

**CONTENT-ADAPTIVE CROSS-LAYER OPTIMIZED
VIDEO PROCESSING USING REAL-TIME FEATURE
FEEDBACK**

A Dissertation
Presented to
The Academic Faculty

by

Joshua W. Wells

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Electrical and Computer Engineering

Georgia Institute of Technology
May 2017

Copyright © 2017 by Joshua W. Wells

CONTENT-ADAPTIVE CROSS-LAYER OPTIMIZED VIDEO PROCESSING USING REAL-TIME FEATURE FEEDBACK

Approved by:

Professor Abhijit Chatterjee, Advisor
School of Electrical and Computer
Engineering
Georgia Institute of Technology

Professor Ghassan AlRegib
School of Electrical and Computer
Engineering
Georgia Institute of Technology

Professor Justin Romberg
School of Electrical and Computer
Engineering
Georgia Institute of Technology

Professor James Hays
School of Electrical and Computer
Engineering
Georgia Institute of Technology

Professor Arijit Raychowdhury
School of Electrical and Computer
Engineering
Georgia Institute of Technology

Date Approved: 20 December 2016

To my parents, Zeb and Karen, for teaching me to believe in myself.

To Anna, for being a constant source of encouragement.

ACKNOWLEDGEMENTS

Through my graduate studies, I have received support and encouragement from a number of individuals. I owe special thanks to my family, fellow students, and friends. My success is as much due to their help as it is my own efforts.

First and foremost, I would like to thank my parents for instilling in me the belief that I can accomplish anything I set my mind to. Thank you to my partner, Anna, for encouraging me through difficult times. Thank you to my bother, Todd, who filled the roles of roommate and (involuntary) study-mate for much of my undergraduate education. Thank you to my sister, Nichole, for always reminding me that no matter how far I have come, there is always room for improvement.

I would like to thank my advisor, Abhijit Chatterjee, for guiding me towards a research area that was both topical and personally motivating. Thank you to my lab-mates for continual guidance and collaboration over the years. A special thanks to Jayaram Natarajan and Debesh Bhatta for many illuminating and thought provoking discussions.

Finally, I would like to thank my friends for their support. Thank you to Josh Pelkey, Pat Graf, and Dan Guyon for being excellent classmates and even better friends. Thank you to Eric Anger and Chris Shearer for being there to share in both achievements and challenges. I am grateful for my connections made during graduate school that will undoubtedly continue on.

TABLE OF CONTENTS

DEDICATION	iii
ACKNOWLEDGEMENTS	iv
LIST OF FIGURES	viii
GLOSSARY	x
SUMMARY	xii
I INTRODUCTION AND BACKGROUND	1
1.1 Introduction	1
1.2 Video Encoding Background	3
1.2.1 Prediction	5
1.2.2 Transform	7
1.2.3 Reconstruction	8
1.3 Object Tracking Background	9
II PRIOR WORK	11
2.1 Conventional Design Improvements	11
2.1.1 Algorithm Optimization	11
2.1.2 Hardware Specialization	14
2.2 Content Adaptation	16
2.2.1 Algorithm	16
2.2.2 Cross-Layer Optimization	19
2.3 Approximate Computing	20
III ADAPTIVE VIDEO ENCODING	23
3.1 Introduction	23
3.2 Video Sensor Pre-compression	25
3.3 Video Input Feedback Controller	27
3.4 System Dynamic Voltage and Frequency Scaling	29

3.5	Design Analysis	30
3.5.1	video input feedback controller (VIFC) Performance	30
3.5.2	Video Quality	31
3.5.3	Power Savings	32
3.6	Conclusion	34
IV	ADAPTIVE OBJECT DETECTION & TRACKING	36
4.1	Introduction	36
4.2	Proposed System Design	38
4.2.1	Region Partitioning	40
4.2.2	Adaptive Sampling	42
4.2.3	Background Modeling	45
4.2.4	Adaptive Control	53
4.3	Results	56
4.4	Conclusion	63
V	ERROR DETECTION AND MITIGATION IN VIDEO ENCODING HARDWARE	65
5.1	Introduction	65
5.2	PISP Design Details	66
5.3	PISP Design Validation	71
5.4	Prediction Signature Processing	73
5.4.1	Intra Prediction System Design	75
5.4.2	Inter Prediction System Design	79
5.5	Transform Signature Processing	82
5.6	Error Detection and Mitigation	84
5.7	Results	85
5.7.1	Experimental Setup	85
5.7.2	Experimental Results	87
5.8	Conclusion	91

VI CONCLUSION	95
6.1 Conclusions	95
6.2 Future Work	98
REFERENCES	100

LIST OF FIGURES

1	Generic motion-compensated hybrid video encoder	4
2	Generic motion-compensated hybrid video decoder	5
3	Proposed video encoder design depicting integration of video input feedback controller, dynamic voltage and frequency scaling controller, and dynamic image sensor	24
4	Multiresolution quadtree structure	27
5	Hypothetical multi-resolution image	28
6	Natural image model	30
7	Prediction accuracy for decimation by a factor of 2	31
8	Decoded video test sequence quality	32
9	Information content of each pre-processed frame relative to non-preprocessed information content	33
10	Estimated dynamic power savings	35
11	Proposed object tracking system architecture	40
12	Hypothetical quadtree configuration	41
13	Hypothetical quadtree configuration—future frame	42
14	Hypothetical quadtree configuration with full-resolution layer	43
15	Weighted learning coefficient	47
16	System scaling response and performance for a single test video sequence	57
17	Average throughput for each test sequence measured in pixels per frame	58
18	Processor time vs frame index	59
19	Object detection area comparison	60
20	Mean squared error of the segmentation mask relative to conventional processing	61
21	Predicted computational complexity compared to observed computational complexity	62

22	Adaptive object detection example showing background model (a), sample scene with object present (b), and quadtree detection structure superimposed on sample scene (c). Colors on the yellow end of the spectrum indicate high probability of an object being present while colors at the blue end of the spectrum indicate the opposite. . .	62
23	Ideal PISP theoretical overview with linear systems. Signature processing produces a signature matching the output of macroblock (MB) processing.	67
24	Realizable PISP Overview. Differential systems, \mathbf{D}_i , are used to provide missing information to signature processing system.	68
25	Proposed video encoder design	70
26	Prediction Signature Processing Model Overview	73
27	Inter prediction signature processing with coordinates in terms of \mathbf{u} for simplicity.	80
28	Transform signature processing design. Two sources of rounding error are captured, inverse transformed, and added to the residual signature to form the reconstructed residual signature.	82
29	Mean signal quality (relative to original image) of PISP method and conventional method for various IDR periods	87
30	Mean quality of various QP values for PISP and conventionally encoded frames with an IDR period of ∞	88
31	Mean execution time of PISP method and conventional method for various IDR periods	89
32	Mean bit-rate of PISP method and conventional method for various IDR periods	90
33	Mean quality of aggregated video sequences with varying BER as a function of time with a fixed quantization parameter (QP) of 16 and a IDR period of ∞	91
34	Qualitative comparison of conventional inter prediction encoding (a–e) and PISP inter prediction encoding (f–j) subjected to $BER = 10^{-6}$	92

GLOSSARY

ASIC	application-specific integrated circuit.
BER	bit error rate.
CAVLC	context-adaptive variable-length coding.
codec	coder-decoder.
CPU	central processing unit.
CS	compressive sampling.
CSTIS	CMOS separable transform image sensor.
DCT	discrete cosine transform.
DSP	digital signal processor.
DVFS	dynamic voltage and frequency scaling.
DVFSC	dynamic voltage and frequency scaling control.
EE	entropy encoder.
FPGA	field-programmable gate array.
GPP	general-purpose processor.
HEVC	High Efficiency Video Coding.
IDCT	inverse discrete cosine transform.
IDR	instantaneous decoder refresh.
IE	intra estimation.
IP	intra prediction.
LBC	linear block code.
MB	macroblock.
MC	motion compensation.
MCHVD	motion-compensated hybrid video decoder.
MCHVE	motion-compensated hybrid video encoder.
ME	motion estimation.

MoG	mixture of Gaussians.
MP	multi-processor.
MPSoC	multi-processor system-on-chip.
MV	motion vector.
nint	nearest integer.
PCCS	pre-compression camera sensor.
PCF	pre-compressed frame.
PE	processing element.
PISP	parallel independent signature processing.
PSNR	peak signal-to-noise ratio.
PVE	predictive video encoding.
QoS	quality of service.
QP	quantization parameter.
RAM	random access memory.
SAD	sum of absolute differences.
SDRAM	synchronous dynamic random access memory.
SIFT	scale-invariant feature transform.
SMP	symmetric multiprocessor.
SoC	system-on-chip.
SURF	speeded-up robust features.
VIFC	video input feedback controller.
VLC	variable-length code.

SUMMARY

The objective of this research is to design a low-power video processing system capable of minimizing power consumption through graceful reduction of the quality of the processed signal. Methods for post-processing signal analysis are proposed for determining the presence and quality of features essential to the high-level application goals. Two applications, video encoding and object tracking, are selected for study based on their high demand on mobile platforms. Methods for scaling the complexity of the entire system are proposed by simultaneously scaling down the input signal and algorithms to focus computational effort on information salient to the application. A cross-layer control system is proposed for determining the optimal complexity scaling and dynamic voltage and frequency scaling. The control system will be charged with providing dynamic voltage and frequency scaling control information for accurate prediction of imminent throughput requirements, allowing for minimal power consumption while narrowly achieving the high-level application goals.

CHAPTER I

INTRODUCTION AND BACKGROUND

1.1 Introduction

Demands of popular video processing applications, such as video encoding and object tracking, are continually creating design challenges for hardware development. Increasing resolution and reduced operating power are the primary competing challenges. Conventional design methods, including device scaling, application-specific integrated circuit (ASIC) design, and algorithm optimization, are encountering limitations preventing continued improvement. For many video encoding applications, a significant portion of the sampled information is trivial to the output signal in the context of the application. If the relevant information in the input signal can be identified prior to processing, a substantial amount of energy may be saved with a controlled impact on signal quality. This research identifies methods for identifying application salient features in video signals and proposes adaptive video processing designs that reduce power and throughput while maximizing signal quality within constraints.

