Aggregate Power Consumption Modeling of Live Video Streaming Systems

Yousef O. Sharrab and Nabil J. Sarhan
Electrical and Computer Engineering Department & Wayne State Media Research Lab
Wayne State University
Detroit, MI 48202
{yousef.sharrab,nabil}@wayne.edu

ABSTRACT

Power consumption of video streaming systems has become a major concern, especially in battery-powered devices, such as video sensors. Power is usually dissipated in each one of the major phases of the streaming process: capturing, encoding, and transmission. This paper develops models for power consumption in each of these phases and validates them with extensive experiments, focusing primarily on H.264 video encoding. For comparative purposes, we also study MJPEG and MPEG-4 video codecs. In addition, we analyze the impacts of the main H.264 video compression parameters on power consumption and bitrate. These parameters include quantization parameter, number of reference frames, motion estimation (ME) range, and ME algorithm.

Categories and Subject Descriptors

H.4.3 [**Information Systems Applications**]: Communications Applications

General Terms

Algorithms, Design, Experimentation

Keywords

Power Consumption Modeling, H.264, Streaming Video, Video Capturing, Video Encoding, Video Transmission.

1. INTRODUCTION

Power consumption has become a major concern in live video streaming systems, especially those employing battery-operated devices, such as automated video surveillance and wireless sensor networks. 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 video streaming systems, energy is consumed at the source in each of the three main phases: capturing, encoding, and transmission. Power consumption at the receivers (such as monitoring stations) may be important but to much lower degrees. In a streaming environment, the three phases operate in a pipelined fashion.

MMSys'13, February 26-March 1, 2013, Oslo, Norway.

This paper analyzes power consumption in the capturing, encoding, and transmission phases. It develops models for all these phases and then validates them individually and in terms of the aggregate power consumption. The developed models are based on 500 different experiments, each of which is repeated at least 3 times, totaling more than 1,500 experiments. Prior studies focused primarily on one aspect/phase. In particular, study [8] developed a power consumption model for video capturing for a specific on-chip vision circuit. For wireless video sensor networks (WVSNs), transmission power consumption has been analyzed in [2, 11]. Encoding power consumption has been studied by [10] and [18]. Study [10] developed a Power-Rate-Distortion (P-R-D) framework for a generic video encoder (that applies to H.263), but it did not analyze the effects of spatial resolution and frame rate. Paper [18] measured the power consumption of an H.263 encoder, but the experiments were limited to QCIF resolution and a 10 fps frame rate. Moreover, it did not consider varying the spatial and temporal resolutions and other encoding parameters. None of these two papers analyzed H.264 or considered the capturing power consumption. H.264 has many new features and algorithms, especially in intra-prediction and inter-prediction. In addition, the impacts on power consumption of important parameters (such as quantization parameter, number of reference frames, search range, and motion estimation algorithms) were not analyzed in prior work (up to our knowledge). Other work on encoding includes developing a cross-layer approach in [3] to tradeoff between coding and communication power consumptions. Furthermore, much work considered improving video encoding, such as using a statistical approach to reduce the computation times of the most computationally intensive components of video coding [9], early detection of all-zero integer transform coefficients [30], and a hexagon-based search (HEXBS) pattern for fast motion estimation (ME) [33].

This paper has been motivated by our ongoing work on the power-aware design of automated video surveillance systems, which requires accurate, simple, and appropriate power consumption models. The main contributions of this paper can be summarized as follows. (1) We analyze the three main phases in live video streaming systems and provide accurate and simple power consumption models. (2) For video encoding, we develop a model for the H.264 standard, considering important factors, such as mode selection, the number of reference frames, and sub-pixel ME search. The developed full mode selection model can be used to find the total number of operations required by H.264 encoding and the number of operations of each type (add, multiply, divide, etc.). (3) We analyze the impacts of important encoding parameters on power consumption, including quantization parameter, number of reference frames, ME algorithms, and ME range. Since tuning the parameters is often based on an energy and bitrate tradeoff, we develop models for the

bitrate as well. (4) We compare the power consumption of each phase and study other encoders, including MPEG-4 and MJPEG, for comparative purposes. (5) We show that the overall computation complexity for all phases can approximately be modeled as a linear function of the pixel rate. The pixel rate is the product of the spatial and temporal resolutions of the raw video. (6) We conduct extensive real experiments.

The rest of the paper is organized as follows. Section 2 discusses background information and related work. Section 3 discusses the setup of experiments and modeling methodology. Section 4 develops various models and presents and analyzes the main results.

2. BACKGROUND INFORMATION AND RELATED WORK

As shown in Figure 1, live video streaming systems consists of three main phases at the source side: capturing, encoding, and transmission. Next discuss each of these phases.

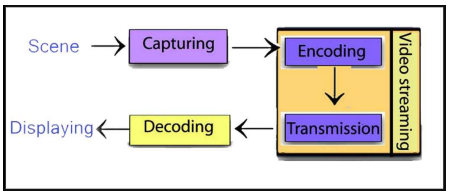


Figure 1: Live Video Streaming Block Diagram

2.1 Video Capturing Power Consumption

Cameras include image sensors, which are silicon devices that capture images. These sensors come in two main classes: *Charge-Coupled Devices* (CCD) and *Complementary Metal Oxide Semiconductor* (CMOS). CCD sensors capture light onto an array of light-sensitive diodes, with each diode representing one pixel. Each diode converts the light photons into a charge. Subsequently, the charges are moved and amplified in full rows. CMOS works like CCD, but each pixel has its own voltage amplifier. In addition, each pixel can be read directly on an $x-y$ coordinate system rather than raw-by-raw as in CCD.

Study [8] presented a vision system based on a smart sensor, called PARISI (Programmable Analog Retin-like Image Sensor I). The experiments were based on a sensor circuit with 16×16 pixels and 16 analog processing units. Study [5] characterized the power consumption of the PARIS-based vision system. In particular, the total power consumption of NxN resolution and N analog processing units was shown to be given by

$$W_s = c_a.N^2 + c_b.N, \quad (1)$$

where c_a, c_b are constants. Study [29] suggested that power consumption of video sensors depends on the frame size and frame rate, but without conducting actual experiments.

This paper considers the popular CMOS sensors and develops a power consumption model based on extensive experiments. Although, we experiment with CMOS, we believe the same model can be applied to CCD with different parameters because of the similarity in the structure and operation.

2.2 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, 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).

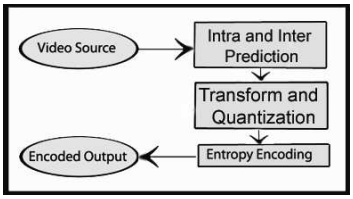


Figure 2: Video Encoding Block Diagram

Before the transmission, the video streams may be adapted. With video rate adaptation, the videos are shaped or transformed to meet the bandwidth constraint [7, 16, 15]. At the receiver side, the video is decoded and rendered.

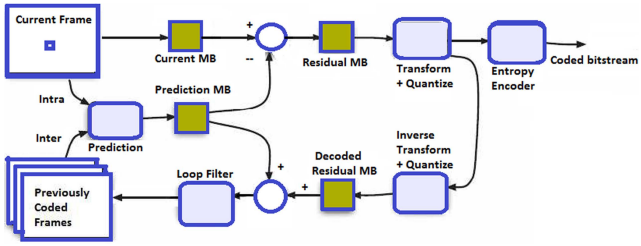


Figure 3: Block Diagram of H.264 Encoder

As H.264 is the primary focus of this paper, let us now discuss it in more detail. H.264 employs many features for more efficient compression and better flexibility with the network environment [31]. Figure 3, shows the processing stages of H.264. At this high level, MPEG-4 compression is similar but does not include the loop filter stage. The processing stages can be described as follows.

