standards. These high-resolution videos consume high bandwidth especially in real-time systems and limited bandwidth channels, such as wireless systems. That demand urged the video encoding community to develop the new encoding standard called High-Efficiency Video Coding (HEVC) to improve the coding efficiency while retaining the quality as in H.264 encoding standard [22, 79].

In comparison to H.264, HEVC offers about double the data compression ratio at the same level of video quality. It supports resolutions up to 8192×4320. HEVC introduces many different techniques in order to improve the coding efficiency, including the introduction of an adaptive quad tree coding [80, 22, 81, 82, 31, 30, 29, 34, 35, 36, 37, 32]. The improved compression performance is the output of high computational algorithms due to the newly introduced techniques such as adaptive quad tree structure, extra intra-prediction modes, and the comprehensive Rate-Distortion Optimization (RDO) calculations in such a structure.

HEVC data structure includes the *Largest Coding Unit* (LCU), *Coding Unit* (CU), *prediction unit* (PU), and *transform unit* (TU). Frames in HEVC are partitioned into LCUs of (64×64) sizes in the adaptive quad tree structure. If a CU of size (64×64) splits, it is divided into four CUs of sizes (32×32). In addition, each CU of size (32×32) can be subdivided into four CUs of sizes of (16×16). Furthermore, each CU of size (16×16) can be subdivided into four additional CUs with sizes of (8×8) [80]. In the original HEVC encoder, *Rate Distortion Optimization* RDO algorithm is used for the partitioning of the *Largest Coding Unit* (LCU) into CUs. Unfortunately, the computation complexity of RDO is extremely high for real-time application which opens the door for sub-optimal LCU partitions that reduce the encoding time [82].

In order to reduce the encoding time, many studies have been proposed with different techniques. The different studies can be categorized into the following approaches: dept-based, machine learning, prediction based on residuals, entropy-based, and total number of blocks. Study [33] proposed a depth-based algorithm to exploit spatial and temporal correlations for fast CU size decision. The decision of splitting or terminating is based on the depth of the spatial and temporal neighbors. The authors of [33] claim a reduction in encoding time of 43% compared to the original HM5.0 encoder for the HD test sequences.

Using *Support Vector Machine* (SVM), study [34] proposed a CU splitting fast termination algorithm. CU splitting is modeled as a binary classification problem, on which SVM is applied. The paper claims that the proposed algorithm can achieve about 41.9 % time saving compared with the HEVC reference software. Study [36] proposed CU early-termination algorithm that takes advantage of the correlations between the *Mean Square Error* (MSE) of prediction residuals and the splitting decision in the current CU level. According to the paper, the proposed algorithm achieves up to 34.83% average encoding time reduction.

Study [25] suggested an approach based on how much information is contained within the block, which is measured by a metric called **entropy**. Entropy-based algorithm improved the encoding speed to be faster than all other existing algorithms that intend to do so. The encoding speed is 3.5 faster than the original RDO algorithm with acceptable average bitrate.

In trying to avoid the limitation of the depth based partitioning approach [33], study [39] used *Total number of Blocks* (TnB), where it made the decision of splitting/termination based on the total number of blocks in the neighbors CUs. TnB is based on the number of sub CUs that contained within a certain CU. The author claimed that the total number of blocks provides more insight into $CU_{current}$ structure of the neighbors which make the decision of termination/splitting more accurate compared to the RDO method than the depth approach. The author claimed a reduction of 44.89% in encoding time compared to RDO that is implemented in the original software.

Although the idea of TnB is acceptable, the study was limited in many aspects. First, from performance evaluation perspective, the experiments were based on one value of QP, which does not show the effect of changing the QP. In addition, it does not consider Bjontegaard's metric although it is very popular in finding the performance of encoders. The sequences are mostly QCIF and CIF(15 out of 18 are QCIF and CIF), only one sequence was of HD resolution. The selection of such range of sequences not only lack wide distribution but also does not cover the resolutions that urge for HEVC development. In addition, QCIF and CIF are not the right sequences for an algorithm that make the decision based on the neighbor decision history, because the existing neighbors are not enough. Second, the so many schemes listed as contribution are more like the tasks completed searching for the right algorithm, no conclusion for which scheme perform better. The most important issue is the reduction in encoding time (or encoding speed), no scheme of [39] study outperform the entropy-based algorithm. Many items in the schemes are repeated and some items are included in others. The main scheme which is the total number of blocks is not studied well on wide range of HD sequences and QPs with the right metrics to find the thresholds of termination and splitting at all depths. Third, the study introduced two schemes with the name (Hybrid). Both schemes are based on the same items customized in a way to maximize the performance for the encoding of the sequences in the study. Unfortunately, these sequences are not chosen from high definition were HEVC is developed for. Fourth, [39] study schemes are based on statistical analysis. The thresholds in these schemes are based on one QP and low-resolution sequences. It is not clear how these thresholds can change based on QP and resolution. Finally, the dataset that is used in developing the algorithms and their thresholds are the same dataset that are used for testing.

Our approach is different from depth-based approach [33] by considering same size neighbors instead of considering 64×64 blocks neighbors. In addition, it is different from depth and TnB [39] by considering entropy-based conditions and entropy-based weights for the neighbors' contribution to the terminating/splitting decision. It is different from entropy-based approach [25] by considering spatial and temporal neighbors in addition to the conditions that are based on the entropy value of the blocks.

In this chapter, we concentrate only on a frame structure and partitioning it to smaller blocks. We focus on reducing the computational complexity of the adaptive quad tree coding by predicting the optimal LCU partition. This prediction increases the encoding speed implemented in HEVC while preserving the coding efficiency and video quality as in HEVC with Rate Distortion Optimization (RDO) option.

In this dissertation, we refer to HEVC with RDO option for partition, simply as RDO. We develop an algorithm that predicts the size of the block without iterating through the exhaustive RDO method. Our Algorithm decides to split the block or not based on the correlation between the content of the block and the content of the previously adjacent encoded blocks in space and time. The algorithm prediction is based on the weighted average of the decision of those adjacent blocks (called spatial and temporal neighbors). To prevent error propagation, we introduce other content conditions that have to be satisfied. The content conditions are based on the entropy of the block and its neighbors. We demonstrate the effectiveness of the proposed algorithm in comparison with RDO and entropy-based approach through extensive experiments. In addition, we compare our algorithm with TnB method discussed above.

The experiments are conducted on 17 different video sequences of resolutions ranging from WQVGA (416×240) up to UHD (3840×2160) with up to 6 different QPs. These sequences are in the raw YUV color space format. The information contents and levels of motion in such sequences cover a wide range of details and mobility which cover different spatial and temporal redundancy. The considered performance metrics include the average encoding speed, the average *Encoding Speedup Enhancement* (ESE), the average *Encoding Time Reduction* (ETR), the average bitrate, the average *Peak Signal-to-Noise Ratio* (PSNR), and *Bjontegaard's Delta*(BD) metric of both bitrate and PSNR.

The rest of this chapter is organized as follows. Section 5.2 discuss the proposed algorithm. Subsequently, section 5.3 discusses the performance evaluation methodology. Section 5.4 presents and analyzes the main results. Finally, section 5.5 present the conclusion.

5.2 Proposed Algorithm

We develop an algorithm, called *History and Entropy-based LCU partitioning* (HELP), to reduce the encoding time without considerable loss of coding efficiency and video quality performance. Reducing the encoding time not only increases the encoding speed, but also lowers the power consumption. The algorithm predicts the size of the CU based on the partition of the same size CUs in the spatial and temporal (co-locator) neighborhood that have been processed. In addition to neighborhood partitioning history, another condition has to be satisfied in order to prevent error propagation. Error propagation conditions are based on the entropy of the block currently being processed ($CU_{current}$).

HELP algorithm predicts the partition decision for $CU_{current}$ based on the weighted average of the *Termination Possibility* (TP) of all the spatial and temporal neighbors. The partition decision is to split the block to four blocks or to terminate the process of searching for the optimal partition for $CU_{current}$.

The algorithm predicts to split the block for four blocks or terminate the process of searching for the optimal partition of $CU_{current}$ based on the weighted average of TPs of all the spatial and temporal neighbors, which we called *Termination Possibility Average* (TPA). TP is defined as the likelihood that $CU_{current}$ will terminate or split based on the decision that have been made for the neighbor block. The neighbor block is each processed block of the same size that is either temporally co-located or spatially share an edge or a corner with $CU_{current}$.

The algorithm uses the factor TPA to make the decision of terminating or splitting. The TPA of current block $CU_{current}$ is based on the decision of partition taken for its neighbors. This factor combined with $CU_{current}$ entropy value is the basis for our termination or splitting decisions. TPA is defined formally as follows:

$$TPA = \frac{1}{\sum_{i=1}^{N} W_i} \times \sum_{i=1}^{N} W_i \times TP_i, \tag{5.1}$$

where W_i is the weight defined in Equation 5.2, N is the number of spatial and temporal neighbors, and TP_i is termination possibility for the neighbor i. As shown in Figures 5.1 and 2.2, the variable i can be 1, 2, 3, 4, 5, or 6 for spatial and temporal neighbors Co-located, Left, Above Left, Above, Above Right, and Under Left, respectively.

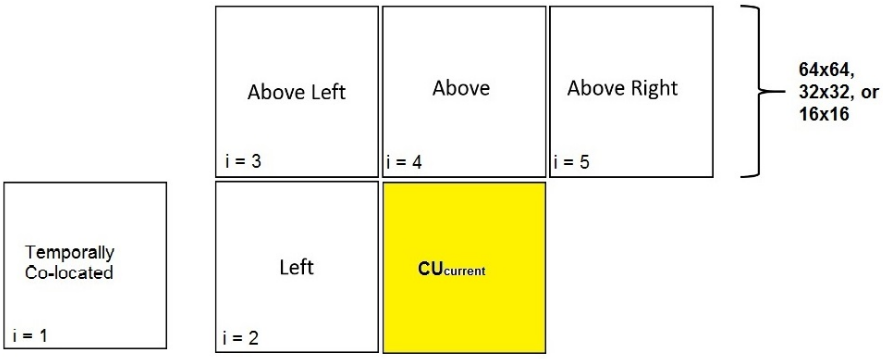


Figure 5.1: Illustration of Spatial and Temporal Neighbors

The termination possibility for each neighbor TP_i is equal to 0 if CU_i has smaller CUs, and it is equal to 1 otherwise. Thus, TP_i can be represented as follows:

$$TP_i = \begin{cases} 0 & \text{if } CU_i \text{ has smaller CUs} \\ 1 & \text{otherwise}, \end{cases}$$

where i represents spatial and temporally co-located neighbors of the same size as of $CU_{current}$.

$$W_i = \frac{1}{1 + |Entropy_{current} - Entropy_i|}, \tag{5.2}$$

where W represents the weight, $|Entropy_{current} - Entropy_i|$ is the absolute value of $(Entropy_{current} - Entropy_i)$. For example, if $Entropy_{current}$ = 1 and $Entropy_1$ = 3, then the weight for neighbor at location i is equal to $1/3$ (i.e. W_i = $1/(1 + |1 - 3|) = 1/3$). The weight value is bounded in the closed real interval [0, 1] to map the correlation strengths from no correlation to the highest correlation, respectively. The maximum weight has the highest correlation (W_{max} = $1/(1 + 0) = 1$) and the minimum weight has no correlation (W_{min} = $1/(1 + \infty) = 0$). The entropy value for CU ($Entropy_{cu}$) can be calculated based on Equation (2.3) in Chapter 2.

Figure 5.2 illustrates the concept of TP_i. The figure shows a 64×64 CU which is divided into 4 CUs, 3 of them are 32×32. The fourth CU which is the bottom left is divided into 4 CUs, 3 of them are 16×16. The fourth CU which is the top left is divided into 4 CUs each of them is 8×8.

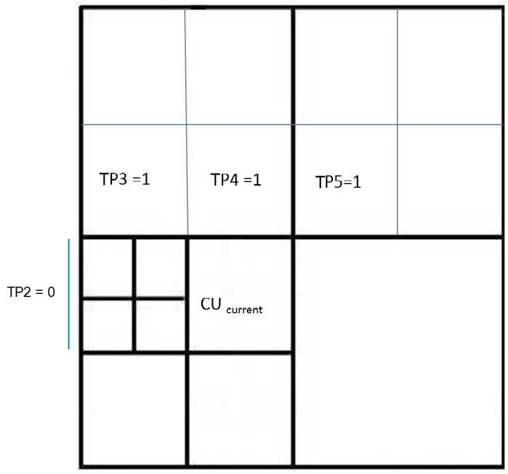


Figure 5.2: Illustration Of The Concept OF TPi[$CU_{current}$ is 16×16]

The weighted average of TPs for all the neighbors not only depends on the value of TP for each neighbor, but also it depends on the weight (W) between that neighbor and $CU_{current}$. The higher the weight the more TP of that neighbor will affect the termination decision. For example, if the weights for CUs in Figure 5.2 are [W2 = 1.0 , W3 = 0.3, W4 = 0.1, W5 = 0.1]. $TPA_{CUcurrent} = [1/(W2+....+W5)] \times [W2 \times TP2 + ... + W5 \times TP5] = (1/1.5) \times (1.0 \times 0 + 0.3 \times 1 + 0.1 \times 1 + 0.1 \times 1) = 0.33$. In this example, we notice that the neighbor that has the highest correlation ($W = 1$) dominates the decision. The final decision is to split if the error propagation condition satisfied although there are three neighbors does not have smaller CUs, but they have lower correlation with $CU_{current}$.