State-of-the-art adaptive methods for video processing typically analyze each frame of an input signal before it is processed. This feedforward approach can allow the system to estimate the amount of computational effort required to process the frame and scale the capabilities of the hardware to match, saving power. However, due to the magnitude of information in the input signal, the types of analyses performed are restricted to being relatively simple. Furthermore, the analyses performed on the input signal, though they may help reduce the number of operations performed on the signal while processing, do not necessarily align with the goals of the high-level

application.

The proposed research adapts the video processing algorithm and hardware to the content of the input signal in an effort to minimize power consumption while meeting the essential goals of the application. Features aligned with the goals of the high-level application being performed on the video signal are used as feedback to control the resolution and processing method for the video signal. The presented results demonstrate the benefits of analyzing post-processed features of the signal to control the distribution of computational effort in the input signal. The results show the ability of the system to concentrate computational effort and power on regions of saliency, reducing effort for non-salient regions. This draws on the principle that the salient information is often a fraction of the total information that is sampled and processed. The goal of the research is to determine features of processed signals that can be used as a quality metrics for use in determining how computational effort should be used, based in the high-level goals of the application. Control systems are proposed with the goal of optimally trading quality for power, providing just enough quality to meet predefined constraints.

Advances in image sensing technology are leveraged for the proposed dynamic methods of data processing. The image sensor described in [61] demonstrates a design for sensing in images directly using any 2-D separable basis. Similarly, [17] demonstrates design methods for block-based compressive sampling (CS) for images. These types of variable sensing methods allow for the algorithm to extend control to the sensor itself.

Two popular video processing applications are selected as a basis for the presented research: video encoding, and object detection and tracking. The goal of video encoding is to compress a sampled video signal by finding redundant information in the signal and then describing the signal in a way that reduces the redundant information. The goal of object detection and tracking is to identify regions in a video

scene that represent objects of interest and determine how the objects are moving from one frame to the next. Both of these applications attempt to filter out ambient information relative to the goals of the application. Video encoding filters out redundancies, and object detection and tracking filters out non-important background information leaving basic information about the objects present in the video. The presented research demonstrates adaptive methods for handling each of these applications types and further demonstrates methods of video encoding error tolerance to preserve quality in the face of continued hardware device scaling.

The remainder of this chapter provides background on the two considered video processing applications. A background on MCHVEs is described to provide the reader with the necessary information to understand design challenges and proposed solutions in Chapters 3 and 5. Likewise, a background in methods for object detection and tracking is necessary to understand the research presented in Chapter 4.

1.2 Video Encoding Background

There are many types of coder-decoders (codecs) and associated video encoders. The class of video encoders considered in this research is the MCHVE, due to its popularity. MCHVEs use previously coded information in the video stream as a basis for describing later information in stream. The previously coded basis information may come from part of the current frame being encoded, or from part of some previous frame. Encoding frames based on previously coded information exploits the high spatiotemporal correlation of video signals to effectively compress the signal. Rather than sending each sampled coefficient, the difference between some sampled information and some previously coded information can be signaled, along with a *prediction mode* that describes how the basis information should be obtained from the previously coded video signal. After a video stream has been encoded, a decoding operation can be performed that reconstructs parts of the video sequence, iteratively

using reconstructed parts to, in turn, reconstruct future parts of the video sequence. In general, the compression is lossless, meaning the reconstructed pixel values are not guaranteed take on the original, sampled values, but the reconstructed values will be close.

A more specific video encoder architecture is shown in 1. Each frame to be encoded is divided into a discrete number of non-overlapping, contiguous blocks, typically called MBs. Each MB contains the same number of samples (pixels) and each MB is encoded individually (but not necessarily independently). The MBs, are generally processed in raster order within each frame. Each sampled MB, $\mathbf{b}_s[m]$, is compressed by the prediction and transform systems to produce a sparse block of coefficients, $\mathbf{b}_{enc}[n]$. These coefficients are reordered and compressed to a bit-stream format by an entropy coder, however, the presented research focuses on elements of the video encoding system that are independent of the entropy coding process. Therefore, the entropy encoder is omitted for simplicity.

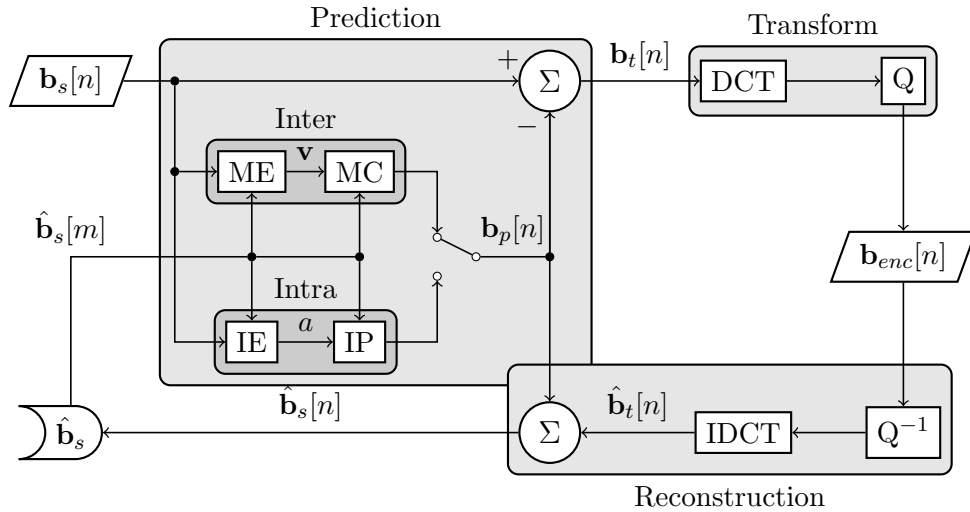


Figure 1: Generic motion-compensated hybrid video encoder

Each block of the sampled video is processed in three primary stages of the video encoder: prediction, transform, and reconstruction, as shown in Figure 1. The prediction system primarily removes temporally redundant information from blocks as

they are processed, and the transform system reduces spatially redundant information. The reconstruction operation performs the inverse functions of the prediction and transform systems, effectively producing a block that closely resembles the original sampled block, $\mathbf{b}_s[n]$. Each encoded MB is transmitted to the decoder(s) along with instructions (\mathbf{v} or a from Figure 1) indicating modes of operation for decoding.

The decoding process, shown in Figure 2, performs an inverse transform followed by the adding of a prediction of type signaled by the encoder. The result is the reconstructed block, $\hat{\mathbf{b}}_s[n]$, which will differ slightly from the original block, $\mathbf{b}_s[n]$, due to quantization error induced by the encoder. The decoder effectively implements the exact reconstruction operation used in the encoder. The prediction system, transform system, and reconstruction system are discussed in more detail in the Sections 1.2.1, 1.2.2, and 1.2.3, respectively.

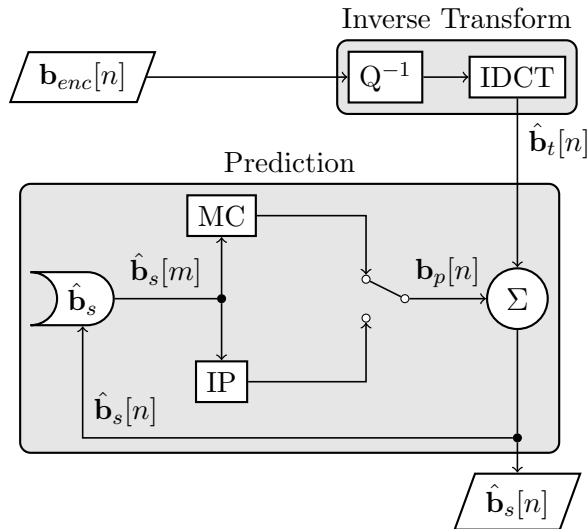


Figure 2: Generic motion-compensated hybrid video decoder

1.2.1 Prediction

The prediction system depicted in 1 takes a sampled MB and subtracts from it some similar prediction based on previously encoded and reconstructed MBs. The prediction system functions by evaluating a number of pre-defined prediction modes and

selecting the mode that produces the greatest compression ratio (or another desirable metric) [79]. There are two primary types of modes for prediction: intra prediction and inter prediction. As indicated in Figure 1, either intra prediction or inter prediction can be selected and intra estimation (IE) or motion estimation (ME) will be used to select the best performing mode. In either case, previously encoded MBs are used as a basis for describing the MB currently being encoded. Inter prediction uses MBs encoded in previous frames while intra prediction uses previously encoded MBs in the current frame. After a prediction method has been selected, the selection is signaled via a (intra prediction mode) or \mathbf{v} (inter prediction motion vector) to the motion compensation (MC) or intra prediction (IP) system depending on which prediction mode is being used. Either the MC or IP system will create the prediction, \mathbf{b}_p , which will be subtracted from the sampled MB to produce the residual MB, \mathbf{b}_t . Ideally, the prediction will be very similar to the sampled block, creating a residual block with minimal energy. This same prediction will be added back after the residual MB has been transformed and inverse transformed.

The Inter prediction system performs two primary operations, ME and MC. The ME system shown in Figure 1 searches areas of the previous frame for a group of contiguous pixels that is highly correlated to the current MB being encoded. This search is necessary to account for moving objects in the video. An independent search is performed for each block to be encoded. Given the sheer volume of data, this is the most computationally intense operation of the entire encoder [40]. The location of the best matching area from the previous frame is described with a vector, \mathbf{v} . Once the best vector is selected, it is used as input to the MC system to formally produce a prediction, $\mathbf{b}_p[n]$. The prediction is effectively the reconstructed pixel values at the frame and spatial location indicated by \mathbf{v} . When MB is encoded into a bitstream, the motion vector is included to signal to the decoder how to generate the correct prediction.