Intra and Inter Prediction (Estimation) Stage
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. Since MJPEG uses only intra compression, it does not use reference frames. In contrast, MPEG-4 allows one reference frame. In Subsection 4.2.1, we analyze the impact of the number of reference frames on power consumption and bitrate.

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 MJPEG and MPEG-4, the minimum block size is 8×8.

An H.264 encoder can choose from many different intra and inter modes when coding a macroblock. Within each inter mode, the encoder has a wide choice of possible MVs, leading to a huge number of options for coding a macroblock [22]. 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. The bitrate cost r and distortion cost t are combined into a single cost J:

$$J = t + gr. \tag{2}$$

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 (qp) is given by

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

Further details can be found in [22].

Transform Stage
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 MJPEG and MPEG-4 use an 8×8 DCT.

Quantization Stage
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 [6]. In Subsection 4.2.1, we analyze the impact of the quantization parameter.

Loop Filter Stage
A deblocking filter is applied to every decoded macroblock in order to smooth block edges. It is applied after the inverse transform in the encoder. No built-in deblocking filter is used in MPEG-4 or MJPEG.

Entropy Coding Stage
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.

Related Work on Video Encoding Power Consumption
Much of the work on H.264 dealt with managing the computation complexity [26, 30, 33]. Papers [10] and [18] studied the encoding power for certain encoders. Study [10] developed a Power-Rate-Distortion (PRD) framework specifically for a generic video encoder (that applies to H.263). That framework extended the traditional RD analysis by including a third dimension, which is the power consumption. That study jointly optimized the PRD performance by adjusting the number of sum of absolute differences (SAD) computations and the number of DCT computations. It did not analyze how the complexity is affected by the spatial resolution, quantization parameter, number of reference frames, and motion estimation (ME) search range. In [18], a model was developed for H.263. That paper measured 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. Study [20] considered two types of optimization scenarios, which determine how to assess the optimized encoding parameters in order to (i) to minimize the total power consumption (for transmission and encoding) for a given video quality level and (ii) to maximize video quality for a given power consumption level. That study, however, was limited to ME search algorithms. In particular, it considered and compared 12 different ME algorithms (modes) and measured the bitrate, distortion, and cycles per second for each one of these modes. A PRD model of a hardware-based encoder was introduced in [14] using the power scalable architecture of an H.264. That study was limited to analyzing the integer and fractional ME search range and I-frame period.

In [25], a joint power-distortion optimization scheme for real-time H.264 video encoding under the power constraint was proposed. That study divided the encoding modules into basic operation units, such as the *sum of absolute differences* (SAD) operations. It measured the encoding complexity of basic operation units by summing up the required processor cycles. For example, a 4×4 SAD operation requires 353 processor cycles. It considered only the ME search algorithm and did not study the spatial and temporal effects.

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. In this paper, as in [3, 9, 10, 18, 24, 30, 33], we consider DVS in our model developments.

Other work on encoding includes developing a cross-layer approach in [3] to tradeoff between coding and communication power consumptions. Furthermore, much work considered improving video encoding. Examples include using a statistical approach to reduce the computation times of the most computationally intensive components of video coding [9], early detection of all-zero integer transform coefficients [30], and hexagon-based search (HEXBS) pattern for fast ME [33].

2.3 Video Transmission Power Consumption

Transmission power consumption is affected mainly by technology or platform, distance, path-loss or environment, and bitrate. The main factors that impact the power consumptions in a WiFi 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) [1].

Most of the recent work on transmission power consumption were focused on ad-hoc wireless sensor networks. For example, study [2] derived upper bounds on the lifetime of sensor networks. In addition, study [11] examined the resource utilization behavior of a wireless video sensor and analyzed its performance under resource constraints.

In this paper, we simplify the wireless/wired power consumption model developed in [2] and adapt it to our environment. We are concerned with primarily the impact of video parameters on the transmission power consumption and thus the impact of various technologies will translate to constants in the model.

3. EXPERIMENTAL SETUP AND MODELING METHODOLOGY

This paper develops general models of the power consumption of the three phases in the video streaming systems: capturing, encoding, and transmission. We validate the developed models by con-

ducting extensive experiments. Figure 4 shows the experimental setup. A Dell Inspiron 1525 laptop with an Intel Core 2 Duo CPU dual-core processor (Model T5750) running at 2.00 GHz with 3.00 GB memory and an external video camera (Logitech Webcam Pro 9000) is used to capture a video rendered on a desktop computer. The external camera is directed to that desktop computer, which plays a specific movie (from the beginning to the end) to ensure that the experiments can be repeated without changes in the video content. The rendered video includes scenes of five children running and playing in a zoo, with much details and fast movements. As discussed later, the actual video content does not change the model, but only the model parameters. 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 MJPEG 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 laptop computer.

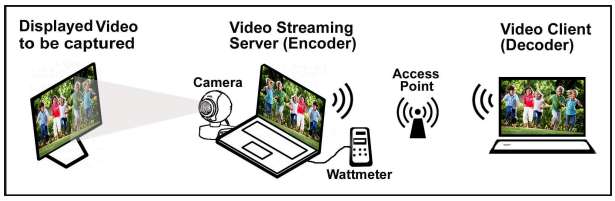


Figure 4: Experimental Setup

The consumed power is measured by an advanced power meter: "Watts Up? Pro ES AC". We measure the power consumed by the streaming server for H.264, MPEG-4 and MJPEG encoding. For encoding, we vary the spatial (i.e., frame size) and temporal (i.e., frame rate) resolutions generally from 160×120 to 800×600 and from 1 to 30 fps, respectively. For H.264, we also study the effect of varying the quantization parameter, the number of references, ME algorithm, and ME range. We use the default encoding parameters in both X.264 and FFmpeg except for those that are under study.

In each experiment, the video is played for 22 minutes and 41 seconds. The reported power is the average power consumption during the whole video period. Each reported value is the average of 1361 power readings, each of which is obtained during one second of the video.

To minimize the effect of other processes while running the experiments, we run the laptop 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.

4. POWER CONSUMPTION MODELING, RESULTS PRESENTATION, AND ANALYSIS

In this section, we develop various power consumption models.

Table 1: Descriptions of Used Symbols

Symbol	Description
W	Power Consumption (Watt)
X	Computation Complexity (basic operation)
r	Bit rate (Kbit/s)
L	Pixel rate (pixel/s)
F	Frame rate (frame/s) Rate
E	Encoding rate (bit/frame)
c	Constant
NxM	Frame dimensions in pixels
PxQ	MB dimensions in pixels
K	Number of Analogue to Digital (A/D) Units
Y	Number of bits/frame
R	Number of reference frames for ME
d	Distance between sender and receiver (meter)
n	Path-loss index in transmission
S, S'	Displacement in pixels for ME
qp	Quantization parameter
i	Number of I frames in GOP
p	Number of P frames in GOP
b	Number of B frames in GOP
J	A joint cost (db)
t	Distortion (db)
g	Lagrange multiplier
N	Number of operations
v	Voltage (volt)
f	Frequency (Hz)
V	Number of MVs in a MB

Table 1 summarizes the symbols used in this section.