The HELP algorithm partitions the LCU based on the following: TPA, the value of the entropy for $CU_{current}$ is compared to the average of the entropy of all possible partitions in the LCU, the entropy value of the temporally co-located block, and the entropy value of each of the spatial neighbors. Figure 5.3 shows the pseudocode of our proposed HELP algorithm.

```
 if ( Depth $< 3$ AND CU Neighbors $> 4$) //Current CU should be 16x16 or larger and it should have at least 5 neighbors
// Initialize .....

    $depthFactor = 0.1 \times Depth$; // The probability of termination is higher for high depth value
    $tpFactor = TPA - 0.5$; // For TPA $>= 0.70$, Higher TPA value leads to more terminations. For TPA $<= 0.30$, Lower TPA leads to more splittings
    $dtFactor = depthFactor + tpFactor$; // We add the above two factors to simplify the algorithm

// Terminate Conditions.....
  if ((TPA $>= 0.70$ ) AND // This is the main condition to terminate, it should be always satisfied to terminate
   ( (entropy of inspected CU $<= 1.2+$ dtFactor) OR // This is the 1st Error Propagation Prevention Conditions (EPPC), one EPPC is enough to terminate
    (abs(Entropy Of Inspected CU $-$ Average Entropy) $<= (0.15\times$ Average Entropy $+$ dtFactor)) OR // 2nd EPPC
    (abs(Entropy Of Inspected CU $-$ CU Colocated Entropy) $<= (0.5+$ dtFactor)) OR // 3nd EPPC
    ((Entropy Of InspectedC U $+$ Average Entropy) $<= (3.5+$ dtFactor)) )) // 4nd EPPC
  Terminate // If the main condition and one of the EPPCs satisfies, terminate the search for optimal size
//Split Conditions .....
  else if $(TPA <= 0.30)$ AND // Main split condition
   (((Entropy Of Inspected CU $>= (3.0+$ dtFactor)) OR // 1st EPPC
    $(abs($Entropy Of Inspected CU $-$ Average Entropy) $>= (0.15\times$ Average Entropy $+$ dtFactor)) OR // 2nd EPPC
    ((Entropy Of Inspected CU $+$ Average Entropy) $> (6.0+$ dtFactor)) ) ) // 3nd EPPC
  Split // If the main condition and one of the EPPCs satisfies, split the current CU to $4$ CUs
 else Do Full RDO // Use the original RDO method if the above conditions does not satisfy
//The constants are based on [25] statistics tuned by us to fit HD, depth factor, and TPA value. The 0.15 means splitting will not decrease the entropy if $CU_{current}$ entropy and the entropy of the possible smaller CUs are close to each other, 3.0 means the entropy is high, which means splitting will decrease the entropy, and 1.2 means the entropy is low, which predict termination. The constants 3.5 and 6.0 are based on pure statistics by [39] tuned to HELP algorithm.
```

Figure 5.3: Pseudocode of the HELP Algorithm

HELP is different from the depth-based approach [33] by considering same size neighbors instead of considering 64×64 blocks neighbors. In addition, it is different from depth and TnB [39] by considering contents based weight for the neighbors and entropy based conditions. It is also different from entropy-based algorithm [25] by considering spatial and temporal neighbors in addition to the conditions which is based on the entropy value of the blocks. The conditions are based on [25] and [39] studies. It is different from these studies by adapting to depth and TPA value. The termination or splitting in HELP has to satisfy TPA threshold and one of the entropy conditions which introduced to prevent error propagation. Finally, HELP uses the entropy as a second parallel condition, whereas [25] and [39] use them as first conditions to make the decision.

5.3 Performance Evaluation Methodology

We use the following performance metrics to compare various algorithms: *Encoding Speed Enhancement* (ESE), *Bjontegaard Delta-PSNR* (BD-PSNR), *Bjontegaard Delta-Rate* (BD-Rate), *Peak Signal to Noise Ratio*($PSNR$), *Encoding Time Reduction* (ETR), the average bitrate, and the average *Peak Signal-to-Noise Ratio* (PSNR).

HEVC reduces the bitrate to half for the same quality but with very slow encoding speed. Thus, the encoding speed is the main attribute to evaluate HEVC enhancement algorithms. We propose *Encoding Speed Enhancement* (ESE) metric, which measures the relative enhancement of the proposed algorithm to the RDO. ESE can be determined as follows:

$$ESE = \frac{EA_{encodingSpeed} - RDO_{encodingSpeed}}{RDO_{encodingSpeed}}, \tag{5.3}$$

where ESE is encoding speed enhancement of EA over RDO, EA is the algorithm being compared with RDO, such as TnB or HELP.

As PSNR is not to compare encoding algorithms in both quality and coding efficiency [83], we

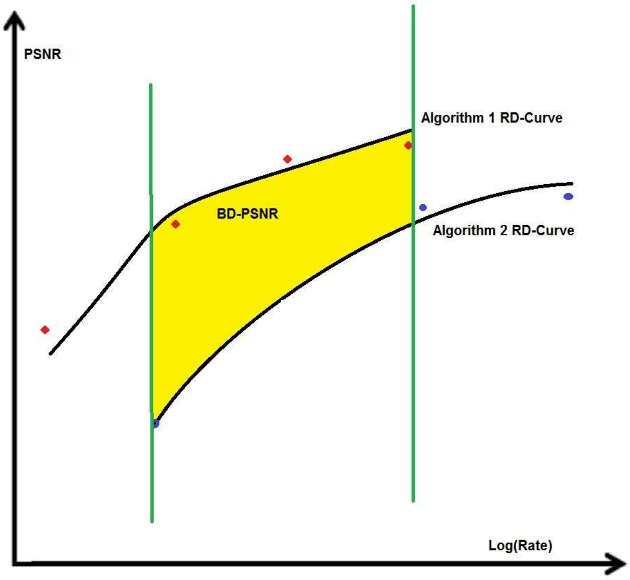


Figure 5.4: Illustration of Bjontegaard BD-PSNR

use a popular metric for evaluation of codecs called Bjntegaard model. The *Bjntegaard Delta-PSNR* (BD-PSNR) metric is used to measure the average PSNR differences between two RD curves obtained from encoding videos of varying bitrates. BD-PSNR measures in dB the average PSNR difference for the same bitrate, while *Bjontegaard Delta-Rate* (BD-Rate) measures the average percent in bitrate difference for the same PSNR.

BD-PSNR can be approximated by the difference between the integrals of the fitted two R-D curves of the algorithms under comparison divided by the integration interval. For bitrate reduction, we use BD-Rate metric which is the average bitrate difference between the two R-D curves as approximated in [84, 85, 86].

The BD-PSNR between two RD curves is calculated by the difference between the area under these curves divided by the logarithm of the bitrate interval. Formally, we can express BD-PSNR as follows:

$$BD - PSNR \approx \frac{1}{r_H - r_L} \int_{r_L}^{r_H} (D_2(r) - D_1(r))dr, \tag{5.4}$$

where BD-PSNR computed between the two fitted *Rate-Distortion* (RD) curves $D_1(r)$ and $D_2(r)$, respectively, and r_L and r_H, determines the higher starting and the lower end rates of the two curves. They can be calculated as follows:

$$r_L = max(min(r_{1,1}, ..., r_{1,N_1}), min(r_{2,1}, ..., r_{2,N_1})), \tag{5.5}$$

and

$$r_H = min(max(r_{1,1}, ..., r_{1,N_1}), max(r_{2,1}, ..., r_{2,N_1})). \tag{5.6}$$

where r refers to the logarithm of the bitrate ($r = log(R)$). Figure 5.4 shows graphically how to calculate BD-PSNR (BD_{PSNR}).

The average bitrate *Bjontegaard Delta-Rate* (BD-Rate) between two RD curves is the horizontal area under the curves divided by the PSNR interval, which can be approximated as

$$BD - Rate \approx \ 10^E - 1, \tag{5.7}$$

where

$$E = \frac{1}{D_H - D_L} \int_{D_L}^{D_H} (r_2(D) - r_1(D))dD, \tag{5.8}$$

where D represents the distortion in terms of $PSNR$, BD-Rate computed between the two fitted *Rate-Distortion* (RD) curves $r_1(D)$ and $r_2(D)$, respectively, and D_L and D_H determines the higher starting and the lower end PSNRs of the two curves. They can be calculated as follows:

$$D_L = max(min(D_{1,1}, ..., D_{1,N_1}), min(D_{2,1}, ..., D_{2,N_1})), \tag{5.9}$$

and

$$D_H = min(max(D_{1,1}, ..., D_{1,N_1}), max(D_{2,1}, ..., D_{2,N_1})). \tag{5.10}$$

The $PSNR$ between an original frame A and the corresponding encoded frame B can be given as follows:

$$PSNR(dB) = 10 \times \log \frac{MAX^2}{MSE}, \tag{5.11}$$

where MSE and MAX represent the Mean-Square Error, and the maximum possible pixel value of the image, respectively. When each pixel is represented as 8 bits, $MAX = 255$. MSE can be given by

$$MSE = \sum_{i=1}^{n} \sum_{i=1}^{m} \frac{(A_{ij} - B_{ij})^2}{n \times m}, \tag{5.12}$$

where m and n represent the width of the image in pixels and the height of the image in pixels, respectively.

Another important attribute of HEVC partitioning algorithms is the encoding time. The performance of any proposed algorithm can be measured in terms of *Encoding Time Reduction* (ETR). We measure ETR by the difference between the encoding time of the proposed algorithm and RDO relative to the RDO encoding time. In Table 5.4, we use ETR for evaluation. ETR is defined as follows:

$$ETR = \frac{EA_{encodingTime} - RDO_{encodingTime}}{RDO_{encodingTime}}, \tag{5.13}$$

where ETR is encoding time reduction of RDO over EA, EA is the algorithm being compared with RDO, such as TnB or HELP.

As shown in Table 5.1, we use low delay main with pattern IBBB configuration, which use a GOP of all B frames except the first, which is I frame [87].

Table 5.1: Unit Definition and Coding Structure

```
==================================== Unit definition ====================================
MaxCUWidth                 :64    # Maximum coding unit width in pixel
MaxCUHeight                :64    # Maximum coding unit height in pixel
MaxPartitionDepth          :4     # Maximum coding unit depth
QuadtreeTULog2MaxSize      :5     # Log2 of maximum transform size for quadtree-based TU coding (2...6)
QuadtreeTULog2MinSize      :2     # Log2 of minimum transform size for quadtree-based TU coding (2...6)
QuadtreeTUMaxDepthInter    :3
QuadtreeTUMaxDepthIntra    :3
=================================== Coding Structure ===================================
IntraPeriod                : -1   # Period of I-Frame ( -1 = only first)
DecodingRefreshType        : 0    # Random Accesss 0:none, 1:CDR, 2:IDR
GOPSize                    : 4    # GOP Size (number of B slice = GOPSize-1)
```

We implement the algorithm in the HEVC Test Model, specifically HEVC HM 13.0 [88]. To minimize the effect of other processes while running the experiments, we run the computer with a bare minimum set of processes and drivers. In addition, each experiment is repeated three times on each of the three computers. We consider the results of the maximum encoding speed of each computer, we show the encoding speed results of M4800 computer in all experiments except when we compare the encoding speed on different computers Figure 5.14. The quality and the bitrate are deterministic and they are the same whether with the different experiment on the same computer or on different computers.

The three computers have the following configuration: (1) Dell Precision M4700 with x64-based PC, Intel(R) Core(TM) i7-3840QM CPU @ 2.80GHz, 4 Core(s), 8 Logical Processor(s), 16.0 GB Installed Physical Memory (RAM), and 64-bit Microsoft Windows 8.1 pro. (2) Dell Precision M4800 with x64-based PC, Intel(R) Core(TM) i7-4810QM CPU @ 2.80GHz, 4 Core(s), 8 Logical Processor(s), 32.0 GB Installed Physical Memory (RAM), and 64-bit Microsoft Windows 7 Enterprise. (3) HP EliteBook 820 G1 with x64-based PC, Intel(R) Core(TM) i5-4310U CPU @ 2.00GHz, 2 Core(s), 4 Logical Processor(s), 8.0 GB Installed Physical Memory (RAM), and 64-bit Microsoft Windows 7 Enterprise.

A description of the main characteristics of each of the used sequences [89, 90] in testing the algorithm is shown in Table 5.2. To determine the thresholds of the algorithms, we used a subset of this sequences. These sequences are crop_Traffic, Basket_ball_Drill, and Basket_ball_Pass sequences.

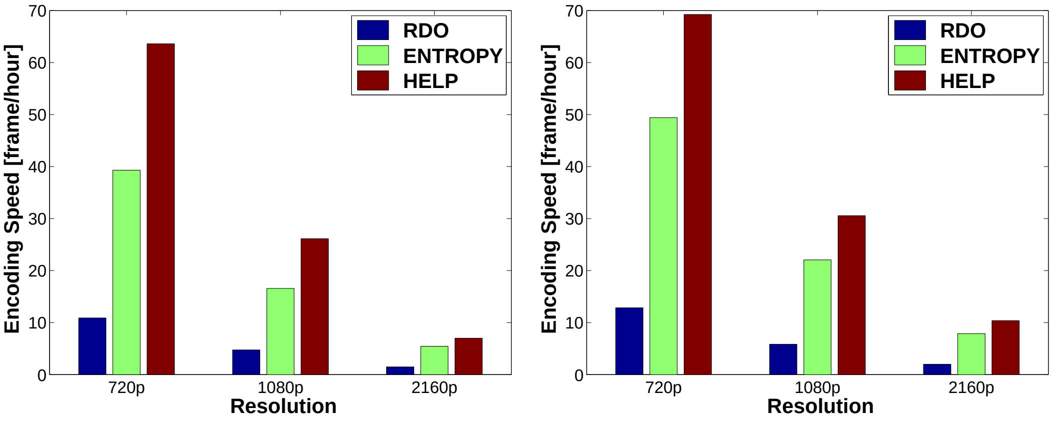
Table 5.2: Characteristics of the Used Standard Video Sequences

Sequence Name	Number of Frames	Resolution	QP	Description
HEVC Test Sequences				
crop_Traffic	150	2560 x 1600	22, 27, 32, 37, 42, 47	HEVC test sequence class A
Basket_ball_Drill	150	832 x 480	22, 27, 32, 37, 42, 47	HEVC test sequence class C
RaceHorses_832x480_30	97	416 x 240	32, 37, 42, 47	HEVC test sequence class C
Basket_ball_Pass	150	416 x 240	22, 27, 32, 37, 42, 47	HEVC test sequence class D
BlowingBubbles_416x240_50	97	416 x 240	32, 37, 42, 47	HEVC test sequence class D
Other Encoders Test Sequences				
Tennis	150	1920x1080	22, 27, 32, 37, 42, 47	Full HD
ducks_take_off_420_720p50	50	1280 x 720	32	HD
ducks_take_off_1080p50	50	1920 x 1080	32	Full HD
ducks_take_off_2160p50	6	3840 x 2160	32	Ultra HD
park_joy_off_420_720p50	50	1280 x 720	32	HD
park_joy_1080p50	50	1920 x 1080	32	Full HD
park_joy_2160p50	6	3840 x 2160	32	Ultra HD
720p50_mobcal_ter	60	1280 x 720	37	HD
720p50_parkrun_ter	60	1280 x 720	37	HD
elephants_dream_720p24	60	1280 x 720	37	HD
life_1080p30	60	1920 x 1080	37	Full HD
big_buck_bunny_1080p24	60	1920 x 1080	37	Full HD