The Intra prediction system performs two operations similar to the two in the Inter prediction system. The IE system searches for good inter prediction modes based on previously encoded blocks in the current frame. Generally, all prediction modes are tested before selecting one. Specifically, immediately adjacent pixels of MBs located to the immediate left, above, and above left are available as sources for the predictions. Predictions generally project border pixel values across the block at various angles or average the information from this set of pixels to produce estimates of the pixel values of the block currently being encoded. The variable a indicates the predefined prediction mode selected. Like with a motion vector in inter prediction, the prediction mode is combined with the encoded MB data when it is ultimately encoded to a bitstream.

The result of both inter prediction and intra prediction is a predicted MB that, hopefully, is similar to the sampled MB. This prediction is subtracted from the sample MB data, reducing the overall energy in the MB which is helpful in terms of compression for the following stages of encoding.

1.2.2 Transform

The transform system is responsible for transforming the MB signal into a likely sparse domain to reveal opportunities for compression. The coefficients are also quantized to further increase compression.

In general, the transform used on the MB is a two dimensional frequency based transform. In the case of the H.264 [1] standard, each MB is usually divided into sub-block of size 4×4 and then a 2-D discrete cosine transform (DCT) based transform is applied to each sub-block. Other transform options exist, such as additionally using a Hadamard transform on a portion of the coefficients in certain cases. In general, a transform similar to the 2-D DCT is used on the MB.

Next, the transformed coefficients are quantized by the system, Q . Ideally, a

number of the higher-frequency coefficients will be quantized to 0, and effectively resulting in compression. In general, 0-quantized coefficients are not specifically signaled to the decoder. The non-zero coefficients are finally encoded using an entropy encoder, which has been omitted for simplicity. When combined with the prediction mode (\mathbf{v} or a), the decoder has all necessary information to reconstruct each MB. The extent to which coefficients are quantized is determined by the user or control algorithm as a quantization parameter.

The output of the transform system is a MB with sparse coefficients. The larger coefficients are likely to be the ones that represent lower spatial frequencies. Therefore, the final, hidden stage of the encoding process is to reorder the coefficients based on frequency magnitude and then perform entropy encoding on the reordered coefficients to exploit the sparsity of the signal.

1.2.3 Reconstruction

The reconstruction operation performs the same exact operation that will be performed on the decoder. It performs an inverse quantization operation, an inverse discrete cosine transform (IDCT), and a prediction summation. The reconstruction is necessary to produce the decoded block, $\hat{\mathbf{b}}_s$, that will be potentially used for creating future predictions.

The first step in the reconstruction process is to perform an inverse quantization (or scaling) of the coefficients to produce coefficients similar to those output by the DCT operation of the transform system. The coefficients will not be exactly the same because of the original quantization operation. The coefficients can only be scaled back close to their original values. The extent of the error induced by quantization depends on the the quantization parameter selected during the encoding process.

After block coefficients have been inverse quantized, each MB is inverse transformed, relative the the transform type(s) used in the transform system. The output

of the inverse quantization process is the residual estimate, $\hat{\mathbf{b}}_t$, a reconstruction of \mathbf{b}_t . Again, due to quantization, the two blocks will not necessarily be equal.

The same prediction subtracted from the original sampled pixels is added back to $\hat{\mathbf{b}}_t$ to produce $\hat{\mathbf{b}}_s$. This represents the final block that will be stored in memory and used in the future to predict future blocks. This block may be used immediately for intra prediction, or it may be used some time later for inter prediction.

The reconstructed block, $\hat{\mathbf{b}}_s$, will probably not be exactly the same as \mathbf{b}_s due to the quantization operation and the fact that the type of compression being performed is lossy. However, the reconstructed values should match the reconstructed values of the decoder exactly. This is, in fact, the reason for the reconstruction operation in the encoder. The decoder will also be using previously decoded blocks as a basis for decoding future blocks. Therefore, this basis needs to be identical on the encoder and decoder to guarantee that quantization differences do not accumulate in decoded blocks. This concept is referred to as decoder drift and implies that the reference memory of the decoder drifts from the reference memory of the encoder, causing degradation in signal quality.

1.3 Object Tracking Background

A large number of object detection and tracking methods have been proposed and refined over time. Of the proposed methods, two are of particular importance: point detection and background modeling [82]. These methods may be used independently or together, depending on the needs of the high-level application.

Point detection finds interest points within an image based on various qualities such as texture changes associated with boundaries of objects. The scale-invariant feature transform (SIFT) algorithm [48] is popular and very robust, however, it has a high computational cost and memory requirement making low-power, real-time operation difficult. In the interest of reducing the computational complexity of the

algorithm, many alternatives have been proposed. Matching methods described in [59, 18, 48, 25, 58] trade matching accuracy for matching complexity. Multiple methods for speeding up the Gaussian blur function, a necessary step in the SIFT algorithm, were proposed in [26, 19, 5, 80]. Further improvements in speed were proposed in [15], allowing for steps of the Gaussian blur function to be performed in parallel, however, this does not necessarily allow for a decrease in algorithmic complexity.

Background modeling algorithms allow for segmentation between a background scene and foreground objects within the scene. Efficient background subtraction can reduce the amount of work performed on each frame of a video sequence for object detection and tracking. Although some improvements in background modeling have been made, conventional background modeling algorithms have the same order of computational complexity [55]. Therefore, one of the most popular methods for performing background subtraction is one of the most robust methods: the mixture of Gaussians (MoG) method [67, 68]. Using this method, each pixel of the background scene is modeled using a number of Gaussian distributions. An observed value that does not fall into one of the background distributions is determined to be part of a foreground object. Due to the modeling of individual pixels, computation can easily exceed the capabilities of hardware platforms. Performing object detection using the MoG method (with other conventional methods being slightly less) requires more than 12 GOPS to process a 720p video at 30 fps [55]. A novel solution to the problem proposed by Cevher et al. [7] uses compressive sensing to perform background subtraction. The existence of an object can be detected directly in the CS [21] domain and when an object is present, a reconstruction is performed to produce the background segmentation mask. This method has the potential to greatly reduce computational complexity depending on the video input.

CHAPTER II

PRIOR WORK

Previous work in low-power video processing is divided into three primary categories according to the method of power savings: conventional design improvements, content adaptation, and approximate computing. The first refers to the expected, incremental improvement of existing designs through algorithm optimization and more efficient hardware designs using conventional design techniques. Content adaptation refers to the ability of a system to modify its operation based on the content of the sampled signal. Approximate computing methods trade quality and precision for power savings. Each of these research areas is reviewed in the following sections.

2.1 Conventional Design Improvements

As the demand for efficient video processing capability has increased, technological advances have been made through the creation of more efficient algorithms and more specialized hardware. In the following subsections, methods for iterative algorithm improvement are reviewed, followed by a review of advances in specialized hardware development for video processing.

2.1.1 Algorithm Optimization

Dynamic power consumption can be reduced by simplifying the algorithmic complexity for a video processing system. Algorithm improvements for video encoding have concentrated primarily on the mode selection functions, which are responsible for the majority of the computational effort of the encoder. Algorithm improvements in object tracking, on the other hand, have been realized by the introduction of generally new methods. Each of these topics are discussed in detail in Section 2.1.1.1 and

2.1.1.2, respectively.

2.1.1.1 Video Encoding

MCHVEs consist of two primary stages: prediction and transformation. The prediction stage produces a prediction of a group of sampled pixels based on previously encoded pixels. The transformation stage transforms the difference between the prediction and the sampled pixels using an energy compacting transform such as the DCT. In modern standards such as H.264 [1] and High Efficiency Video Coding (HEVC) [2], the complexity of the transformation algorithm has been greatly reduced by the introduction of an efficient, integer-based transform [49]. However, the prediction stage continues to grow in complexity as new standards define additional prediction modes.

The prediction mode selection process is historically the most computationally complex operation performed by video encoders. Specifically, ME has been the most computationally complex operation in the video encoder [40]. Initial algorithms performed a full search, evaluating each possible motion vector (MV). More advanced algorithms reduced effort by concentrating on efficient exploration of the mode space to quickly find a near optimal MV. The three-step search, proposed by Koga et al. in 1981, narrowed the search range with each of the three iterations of the algorithm [39], comparing the performance of nine different candidate MVs at each stage. A similar, more generalized algorithm using five comparisons with a variable number of stages was proposed the same year by Jain et al. [34]. Improvements to these methods were made in [66, 24, 44, 60, 47, 23, 85]. All proposed search algorithms proposed a patterned search with a fixed amount of computation to deliver a good MV. Tsai and Hang [72] later proposed an algorithm combining the methods of previous these heuristic algorithms to create a systematic algorithm with near optimal performance.

The most recent advances in reducing ME algorithmic complexity focus on modifying search methods based on statistical information related to the sampled video signal. This state of the art research is discussed in Section 2.2.1.

2.1.1.2 Object Detection

A large number of object detection and tracking methods have been proposed and refined over time. Of the proposed methods, two are of particular importance: point detection and background modeling [82]. These methods may be used independently or together, depending on the needs of the high-level application.

Point detection finds interest points within an image based on various qualities such as texture changes associated with boundaries of objects. The SIFT algorithm [48] is popular and very robust, however, it has a high computational cost and memory requirement making low-power, real-time operation difficult. In the interest of reducing the computational complexity of the algorithm, many alternatives have been proposed. Matching methods described in [59, 18, 48, 25, 58] trade matching accuracy for matching complexity. Multiple methods for speeding up the Gaussian blur function, a necessary step in the SIFT algorithm, were proposed in [26, 19, 5, 80]. Further improvements in speed were proposed in [15], allowing for steps of the Gaussian blur function to be performed in parallel, however, this does not necessarily allow for a decrease in algorithmic complexity.