4.1 Modeling of the Power Consumed by Video Capturing

To model the power consumption of video capturing in general video sensors (CMOS and CCD), let us first start by generalizing Equation (1) to a general mesh of photo-diodes 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

$$W_s = c_i.N.M + c_b.K, \tag{4}$$

where c_i and c_b are constants. Equation (4) 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 considering the temporal resolution. Thus, the total capturing power consumption W_C can be expressed as follows:

$$W_c = F.W_s = F.(c_i.N.M + c_b.K), \tag{5}$$

where F is the frame rate. As indicated by Equation (5), the main players in the capturing power consumption are the spatial and temporal resolutions. The impacts of a specific sensor type (CMOS or CCD), technology, and/or implementation translate to (changing the values of) constants in the model. Our experiments confirm that Equation (5) applies but with an additional constant:

$$W_c = F.(c_i.N.M + c_b.K) + c_j, \tag{6}$$

where c_j is a constant specifying the power consumed by the sensor when no capturing takes place.

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:

$$W_c \approx c_i.F.N.M + c_j. \tag{7}$$

Equation (7) 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 as a function of the pixel rate can be given by

$$W_c \approx c_i.L + c_j, \tag{8}$$

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.

Figure 5(a) compares the model (Equation (6)) and the simplified model (Equation (8)) with actual experimental data when both the spatial and temporal resolutions are varied. The results show that the model in both forms accurately represents the real behavior. Figures 5(b) and 5(c) validate the model when only the temporal resolution or spatial resolution is varied, respectively.

4.2 Modeling of the Power Consumed by Video Encoding

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 match inside a reference frame for each of these blocks, using a predefined distortion criterion. The best match 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 [27]. 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, $(2.S+1)^2$ 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 macroblock (MB) size of $P \times Q$ and R reference frames, the integer motion estimation (ME) computation complexity X_i can be given by

$$X_i = F.\frac{N.M}{P.Q}.R.(2.S+1)^2.(2.P.Q - 1 + V.X_{MV})$$

$$\approx (4.F.N.M.R.S^2)(2 + \frac{V.X_{MV}}{P.Q}), \tag{9}$$

where $(2.P.Q-1)$ 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. Based on the *single equal reference condition* method [28], the computational complexity for computing a MV (X_{MV}) includes 3 multiplies, 3 additions, 24 shifts, 1 median of 3 MVs, and 2 subtractions. Equation (9) generalizes the equation in [27] to handle multiple reference frames and consider the effect of computing MVs. The term $V.X_{MV}$ shows the effect of block size on computational complexity. As an approximation, it can be ignored leading to

$$X_i \approx 8.F.N.M.R.S^2, \tag{10}$$

Table 2: Per-Pixel Computation Complexity of Interpolation for Fractional Pixel ME

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

Let us now develop a ME complexity model for encoders supporting sub-pixel search (such as H.264 and MPEG-4/ASP). 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 [17]. The complexity depends on the number of operations for interpolating in-between pixels in the block (i.e., pixels at non-integer locations) [22]. Table 2 shows the number of operations for interpolation. 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 [32]. Note 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. With S' representing the range of the sub-pixel search in pixels and X_p representing the number of operations for pixel interpolation, the computation complexity X_s of fractional pixel ME can be given by

$$X_s = F.\frac{N.M}{P.Q}.(2.S' + 1)^2.(2.P.Q - 1 + P.Q.X_p)$$

$$\approx 4.(2 + X_p).F.N.M.S'^2, \tag{11}$$

Therefore, the total (integer and fractional) ME computation complexity for full search X_c can be given by

$$X_c = X_i + X_s$$

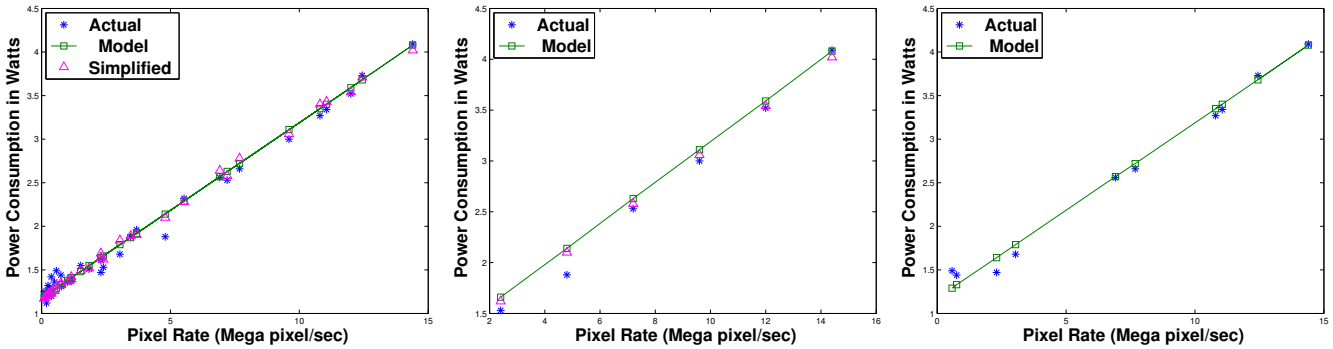
$$= (4.F.N.M.R.S^2)(2 + \frac{V.X_{MV}}{P.Q}) + 4.(2 + X_p).F.N.M.S'^2,$$

$$\approx 4.L.(R.S^2.(2 + \frac{V.X_{MV}}{P.Q}) + (2 + X_p).S'^2), \tag{12}$$

where L represents the pixel rate (in pixel/sec).

These equations indicate that the computation complexity of ME is linearly proportional to the frame rate, frame size, and ME area. They also indicate that the computation complexity is linearly proportional to the number of reference frames.

The power consumption W_e dissipated as a result of encoding a raw video that is captured by a camera is a function of computation



(a) Spatial and Temporal Effects (b) Temporal Effect at 800x600 (c) Spatial Effect at 30 fps

Figure 5: Video Capturing Power Consumption [c_i=2.014.10^{-7} joule/pixel, c_j=1.175 Watt]

complexity of the video encoder X_e. As discussed in Subsection 2.2, the computation complexity of encoding one frame is primarily the sum of the complexities for (i) intra-prediction and inter-prediction, (ii) transformation, quantization, and their inverses, and (iii) entropy encoding. Consequently, as in [10], the encoding computation complexity X_e for one frame is given by

$$X_e = m_{mb}.m_{mec}.X_{mec} + m_{nzmb}.X_{nzmb} + E_e.X_{bit}, \quad (13)$$

where m_{mb} represents the number of macroblocks (MBs) in a frame, m_{mec} represents the number of intra and inter prediction operations per MB, X_{mec} represents the computation complexity of one operation from the intra and inter prediction operations (such as the sum of SAD operations), 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, E_e represents the encoding rate in bit/frame, and X_{bit} is the computation complexity per bit for entropy coding. Note that a nonzero MB is a MB that has nonzero transform coefficients after quantization. Only nonzero MBs will go through the transformation and quantization processes.

For inter-prediction, the first term in Equation (13) can be computed using our ME complexity model given in Equation (12):

$$m_{mb}.m_{mec}.X_{mec} = \frac{X_c}{F}. \quad (14)$$

The complexity for intra-prediction will be analyzed in Subsection 4.2.1.