5.4 *Result Presentation and Analysis*

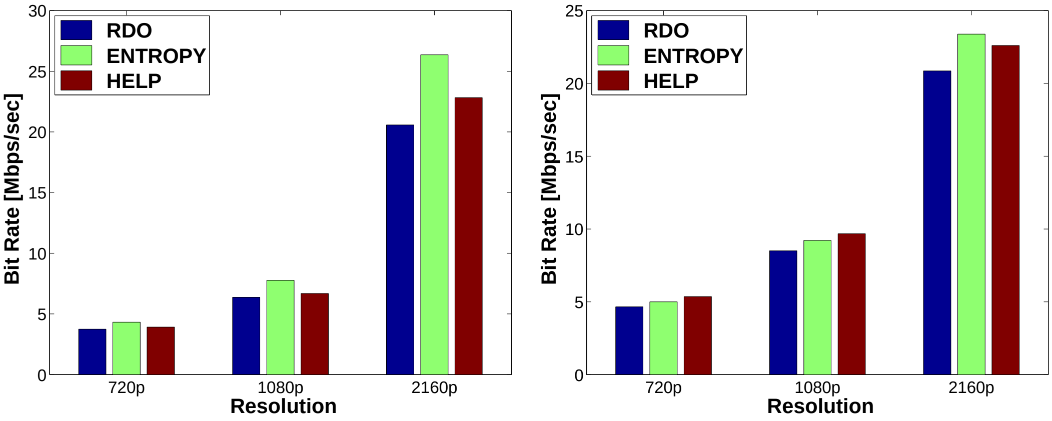
In the following results, we refer to entropy-based algorithm [25] as ENTROPY. Figures 5.5, 5.6, and 5.7 show the encoding speed, the bitrate, and the PSNR, respectively. The figures demonstrate the encoding performance of the RDO, ENTROPY, and HELP algorithms on sequences (Ducks_take_off and Park joy) each at 720p, 1080p, and 2160p resolution. The encoding speed enhancement ESE for the ENTROPY over RDO is 2.72 in average, while it is 4.31 in average for HELP over RDO (Table 5.4). ESE of 4.31 means 5.31 times faster. The figures show that there is no significant increase in bitrate of applying HELP algorithm and the decrease in PSNR is negligible. Both the coding efficiency and quality is better with HELP than ENTROPY.

Figure 5.8 shows the encoding speed for different quantization parameters, specifically at $QP =$



(a) Ducks_take_off (b) Park_joy

Figure 5.5: Comparing Encoding Speed vs. Resolution with Different Algorithms



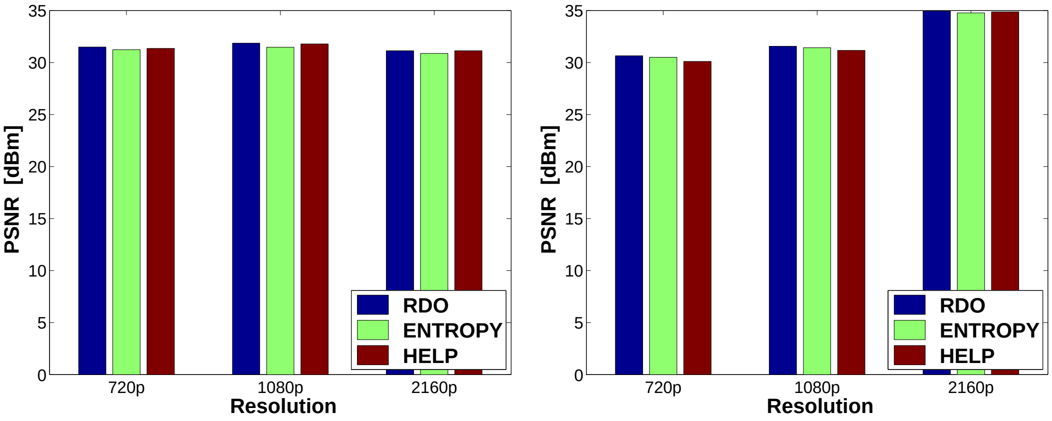
(a) Ducks_take_off (b) Park_joy

Figure 5.6: Comparing Bitrate vs. Resolution with Different Algorithms

$(32, 37, 42, 47)$. The figure demonstrates (based on Table 5.4) that in general HELP encoding speed is 5 times the encoding speed of RDO for all sequences at all QPs. The encoding speed for ENTROPY over RDO is 3.5. $ESEs$ are 4 and 2.5 for HELP and ENTROPY over RDO, respectively.

Figure 5.9 shows the bitrate versus quantization parameter at $QP = (32, 37, 42, 47)$. The figure demonstrates that HELP outperforms ENTROPY in coding efficiency.

Figure 5.10 shows the quality for different quantization parameters, specifically at $QP = (32, 37, 42, 47)$. We note that HELP algorithm outperforms ENTROPY in terms of quality in most of the sequences and



(a) Ducks_take_off (b) Park_joy

Figure 5.7: Comparing PSNR vs. Resolution with Different Algorithms

QPs in most of the sequences at most of the QPs.

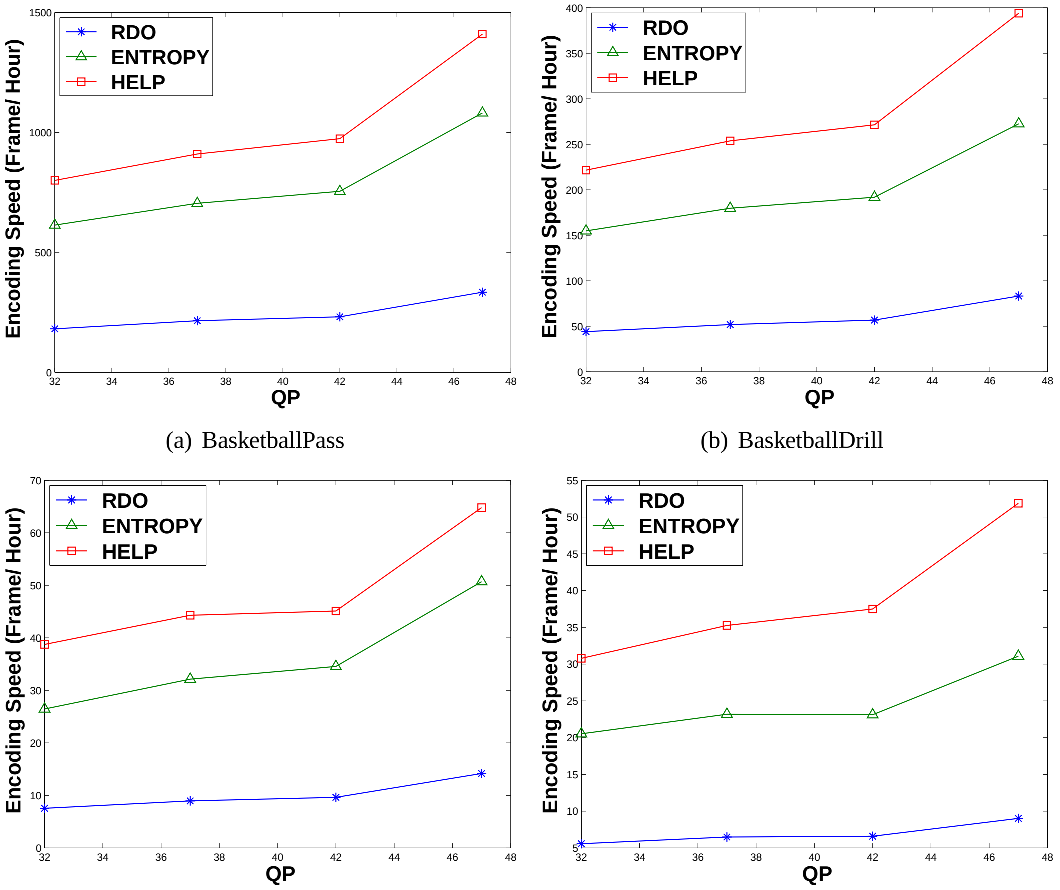
We compare various partitioning algorithms in terms of encoding speed for different values of QP. Figure 5.11 shows the encoding speed versus bitrate at $QP = (32, 37, 42, 47)$. The figure demonstrates that the encoding speed of HELP is around 1.5 in average faster than ENTROPY.

Figure 5.12 shows the quality versus bitrate for 4 sequences at $QP = (32, 37, 42, 47)$. In general in all the sequences at most of the QPs HELP outperforms ENTROPY in terms of quality.

Figure 5.13 shows Y-PSNR, U-PSNR, V-PSNR, and *Bjontegaard Delta* (BD-PSNR) of ENTROPY to RDO, HELP to RDO, and HELP to ENTROPY for Tennis sequence at $QP = (32, 37, 42, 47)$. The figure shows clearly how HELP maintains a close quality and bitrate to RDO. The figure also demonstrates how HELP outperforms ENTROPY in both quality and coding efficiency.

In Figure 5.14, we compare the encoding speed of HELP, ENTROPY, and RDO algorithms on three different computers for of Tennis sequence. The figure demonstrates that HELP algorithm outperforms ENTROPY on the three computers in terms of encoding speed. The bitrate and the quality have the same values on each of the three computers, therefore we do not show them.

Based on Table 5.4, we notice that HELP's *Encoding Speed Enhancement* (ESE) is 4, which means



(a) BasketballPass

(b) BasketballDrill

(c) Tennis

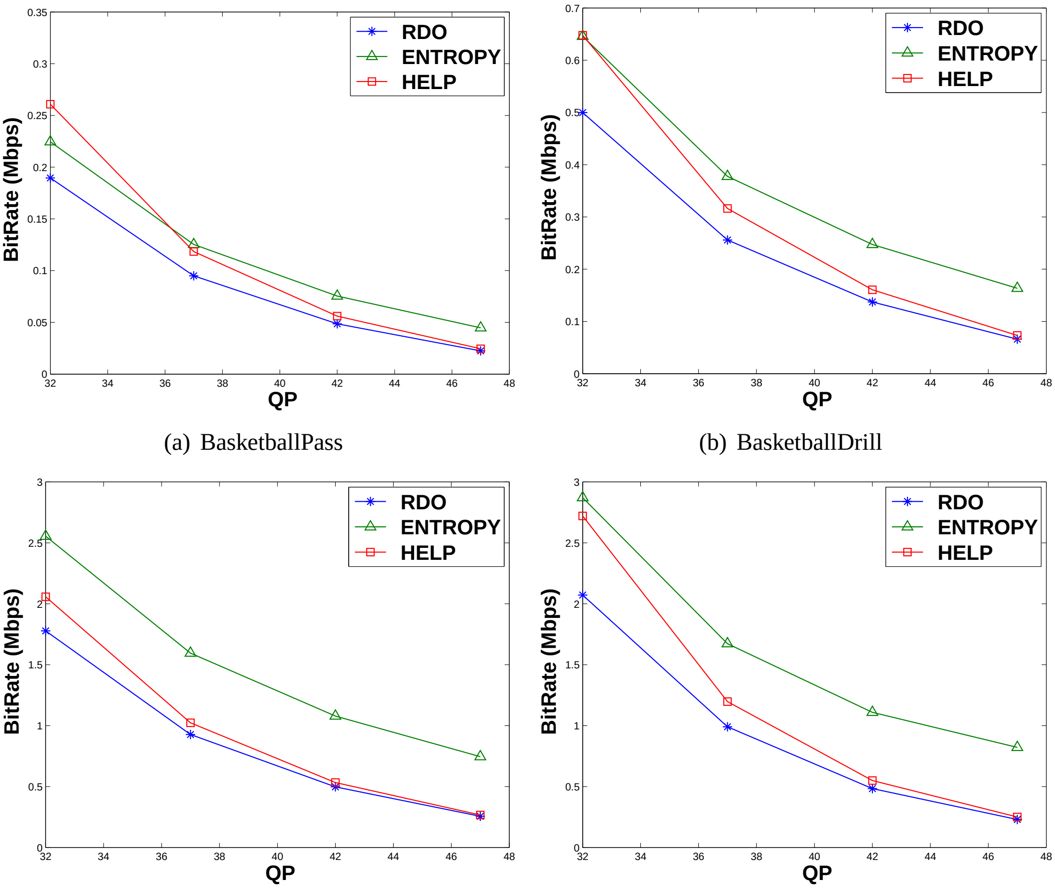
(d) Traffic

Figure 5.8: Comparing Encoding Speed vs. QP

HELP encoding is 5 times faster than RDO algorithm in average, while ENTROPY's ESE is about 2.5 over RDO. We calculate the averages of ESE based on 46 different sequences and QPs combination.

The average of *Bjntegaard Delta-PSNR* (BD-PSNR) is 1.416 more for HELP than ENTROPY and the average *Bjntegaard Delta-rate* (BD-rate) is 34.25 less for HELP than ENTROPY. BD averages are shown in Table 5.3. Encoding speed enhancement is calculated based on Equation 5.3.