Background modeling algorithms allow for segmentation between a background scene and foreground objects within the scene. Efficient background subtraction can reduce the amount of work performed on each frame of a video sequence for object detection and tracking. Although some improvements in background modeling have been made, conventional background modeling algorithms have the same order of computational complexity [55]. Therefore, one of the most popular methods for performing background subtraction is one of the most robust methods: the MoG method

[67, 68]. Using this method, each pixel of the background scene is modeled using a number of Gaussian distributions. An observed value that does not fall into one of the background distributions is determined to be part of a foreground object. Due to the modeling of individual pixels, computation can easily exceed the capabilities of hardware platforms. Performing object detection using the MoG method (with other conventional methods being slightly less) requires more than 12 GOPS to process a 720p video at 30 fps [55]. A novel solution to the problem proposed by Cevher et al. [7] uses compressive sensing to perform background subtraction. The existence of an object can be detected directly in the CS [21] domain and when an object is present, a reconstruction is performed to produce the background segmentation mask. This method has the potential to greatly reduce computational complexity depending on the video input.

2.1.2 Hardware Specialization

The performance of video processing hardware has evolved as demands have increased for performance and power efficiency. Conventional general-purpose processors (GPPs) are not well suited for processing the large volume of data associated with video processing applications, especially when power needs to be minimized. GPPs consume power well above the design constraints of mobile, energy constrained devices [4]. Improvements of conventional designs have been made by developing specialized hardware for specific video processing functions (such as ME) or by developing specialized hardware for an entire video processing system (such as an entire video encoder). Prior work in the conventional hardware specialization entire systems and for modules are discussed in this section.

MPEG-4 Encoder Designs A review of hardware architectures for MPEG-4 [22] is discussed by Tseng et al. in [73]. Full system designs discussed in the review include [70, 57, 27, 53, 3]. The proposed designs are capable of performing video encoding in

the range of 60-160 mW using custom architectural designs for the MPEG-4 codec. These designs specialize in creating ASICs for frequently used encoder functions of high complexity, such as ME.

H.264 Encoder Designs With the introduction of the H.264 standard [1], a large increase in complexity was introduced from MPEG-4 [13]. Chien et al. introduced a system level hardware design featuring accelerated sum of absolute differences (SAD) architecture for faster ME in addition to ASICs designs for primary H.264 functions (integer and fractional ME, intra prediction, entropy coding, and deblocking) with dedicated buses linking the functional units and dedicated memories. The design more closely resembles the system design of the video encoder and saves power minimizing inefficient, generalized hardware such as large, shared buses and large shared memories. The design was further refined by Huang et al. [32] and Chen et al [10, 11] to be able to trade off quality for power consumption by introducing a specialized memory hierarchy and a pipelined design with specialized processing elements.

SIFT Designs There have also been hardware developments for object detection and tracking. Chiu et al. proposed system-level hardware design for implementing a modified version of the SIFT algorithm in [15]. Aligning the hardware to the algorithm allowed for real-time, power-efficient operation.

Motion Estimation Designs In general, algorithmic advances in ME have proven more successful than hardware [73]. The few attempts at hardware acceleration of ME are designed to be used with the full search algorithm and are impractical. Notable advances have been made through cross-layer optimization, which is discussed in Section 2.2.2.

Intra Encoder Designs Prior work in intra prediction hardware has produced ASIC designs for accelerating the intra prediction process. Huang et al. [33] proposed a hardware design that optimizes memory access and pipelines the prediction process. Later, Kuo et al. [41] proposed a similar hardware design capable of performing intra prediction for 1080p video sequences in real-time, also using pipelining.

Video Memory Designs The massive amount of data collected and processed by video processing systems is a key challenge for low-power operation. Stored pixel information may be accessed frequently and often with different patterns, encouraging the design of specialized memory architectures. Kim and Park [37] proposed an array address-translation technique capable of improving synchronous dynamic random access memory (SDRAM) access energy by minimizing the number of row activations and number of commands on the address bus.

2.2 Content Adaptation

Adapting video processing algorithms and hardware to the characteristics of the video signal being processed is one of the more recent advances in the field. This section reviews methods of adaptation for algorithms, hardware, and cross-layer optimization of both.

2.2.1 Algorithm

Content-adaptive algorithms are highly researched in video encoding systems as a method to help reduce computational complexity. There are four primary categories of adaptive algorithms for video encoding in the literature: ME, inter mode selection, early skip mode detection, and energy budgeting. The research in each of these categories simplifies a complex search process based on characteristics of the input video signal. Either by analyzing the signal before encoding is performed, or by analyzing encoded information, algorithms are modified at run-time to give preference

to more likely coding modes.

Motion Estimation Adaptive ME has been a very important area of research with the increasing complexity of encoding standards. In general, adaptive ME algorithms attempt to select search methods that perform better statistically with the current video signal. In [56], Ng et al. discovered that some common search algorithms perform better than others when there is significant motion in a frame. The authors proposed an method for switching between search patterns based on the video content. Search patterns are adapted as more information is gathered, allowing for high accuracy and reduced computation. Similarly, Tsai and Hang proposed an adaptive switching technique using two different search patterns [72]. Additionally, the authors discuss a method for adaptively adjusting the starting point for each search and a method for terminating searches early. A different form of early termination is also discussed in [16]. The authors propose an a search algorithm for the HEVC standard that adaptively avoids fine-grained ME searches when the search is unlikely to produce further refinement. The constraint placed on the search granularity is based on previous observations. Algorithmic complexity is reduced as a result, while the coding performance incurs a minor, temporary penalty. Additionally, Shen et al. [65] describe an HEVC ME algorithm that adapts to the video signal content to reduce the complexity of the algorithm using spatiotemporal correlation.

Inter Mode Selection Adaptive inter mode selection has also been an important area of research for video encoding. Rather than selecting between better performing search patterns and algorithms, prediction modes are ranked in terms of probability given characteristics about the video sequence. Shafique et al. [64] propose an adaptive mode selection algorithm that excludes both intra and inter prediction modes for H.264. Modes are excluded by analyzing the properties of incoming frames and

analyzing the selected QPs of previously encoded frames. Similarly, with the introduction of the HEVC standard [2], a new opportunity for ME complexity reduction was found. Correa et al. [16] proposed a complexity reducing method by constraining the tree depth—an added feature of the HEVC standard—to be searched when regions exhibited stationary characteristics. Many of the additional MV signaling modes can be ruled out under certain circumstances.

Early Skip Mode Detection With the introduction of H.264 standard [1], a skip mode was introduced where no MV is specifically indicated and the most likely MV based on other, nearby MVs is used. Various methods of early skip mode detection were proposed in [78, 83, 42, 84]. Early detection of a skip mode allowed ME operations to be bypassed completely for some blocks. If the default, skip mode motion vector performs above some threshold, other MV candidates are ignored and the default MV is accepted.

Energy Budgeting Energy budgeting has been proposed in the literature as a method for maximizing encoded video quality under power constraints. Energy budgeting methods attempt to predict which regions of each sampled frame will contain relatively high and low amounts of work and budget the limited computational resources to spatial regions such that the quality of the processed signal is maximized. He et al. described a power-rate-distortion model in [28] that can determine the appropriate bit rate a particular video should be encoded at to cause power consumption at a specific rate. The video signal and system are modeled so that quality can be traded for energy. In [63] and [46], an adaptive ME framework is proposed that budgets ME effort based on the anticipated needs of each MB. Both methods assign MBs to classes associated with varying ME computational need. Frames and MBs are categorized into classes based on prior performance and textures. Based on the amount of available energy and user imposed constraints, an ME energy budget

is calculated and distributed across MBs based on the anticipated need of each MB, allowing for the graceful degradation of video quality under a given energy budget. More recent improvements have been published that allow adaptation of the ME algorithm to optimize quality given an energy constraint [45]. The algorithm adapts to the content of the video signal to determine which blocks are likely to require more effort.

2.2.2 Cross-Layer Optimization

Cross-layer optimization for video processing is a new and important area of research in the literature. Attention is given to the co-optimization of multiple design parameters spanning multiple conventional design layers. The result is power saving designs that are more oriented to applications. In this section, examples from the literature discussing general video processing designs, video encoding designs, and object detection and tracking designs are reviewed.

Low-power, cross-layer optimized designs for video processing focus on minimizing power consumption through changing operating parameters across design layers. Often, this is performed online. Using audio processing as an example, Jun and Jones [36] presented a method for allowing a signal processing system to fall into a very low-power state and progressively trigger more complex (higher power) states as signals of interest are identified. Muthukaruppan et al. [52] proposed a design method for multi-processor system-on-chips (MPSoCs) applicable to video processing that explores the energy locus of four design parameters: DVFS, processor customization, cache customization, and task mapping. Similarly, Javaid et al. [35] present a pipelined MPSoC video processor that is capable of continually adapting its architecture and processing capability to match the complexity of the video encoding task. Each frame is analyzed to predict the amount of processing that will likely be needed and the proper amount of processing elements are reserved. Part way

through the encoding of each frame, its progress is checked against the model and further adjustments are made to the architecture. Processing elements are activated and deactivated using clock/power gating.

There have also been video encoding specific advances in cross-layer, low-power optimization. The H.264 encoding system proposed by Chen et al. [11] demonstrates cross-layer optimization by allowing the power management system to monitor the progress of the algorithm and clock gate processing elements when they are idle.

Recent cross-layer advances in object detection are found in the literature. Chippa et al. [14] proposed a cross-layer, adaptive processor for the purpose of recognition and mining. The proposed application of the design is image processing—specifically, image recognition and segmentation. The design is scalable at the algorithm, architecture, and circuit levels.

2.3 Approximate Computing

Approximate computing methods save power by trading off accuracy of processed results. In general, a controlled amount of error is allowed in the hardware, typically as a result of voltage over-scaling. Various methods are proposed in the literature for the detection and mitigation of the tolerable errors. This works well with video processing applications as there is often a wide range of acceptable results in terms of processed, output signals. Four general areas of application of approximate computing are discussed from the literature: digital signal processor (DSP), functional units/processing elements, video memory, and video encoding.