The average computation complexity X_{ea} per frame for the different frame types of a group of pictures GOP, considering the averages of the number of I, P, B frames in a video segment, can be calculated by averaging X_{ei} , X_{ep}, and X_{eb}:

$$X_{ea} = (i.X_{ei} + p.X_{ep} + b.X_{eb})/(i + p + b), \quad (15)$$

where i, p, b are the numbers in the GOB of I, P and B frames, respectively. The average computation complexity X_e per second, for the different frame types of a group of pictures GOP is

$$X_e = F.(i.X_{ei} + p.X_{ep} + b.X_{eb})/(i + p + b), \quad (16)$$

where F is the frame rate. As in [10], the power consumption W_e for the encoder can be expressed as

$$W_e = c_{eff}.(X_e)^3, \quad (17)$$

where c_{eff} is the effective switched capacitance of a processor with an energy-scaling feature, such as Dynamic Voltage Scaling (DVS) (discussed in Subsection 2.2). Substituting X_e from Equation (16) in Equation (17) yields

$$W_e = c_{eff}.[F.(i.X_{ei} + p.X_{ep} + b.X_{eb})/(i + p + b)]^3. \quad (18)$$

From Equation (18), we notice that the consumed encoding power depends on the video parameters (spatial and temporal resolutions), video content, encoding 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 parameter, and ME range). The video content coupled with the encoding algorithm and parameters (such quantization parameter) impact the number of nonzero MBs (shown in Equation (13)).

Next, we develop power consumption models for of H.264, MPEG-4, and MJPEG.

4.2.1 Modeling for H.264 Encoder

H.264 has high computational complexity, mainly due to its ME, complex prediction, and RDO mode selection [23].

Let us now develop a complexity model for full mode selection, covering both intra and inter predictions. Due to adaptive block sizes, mode selection operates on multiple variable block sizes, different intra prediction modes, and ME vectors. For each macroblock (MB), full mode selection finds the mode combination with the least RDO cost J (discussed in Subsection 2.2) among all possible mode combinations. For a specific MB and a specific mode 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 mode combination and then the mode combination with the minimum cost will be selected for the MB. The whole process is repeated for each MB in the frame.

As an illustrating example, in intra mode selection in H.264, the number of mode combinations for one MB (16×16 pixels) is $N8\times(16\times N4+N16)$, where $N8$, $N4$, and $N16$ 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 $4 \times (16 \times 9 + 4) = 592$ RDO calculations [13]. For inter-mode selection, 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 mode combinations. To select the best mode for one MB in inter-prediction, the encoder performs $16 + 8 + 8 + 4 + 2 + 2 + 1 = 41$ RDO operations.

Intra-prediction and inter-prediction require a total of $592 + 41 = 633$ RDO calculations. Due to this high complexity, inter-mode, intra-mode, and intra/inter mode selections have been active areas of research [12] (and references within).

Based on the aforementioned discussion, the full mode selection complexity X_m can be given as follows:

$$X_m = F.\frac{N \times M}{16 \times 16} \times \sum_{i=1}^{m} (N_{predi} + N_{ei} + N_{di} + N_{Di} + N_{Ji}) + N_{cost}, \quad (19)$$

where m represents number of mode combinations for a MB, N_{predi} represents the number of operations to compute the prediction MB for mode combination i, N_{ei} represents the number of operations to compute the residual, transform, quantization, and entropy coding for mode combination i, N_{di} represents the number of operations to compute the inverse quantization, inverse transform, and reconstruction for mode combination i, N_{Di} represents the number of operations to compute the distortion for mode combination i, N_{Ji} represents the number of operations to compute the cost J for mode combination i, and N_{cost} represents the number of operations to compute the minimum cost among all mode combinations for the MB.

N_{predi} in the case of inter-prediction modes involves ME and thus Equation (12) can be used: $N_{predi} = 4.L.(2.R.S^2 + (2 + X_p).S'^2)$. For intra modes, however, N_{predi} can be found as follows:

$$N_{predi} = N_{Bi}.N_{li} + 2 \times 4 \times N_{ci}, \quad (20)$$

where N_{Bi} represents the number of blocks in the MB, N_{li} represents number of operations to compute the Luma prediction block, and N_{ci} represents the number of operations to compute the Chroma prediction block. Note that there are one Luma and two Chroma components. Since the Chroma block size is 8×8 pixels, a MB will have 4 Chroma blocks in each Chroma component. We develop Tables 3, 4, and 5 to assist in computing N_{Bi}. Tables 3 and 4 show N_{li} for 4×4 and 16×16 Luma blocks, respectively. Table 5 shows N_{ci} for the 8×8 Chroma block. The tables also include a brief description of each intra mode. More detailed descriptions of these modes can be found in [22].

Without unnecessarily detailing their equations, the terms N_{ei}, N_{di}, N_{Di}, and N_{Ji} can be computed by summing their individual components. We develop Table 6 to determine the complexities of various steps in mode selection.

Note that full mode selection captures almost all aspects of H.264 encoding. The developed full mode selection model in Equation (19) can be used to find the total number of operations required by H.264 encoding and also the number of instructions of each type (add, multiply, divide, etc.) as different instruction types may have different execution times (or different numbers of required CPU cycles).

In H.264 encoding, the overall computation complexity is the sum of the intra-prediction (intra estimation) and ME complexity, the transform (and inverse transform), the quantization (and inverse quantization), loop filter, and entropy coding [33, 10]. Let us now discuss how the overall encoding complexity can be modeled in terms of the input video. From Equation (18), we notice that the estimation complexity is a function of the number of MBs and the frame rate, and thus the pixel rate. This is true although we analyze the full search approach, the main parameters that affect computation complexity will not be affected by optimization technique that stop the search early based on some statistics or other algorithms such as fast intra/inter prediction algorithms. The computation complexity of the transform and quantization and their inverses is directly proportional to the number of nonzero MBs in the frame,

Table 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	4×4 copy operations
Mode 1	Horizontal: the left column's samples are extrapolated horizontally	4×4 copy operations
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 + 4×4 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	$4 \times 4 \times$(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

Table 4: Number of Operations to Compute a 16×16 Luma Prediction Block

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

which is directly proportional to the bitrate [19, 30]. The complexity of entropy encoding is directly proportional to the bitrate [10]. 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. Thus, building on Equations (12), (13),(14), (18) and (19), the power consumed by H.264 can be expressed as

$$W_h = (c_2 \times r + c_3 \times L + c_4)^2 \times (c_2 \times r + c_3 \times L), \quad (21)$$

where c_2, c_3, and c_4 are constants, r is the bitrate in $kbit/s$, and L is the pixel rate in $pixel/s$. We include c_4 to better capture the linear relationship between the voltage and the frequency in DVS circuits. As in [4, 24], the voltage (v_{DVS}) and frequency (f_{DVS}) relationship is given by

$$v_{DVS} = c_1 \times f_{DVS} + c_2, \quad (22)$$

where c_1 and c_2 are constants. We notice in Figure 6(b) that the relationship between pixel rate and bitrate is linear, Thus, Equation (21) can be simplified to

$$W_h \approx L \times (c_a \times L + c_b)^2, \quad (23)$$

where c_a and c_b are constants. Figure 6(a) shows that the model is accurate compared with actual experimental results.