In Table 5.5, we compare HELP algorithm with TnB and RDO. The encoding speeds are 25.64, 50.01, and 125.21, for RDO, TnB, and HELP, respectively. These encoding speeds mean encoding speed enhancement ESE of 1.10 and 4.14 for TnB and HELP, respectively. In terms of encoding time reduction, we notice -0.52 and -0.80, for TnB and HELP, respectively. There is a little increase in



(a) BasketballPass (b) BasketballDrill

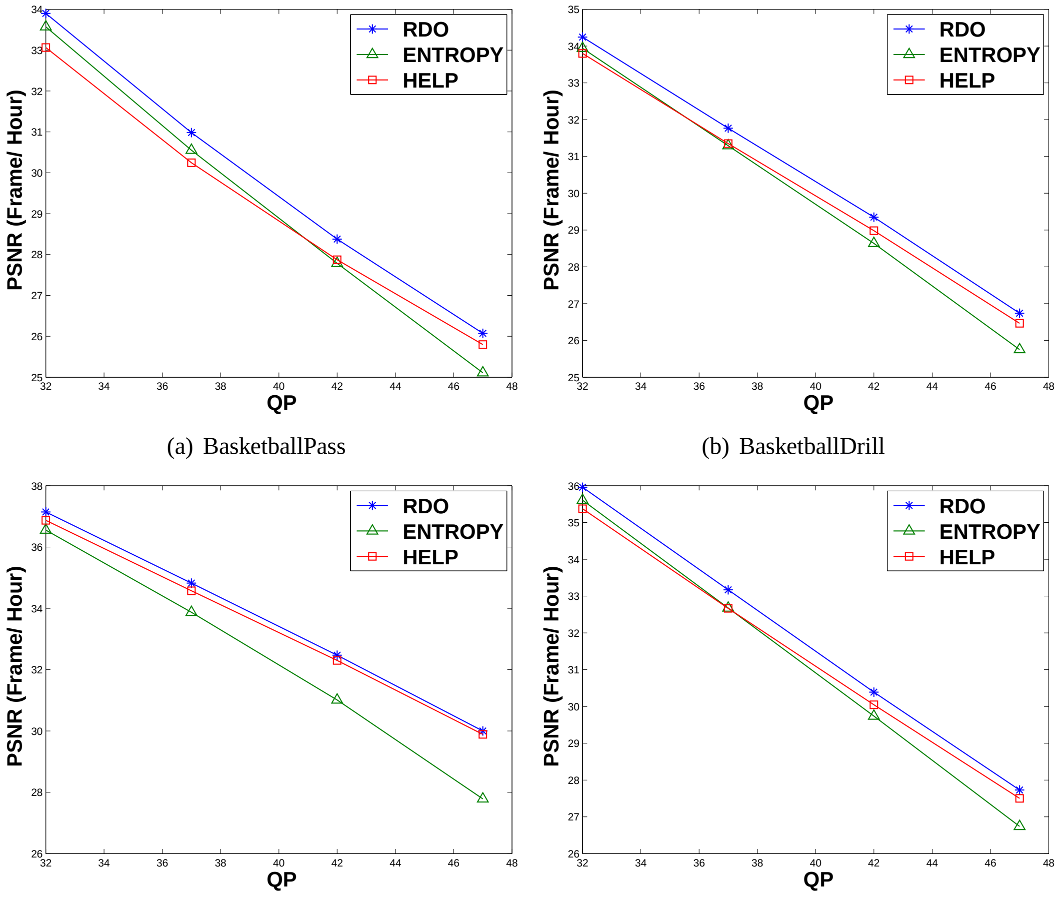
(c) Tennis (d) Traffic

Figure 5.9: Comparing Bitrate vs. QP with Different Algorithms

bitrate and an unnoticeable decrease in quality. The encoding speed is 5 and 2 times that of the RDO for HELP and TnB, respectively.

Figures 5.15, 5.16, and 5.17 show the encoding speed, the bitrate, and the PSNR versus quantization parameter at $QP = (32, 37, 42, 47)$, respectively. Figure 5.15 demonstrates (based on Table 5.5) that in general HELP encoding speed is 2.5 times the encoding speed of TnB for all sequences at all QPs. The difference in PSNR is not noticeable neither for machines in computer vision systems nor for a human eye. The difference in bitrate does not worth the big difference in encoding speed, especially in real-time applications.

In terms of encoding time reduction, Study [39] demonstrated that Hybrid-2 performs better than



(a) BasketballPass

(b) BasketballDrill

(c) Tennis

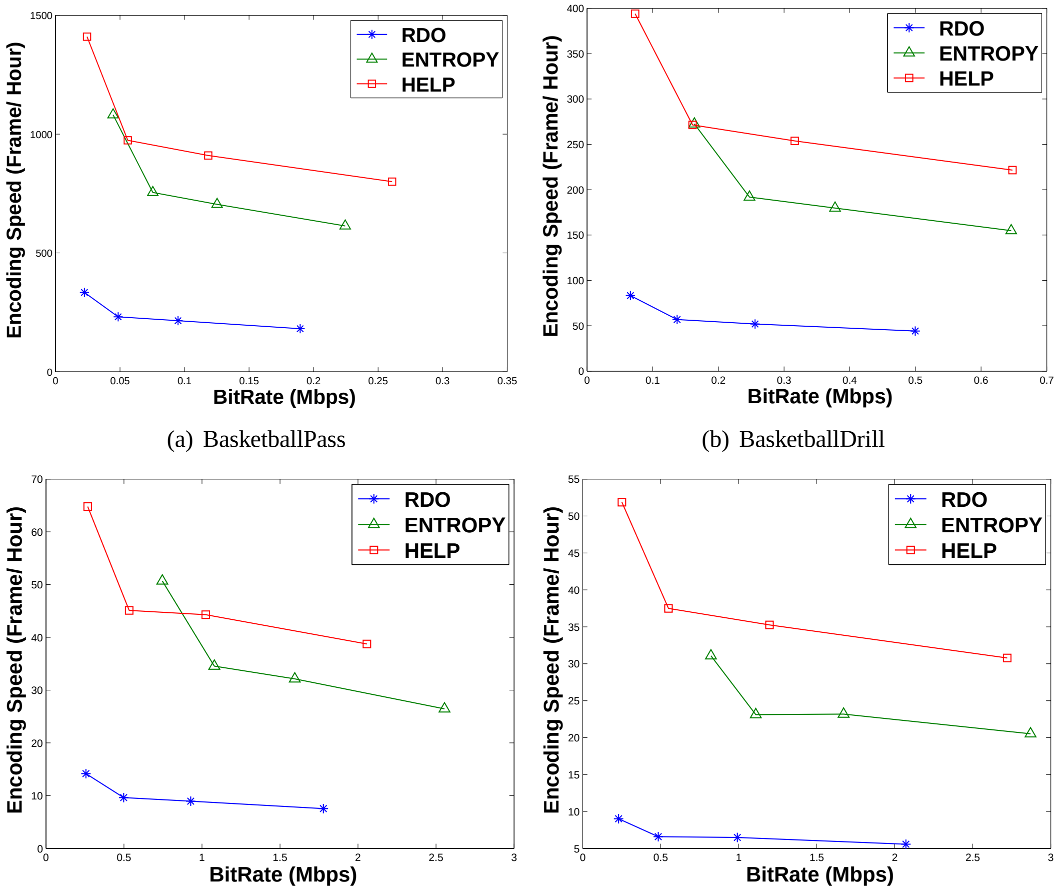
(d) Traffic

Figure 5.10: Comparing PSNR vs. QP with Different Algorithms

all the previous approaches that the author aware off. In Table 5.6, we compare HELP algorithm with Hybrid-2 and RDO. The encoding speeds are 8.59, 25.54, and 47.05, for RDO, Hybrid2, and HELP, respectively. These encoding speeds mean encoding speed enhancement ESE of 1.85 and 4.27 for Hybrid2 and HELP, respectively. In terms of encoding time reduction, the performance are -0.68 and -0.81, for Hybrid2 and HELP, respectively.

5.5 Conclusions

In this chapter, we have proposed a new algorithm to partition LCU in HEVC. The proposed algorithm highly enhances the encoding speed with an acceptable degradation in coding efficiency and quality. Based on the results, the proposed algorithm leads to encoding speed of 5 times that of RDO



(a) BasketballPass

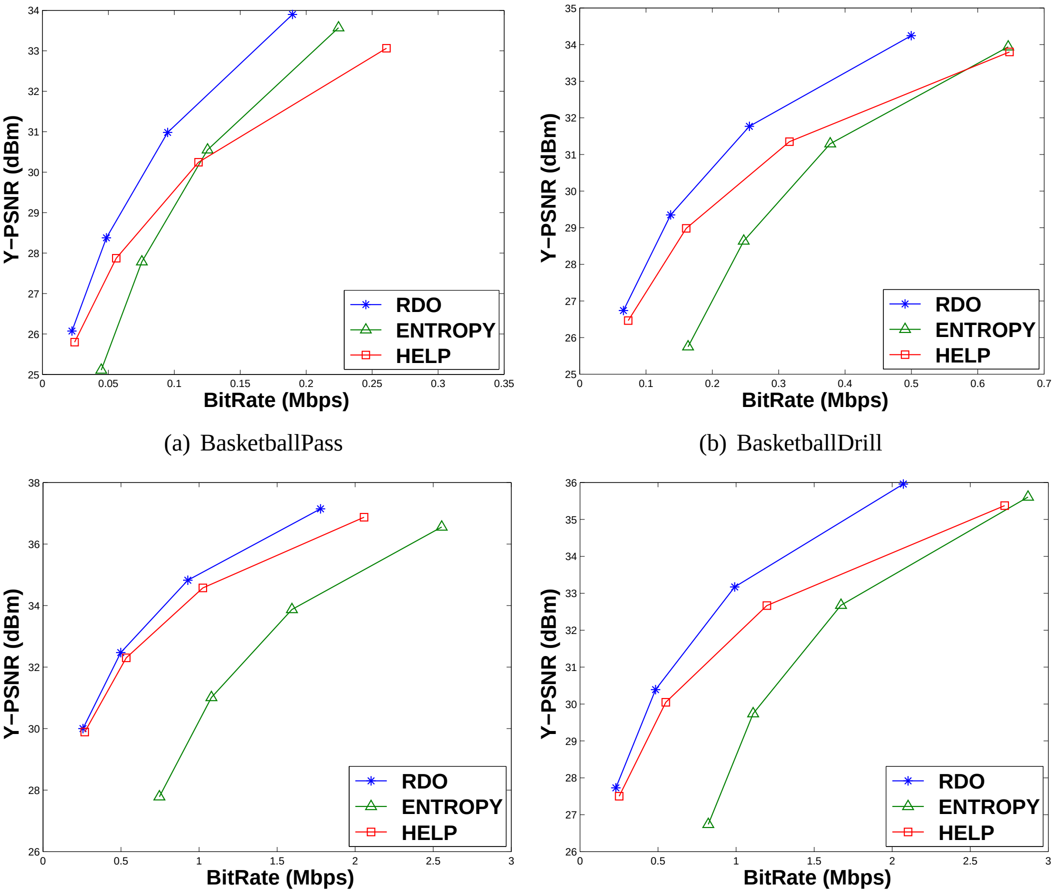
(b) BasketballDrill

(c) Tennis

(d) Traffic

Figure 5.11: Comparing Enc. Speed vs. Bitrate at QP = 32, 37, 42, 47

(4 times faster), with an acceptable decrease in quality. Therefore, HELP algorithm is a technique that worth further investigation. In our knowledge, no other study approach as this performance.



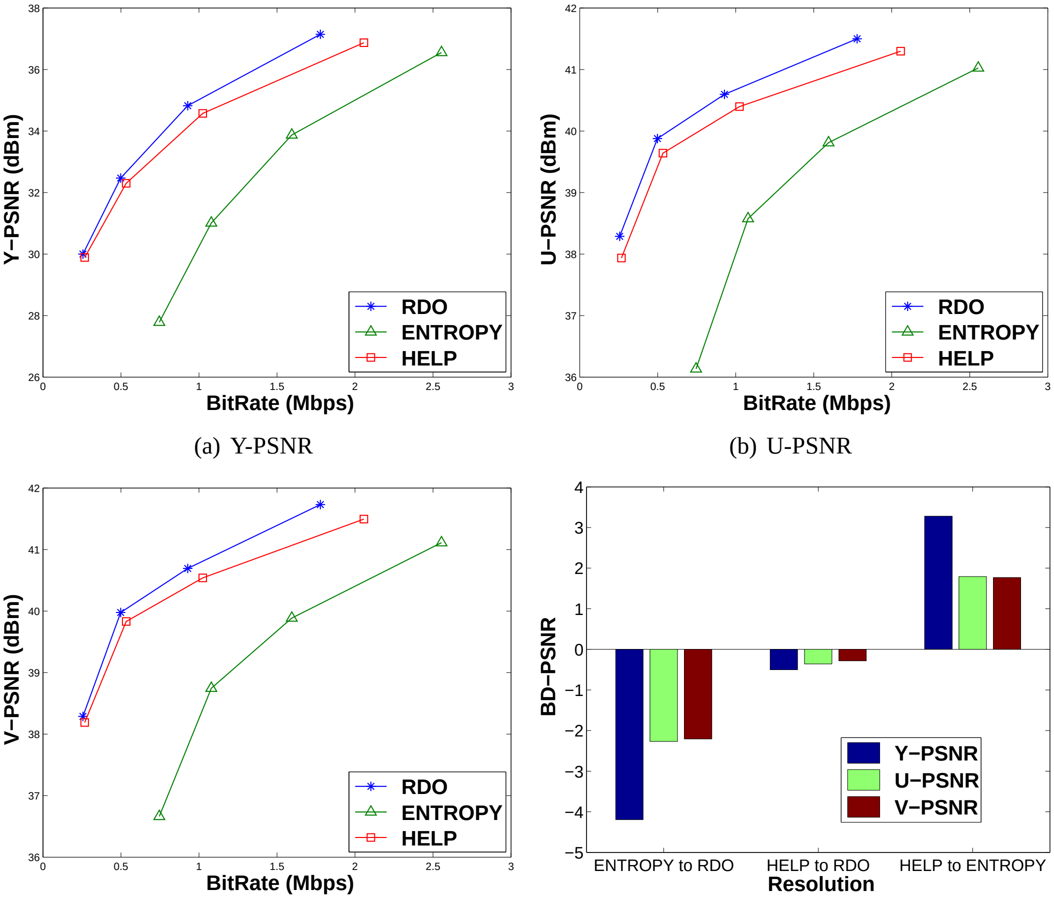
(a) BasketballPass

(b) BasketballDrill

(c) Tennis

(d) Traffic

Figure 5.12: Comparing PSNR vs. Bitrate with Different Algorithms at QP = 32, 37, 42, 47