A general DSP degradable filter is proposed by Hegde et al. [30, 31]. In this research, a voltage over-scaled filter design is presented that uses an adapting model for the output signal to check for errors caused by voltage over-scaling. When the computed result diverges from the model, an error is signaled and the expected, modeled value is used instead of the computed value.

Functional unit scalability was proposed by Narayanan et al. [54] to allow for less precise, degraded operation while consuming less power. In this research, redundant functional units are available to produce degraded calculations on data at a variable supply voltage. The proposed scalable units provide graceful degradation of results with decreasing voltage. If coupled with the appropriate control, the system can adapt to operate at reduced power at the cost of reduced quality. When exact results are required, the processor can be signaled to use a conventional functional unit. Similarly, Mohaptra et al. [50] proposed low-level functional units with scalable accuracy. As the voltage is decreased below the threshold voltage, the accuracy of the functional units is reduced. With an appropriate model and control method, the functional units can be used to reduce power at the expense of quality.

Research has also been performed regarding the storage of video information. Memory access is a significant consumer of power in video processing systems. Chang et al. [8] proposed a hybrid memory architecture suitable video applications where higher order luma bits were stored in 8T cells and the remaining pixel information was stored using 6T cells. Power savings were realized by aggressively scaling the voltage of the memory. Applications resilient to exact pixel representation can use the proposed design to trade accuracy for power savings.

Video encoding specific approximate computing research is present in the literature. Huang et al. [33] proposed a software-based solution that increases throughput by dropping the least two significant bits of pixel values and performing multiple lower resolution differences with a single instruction issue. Additionally, blocks are subsampled to reduce computational complexity. When coupled with the appropriate hardware platform, the proposed method provides a coarse method of power savings at the expense of accuracy.

There are many examples in the literature of ME power savings through approximate computing. As early as 2000, He et al. [29] proposed obtaining less accurate

MV by using pixels with reduced accuracy. The less significant bits of the pixels were dropped to allow for a lower-power computation. Later, Varatkar and Shanbhag [77] proposed ME using algorithmic noise-tolerance. The voltage of the ME system is overscaled to the point that errors are induced in the results. Large errors are detected and corrected algorithmically. Similarly, Dhoot et al. [20] proposed a low-power ME method where reliable and voltage over-scaled hardware are present. The reliable hardware processes a small, central search area, while the over scaled hardware operates on a larger search area. The search range adapts to the input signal and performance of the system, allowing for power savings.

CHAPTER III

ADAPTIVE VIDEO ENCODING

3.1 Introduction

The goal of video encoding is to compress a video signal by reducing the amount of redundant information contained in it, increasing entropy. Modern video encoding standards such as H.265, H.264, and MPEG-4 are capable of producing excellent compression, but at the cost of high power consumption in the video encoder. Performing real-time video encoding with limited energy supply is a major issue, particularly for mobile camera systems.

Prior work in low-power video encoding has concentrated on improving the energy efficiency of specific processing elements (PEs) within the video encoder system. The ME kernel typically accounts for 60-80% of the computational load of a video encoder [40]. Accordingly, prior works in low-power video compression have focused on the ME kernel or other individual PEs to conserve energy [9, 38, 81, 53, 51, 12, 75, 76, 77]. Notably, in [75], an error-tolerant, voltage over-scaled ME design is proposed. Past work has been effective at either reducing the workload of the ME kernel or reducing its voltage directly at the expense of video precision. Although meaningful work has been done to increase performance of the various PEs of a video encoder, little progress has been made to *reduce the power consumption of the encoder as a system*.

The proposed design *reduces the power consumed by the entire video encoding system by extending algorithm control to the image sensor sensor*. Rather than processing all data captured by the camera, *raw data is pre-compressed at the sensor* without direct knowledge of the actual entropy of the signal, resulting in a smaller signal being presented to the core encoding system. There are three major areas of

focus of the proposed design:

- Multi-resolution frame description
- Video Input Feedback Controller
- Dynamic Voltage and Frequency Scaling Control

Integrated into the typical video encoder design as shown in Figure 3, each of these elements performs an important role in pre-compressing a video signal and reducing system power consumption.

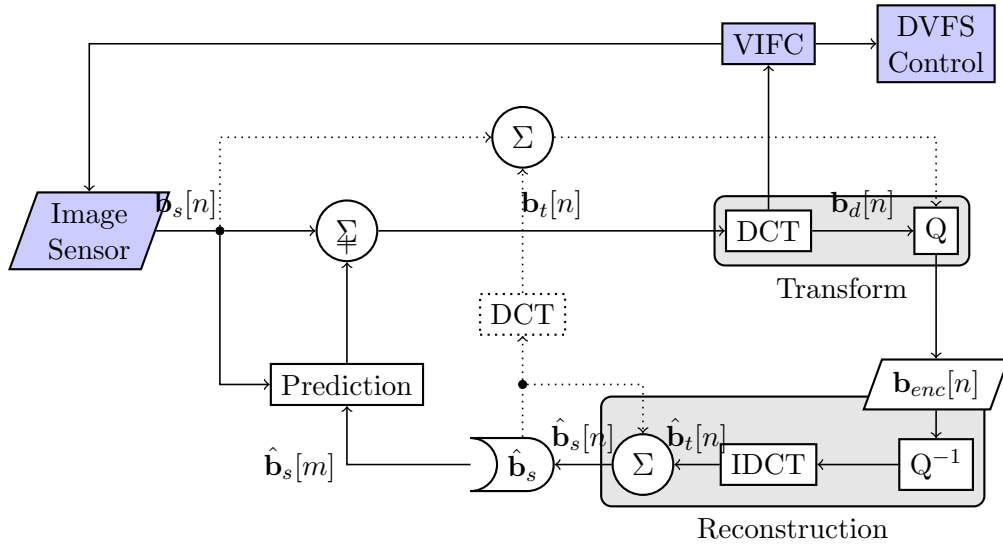


Figure 3: Proposed video encoder design depicting integration of video input feedback controller, dynamic voltage and frequency scaling controller, and dynamic image sensor

Pre-compression of the signal is performed (for certain frame regions) directly on the camera sensor by sensing and transforming the signal together [61]. Instead of the camera producing a typical full-resolution frame, it outputs a multi-resolution representation of the frame. Regions with suspected low entropy are sensed with lower resolution while more dynamic, high-entropy regions are sensed with higher resolution without loss of information or loss of detail in the decoded video sequence.

The VIFC dynamically predicts the entropy of future frames based on previous, encoded frames. The VIFC outputs a control signal for the camera sensor that describes the multi-resolution frame configuration. Resolution levels are constrained by a quadtree structure that guarantees complete, non-overlapping frame coverage.

The dynamic voltage and frequency scaling control (DVFSC) is responsible for controlling the voltage and clock frequency of all core PEs of Figure 3. The DVFSC changes the DVFS of each PE before each frame is encoded. This is possible and necessary because the timing requirements for each frame change based on its content. The less entropy there is in a frame is, the smaller the size of the pre-compressed frame (PCF) is. With less information to process by the core PEs of the encoder, the slower the information can be processed and still meet the timing restrictions of the frame rate. Each PE is scaled so that the video encoder takes just under the frame period to encode each PCF. The result is a video encoder that is capable of large reductions in throughput and power based on the content of the video.

3.2 Video Sensor Pre-compression

With conventional video encoding methods, after a given block has been processed by the prediction and transform systems, the redundant data is revealed as low-magnitude high-frequency coefficients. Through quantization, these coefficients are eliminated from the encoded signal. The elimination of these coefficients is equivalent to applying a spatial low-pass filter to the residual block. The effect of typical video encoding can be described by

$$X[k, l] = \text{DCT}\{b_s[m, n] - b_p[m, n]\} \cdot H[k, l] \quad (1)$$

where x , b , b' , and h are the compressed block, raw input block, reference block, and low-pass filter respectively. The same result can be achieved by first decimating the raw input from the camera as represented by

$$X[k, l] = \text{DCT}\{b[m, n] * h[m, n] - b'[m, n] * h[m, n]\} \quad (2)$$

To perform the pre-compression before the signal is passed through the prediction and transform systems, an estimate of the magnitude of the DCT coefficients is required which will be covered in Section 3.3.

Using CMOS separable transform image sensor (CSTIS), the pre-compression of blocks can be moved all the way to the input sensor itself [61]. Pre-compressed blocks that have been sensed in this way are already described in the DCT domain. This means the typical digital DCT function of the encoder can be bypassed. However, a consequence of sensing a pre-compressed block in the DCT domain is that normal ME and MC are no longer possible. This is of little consequence because as will be shown in Section 3.3, pre-compressed regions of frames are relatively static and can be assumed to have null motion vectors.

Non-contributing high-frequency content is removed before the video signal is passed to the temporal model by sensing the signal in the DCT domain for low frequency coefficients only. If a sensed block is allowed to have a completely variable number of low frequency coefficients included, blocks of varying sizes will be produced which requires an undesirable, complicated design for the PEs of the encoder. To mitigate this problem, effective decimation of blocks is limited to a quadtree structure as shown in Figure 4 allowing the PEs to remain unchanged. For a block to be effectively decimated by the smallest increment, it must be concatenated with three of its neighbors according to the quadtree structure and then decimated by a factor of 2. For example, in Figure 4, blocks A, B, E, and F can be combined and decimated by a factor of 2 to yield block Q. Higher levels of decimation are allowed as long as they adhere to the quadtree structure meaning that all sibling blocks must be collapsed to a single parent. For example, in Figure 4, blocks A-P can be combined and decimated by a factor of 4 to yield block U. The quadtree structure allows the input frame to be represented as a multi-resolution image where areas predicted to have more high-frequency residue can be represented with more resolution and other

areas with less as illustrated by Figure 5. Each block bounded by the black lines in the figure represents the same amount of information – a single block. To make viewing easier, the blocks have been interpolated to fill the whole area they represent. Blocks that occupy a larger area like the one in the top left quadrant contain less detailed information.