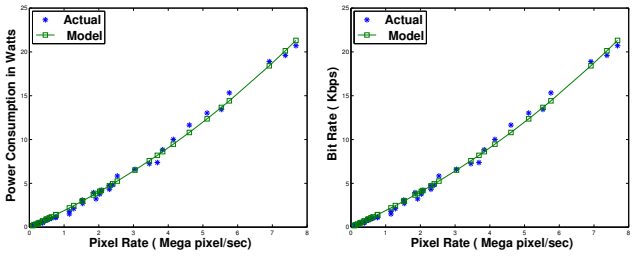
Table 5: Number of Operations to Compute an 8×8 Chroma Prediction Block

Mode	Description	# Operations
Mode 0	Vertical: copy row	8×8 copy operations
Mode 1	Horizontal: copy column	8×8 copy operations
Mode 2	DC: average of 32 values of macroblock	(16-1) add + 1 div + 8×8
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)

Table 6: Complexity of Various Steps in Mode Selection (Per Macroblock)

RDO Step	Description	# Operations
N_{residi}	# ops to compute residual	(4×4 - 1 add)$\times$ $4 \times 4 \times 16$
N_{trfmi}	# ops to compute transform ($Y = AXA^T$): 1 transpose and 2 4×4 matrix multiplications by blocks in macroblock	($2 \times$(4 mul + 3 add)$\times$ 4×4 entries + 16)$\times$16
N_{quanti}	# ops to compute quantization	(4 mul + 3 add)$\times$ $4 \times 4 \times$ 16 entries
N_{enti}	# ops to compute entropy coding	a function of bit rate
$N_{Invquanti}$	# ops to compute inverse quantization	(4 mul + 3 add)$\times$ $4 \times 4 \times$ 16 entries
$N_{Invtrfmi}$	# ops to compute inverse transform $Z = A^T YA$: 1 transpose and 2 4×4 matrix multiplications by blocks in macroblock	($2 \times$(4 mul + 3 add)$\times$ 4×4 entries + 16)$\times$16
$N_{reconsti}$	# ops to compute the reconstructed macroblock	(4×4 - 1 add)$\times$ $4 \times 4 \times 16$
N_D	# ops to compute Distortion and the Sum of Squared Distortion (SSD) between the original and the reconstructed macroblock	($2 \times 4 \times 4$-1 + 4×4)$\times$ 16
N_J	# ops to compute the single cost for the mode combination: $J = t + gr$	(1 add + 1 mul)$\times 4 \times 4 \times$ 16
N_{cost}	# ops to find the minimum cost among all mode combinations for the macroblock	(1 initialize+599 $\times$(1 compare + 1 equal))$\times$ 16

Figure 7 shows the power consumption for H.264 encoding at frame rates of 15, 20, 25, and 30 $frames/s$ with varying spatial resolutions, whereas Figure 8 shows the effect of varying the frame rate on power consumption at different spatial resolutions.



(a) Power Consumption (b) Bitrate

Figure 6: Power Consumption of H.264 Encoding: Spatial and Temporal Effects [c_a=$2.8x10^{-11}$/pixel, c_b=$13.8x10^{-4}$]

Impact of the Number of Reference Frames

As discussed earlier, H.264 enhances the compression ratios by allowing the use of multiple reference frames.

Based on Equations (12), (13), and (18), let us now model the power consumption and the bitrate as functions of the number of reference frames. The number of reference frames impacts ME, the number of nonzero MBs, and entropy coding, with the first being the most significant factor. Neglecting the variations in the other two factors, the power consumption W_R and bitrate r_R can be expressed as

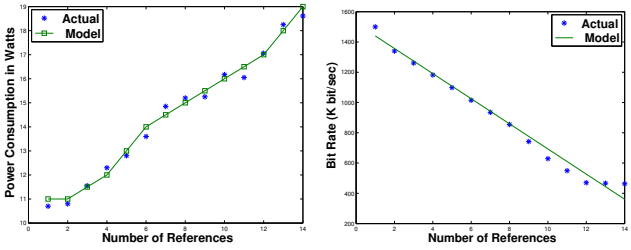
$$W_R = (c_m \times R + c_n)^3 \tag{24}$$

and

$$r_R = b = c_s \times R + c_t, \tag{25}$$

where R is the number of reference frames and c_s, c_m, c_n, and c_t are constants.

Figure 9 compares the actual experimental data with the analytical models and shows that models are reasonably accurate.



(a) Power Consumption (b) Bit Rate

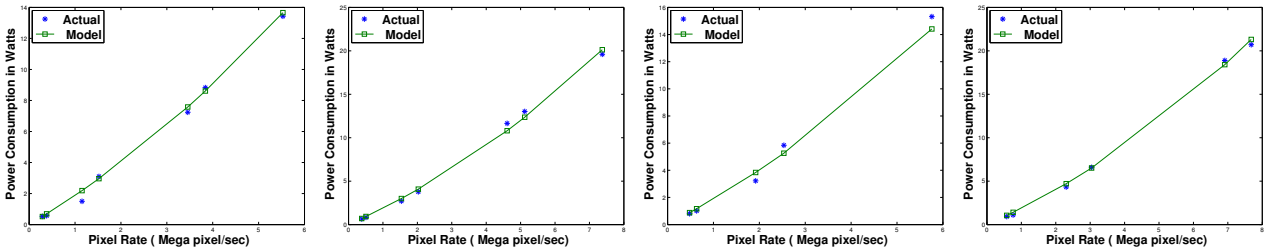
Figure 9: Impact of Number of Reference Frames in H.264 [c_m=0.0333, c_n=2.14, c_s=$-$83.03, c_t=1523.361]

Impact of the ME Algorithm and Range

Let us now analyze the following five ME search algorithms used in the X.264 codec. (1) *Diamond* (DIA)– It is the simplest search, consisting of starting at the best predictor, checking the MVs at one pixel upwards, left, down, and to the right, and picking the best. The process is repeated until it no longer finds any better MV. (2) *Hexagon* (HEX)– It consists of a similar strategy as Diamond, except that it uses 6 surrounding points. (3) *Uneven Multi-Hex* (UMH)– It searches a complex multi-hexagon pattern in order to avoid missing harder-to-find MVs and thus it is slower than HEX. (4) *Exhaustive* (ESA)– It is a highly optimized search of the entire ME search space. It searches every single MV in the area and thus it is slower than UMH. (5) *Transformed Exhaustive* (TESA)– It is attempts to approximate the effect of running a Hadamard transform comparison at each MV and is a little slower than ESA.

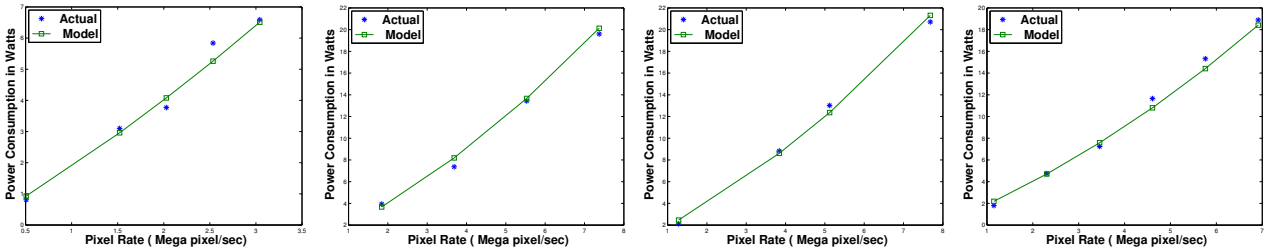
Figures 10(a) and 10(b) compare the five ME algorithms in terms of power consumption and bitrate, respectively. The results can be summarized as follows. (1) DIA yields the least power consumption, but results in the highest bitrate among the five ME algorithms. (2) HEX has the best power consumption and bitrate tradeoff. (3) UMH has higher power consumption and a higher bitrate than DIA. (4) ESA consumes more power than UMH but without a significant benefit in the bitrate for the same quality. (5) TESA consumes a little bit more power than ESA and provides a lower bitrate for the same quality.