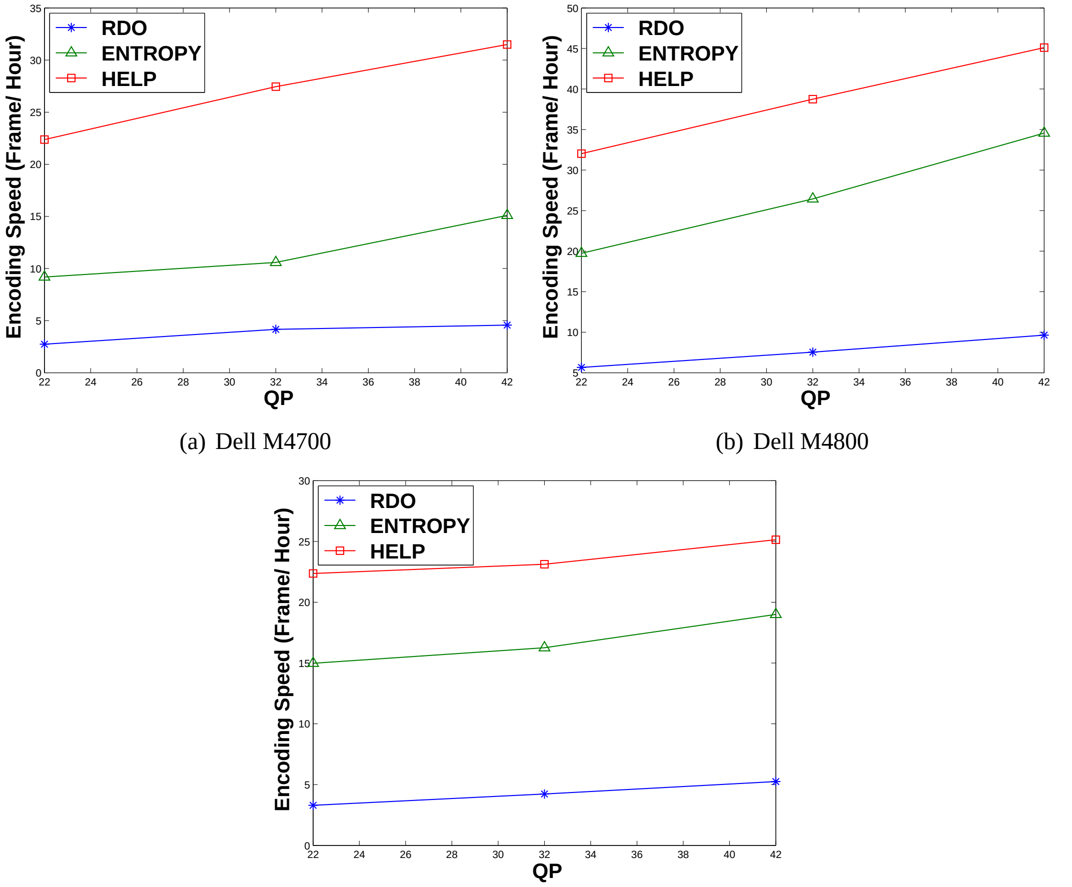
(a) Y-PSNR

(b) U-PSNR

(c) V-PSNR

(d) Bjontegaard Delta (BD-PSNR)

Figure 5.13: Comparing YUV-PSNR vs. Bitrate at QP = 32, 37, 42, 47



(a) Dell M4700

(b) Dell M4800

(c) HP EliteBook

Figure 5.14: Comparing Encoding Speed vs. QP on Three different Computers, QP = 22, 32, 42

Table 5.3: Comparing the performance of HELP, ENTROPY, and RDO [QP = (32,37,42,47)]

Comparing the performance of HELP compared to ENTROPY				
Sequence	BD-PSNR	BD-RATE	$ESE_{ENTROPY}$	ESE_{HELP}
crop_Traffic	1.744166	-42.287112	2.64	4.59
Basket_ball_Drill	1.223721	-30.498920	2.44	3.98
Basket_ball_Pass	0.414357	-13.296025	2.32	3.34
Tennis	3.279842	-52.946821	2.55	4.07
Average	1.416	-34.25	2.5	4.0

BD-PSNR and BD-Rate of HELP and ENTROPY compared to RDO				
	BD-PSNR		BD-Rate	
Sequence	(ENTROPY to RDO)	(HELP to RDO)	(ENTROPY to RDO)	(HELP to RDO)
crop_Traffic	-3.432821	-0.978797	122.299820	32.377698
Basket_ball_Drill	-2.690911	-1.019517	87.363419	33.589731
Basket_ball_Pass	-2.013280	-1.206478	60.399920	44.395363
Tennis	-4.194577	-0.502971	142.033234	15.284985
Average	-3.09	-0.92	102.94	31.4

Table 5.4: Results of the Proposed HELP Algorithm Compared to ENTROPY and RDO.

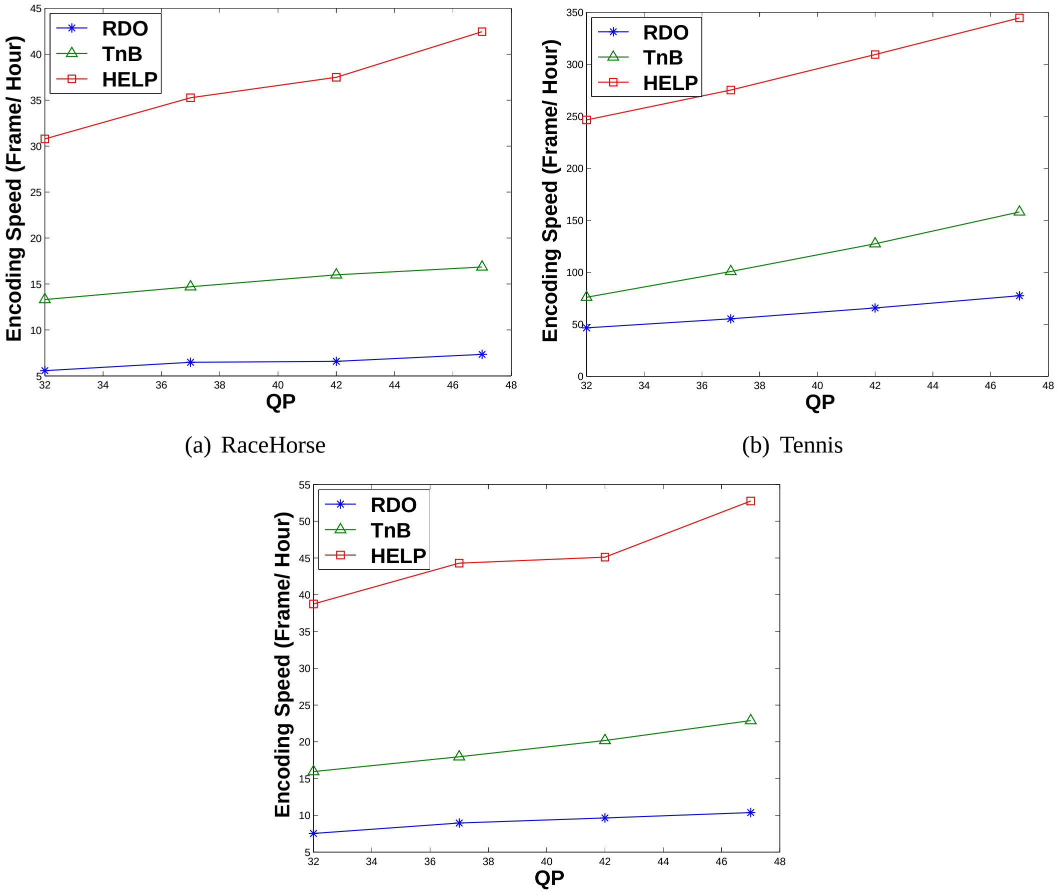
		Enc. Time (sec)			Bitrate			Y-PSNR			Enc. Speed			Enc. Speed. Enh. (ESE)	
Sequence	QP	RDO	Entropy	HELP	RDO	Entropy	HELP	RDO	Entropy	HELP	RDO	Entropy	HELP	Entropy	HELP
Tennis	22	95372.141	27351.042	16856.919	8190.7104	9638.992	9940.888	41.4959	41.3729	41.3502	5.66	19.74	32.03	2.49	4.66
Tennis	27	80365.181	22452.981	14696.994	3707.112	4711.03	4470.504	39.4665	39.1724	39.2469	6.72	24.05	36.74	2.58	4.47
Tennis	32	71610.691	20415.191	13936.386	1777.9552	2554.211	2057.2112	37.1438	36.5523	36.872	7.54	26.45	38.75	2.51	4.14
Tennis	37	60299.066	16806.003	12192.248	927.9216	1594.982	1023.8992	34.8245	33.8703	34.5745	8.96	32.13	44.29	2.59	3.95
Tennis	42	55990.459	15626.327	11970.469	498.2128	1078.824	533.7456	32.4712	31.0107	32.2998	9.64	34.56	45.11	2.58	3.68
Tennis	47	58129.505	13094.148	10239.249	256.336	745.9488	267.2448	30.0009	27.7847	29.8909	9.29	41.24	52.74	3.44	4.68
ducks_take_off_420_720p50	32	12212.131	3390.473	2083.14	3747.768	4320.389	3916.7424	31.4885	31.2376	31.3579	14.74	53.09	86.41	2.60	4.86
ducks_take_off_1080p50	32	27995.864	7957.518	5086.473	6373.6704	7772.87	6686.2752	31.8648	31.4741	31.7869	6.43	22.62	35.39	2.52	4.50
ducks_take_off_2160p50	32	10602.459	2900.2002	2252.86687	20573.92	26351.64	22821.56	31.1385	30.8749	31.1333	2.04	7.45	9.59	2.66	3.71
park_joy_420_720p50	32	10226.912	2659.2564	1897.73793	4653.9936	4996.147	5355.1344	30.6486	30.5038	30.1065	17.60	67.69	94.85	2.85	4.39
park_joy_1080p50	32	22499.295	5956.4766	4300.2694	8502.7824	9213.998	9678.0672	31.5578	31.4219	31.1661	8.00	30.22	41.86	2.78	4.23
park_joy_2160p50	32	7877.3388	2004.5917	1519.34389	20855.24	23379.2	22598.44	34.9601	34.7709	34.8733	2.74	10.78	14.22	2.93	4.18
720p50 mobcal ter	37	7225.816	2083.384	1292.556	228.66	388.612	223.572	28.7321	28.543	28.5329	24.91	86.40	139.26	2.47	4.59
720p50_parkrun_ter	37	9988.401	2775.666	1797.232	2129.42	2341.968	2212.58	26.2494	26.0697	25.96	2.16	7.78	12.02	2.60	4.56
elephants_dream_720p24	37	7445.16	1758.451	1574.932	65.2128	120.3904	66.2176	50.4844	49.9978	50.4283	2.90	12.28	13.71	3.23	3.73
sintel_trailer_2k 1080p24	37	9291.08	1432.145	1427.871	302.732	315.968	314.744	58.7828	57.6284	57.6474	2.32	15.08	15.13	5.49	5.51
life 1080p30	37	19241.729	5334.227	3113.407	6293.164	10364.32	6431.144	28.8134	28.3042	28.6723	9.35	33.74	57.81	2.61	5.18
big_buck_bunny_1080p24	37	18213.203	3926.371	3747.703	195.8656	243.5072	193.6032	44.0719	43.7404	44.021	9.88	45.84	48.03	3.64	3.86
BasketBallPass	22	3957.883	1164.832	872.301	786.688	855.0528	1206.0528	41.0869	40.9207	40.5263	136.44	463.59	619.05	2.40	3.54
BasketBallPass	27	3205.036	956.03	719.753	390.1824	436.0752	583.3552	37.3088	37.0794	36.5698	168.48	564.84	750.26	2.35	3.45
BasketBallPass	32	2979.208	879.956	675.043	189.6192	224.4992	260.8464	33.8989	33.5703	33.0646	181.26	613.67	799.95	2.39	3.41
BasketBallPass	37	2511.538	766.35	593.253	94.976	125.264	118.384	30.982	30.5519	30.2448	215.01	704.64	910.24	2.28	3.23
BasketBallPass	42	2334.305	715.516	554.309	48.5856	75.4288	56.0512	28.3772	27.7852	27.8724	231.33	754.70	974.19	2.26	3.21
BasketBallPass	47	1943.857	615.452	471.858	22.4192	44.6912	24.4768	26.0727	25.107	25.7986	277.80	877.40	1144.41	2.16	3.12
BlowingBubbles_416x240_50	32	2067.557	624.844	500.652	375.699	450.334	453.1876	31.5452	31.3166	30.8273	87.06	288.07	359.53	2.31	3.13
BlowingBubbles_416x240_50	37	1761.922	553.286	426.442	169.3773	229.0186	188.1361	28.7605	28.348	28.274	102.16	325.33	422.10	2.18	3.13
BlowingBubbles_416x240_50	42	1526.887	512.196	376.116	76.9526	122.8247	83.2495	26.2705	25.7176	26.0033	117.89	351.43	478.58	1.98	3.06
BlowingBubbles_416x240_50	47	1361.649	444.663	340.33	33.4268	66.2268	36.6351	24.1875	23.406	24.0144	132.19	404.80	528.90	2.06	3.00
RaceHorse_832x480_30	32	11586.031	3288.439	2190.4	1244.207	1463.866	1693.2	32.0739	31.8351	31.2822	15.54	54.74	82.18	2.52	4.29
RaceHorse_832x480_30	37	9758.988	2756.51	1962.031	557.8713	746.3332	685.5513	29.0911	28.7356	28.353	18.44	65.30	91.74	2.54	3.97
RaceHorse_832x480_30	42	8215.553	2334.622	1745.518	242.4866	411.8375	276.2004	26.7241	26.152	26.2637	21.91	77.10	103.12	2.52	3.71
RaceHorse_832x480_30	47	6967.909	2035.242	1567.1	108.3142	255.021	118.174	25.0192	23.8818	24.7504	25.83	88.44	114.86	2.42	3.45
BasketballDrill	22	17037.458	4845.346	3189.389	2237.704	2552.163	3078.3328	40.318	39.8298	39.8298	31.69	111.45	169.31	2.52	4.34
BasketballDrill	27	13617.474	3861.338	2663.73	1051.2544	1248.694	1410.0896	37.1319	36.8684	36.6297	39.65	139.85	202.72	2.53	4.11
BasketballDrill	32	12236.089	3488.113	2437.079	499.7248	645.8128	647.7488	34.2431	33.9392	33.7991	44.13	154.81	221.58	2.51	4.02
BasketballDrill	37	10394.483	3004.389	2127.887	255.8688	377.5888	316.1712	31.7678	31.2921	31.3494	51.95	179.74	253.77	2.46	3.88
BasketballDrill	42	9504.908	2814.418	1989.936	137.224	247.28	160.7024	29.3486	28.6372	28.9833	56.81	191.87	271.37	2.38	3.78
BasketballDrill	47	7992.393	2432.809	1690.403	66.0912	163.3712	73.1968	26.7375	25.7502	26.464	67.56	221.97	319.45	2.29	3.73
Traffic	22	136129.39	35495.018	24332.247	13839.8416	15471.31	17832.435	41.6225	41.4666	41.0539	3.97	15.21	22.19	2.84	4.59
Traffic	27	105110.13	27945.961	19011.845	4845.904	5915.118	6553.4832	38.7311	38.5042	38.164	5.14	19.32	28.40	2.76	4.53
Traffic	32	97001.769	26316.265	17544.513	2071.6208	2870.504	2720.5008	35.9579	35.605	35.3743	5.57	20.52	30.78	2.69	4.53
Traffic	37	83346.158	23296.681	15315.877	991.0496	1672.301	1197.1808	33.1703	32.6754	32.6663	6.48	23.18	35.26	2.58	4.44
Traffic	42	81936.19	23363.896	14404.968	484.2352	1109.446	549.784	30.3891	29.7383	30.0456	6.59	23.11	37.49	2.51	4.69
Traffic	47	73550.12	21370.914	12719.857	230.128	822.1568	251.2272	27.7297	26.7386	27.5009	7.34	25.27	42.45	2.44	4.78
		Average													
		Enc. Time			Bitrate			Y-PSNR			Enc. Speed			Enc. Speed Enh.	
		RDO	Entropy	HELP	RDO	Entropy	HELP	RDO	Entropy	HELP	RDO	Entropy	HELP	Entropy	HELP
Avg. All Median. All		29377.76 10498.47	8041.08 2952.29	5463.83 2105.51	2734.82 498.97	3380.35 838.60	3167.41 615.55	33.47 31.66	32.95 31.30	33.08 31.32	50.21 15.14	166.62 53.91	223.45 84.29	2.62 2.52	4.05 4.07
			Enc. Time Red. (ETR)												
Avg. Median			-0.73 -0.72	-0.81 -0.80											

Table 5.5: Results of the Proposed HELP Algorithm Compared to TnB and RDO.