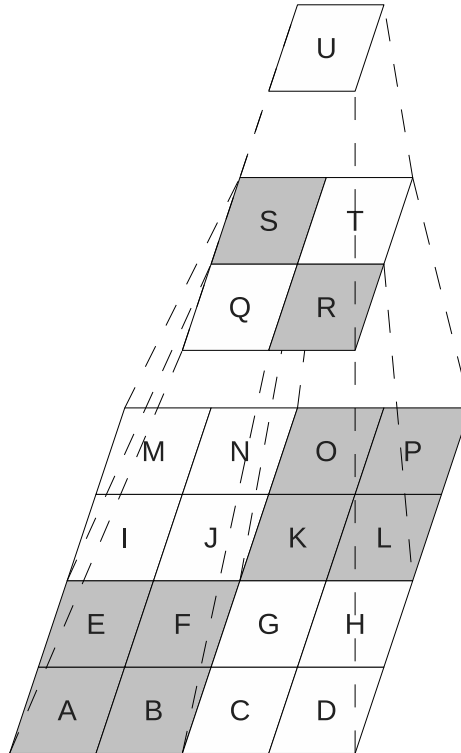


Figure 4: Multiresolution quadtree structure

3.3 Video Input Feedback Controller

The pre-compression camera sensor (PCCS) described in Section 3.2 requires a control signal indicating the quadtree structure that should be applied to the next frame. This is accomplished by analyzing the spectrum of each encoded block in the current PCF. A model is fit to the spectrum of the block that predicts the amount of extra detail or missing detail in the residual block. Based on the model parameters, a decision is made to increase, decrease, or keep the resolution level associated with the

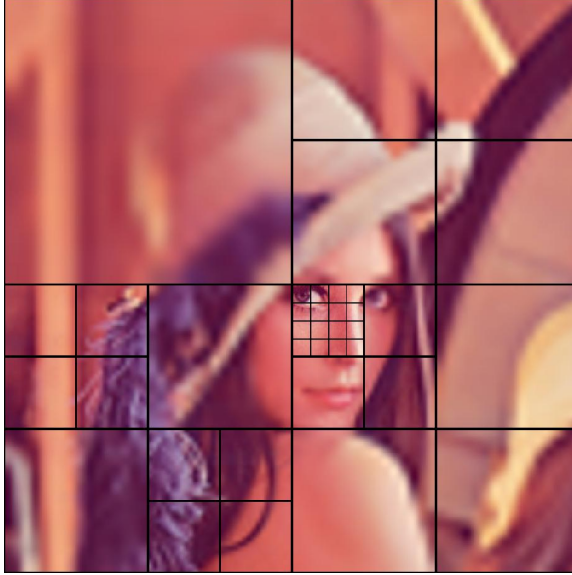


Figure 5: Hypothetical multi-resolution image

corresponding area of the next frame.

The power spectrum of a natural image can be modeled using an exponential function of the form

$$|P_X(j\omega)| = \beta(\omega + 1)^\alpha \quad (3)$$

where β and α are the parameters for the model and ω is the spatial frequency [74]. The expected magnitude of the spectrum decays exponentially with respect to increasing frequency as shown in Figure 6. Once the two parameters for the model are determined, the spectrum can be extended beyond sampled frequencies to determine how much error is predicted to exist in each block. In the figure, ω_c represents the cutoff frequency of a sampled region of a multi-resolution frame. The shaded high frequency coefficients are considered error and can be measured as peak signal-to-noise ratio (PSNR). The controller attempts to keep the PSNR of the signal at a specified level by adjusting the cutoff frequency so that it produces an error just below the predefined goal.

Given the coarseness of available values for ω_c and the fairly limited range of expected values for parameters α and β , pre-computing PSNR values and storing

them in a look-up table is possible. This way, the overhead of the proposed control can be reduced by avoiding repeated calculations. The goal of the image resolution control kernel is minimize overall resolution on the input image while maintaining a given PSNR.

3.4 System Dynamic Voltage and Frequency Scaling

The PCCS modulates the size of the signal for each frame presented to the video encoder. The video encoder also has a fixed frame rate. Since the size of the input signal is modulated, but the amount of time needed to process the signal is static, the video encoder may process smaller frames (less information) at a slower speed without error. Next, DVFS is used to scale back the power consumption of the encoder for all blocks (except for the control and decimation blocks). When the voltage and frequency are scaled down, PE delay time scales up. Based on the hardware implementation of the encoder, a table is computed with optimal voltage and frequency values for different frame sizes.

The DVFSC kernel is responsible for setting the proper voltage and clock frequency for each PE in the video encoder just prior to the encoding of a frame. Transistor-level simulations of each processing element reveal the relationship between scaled voltage, frequency, and delay on the critical path. In each PE, a static amount of work is required for each block processed. For a given critical path delay, the amount of time required to process a single block is known. The VIFC sends a control signal to the DVFSC indicating the number of blocks to be encoded in the next frame. The DVFSC uses a lookup table to translate the block count into a voltage and frequency level for each of the PEs and sets the voltage just prior to the encoding of the frame. Essentially, voltage and frequency are scaled down as the block count is scaled down from its largest count (full resolution.)

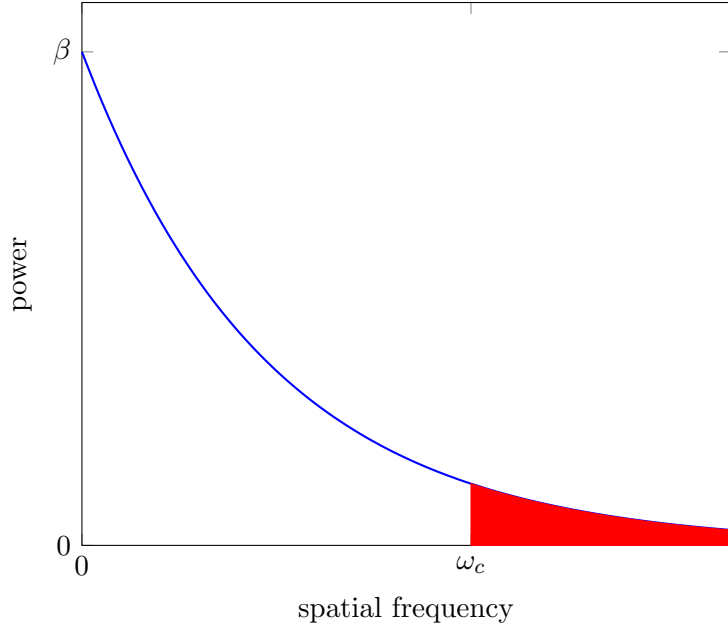


Figure 6: Natural image model

3.5 *Design Analysis*

Three topics must be addressed regarding the validation of the proposed design. First, the VIFC prediction accuracy must be measured. If the VIFC is not capable of producing valid predictions, either power consumption or video quality will suffer. Second, the video quality must be analyzed. When a less than nominal resolution level is predicted, error will be introduced into the encoded signal which will affect the quality of the decoded sequence. Lastly, the power savings from the proposed technique must be evaluated. The savings will depend on the actual video content, so multiple test video sequences are used.

3.5.1 VIFC Performance

As discussed in Section 3.3, the VIFC predicts how much information in a transformed block is missing or alternatively, unnecessary. To evaluate the accuracy of the predictor, a sampling of frames is taken from the test video sequences and indiscriminately decimated by a factor of 2, meaning that groups of four blocks are combined

into a single block. Prediction blocks are subtracted from the decimated blocks and the resulting residual blocks are transformed and passed to the VIFC. The VIFC predicts the amount of missing information in each residual test block. The directly computed (correct) amount of error in each block is compared to the predicted error, measured in PSNR as defined by

$$PSNR = 10 \cdot \log_{10} \left(\frac{MAX_I^2}{MSE} \right) \quad (4)$$

$$MSE = \frac{1}{MN} \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} |x_1[m, n] - x_2[m, n]|^2$$

where MAX is the maximum unsigned integer possible using 8 bits; $x_1[m, n]$ and $x_2[m, n]$ are the two signals being compared. The difference between the two values represents the prediction error. A histogram of the VIFC performance is shown in Figure 7. The results indicate that the proposed VIFC design is valid.

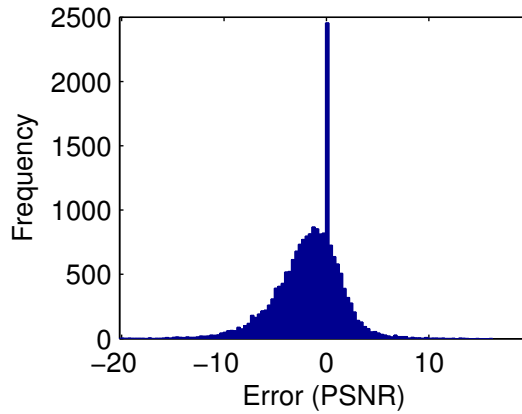


Figure 7: Prediction accuracy for decimation by a factor of 2

3.5.2 Video Quality

Well known test video sequences in the signal processing community were used to evaluate the proposed design in terms of quality. Each raw test sequence was dynamically pre-compressed, simulating the method described in Section 3.2. The reduced signal was processed as depicted in Figure 1 with the exception of bypassing the quantizer.

The quantizer is bypassed to avoid mistaking quantization error for pre-compression error. Each frame is decoded and evaluated against the original, raw test sequence. In each test signal, the VIFC is tasked with predicting the resolution for each region of the frame that will maintain a PSNR of at least 35dB. The resulting PSNR of the test sequences are shown in 8. The VIFC does an excellent job at maintaining the required signal quality. It should also be noted that modern video encoding methods are capable of producing an encoded sequence with a quality similar to the proposed encoder, but no power savings are achieved with modern methods.