The ME range ($merange$) is another important parameter. It controls the maximum range of the ME search in pixels. Increasing the ME range can decrease the bitrate at the expense of significantly slowing down the encoding process and increasing power



(a) 15 fps (b) 20 fps (c) 25 fps (d) 30 fps

Figure 7: Power Consumption of H.264 Encoding: Spatial and Temporal Effects [c_a=$2.8x10^{-11}$/pixel, c_b=$13.8x10^{-4}$]



(a) 352x288 (b) 640x360 (c) 640x400 (d) 768x480

Figure 8: Power Consumption of H.264 Encoding: Spatial and Temporal Effects [c_a=$2.8x10^{-11}$/pixel, c_b=$13.8x10^{-4}$]

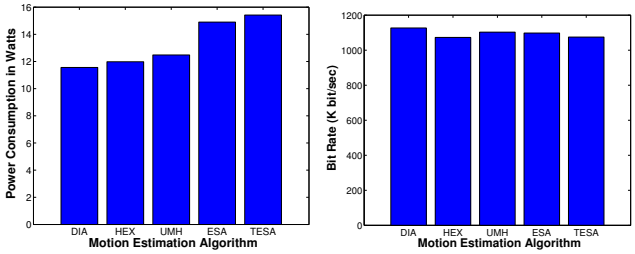
consumption. Figures 11(a) and 11(b) show the effects of the ME range on power consumptions and bitrate, respectively. Based on Equations (12), (13), and (18), we can model the power consumption W_D and the bitrate r_D as functions of $merange$ as follows:

$$W_D = (c_d \times S^2 + c_e)^3 \tag{26}$$

and

$$r_D = c_f \times S + c_g, \tag{27}$$

where S is the ME range.



(a) Power Consumption (b) Bit Rate

Figure 10: Impact of ME Algorithm in H.264

Impact of the Quantization Parameter (QP)

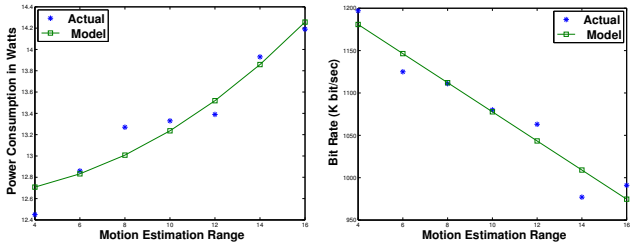
Figures 12(a) and 12(b) show the impact of QP on power consumption and bitrate, respectively. By applying Equations (18) and (22) and using curve fitting, we get

$$W_q = (c_{L1}/qp^c + c_{L3})^2 \times (c_{L2}/qp^c + c_{L4}) \tag{28}$$

and

$$r = (c_{r1}/qp^c), \tag{29}$$

where qp is the quantization parameter and c_{L1},c_{L2}, c_{L3}, and c_{r1} are constants. In Equation (28), we introduce c_{L4} to represent the



(a) Power Consumption (b) Bit Rate

Figure 11: Impact of ME Range in H.264 [TESA Algorithm,c_d = 0.00038, c_e = 2.33, c_f = −17.18, c_g = 1249.5]

ME computation complexity, which does not vary with QP. The complexities, however, of intra prediction, transform, quantization, and entropy coding are inversely proportional to QP and directly proportional to the bitrate. These results, which are validated by extensive experiments, show that QP has great impacts on power consumptions and bitrate. Figures 12(a) and 12(b) show that the models are accurate.

By substituting Equation (29) in Equation (28), we can model the power consumption as a function of the bitrate:

$$W_q = (c_{b1}.r + c_{b3})^2.(c_{b2}.r + c_{b4}). \tag{30}$$

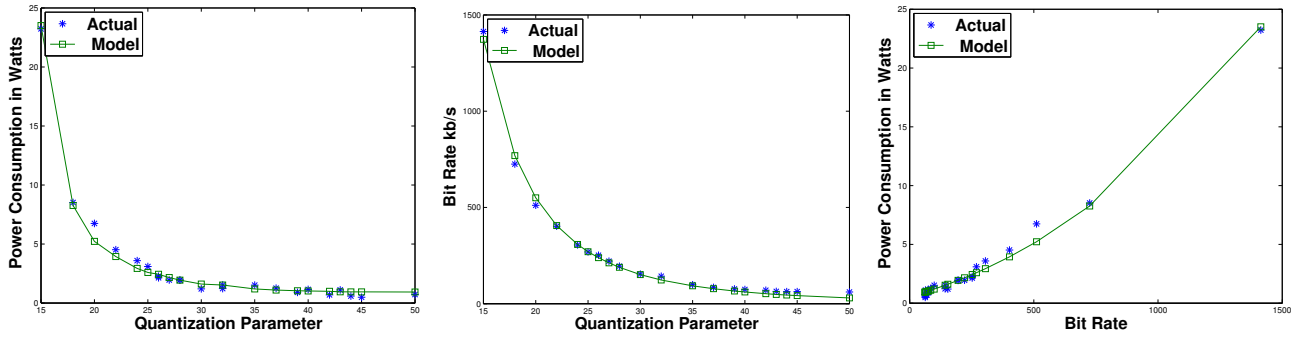
This equation further confirms our aforementioned conclusions.

4.2.2 *Modeling for MPEG-4 Encoder*

Figure 13(a) shows the power consumption in the case of $MPEG$-4 encoding. The model is that same as that of H.264, but with different constants:

$$W_m = L.(c_{ap}.L + c_{bp})^2, \tag{31}$$

where W_m is the $MPEG$-4 encoding power, and L is the pixel

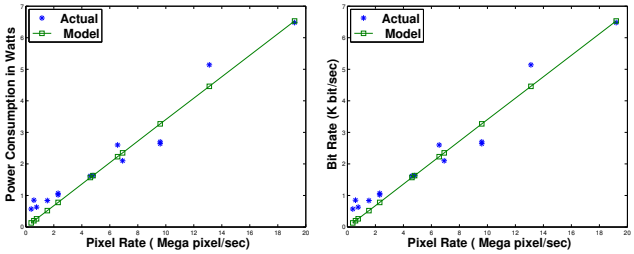


(a) Power Consumption Vs. QP (b) Bit Rate Vs. QP (c) Power Consumption Vs. Bit Rate

Figure 12: Impact of Quantization Parameter in H.264 [(a) c_{L1}=31579.85, c_{L2}=509, c_{L3}=9.595, c_{L4}=0.00555, c=3.18; (b) c_{L1}=31579.85, c=3.18; (c) c_{b1}=0.001078, c_{b2}=0.001084, c_{b3}=2.2637, c_{b4}=0.10517]

rate in $pixel/sec$, and c_{ap} and c_{bp} are constants.

Figure 13(b) indicates that bitrate and pixel rate relationship is linear, with three times the slope of that in the case of H.264.