		Enc. Time (sec)			Bitrate			Y-PSNR			Enc. Speed			Enc. Speed. Enh. (ESE)		Enc. Time Red.	
Sequence	QP	RDO	TnB	HELP	RDO	TnB	HELP	RDO	TnB	HELP	RDO	TnB	HELP	TnB	HELP	TnB	HELP
Tennis	32	71610.691	33865.491	13936.386	1777.9552	1821.579	2057.2112	37.1438	37.0999	36.872	7.54	15.95	38.75	1.11	4.14	-0.53	-0.81
Tennis	37	60299.066	30049.318	12192.248	927.9216	936.5648	1023.8992	34.8245	34.7861	34.5745	8.96	17.97	44.29	1.01	3.95	-0.50	-0.80
Tennis	42	55990.459	26754.249	11970.469	499.6656	504.088	533.7456	32.4712	32.4548	32.2998	9.64	20.18	45.11	1.09	3.68	-0.52	-0.79
Tennis	47	52002.561	23592.374	10239.249	256.336	259.8496	267.2448	30.0009	29.9609	29.8909	10.38	22.89	52.74	1.20	4.08	-0.55	-0.80
													Avg.	1.10	3.96	-0.52	-0.80
Traffic	32	97001.769	40578.323	17544.513	2071.6208	2186.275	2720.5008	35.9579	35.8088	35.3743	5.57	13.31	30.78	1.39	4.53	-0.58	-0.82
Traffic	37	83346.158	36732.308	15315.877	991.0496	1016.787	1197.1808	33.1703	33.0679	32.6663	6.48	14.70	35.26	1.27	4.44	-0.56	-0.82
Traffic	42	81936.19	33750.004	14404.968	484.2352	491.3152	549.784	30.3891	30.3443	30.0456	6.59	16.00	37.49	1.43	4.69	-0.59	-0.82
Traffic	47	73550.12	32055.544	12719.857	230.128	229.072	251.2272	27.7297	27.7044	27.5009	7.34	16.85	42.45	1.29	4.78	-0.56	-0.83
													Avg.	1.35	4.61	-0.57	-0.82
ducks_take_off_420_720p50	32	12212.131	5621.158	2083.14	3747.768	3789.293	3916.7424	31.4885	31.4616	31.3579	14.74	32.02	86.41	1.17	4.86	-0.54	-0.83
ducks_take_off_1080p50	32	27995.864	12586.495	5086.473	6373.6704	6435.437	6686.2752	31.8648	31.8535	31.7869	6.43	14.30	35.39	1.22	4.50	-0.55	-0.82
ducks_take_off_2160p50	32	10602.459	4722.371	2252.86687	20573.92	20601.48	22821.56	31.1385	31.1361	31.1333	2.04	4.57	9.59	1.25	3.71	-0.55	-0.79
													Avg.	1.21	4.36	-0.55	-0.81
park_joy_420_720p50	32	10226.912	6966.012	1897.73793	4653.9936	4719.067	5355.1344	30.6486	30.5916	30.1065	17.60	25.84	94.85	0.47	4.39	-0.32	-0.81
park_joy_1080p50	32	22499.295	14205.014	4300.2694	8502.7824	8635.56	9678.0672	31.5578	31.5093	31.1661	8.00	12.67	41.86	0.58	4.23	-0.37	-0.81
park_joy_2160p50	32	7877.3388	3373.226	1519.34389	20855.24	21096.92	22598.44	34.9601	34.9499	34.8733	2.74	6.40	14.22	1.34	4.18	-0.57	-0.81
													Avg.	0.80	4.27	-0.42	-0.81
RaceHorse_832x480_30	32	11586.031	7101.616	2190.4	1244.207	1311.756	1693.2	32.0739	31.8869	31.2822	15.54	25.35	82.18	0.63	4.29	-0.39	-0.81
RaceHorse_832x480_30	37	9758.988	5359.717	1962.031	557.8713	571.9522	685.5513	29.0911	28.903	28.353	18.44	33.58	91.74	0.82	3.97	-0.45	-0.80
RaceHorse_832x480_30	42	8215.553	4236.64	1745.518	242.4866	243.2313	276.2004	26.7241	26.613	26.2637	21.91	42.49	103.12	0.94	3.71	-0.48	-0.79
RaceHorse_832x480_30	47	6967.909	3417.015	1567.1	108.3142	107.8565	118.174	25.0192	24.9678	24.7504	25.83	52.68	114.86	1.04	3.45	-0.51	-0.78
													Avg.	0.86	3.85	-0.46	-0.79
BlowingBubbles_416x240_50	32	2067.557	1301.48	500.652	375.699	386.1361	453.1876	31.5452	31.4213	30.8273	87.06	138.30	359.53	0.59	3.13	-0.37	-0.76
BlowingBubbles_416x240_50	37	1761.922	965.002	426.442	169.3773	171.7031	188.1361	28.7605	28.6442	28.274	102.16	186.53	422.10	0.83	3.13	-0.45	-0.76
BlowingBubbles_416x240_50	42	1526.887	773.309	376.116	76.9526	77.7979	83.2495	26.2705	26.2226	26.0033	117.89	232.77	478.58	0.97	3.06	-0.49	-0.75
BlowingBubbles_416x240_50	47	1361.649	662.986	340.33	33.4268	33.7278	36.6351	24.1875	24.1632	24.0144	132.19	271.50	528.90	1.05	3.00	-0.51	-0.75
													Avg.	0.86	3.08	-0.46	-0.75
		Average															
		Enc. Time			Bitrate			Y-PSNR			Enc. Speed			Enc. Speed Enh.		Enc. Time Red.	
		RDO	TnB	HELP	RDO	TnB	HELP	RDO	TnB	HELP	RDO	TnB	HELP	TnB	HELP	TnB	HELP
Avg. All		32290.80	14939.53	6116.91	3397.94	3437.61	3781.42	30.77	30.71	30.43	28.87	55.31	126.83	1.04	4.01	-0.50	-0.80

Table 5.6: Results for the Proposed Algorithm Compared to Hybrid2 and RDO.

		Enc. Time (sec)			Bitrate			Y-PSNR			Enc. Speed			Enc. Speed. Enh. (ESE)		Enc. Time Red. %	
Sequence	QP	RDO	Hybrid2	HELP	RDO	Hybrid2	HELP	RDO	Hybrid2	HELP	RDO	Hybrid2	HELP	Entropy	HELP	Hybrid2	HELP
ducks_take_off_420_720p50	32	12212.131	3493.202	2083.14	3747.768	3794.846	3916.7424	31.4885	31.452	31.3579	14.73944228	51.5286548	86.40801866	2.50	4.86	-0.71	-0.83
ducks_take_off_1080p50	32	27995.864	7535.176	5086.473	6373.6704	6440.4	6686.2752	31.8648	31.8481	31.7869	6.429521161	23.88796227	35.38797906	2.72	4.50	-0.73	-0.82
ducks_take_off_2160p50	32	10602.459	2995.439	2252.86687	20573.92	20635.88	22821.56	31.1385	31.1338	31.1333	2.037263164	7.210963068	9.587783587	2.54	3.71	-0.72	-0.79
													Avg.	2.58	4.36	-0.72	-0.81
park_joy_420_720p50	32	10226.912	4508.437	1897.73793	4653.9936	4752.912	5355.1344	30.6486	30.5368	30.1065	17.60062141	39.92514479	94.84976674	1.27	4.39	-0.56	-0.81
park_joy_1080p50	32	22499.295	8824.92	4300.2694	8502.7824	8703.859	9678.0672	31.5578	31.4664	31.1661	8.000250689	20.39678547	41.85784267	1.55	4.23	-0.61	-0.81
park_joy_2160p50	32	7877.3388	2105.021	1519.34389	20855.24	21167.92	22598.44	34.9601	34.9432	34.8733	2.7420428	10.26118029	14.21666296	2.74	4.18	-0.73	-0.81
													Avg.	1.85	4.27	-0.63	-0.81
		Average															
		Enc. Time			Bitrate			Y-PSNR			Enc. Speed			Enc. Speed Enh.		Enc. Time Red.	
Avg. All		15235.67	4910.37	2856.64	10784.5624	10915.97	11842.703	31.94305	31.89672	31.737333	8.59	25.54	47.05	2.22	4.31	-0.68	-0.81

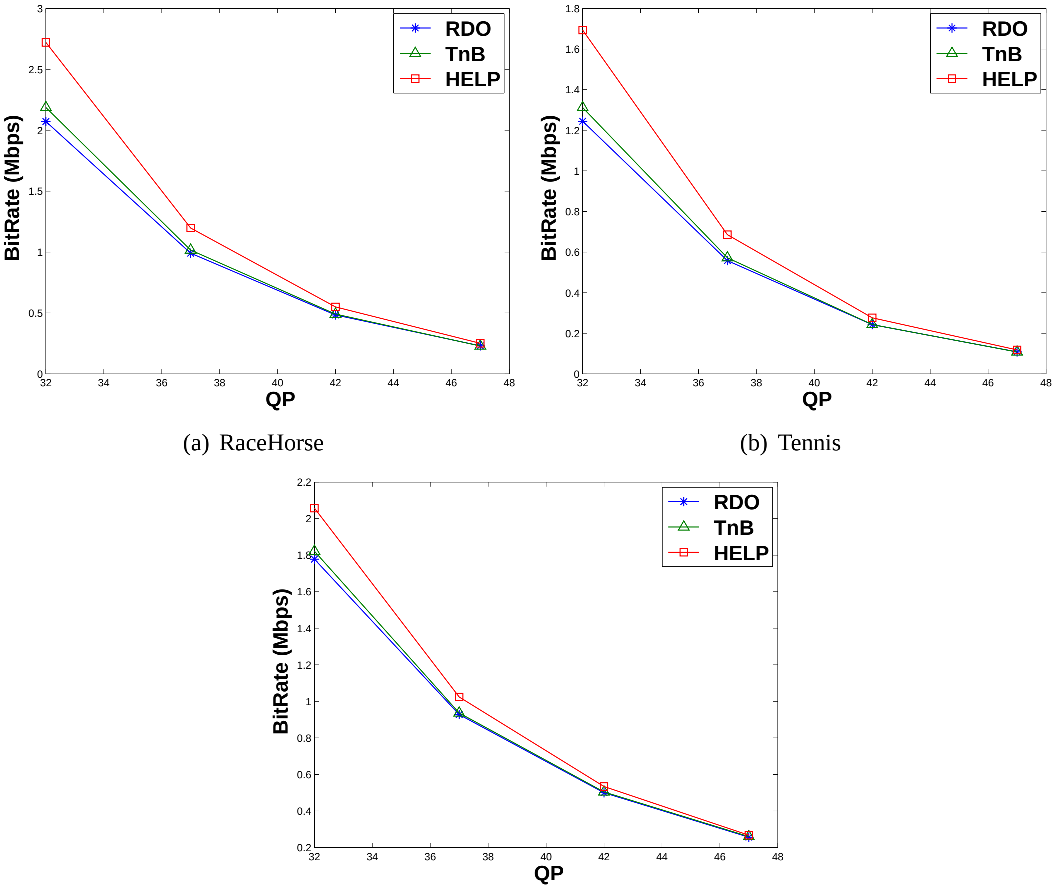


(a) RaceHorse

(b) Tennis

(c) Traffic

Figure 5.15: Comparing Encoding Speed vs. QP [RDO, TnB, and HELP]

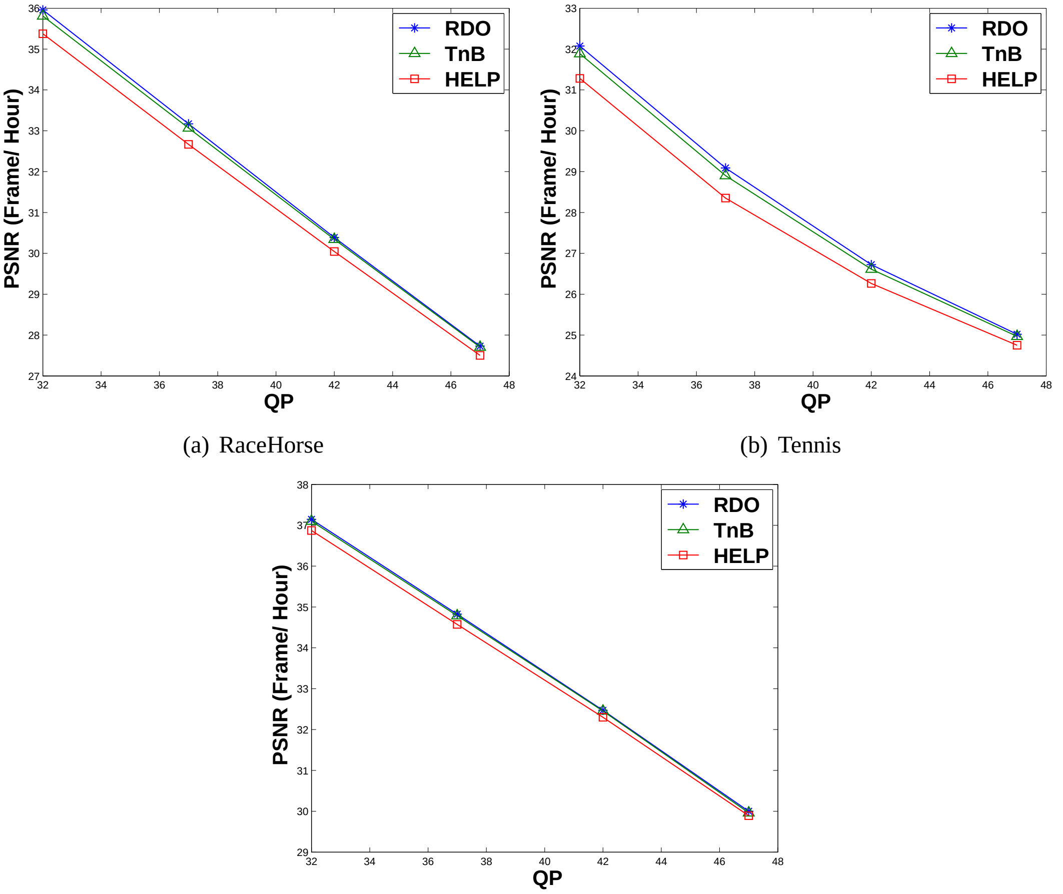


(a) RaceHorse

(b) Tennis

(c) Traffic

Figure 5.16: Comparing Bitrate vs. QP with Different Algorithms [RDO, TnB, and HELP]