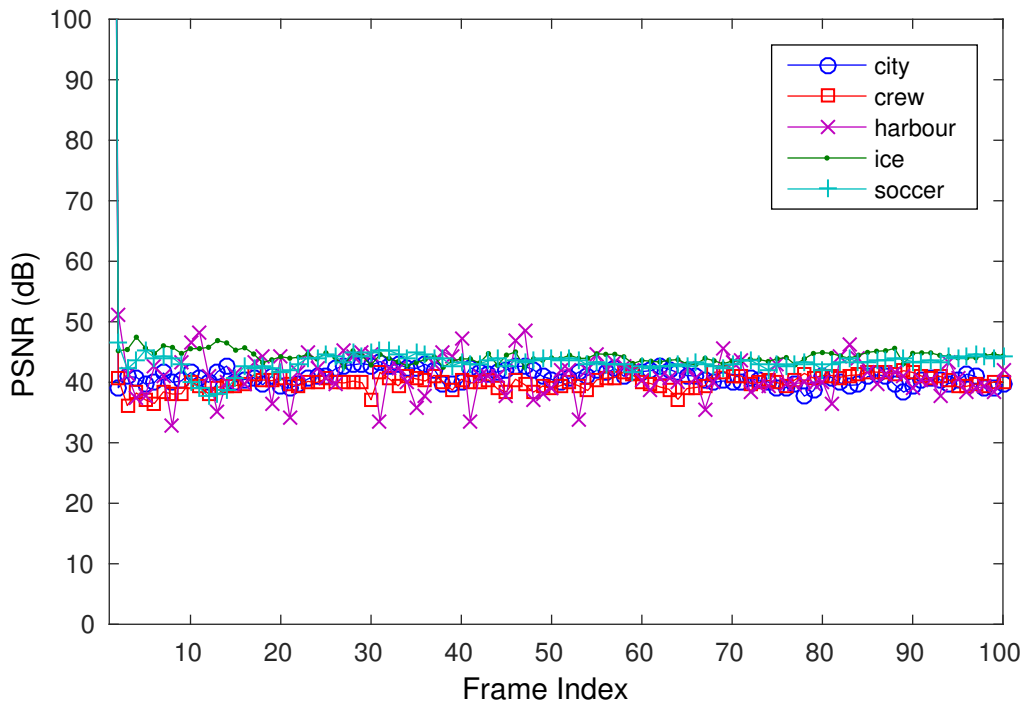


Figure 8: Decoded video test sequence quality

3.5.3 Power Savings

Modern video compression standards do not specify how a signal should be encoded. Instead, they specify what criteria the encoded signal must meet. Therefore, many different hardware implementations have been developed, and as a result, it is very difficult to quantify power savings in terms of Watts. Instead, for the purposes of

this research, estimated power savings are described in terms of the amount of power that can be reduced compared to nominal operation.

Power savings starts with pre-compression. Pre-compression alone is capable of saving dynamic power by reducing the digital signal size and therefore reducing the amount of switching in the circuit for each frame. The extent of the reduction in switching depends on the content of the video. If there is a large amount of temporal and spatial redundancy in the signal, the VIFC will perform well and pre-compression will also be effective, necessary switching in the digital circuit. For each frame in the test sequences, the size of the pre-compressed signal is shown relative to the size of the typical, non-pre-compressed signal. Figure 9 shows relative frame sizes. Due to the difference in the content of each sequence, a different amount of work is required for different frames while the actual quality of each encoded signal remains constant as shown in Figure 8.

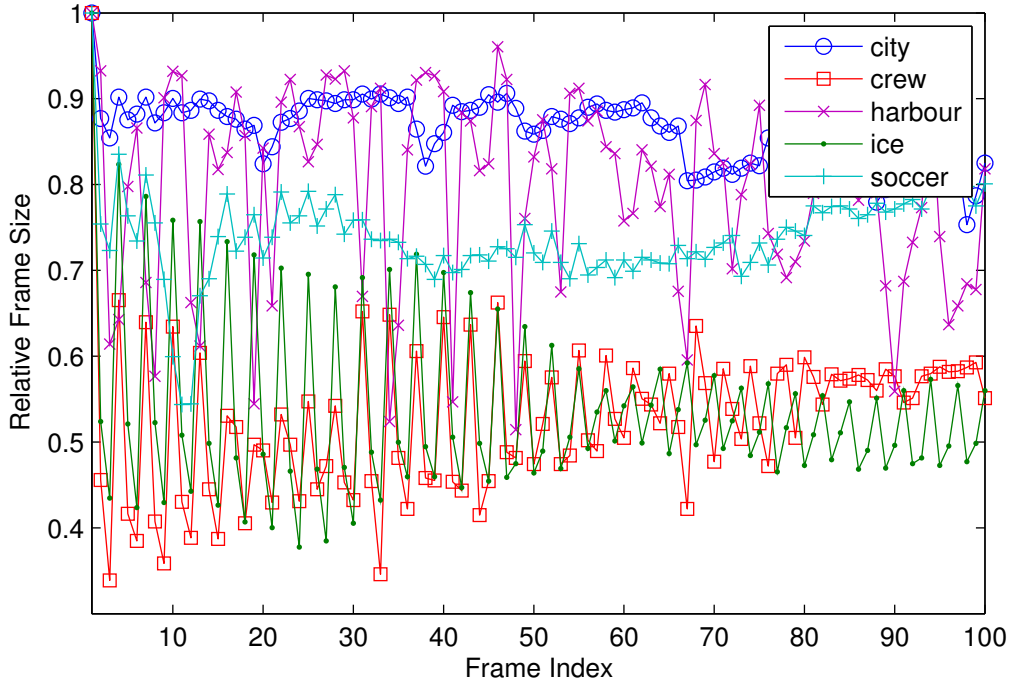


Figure 9: Information content of each pre-processed frame relative to non-preprocessed information content

Although the reduced signals result in reduced switching which decreases the dynamic power consumption of the system, power utilization is not maximized until DVFS is used to reduce the voltage for each PE. As described in Section 3.4, fixed frame rates allow the proposed encoder to perform DVFS on each PE in the encoder. Since there is no standardized encoder design, nominal power savings using DVFS for dynamic power are generalized to an estimated power relative to typical non adaptive encoding. The relative frame size is defined as

$$\gamma = \frac{S}{S_{max}} \quad (5)$$

where S is the frame size of an adaptively acquired frame as shown in Figure 9, and S_{max} is the size of the frame if it were acquired normally (full resolution.) The relative dynamic power of an adaptive encoder can now be computed as

$$P_{Rdyn} = \frac{(V\gamma)^2 f \gamma}{V^2 f} = \gamma^3 \quad (6)$$

where V is the nominal voltage of a non adaptive encoder and f is the average switching frequency. From (6), estimated power results for the sample video sequences are computed. The results are shown in Figure 10. The presented estimates are ideal. They do not account for incremental steps in DVFS but as the number of allowed levels are increased, the savings should approach the computed estimates.

3.6 Conclusion

The presented theory and results demonstrate the potential in power savings for real-time embedded video encoding systems. But to achieve such large savings, the system needs to be considered as a whole. Trying to improve the efficiency of each PE has reached its limits. Although the overall success of the video encoder depends on power efficient operations, the ultimate savings comes from reducing the number of operations needed through the entire system. For a real-time video encoder to truly use minimal power, it must adapt to the content of the video being processed. All

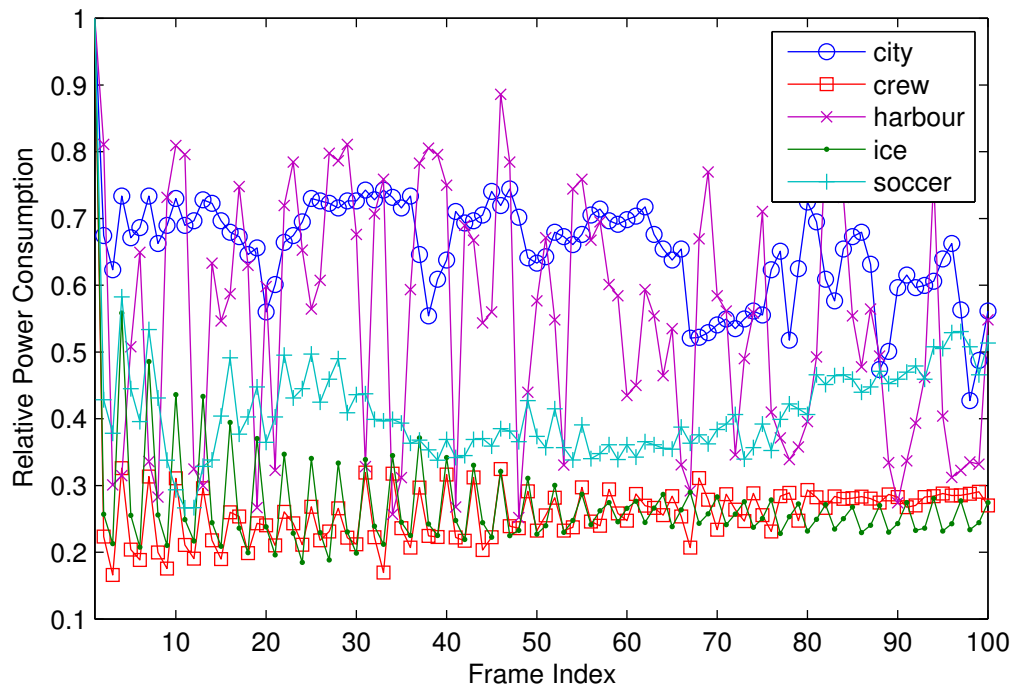


Figure 10: Estimated dynamic power savings

parts of the video encoder system (sensors, algorithm, and hardware) must be able to adapt. The proposed design is capable of dynamically scaling the amount of work performed and power consumed throughout the system so that just enough energy is used to meet the required goals.

CHAPTER IV

ADAPTIVE OBJECT DETECTION & TRACKING

4.1 Introduction

State-of-the-art methods for fixed-camera object detection capture maximal information and perform algorithms that are of constant complexity. This is the case even when objects are not present in the field of view. The goal of the proposed system design is to adaptively reduce power consumption when object detection work can be performed easily (i.e. there are no/few objects present). As the number of objects present in the field of view increases, the proposed system responds with more computing resources and greater power consumption.