(a) Power Consumption (b) Bit Rate

Figure 13: Power Consumption of MPEG-4 Encoding: Spatial and Temporal Effects [c_{ap}=$7x10^{-17}$/pixel, c_{bp}=$5.8x10^{-4}$]

4.2.3 *Modeling for MJPEG Encoder*

Figure 14(a) shows the power consumption of MJPEG encoding. In this case, no ME is performed, but intra prediction for spatial compression is carried out in addition to the transform and the quantization. The computation complexity is better shown as a function of the pixel rate rather than the bitrate, although a linear relationship between the bitrate and the pixel rate is observed for high pixel rates as shown in Figure 14(b). Based on Equation (18), the power consumption model of MJPEG can be give as follows:

$$W_j = (c_v.L + c_w)^3, \tag{32}$$

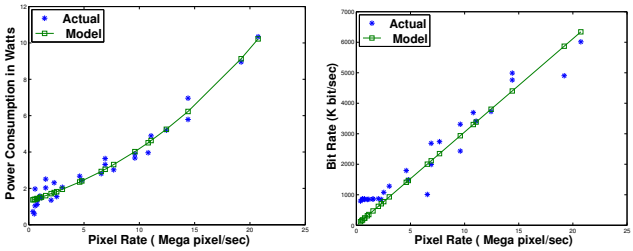
where c_v and c_w are constants, W_j is the MJPEG encoding power, and L is the pixel rate in $pixel/s$.

4.3 Video Transmission Power Consumption Modeling

In the last phase, the video is transmitted to the receiver(s). According to [2, 11, 21], the power W_t consumed in wireless transmission when it is chosen such that the bit error (BER) at the receiver side is very low can be expressed as

$$W_t = (c_x + c_z.d^n).r, \tag{33}$$

where c_x, c_z are wireless model constants, d is the transmission distance, n is the path-loss index, and r represents the transmission bitrate. Equation (33) can be generalized for wired transmission by



(a) Power Consumption (b) Bit Rate

Figure 14: Power Consumption of MJPEG Encoding: Spatial and Temporal Effects [c_v=$0.52.10^{-7}$/pixel, c_w=1.09]

assuming the path-loss index n is zero. Therefore, the transmission power for wired transmission is a special case of (33):

$$W_t wired = c.r, \tag{34}$$

where c is a wire model constant, and r is the transmission bitrate. Equations (33) and (34) indicate that the power consumption of transmitting the video is linearly proportional to the transmission bitrate.

In our experiments using wireless transmission, we confirmed that the model in Equation (33) 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:

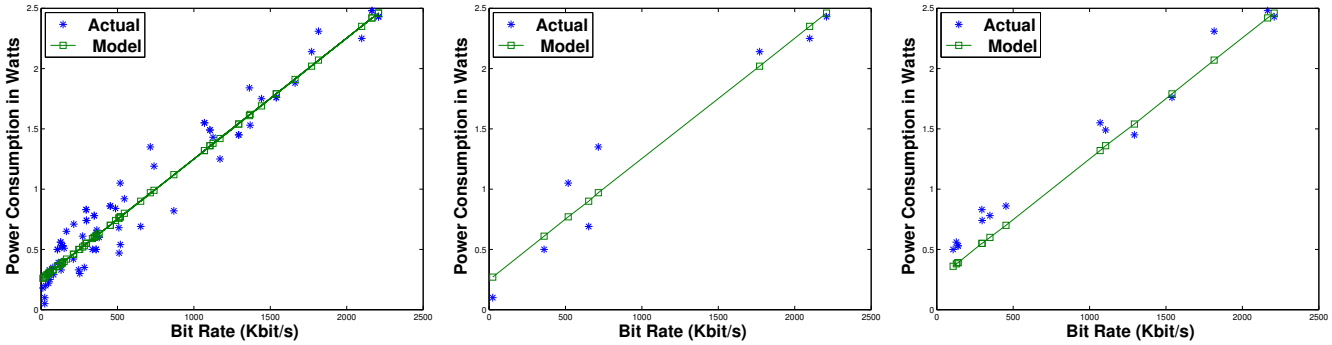
$$W_t = (c_x.r + c_y), \tag{35}$$

where W_t is the wireless transmission power in Watts, and c_x and c_y are constants.

Figure 15(a) validates the model when both the spatial and temporal resolution are varied, whereas Figure 15(b) shows the results when varying only the temporal resolution, and Figure 15(c) shows the results when varying only the spatial resolution. 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 Modeling the Aggregate Power Consumption

Table 7 shows the aggregate power consumption models when



(a) Transmission: Spatial and Temporal Effects (b) Temporal Effect at 800x600 (c) Spatial Effect at 30 fps

Figure 15: Transmission Power Consumption [c_x=0.001 joule/bit, c_y=0.25 Watt]

Table 7: Aggregate Power Consumption Models (Including Capturing, Encoding, and Transmission)

Encoder	Total Power Consumption
H.264	$W_h = L.(c_{ah}.L + c_{bh})^2 + c_{ch}.L + c_{dh}$
MPEG-4	$W_p = L.(c_{ap}.L + c_{bp})^2 + c_{cp}.L + c_{dp}$
MJPEG	$W_j = (c_{aj}.L + c_{bj})^3 + c_{cj}.L + c_{dj}$

each encoder is used. These models include the power consumed in each phase: capturing, encoding, and transmission.

Figure 16 shows that the aggregate power consumption models are highly accurate.

5. CONCLUSIONS

We have analyzed power consumption in the capturing, encoding, and transmission phases of live video streaming systems. In particular, we have developed models for all these phases and have validated them individually and also in terms of the aggregate power consumption. For video encoding, we have developed a model for H.264 standard, considering important factors, such as mode selection, the number of reference frames, and sub-pixel ME search. In addition, we have analyzed the impacts of important encoding parameters on power consumption, such as quantization parameter, number of reference frames, ME algorithm, and ME range. The models have been validated by extensive real experiments.

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. (2) For high spatial and/or temporal resolutions, the video encoding consumes more than 90% of the power, while capturing consumes less than 6% and transmission less than 4%. (3) H.264 consumes more than four times the power consumed by earlier standards. (4) In H.264 encoding, inter and intra predictions consume much more energy than transform, quantization, entropy coding, and decoding. (5) The quantization parameter affects power consumption in an exponential fashion. (6) Other encoding parameters, such as the number of references, search range, sub-pixel search range, and ME algorithm vary the power consumption by up to 10%. (7) Tuning of parameters must be done based on an energy and bitrate tradeoff. (8) The complexities of inter prediction, intra prediction, RDO mode selection, and sub-pixel search are impacted primarily by the temporal and spatial resolutions (pixel rate).

6. ACKNOWLEDGMENT

The authors are grateful to Eman Haidar who helped in running some experiments.

7. REFERENCES

[1] Atheros Communications. Power consumption and energy efficiency comparisons of WLAN products. Tech. Report, available at www.atheros.com, 2004.

[2] M. Bhardwaj and A. P. Chandrakasan. Bounding the lifetime of sensor networks via optimal role assignments. In *Proceedings of IEEE Infocom 2002*, pages 1587–1596, 2002.

[3] C. Blanch, T. Gan, A. Dejonghe, and B. Masschelein. Cross-layer optimization for the scalable video codec over WLAN. In *Proceedings of 17th International Packet Video Workshop*, May 2009.

[4] T. Burd and R. Broderson. Processor design for portable systems. *VLSI Signal Process*, 13(2):203–222, Aug 1996.

[5] A. Dupret, J. Klein, and A. Nshare. A programmable vision chip for CNN based algorithms. In *Proceedings of IEEE International Workshop Chip for Cellular Neural Networks and Their Applications Proceedings*, 2000.