(a) RaceHorse

(b) Tennis

(c) Traffic

Figure 5.17: Comparing PSNR vs. QP with Different Algorithms [RDO, TnB, and HELP]

CHAPTER 6 SUMMARY AND FUTURE WORK

6.1 Summary

This dissertation considers the design of real-time CV systems with live video streaming, especially those over wireless and mobile networks. Such systems include video cameras/sensors and monitoring stations. The cameras should adapt their captured videos based on the events and/or available resources. The monitoring station receives video streams from all cameras and runs CV algorithms for decisions, warnings, control, and/or other actions. Real-time CV systems have constraints in power, computational, and communicational resources. The metric for the performance of CV systems is the accuracy of the system in perceiving or extracting descriptions of physical objects or events from pictures (i.e. detection, recognition, and tracking accuracy of objects and events). We have analyzed and compared the rate-accuracy and rate-energy characteristics of various video rate adaptation techniques in computer vision applications. In addition, we have studied the impacts of different adaptation combinations. Furthermore, we have presented an objective function that provides any desired tradeoff in terms of accuracy, bitrate, and energy consumption. The reported results are based on realistic experiments considering both H.264 and MPEG-4, with standard video sequences and a dataset of 300 actual security, surveillance, news, and speech videos.

Power consumption has also become a major concern in CV systems, especially those employing battery-operated devices. In such systems, prolonging the battery lifetimes is a primary objective due to its great implications in terms of system cost and availability. In such systems, energy is consumed at the source in each of the three main phases: capturing, encoding, and transmission. Due to the limited amount of energy resources available, power consumption efficiency is one of the most challenging design factors. Since video encoding contributes to most of the overall power consumption at the video stations, the encoding parameter settings used at each station determine the encoding power consumption and bitrate of the video. We have developed an aggregate power consumption metric for many-to-one live video streaming systems. We model the video capturing, encoding, and transmission aspects and then provide an overall model of the power consumed by the video cameras and/or sensors. The model can help in the dynamic control of various camera/sensor settings, including resolution, frame rate, and quantization to achieve the best overall tradeoff in terms of power consumption, bitrate, and quality. We also analyze the power consumed by the monitoring station, which is due to video reception, potential video upscaling, and video decoding of all received video streams. The developed model captures the following main parameters: resolution, frame rate, quantization, motion estimation range, and number of reference frames. In addition to modeling the power consumption, we model the output bitrate of video encoding. The bitrate impacts the medium bandwidth, the video quality, and the transmission power consumption. We validate the developed models through extensive experiments. The analysis includes examining individual parameters separately as well examining the impacts of changing more than one parameter at a time. Spatial resolution and quantization parameters are the major contributors to the encoding outcome. Therefore, we analyze the effect of varying these parameters combination on encoding outcome.

We have developed an algorithm, called *History and Entropy-based LCU partitioning* (HELP), to increase the encoding speed of HEVC without considerable loss of coding efficiency and video quality performance. The encoding speed is around 5 times in average, which is about 1.42 the encoding speed of the fastest encoding speed algorithm which we referred to by Entropy-based [25]. The coding efficiency and video quality are still better than entropy in general. The algorithm predicts the size of the CU based on the entropy of the current CU and same size CUs in the spatial and temporal (co-locator) neighborhood that have been processed.

6.2 List of Publications

6.2.1 Published:

- Yousef Sharrab and Nabil J. Sarhan. Accuracy and Power Consumption Tradeoffs in Video Rate Adaptation for Computer Vision Applications. In Proceedings of the *2012 IEEE International Conference on Multimedia & Expo (ICME 2012)*, pages 410 - 415, Melbourne, Australia, July 2012. Acceptance rate: 30%.
- Yousef Sharrab and Nabil J. Sarhan. Detailed Comparative Analysis of VP8 and H.264. In Proceedings of the *IEEE International Symposium on Multimedia (ISM 2012)*, pages 133 - 140, Irvine, California, December 2012. Acceptance rate: 24.8%.
- Yousef Sharrab and Nabil J. Sarhan. Aggregate Power Consumption Modeling of Live Video Streaming Systems. In Proceedings of the *ACM Multimedia Systems (MMSys 2013)*, Oslo, Norway, February 27 - March 1, 2013. Acceptance rate: 23.8%.

6.2.2 Under Review:

- Yousef O. Sharrab and Nabil J. Sarhan, "Adaptation of Live Video Streams in Computer Vision Systems" submitted to *IEEE Transactions on Circuits and Systems for Video Technology* (TCSVT).
- Yousef O. Sharrab and Nabil J. Sarhan, "Modeling and Analysis of Power Consumption in Live Video Streaming Systems" submitted to *ACM Transactions on Multimedia Computing, Communications, and Applications* (TOMM).

6.3 Future Work

Even with our work, which speed up the encoding process five times in HEVC, the encoding speed still not practical, especially in live video streaming. We will enhance HELP algorithm to get even higher encoding speed and smiler coding efficiency and video quality performance to RDO algorithm. Our work will not focus only on splitting prediction, but also it will explore intra-prediction modes and motion estimation. We will use machine learning to predict block sizes and intra-prediction modes that will further enhance the encoding speed without degradation in coding efficiency and quality.

BIBLIOGRAPHY

[1] Milan Sonka, Vaclav Hlavac, and Roger Boyle, *Image processing, analysis, and machine vision*, Cengage Learning, 2014.

[2] Richard Szeliski, *Computer vision: algorithms and applications*, Springer Science & Business Media, 2010.

[3] Pavel Korshunov and Wei Tsang Ooi, "Critical video quality for distributed automated video surveillance," in *Proceedings of the 13th annual ACM international Conference on Multimedia*, Nov. 2005.

[4] Pavel Korshunov, "Rate-accuracy tradeoff in automated, distributed video surveillance systems," in *Proceedings of the 14th annual ACM international Conference on Multimedia*, 2006, pp. 887–889.

[5] Zhihai He, Yongfang Liang, Lulin Chen, Ishfaq Ahmad, and Dapeng Wu, "Power-rate-distortion analysis for wireless video communication under energy constraints," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 15, no. 5, pp. 645–658, May 2005.

[6] Aisha Arar, Amr A El-Sherif, Amr Mohamed, and Victor Leung, "Optimum power and rate allocation in cluster based video sensor networks," in *Proceedings of IEEE International Conference on Computing, Networking and Communications (ICNC)*, 2015, pp. 183–188.

[7] Seong-Ping Chuah, Yap-Peng Tan, and Zhenzhong Chen, "Rate and power allocation for joint coding and transmission in wireless video chat applications," 2015, vol. 17, pp. 687–699.

[8] Bambang AB Sarif, MT Pourazad, P Nasiopoulos, and VCM Leung, "Analysis of power consumption of h.264/avc-based video sensor networks through modeling the encoding complexity and bitrate," in *Proceedings of the International Conference on Digital Society (ICDS)*, 2014.

[9] Waqas A. Latif and Chiu C. Tan, "Smartargos: Improving mobile surveillance systems with software defined networks," in *IEEE Conference on Communications and Network Security (CNS)*,

2015.

[10] Meng Guo, Mostafa H Ammar, and Ellen W Zegura, "V3: A vehicle-to-vehicle live video streaming architecture," *Pervasive and Mobile Computing*, vol. 1, no. 4, pp. 404–424, 2005.

[11] Elias Yaacoub and Fethi Filali, "Cluster based V2V communications for enhanced QoS of SVC video streaming over vehicular networks," in *Proceedings of International Wireless Communications and Mobile Computing Conference (IWCMC)*, 2014.

[12] Bin-Feng Lin, Yi-Ming Lin, Li-Chen Fu, Pei-Yung Hsiao, Li-An Chuang, Shih-Shinh Huang, and Min-Fang Lo, "Integrating appearance and edge features for sedan vehicle detection in the blind-spot area," *IEEE Transactions on Intelligent Transportation Systems*, vol. 13, no. 2, 2012.

[13] US DOT, "Connected vehicle insights - trends in computer vision," in *http://www.its.dot.gov/index.htm*, 2011.

[14] Clint Wheelock, "Tractica research reports," in *https://www.tractica.com/newsroom/press-releases/computer-vision-technology-market-to-reach-33-3-billion-by-2019/*.

[15] Philip Ingram, "How many cctv cameras are there globally?," in *http://www.securitynewsdesk.com/*.

[16] Bambang AB Sarif, Mahsa T Pourazad, Panos Nasiopoulos, and Victor Leung, "Encoding and communication energy consumption trade-off in h. 264/avc based video sensor network," in *IEEE International Symposium and Workshops on a World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, 2013, pp. 1–6.

[17] Shih-Fu Chang and Anthony Vetro, "Video adaptation: concepts, technologies, and open issues," *Proceedings the IEEE*, vol. 93, no. 1, pp. 148–158, 2005.

[18] A. Eleftheriadis and D. Anastassiou, "Constrained and general dynamic rate shaping of compressed digital video," in *Proceedings of the 1995 International Conference on Image Processing*, 1995.

[19] Minjung Kim and Yucel Altunbasak, “Optimal dynamic rate shaping for compressed video streaming,” in *Proceedings of the First International Conference on Networking-Part 2 (ICN)*, 2001, pp. 786–794.

[20] J. Kim, Y. Wang, and S. Chang, “Content-adaptive utility-based video adaptation,” in *Proceedings of the IEEE International Conference on Multimedia and Expo (ICME)*, July 2003, pp. 281–284.

[21] Cheng hsin Hsu and Mohamed Hefeeda, “Video quality for face detection, recognition, and tracking,” *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMCCAP)*, vol. 7, no. 1, January 2011.

[22] Frank Bossen, Benjamin Bross, Karsten Suhring, and Damian Flynn, “Hevc complexity and implementation analysis,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, no. 12, pp. 1685–1696, 2012.

[23] Claude E Shannon and Warren Weaver, *The mathematical theory of communication*, University of Illinois press, 2015.

[24] Claude Elwood Shannon, “A mathematical theory of communication,” *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 5, no. 1, pp. 3–55, 2001.

[25] Mengmeng Zhang, Jianfeng Qu, and Huihui Bai, “Entropy-based fast largest coding unit partition algorithm in high-efficiency video coding,” *Entropy*, vol. 15, no. 6, pp. 2277–2287, 2013.

[26] Thomas Wiegand, Gary J. Sullivan, Gisle Bjntegaard, and Ajay Luthra, “Overview of the H.264/AVC video coding standard,” *IEEE Transaction on Circuits and Systems for Video Technology*, vol. 13, no. 7, 2003.

[27] Iain E. G. Richardson, *The H.264 Advanced Video Compression Standard, Second Edition, WILEY, Published Online*, Vcodex Limited, UK, 2010.

[28] Touradj Ebrahimi and Caspar Horne, “Mpeg-4 natural video coding: An overview,” *Signal Pro-*

cessing: Image Communication, vol. 1, no. 15, pp. 365–385, September 2000.

[29] Guilherme Corrêa, Pedro Assuncao, Luciano Agostini, and Luis A Cruz, "Complexity control of high efficiency video encoders for power-constrained devices," *IEEE Transactions on Consumer Electronics*, vol. 57, no. 4, pp. 1866–1874, 2011.

[30] Chenggang Yan, Yongdong Zhang, Feng Dai, and Liang Li, "Highly parallel framework for hevc motion estimation on many-core platform," in *Data Compression Conference (DCC)*. IEEE, 2013, pp. 63–72.

[31] Changfeng Yan, Ye Zhang, Feng Dai, Juyong Zhang, Luoqing Li, and Qionghai Dai, "Efficient parallel hevc intra-prediction on many-core processor," *Electronics Letters*, vol. 50, no. 11, pp. 805–806, 2014.

[32] Liquan Shen, Zhaoyang Zhang, and Zhi Liu, "Effective cu size decision for hevc intracoding," *IEEE Transactions on Image Processing*, vol. 23, no. 10, pp. 4232–4241, 2014.

[33] Wei-Jhe Hsu and Hsueh-Ming Hang, "Fast coding unit decision algorithm for hevc," in *Proceedings of Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA)*. IEEE, 2013, pp. 1–5.

[34] Xiaolin Shen and Lu Yu, "Cu splitting early termination based on weighted svm," *EURASIP Journal on Image and Video Processing*, vol. 2013, no. 1, pp. 1–11, 2013.

[35] Jie Leng, Lei Sun, Takeshi Ikenaga, and Shinichi Sakaida, "Content based hierarchical fast coding unit decision algorithm for hevc," in *Proceedings of IEEE International Conference on Multimedia and Signal Processing (CMSP)*, 2011, vol. 1, pp. 56–59.

[36] Qin Yu, Xinfeng Zhang, Shiqi Wang, and Siwei Ma, "Early termination of coding unit splitting for hevc," in *Proceedings of Signal & Information Processing Association Annual Summit and Conference (APSIPA ASC), 2012 Asia-Pacific*. IEEE, 2012, pp. 1–4.

[37] K. Choi and E. Jang, "Coding tree pruning based cu early termination," *document JCTVC-F092 of JCT-VC, Torino, IT*, 2011.

[38] Du-Yih Tsai, Yongbum Lee, and Eri Matsuyama, "Information entropy measure for evaluation of image quality," *Journal of digital imaging*, vol. 21, no. 3, pp. 338–347, 2008.

[39] SALEH ABDEL-GADER AMAREEN, "Efficient algorithms for coding unit size selection in hevc using entropy and number of blocks," 2014.