[6] T. Ebrahimi and C. Horne. Mpeg-4 natural video coding: An overview. *Signal Processing: Image Communication*, 1(15):365–385, September 2000.

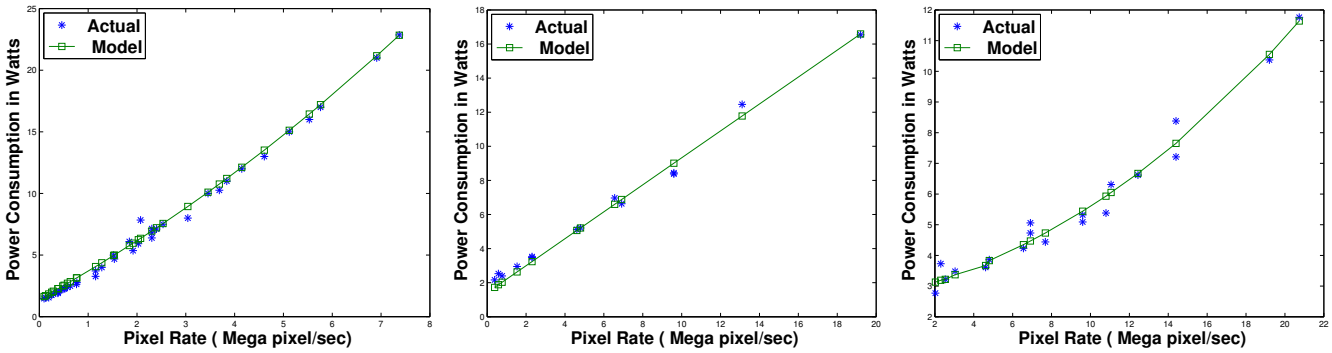
[7] A. Eleftheriadis and D. Anastassiou. Constrained and general dynamic rate shaping of compressed digital video. In *Proceedings of the 1995 International Conference on Image Processing*, 1995.

[8] A. Elouardi, S. Bouaziz, A. Dupret, L. Lacassagne, J. O. Klein, and R. Reynaud. A smart sensor-based vision system: implementation and evaluation. *Journal of Physics D: Applied Physics*, 39(8), Nov. 2006.

[9] B. Erol, F. Kossentini, and H. Alnuweiri. Efficient coding and mapping algorithms for software-only real-time video coding at low bit rates. *CirSysVideo*, 10(6):843–856, September 2000.

[10] Z. He, Y. Liang, L. Chen, I. Ahmad, and D. Wu. Power-rate-distortion analysis for wireless video communication under energy constraints. *IEEE Transaction Circuits System Video Technology*, 15(5):645–658, May 2005.

[11] Z. He and D. Wu. Resource allocation and performance analysis of wireless video sensors. *IEEE Transactions on*



(a) H.264 (b) MPEG-4 (c) MJPEG

Figure 16: Aggregate Capturing, Encoding, and Transmission: Spatial and Temporal Effects [Webcam Pro 9000]

Circuits and Systems for Video Technology, 16(5):590–599, May 2006.

[12] C. Kim and C.-C. J. Kuo. Feature-based intra-/intercoding mode selection for H.264/AVC. *IEEE Transactions on Circuits and Systems for Video Technology*, 17(4):441–453, April 2007.

[13] J. Kim, D. Kim, and J. Jeong. Complexity reduction algorithm for intra mode selection in H.264/AVC video coding. *Advanced Concepts for Intelligent Vision Systems (ACIVS'06)*, 4179:454–465, 2006.

[14] J. Kim, J. Kim, G. Kim, and C.-M. Kyoung. Power-rate-distortion modeling for energy minimization of portable video encoding devices. Aug. 2011.

[15] 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)*, pages 281–284, July 2003.

[16] M. Kim and Y. Altunbasak. Optimal dynamic rate shaping for compressed video streaming. In *Proceedings of the First International Conference on Networking-Part 2 (ICN)*, pages 786–794, 2001.

[17] W. Lin, K. Panusopone, D. M. Baylon, M.-T. Sun, Z. Chen, and H. Li. A fast sub-pixel motion estimation algorithm for H.264/AVC video coding. In *IEEE Transactions on Circuits and Systems for Video Technology, VOL. 21, NO. 2, February 2011*.

[18] X. Lu, T. Fernaine, and Y. Wang. Modelling power consumption of a h.263 video encoder. In *Proceedings of IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 77–80, 2004.

[19] I. Pao and M. Sun. Statistical computation of discrete cosine transform in video encoders. In *Journal of Visual Communication and Image Representation*, June 1998.

[20] W. Pu, Y. Lu, and F. Wu. Joint power-distortion optimization on devices with MPEG-4 AVC/H.264 codec. In *Proceedings of IEEE International Conference on Communications (ICC 2006)*.

[21] T. Rappaport. *Wireless Communications: Principles and Practice*. New Jersey: Prentice-Hall, Inc., 1996.

[22] I. E. G. Richardson. *The H.264 Advanced Video Compression Standrad, second edition, WILEY, Published Online*. Vcodex Limited, UK, 2010.

[23] M. Shafique, B. Molkenthin, and J. 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*, pages 1713–1718, 2010.

[24] T. Simunic, L. Benini, A. Acquaviva, P. Glynn, and G. D. Micheli. Dynamic voltage scaling and power management for portable systems. In *Proceedings of the 38th annual Design Automation Conference*, pages 524–529, 2001.

[25] L. Su, Y. Lu, F. Wu, S. Li, and W. Gao. Complexity-constrained H.264 video encoding. *IEEE Transactions on Circuits and Systems for Video Technology*, 19(4):477–490, April 2009.

[26] Y. H. Tan, W. S. Lee, J. Y. Tham, S. Rahardja, and K. M. Lye. Complexity scalable H.264/AVC encoding. *IEEE Transactions on Circuits and Systems for Video Technology*, 20(9):2010 1271, Septemper 2010.

[27] A. M. Tourapis, O. C. Au, and M. 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.

[28] A. M. Tourapis, F. W. B, and S. L. B. Motion vector prediction with reference frame consideration. In *Proceedings of Applications of Digital Image Processing*, pages 440–1217, 2003.

[29] A. Verdant, A. Dupret, H. Mathias, P. Villard, and L. Lacassagne. Low power motion detection with low spatial and temporal resolution for CMOS image sensor. In *Proceedings of International Conference on Application-Specific Systems, Architectures and Processors*, pages 12–17, 2007.

[30] Y. Wang, Y. Zhou, and H. Yang. Early detection method of all-zero integer transform coefficients. *IEEE Transactions on Consumer Electronics*, 50(3):923–928, 2004.

[31] T. Wiegand, G. J. Sullivan, G. Bjøntegaard, and A. Luthra. Overview of the H.264/AVC video coding standard. *IEEE Transaction on Circuits and Systems for Video Technology*, 13(7), 2003.

[32] X. Xu and Y. He. Improvements on fast motion estimation strategy for H.264/AVC. *IEEE Transactions on Circuits and Systems for Video Technology*, 18(3), March 2008.

[33] C. Zhu, X. Lin, L.-P. Chau, K. pang Lim, H. ann Ang, and C. yin Ong. A novel hexagon-based search algorithm for fast block motion estimation. In *Proceedings of Acoustics, Speech, and Signal Processing*, 2001.