[40] Omar Javed and Mubarak Shah, "Tracking and object classification for automated surveillance," in *Proceedings of the 7th European Conference on Computer Vision-Part IV*, 2002, pp. 343–357.

[41] Xiaojing Yuan, Zehang Sun, Yaakov Varol, and George Bebis, "A distributed visual surveillance system," in *Proceedings of the IEEE Conference on Advanced Video and Signal Based Surveillance (AVSS)*, 2003, p. 199.

[42] W. Niu and J. Long, "Human activity detection and recognition for video surveillance," in *Proceedings of the IEEE Multimedia and Expo Conference (ICME)*, June 2004, pp. 719–722.

[43] Yousef O. Sharrab and Nabil J. Sarhan, "Aggregate power consumption modeling of live video streaming systems," in *Proceedings of the ACM Multimedia Systems Conference*, February 2013, pp. 60–71.

[44] Pavel Korshunov and Wei Tsang Ooi, "Video quality for face detection, recognition, and tracking," *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMCCAP)*, vol. 7, no. 3, August 2011.

[45] Nafise Barzigar, Aminmohammad Roozgard, Pulkit Verma, and Shukang Cheng, "A video super resolution framework using scobep," 2014.

[46] George Lentaris Georgios Georgis and Dionysios Reisis, "Reduced complexity superresolution for low-bitrate video compression," *IEEE Transactions on Circuits and Systems for Video Technology*,

2016.

[47] Abdelhafid Elouardi, Samir Bouaziz, Antoine Dupret, Lionel Lacassagne, Jacques-Olivier Klein, and Roger Reynaud, "Image processing vision systems: Standard image sensors versus retinas," *IEEE Transactions on Instrumentation and Measurement*, vol. 56, no. 5, pp. 1675–1687, 2007.

[48] Robert LiKamWa, Bodhi Priyantha, Matthai Philipose, Lin Zhong, and Paramvir Bahl, "Energy characterization and optimization of image sensing toward continuous mobile vision," in *Proceedings of the ACM Annual International Conference on Mobile Systems, Applications, and Services (MobiSys)*, 2013, pp. 69–82.

[49] C. Sampath Kannangara, Iain. E. Richardson, and A. J. Miller, "Computational complexity management of a real-time H.264/AVC encoder," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 18, no. 9, pp. 1191–1200, July 2008.

[50] Yih Han Tan, Wei Siong Lee, Jo Yew Tham, Susanto Rahardja, and Kin Mun Lye, "Complexity scalable H.264/AVC encoding," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 20, no. 9, pp. 2010 1271, September 2010.

[51] Bambang AB Sarif, Mahsa Pourazad, Panos Nasiopoulos, and Victor CM Leung, "A study on the power consumption of h. 264/avc-based video sensor network," *International Journal of Distributed Sensor Networks*, vol. 2015, 2015.

[52] Swaminathan Vasanth Rajaraman, Matti Siekkinen, and Mohammad A. Hoque, "Energy consumption anatomy of live video streaming from a smartphone," in *IEEE Annual International Symposium on Personal, Indoor, and Mobile Radio Communication (PIMRC)*, 2014, pp. 2013–2017.

[53] Wei Pu, Yan Lu, and Feng Wu, "Joint power-distortion optimization on devices with MPEG-4 AVC/H.264 codec," in *Proceedings of IEEE International Conference on Communications (ICC)*,

2006, pp. 441–446.

[54] Jaemoon Kim, Jungsoo Kim, Giwon Kim, and Chong-Min Kyoung, “Power-rate-distortion modeling for energy minimization of portable video encoding devices,” in *Proceedings of IEEE International Midwest Symposium on Circuits and Systems (MWSCAS)*, August 2011, pp. 1–4.

[55] Li Su, Yan Lu, Feng Wu, Shipeng Li, and Wen Gao, “Complexity-constrained H.264 video encoding,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 19, no. 4, pp. 477–490, April 2009.

[56] Xiaoan Lu, Thierry Fernaine, and Yao Wang, “Modelling power consumption of a h.263 video encoder,” in *Proceedings of IEEE International Symposium on Circuits and Systems (ISCAS)*, 2004, pp. 77–80.

[57] Carolina Blanch, Tong Gan, Antoine Dejonghe, and Bart Masschelein, “Cross-layer optimization for the scalable video codec over WLAN,” in *Proceedings of International Packet Video Workshop*, May 2009, pp. 1–6.

[58] Yingkun Wang, Yuanhua Zhou, and Hua Yang, “Early detection method of all-zero integer transform coefficients,” *IEEE Transactions on Consumer Electronics*, vol. 50, no. 3, pp. 923–928, 2004.

[59] Ce Zhu, Xiao Lin, Lap-Pui Chau, Keng-Pang Lim, Hock-Ann Ang, and Choo-Yin Ong, “A novel hexagon-based search algorithm for fast block motion estimation,” in *Proceedings of Acoustics, Speech, and Signal Processing*, 2001, pp. 1593–1596 vol.3.

[60] Manish Bhardwaj and Anantha P. Chandrakasan, “Bounding the lifetime of sensor networks via optimal role assignments,” in *Proceedings of IEEE INFOCOM*, 2002, vol. 3, pp. 1587–1596.

[61] Zhihai He and Dapeng Wu, “Resource allocation and performance analysis of wireless video sensors,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 16, no. 5, pp.

590–599, May 2006.

[62] Rei-Heng Cheng and Chiming Huang, "The impact of the transmission power range on energy consumption for wireless sensor networks," in *Proceedings of International Conference on Ubiquitous and Future Networks (ICUFN)*, July 2013, pp. 77–81.

[63] Huseyin Cotuk, Kemal Bicakci, Bulent Tavli, and Erkam Uzun, "The impact of transmission power control strategies on lifetime of wireless sensor networks," *IEEE Transactions on Computers*, vol. 63, no. 11, pp. 2866–2879, 2014.

[64] Mohammad Alsmirat and Nabil J. Sarhan, "Cross-layer optimization and effective airtime estimation for wireless video streaming," in *Proceedings of International Conference on Computer Communications and Networks (ICCCN)*, July 2012.

[65] Zhou Wang, Alan C. Bovik, Hamid R. Sheikh, and Eero P. Simoncelli, "Image quality assessment: From error visibility to structural similarity," *IEEE Transactions on Image Processing*, vol. 13, no. 4, pp. 600–612, 2004.

[66] SAE International, "On-board system requirements for v2v safety communications standard: J2945/1_201603," March 2016.

[67] Paul Viola and Michael Jones, "Robust real-time face detection," in *Proceedings of the ICCV 2001 Workshop on Statistical and Computation Theories of Vision (ICCV)*, July 2001, p. 747.

[68] Wu-Chi Feng, Ed Kaiser, Wu Chang Feng, and Mikael Le Baillif, "Panoptes: scalable low-power video sensor networking technologies," *ACM Transactions Multimedia Computer Comm. Application.*, vol. 1, no. 2, pp. 151–167, 2005.

[69] Mohammad Alsmirat and Nabil J. Sarhan, "Cross-layer optimization for automated video surveillance," in *IEEE International Symposium on Multimedia (ISM)*, December 2016, pp. 243–246.

[70] Yousef O. Sharrab and Nabil J. Sarhan, "Accuracy and power consumption tradeoffs in video rate

adaptation for computer vision applications," in *Proceedings of the IEEE International Conference on Multimedia and Expo (ICME)*, July 2012, pp. 410–415.

[71] Muhammad Shafique, Bastian Molkenthin, and Jörg Henkel, "An HVS-based adaptive computational complexity reduction scheme for H.264/AVC video encoder using prognostic early mode exclusion," in *Proceedings of Design, Automation and Test in Europe Conference and Exhibition*, 2010, pp. 1713–1718.

[72] Changsung Kim and C.-C. Jay Kuo, "Feature-based intra-/intercoding mode selection for H.264/AVC," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 17, no. 4, pp. 441–453, April 2007.

[73] Alexis M. Tourapis, Oscar C. Au, and Ming L. Liou, "Predictive motion vector field adaptive search technique (PMVFAST) – enhancing block based motion estimation," in *Proceedings of Visual Communications and Image Processing Conference*, 2001, pp. 883–892.

[74] Weiyao Lin, Krit Panusopone, David M. Baylon, Ming-Ting Sun, Zhenzhong Chen, and Hongxiang Li, "A fast sub-pixel motion estimation algorithm for H.264/AVC video coding," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 21, no. 2, pp. 237–242, February 2011.

[75] Xiaozhong Xu and Yun He, "Improvements on fast motion estimation strategy for H.264/AVC," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 18, no. 3, March 2008.

[76] Jongho Kim, Donghyung Kim, and Jechang Jeong, "Complexity reduction algorithm for intra mode selection in H.264/AVC video coding," in *Proceedings of Advanced Concepts for Intelligent Vision Systems (ACIVS)*, September 2006, pp. 454–465.

[77] Thomas D. Burd and Robert W. Brodersen, "Processor design for portable systems," *VLSI Signal Process*, vol. 13, no. 2, pp. 203–222, Aug 1996.

[78] Ming-Ting Sun and I-Ming Pao, “Statistical computation of discrete cosine transform in video encoders,” *Journal of Visual Communication and Image Representation*, vol. 9, no. 2, pp. 163–170, June 1998.

[79] Gary J Sullivan, J-R Ohm, Woo-Jin Han, and Thomas Wiegand, “Overview of the high efficiency video coding (hevc) standard,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, no. 12, pp. 1649–1668, 2012.

[80] Thaísa L da Silva, Luciano V Agostini, and Luis A Cruz, “Fast hevc intra prediction mode decision based on edge direction information,” in *Proceedings of the European Signal Processing Conference (EUSIPCO)*. IEEE, 2012, pp. 1214–1218.

[81] Thaisa L da Silva, Luciano V Agostini, and Luis A da Silva Cruz, “Speeding up hevc intra coding based on tree depth inter-levels correlation structure,” in *Proceedings of the European Signal Processing Conference (EUSIPCO)*. IEEE, 2013, pp. 1–5.

[82] Zhaoqing Pan, Yun Zhang, Sam Kwong, Xu Wang, and Long Xu, “Early termination for tzsearch in hevc motion estimation,” in *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2013, pp. 1389–1393.

[83] Hamid R Sheikh, Muhammad F Sabir, and Alan C Bovik, “A statistical evaluation of recent full reference image quality assessment algorithms,” *IEEE Transactions on image processing*, vol. 15, no. 11, pp. 3440–3451, 2006.

[84] Gisle Bjontegaard, “Calcuation of average psnr differences between rd-curves,” *Doc. VCEG-M33 ITU-T Q6/16, Austin, TX, USA, 2-4 April 2001*, 2001.

[85] G. Bjntegaard, “Improvements of the bd-psnr model,” *ITU-T SG16/Q6 document VCEG-Al11, Berlin, Germany*, 2008.

[86] Philippe Hanhart and Touradj Ebrahimi, “Calculation of average coding efficiency based on sub-

jective quality scores," *Journal of Visual Communication and Image Representation*, vol. 25, no. 3, pp. 555–564, 2014.

[87] F. Bossen, "Common test conditions and software reference configurations," *document JCTVC-H1100 of JCT-VC, San Jose, CA, USA*, 2012.

[88] K. Schrter, "Hevc test model (hm) software," in *Fraunhofer Heinrich Hertz Institute*. https://hevc.hhi.fraunhofer.de, 2014.

[89] S. Patrick F. Frank R. Martin, K. Lina and M. Tatjana, "Video trace library," *http://trace.eas.asu.edu/yuv/*.

[90] K. S. Wong, "Video traces," *http://web.fsktm.um.edu.my/ koksheik*.

ABSTRACT

VIDEO STREAM ADAPTATION IN COMPUTER VISION SYSTEMS

by

YOUSEF ODEH SHARRAB

May 2017

Advisor: Dr. Nabil Sarhan

Major: Computer Engineering

Degree: Doctor of Philosophy

Computer Vision (CV) has been deployed recently in a wide range of applications, including surveillance and automotive industries. According to a recent report, the market for CV technologies will grow to 33.3 billion by 2019. Surveillance and automotive industries share over 20% of this market. This dissertation considers the design of real-time CV systems with live video streaming, especially those over wireless and mobile networks. Such systems include video cameras/sensors and monitoring stations. The cameras should adapt their captured videos based on the events and/or available resources and time requirement. The monitoring station receives video streams from all cameras and run CV algorithms for decisions, warnings, control, and/or other actions. Real-time CV systems have constraints in power, computational, and communicational resources. Most video adaptation techniques considered the video distortion as the primary metric. In CV systems, however, the main objective is enhancing the event/object detection/recognition/tracking accuracy. The accuracy can essentially be thought of as the quality perceived by machines, as opposed to the human perceptual quality. High-Efficiency Video Coding (HEVC) is a recent encoding standard that seeks to address the limited communication bandwidth problem as a result of the popularity of High Definition (HD) videos. Unfortunately, HEVC adopts algorithms that greatly slow down the encoding process, and thus results in complications in real-time

systems.

This dissertation presents a method for adapting live video streams to limited and varying network bandwidth and energy resources. It analyzes and compares the rate-accuracy and rate-energy characteristics of various video streams adaptation techniques in CV systems. We model the video capturing, encoding, and transmission aspects and then provide an overall model of the power consumed by the video cameras and/or sensors. In addition to modeling the power consumption, we model the achieved bitrate of video encoding. We validate and analyze the power consumption models of each phase as well as the aggregate power consumption model through extensive experiments. The analysis includes examining individual parameters separately and examining the impacts of changing more than one parameter at a time. For HEVC, we develop an algorithm that predicts the size of the block without iterating through the exhaustive Rate Distortion Optimization (RDO) method. We demonstrate the effectiveness of the proposed algorithm in comparison with existing algorithms. The proposed algorithm achieves approximately 5 times the encoding speed of the RDO algorithm and 1.42 times the encoding speed of the fastest analyzed algorithm.

AUTOBIOGRAPHICAL STATEMENT

Yousef Sharrab is Research and Development Engineer at GM Global R&D Labs. He is a Ph.D candidate in the Electrical and Computer Engineering Department at Wayne State University. He received his M.Sc. Degree in Computer Science from New York Institute of Technology and his B.S. degree in Electrical and Computer Engineering from Jordan University of Science and Technology. Yousef Sharrab main research is in Video Stream Adaptation in Computer Vision Systems, Automated Video Surveillance, Systems Simulation and Modeling, Connected Vehicle Systems, and Autonomous Driving.