There are many proposed methods for object detection for the purposes of object tracking [82]. Popular among them is background subtraction. Analyzing what has changed in a scene with a fixed camera is a relatively good method for detecting objects. The conventional method for performing a background subtraction involves modeling each pixel position using a mixture of Gaussian distributions [67]. The overall background model created using these methods works extremely well to create a foreground mask that classifies each pixel as either foreground or background.

Mixed Gaussian background modeling, though effective, is power intensive. One design makes use of the work already performed by a video encoder to discern the objects in the encoded video sequence. Additionally, a method for low-power background subtraction is described in [7] using compressive sampling. The amount of processing necessary to monitoring a background scene is greatly reduced compared to the conventional, mixture of Gaussians method, however, when an object is detected, high-power reconstructions of the foreground image must be computed to locate the

object(s). Additionally, assumptions about the sparsity of the foreground in each frame must be made a priori, allowing for the possibility of object detection failure of too many objects are present in one frame, or if objects are too large relative to the field of view.

CS provides a method for sampling signals below the Nyquist rate, provided the signal is sparse in some representation basis, Ψ [6]. Rather than sampling the signal in the representation domain, sampling is performed in a domain that is incoherent with the representation domain. From [6], it is sufficient to select the sampling basis, Φ , randomly, meaning a matrix of random values is orthogonalized (and potentially normalized). Typically, reconstruction is performed with a linear program to minimize the ℓ_1 norm of the reconstructed signal. In the presented design, this reconstruction is not necessary as all processing is performed in the sampling domain.

Typically, Ψ , is highly rank deficient with $m \ll n$, where m and n are the number of columns and rows of Ψ , respectively. There are typically an infinite number of solutions for the original signal, \mathbf{x} , that could have produced $\mathbf{y} = \Phi\mathbf{x}$. Traditional methods for minimizing the energy in \mathbf{x} do not produce desirable results. Instead, of the possible reconstruction solutions, the one that minimizes $\|\mathbf{x}\|_{\ell_1}$ is selected. Finding the minimum value requires solving a linear program. There are many approaches to solving the linear program [6].

The primary advantage of using CS is the ability to sample signals in a compressed state. This is of particular interest in video processing due to the large amounts of data typically collected and the need to compress the information. In the presented research, CS is used condense the amount of information processed for object detection, and by operating on the data in the CS domain, greatly reduce the power used by the system.

4.2 *Proposed System Design*

Performing conventional object detection and tracking is a power intensive process, which is problematic for highly desired implementation in embedded systems. State-of-the-art methods for fixed-camera object detection model each pixel as a multi-modal Gaussian distribution [68]. With each frame, the models are updated with new, sampled pixel values, and each pixel location is classified as either foreground or background based the model parameters and the current observed pixel value. This process presents a fairly static algorithm that consumes a large, consistent amount of power.

The proposed system design uses a variable number of models—as opposed to one for every pixel. Regions of pixels can contribute to a single model, allowing a single model to cover a varying amount of space in a frame. More complex regions of pixels will use a greater number of models, while less complex regions of pixels will use fewer models. Multi-resolution frames are acquired from the image sensor to match the spatial model density of the frame. A greater number of models in a region will require a greater sampling resolution for that region. Sampling density of the regions varies over time in an effort to provide more detail about foreground objects while performing the minimal amount of sampling required to verify that regions covering background scenes have not changed. In this way, the through put of the entire system is allowed to scale, offering maximal opportunity for power savings.

The proposed design adaptively optimizes power consumption by performing pre-compression at the image sensor to throttle throughput to the minimum levels required to perform adequate object detection and tracking. Multi-resolution frames are captured by the image sensor for processing by the system. The amount of information captured for a region of a frame is proportional to the predicted amount of pertinent information in the frame region. Each frame is analyzed to determine if more or less information may be required for spatial regions in future frames, and

sampling for the next frame is adjusted appropriately.

From frame to frame, the quadtree configuration is allowed to adaptively change, providing more or less detail as required. Initially, the first frame is divided into the blocks of the smallest size allowed, providing the highest resolution. Once sufficient frames are observed to build a high-resolution background model, the complexity of each model is observed. If a group of four sibling nodes all have low-complexity models, their parent node will be promoted to active mode for the subsequent frame allowing for a single model to be used for all pixels covered by the sibling nodes. Similarly, if a single node (that is not a leaf node) observes information that does not match the associated model sufficiently, its child nodes are made active in the subsequent frame to gather more information.

When transitioning from higher-resolution nodes to a lower-resolution node, the parameters for the lower-resolution model are inferred from the higher-resolution models. The method of inference is described in the two following sections. However, higher-resolution model information cannot be inferred from a lower-resolution model. Therefore, all inactive nodes retain their model parameters for the moment they are activated again. In this way, high-resolution information is always kept, but low-resolution information is sampled and processed when regions are stable.

The proposed system architecture shown in Figure 11 minimizes throughput and frame detail based on learned video signal properties and scales voltage and frequency back to just meet the throughput requirements. This differs significantly from current DVFS practices by actively controlling the system throughput rather than trying to predict it. As a result, more aggressive scaling can be performed without the risk of violating real-time constraints. The primary components of a typical object tracking system, which are emphasized in Figure 11, are present, but the background modeling system is notably different and control logic is added to throttle system throughput.

Throughput reduction occurs at the image sensor. Multi-resolution frames are captured that provide greater spatial resolution regions that are likely to contain objects and less resolution for regions that contain static background. With each multi-resolution frame, the background model is updated and the model parameters are used, not only to determine the presence of foreground objects, but also influence the spatial resolution of future frames. Frame resolution is also influenced by the predicted path of object, allowing the system to gather high-resolution information about a region just as an object enters the region.

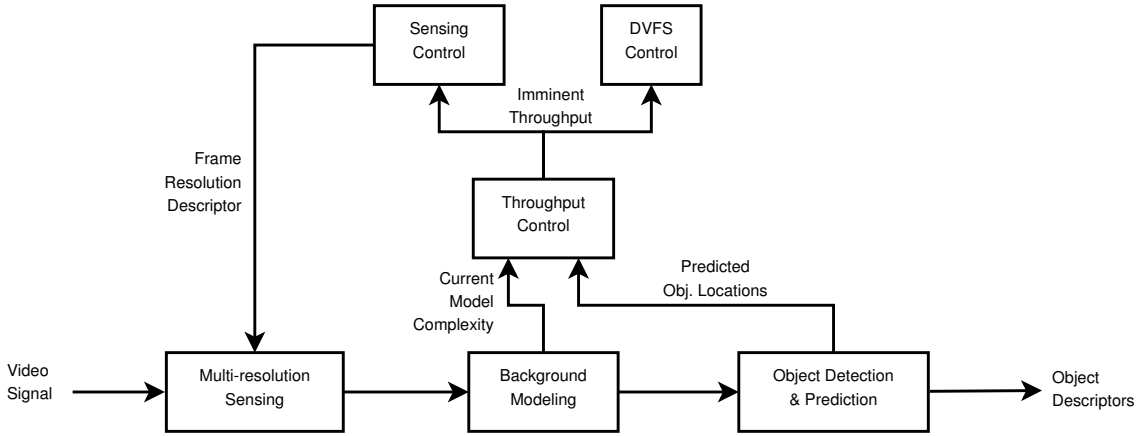


Figure 11: Proposed object tracking system architecture

4.2.1 Region Partitioning

To allow for variable throughput, each frame is divided into non-overlapping regions that correspond to levels of predicted saliency. Region boundaries are allowed to change and adapt to the video signal content with each frame. To reduce computational overhead associated with determining appropriate boundaries for regions, region boundaries are constrained by a hierarchical quadtree structure. At the most granular level, pixels are grouped into non-overlapping blocks of uniform size (typically 8×8) where each of the blocks represents a single region. This is illustrated by level C0 in Figure 12. Contiguous, specific groups of 2×2 blocks can be collapsed into a single block to represent a larger region as illustrated by levels C1 and C2 in

Figure 12.

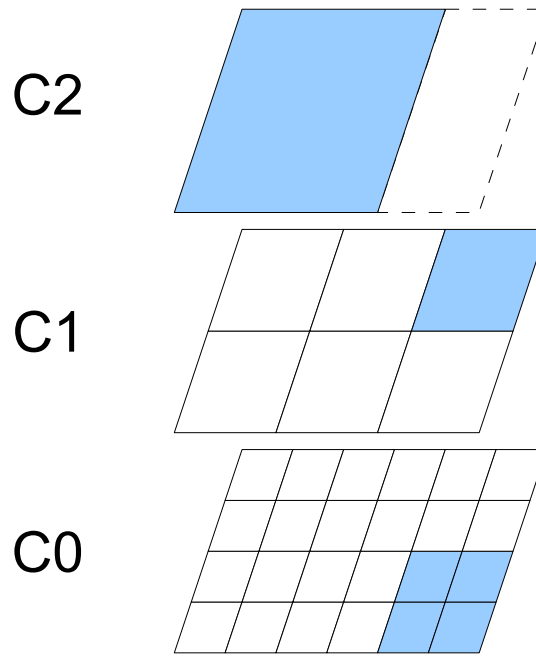


Figure 12: Hypothetical quadtree configuration

Lower levels in the quadtree structure indicate higher resolution and greater information density. Using the quadtree structure allows for the creation of multi-resolution frames that provide spatially varying resolution to correspond with suspected regions of saliency. For example, Figure 12 shows a simple, potential configuration where the bottom right region of the frame represents a region of suspected saliency. Greater resolution is selected for that region using lower-level blocks, while lesser resolution is used for other areas of the frame. The shaded blocks are active blocks for this particular frame. Together, the active blocks provide complete coverage of the frame area and do not overlap. A valid quadtree configuration is one that meets both criteria of complete coverage and non-overlapping space.

It is also useful to note that because of the dimensions of the hypothetical frame in Figure 12, the frame cannot be represented by a single, highest level block. Furthermore, only a portion of the frame can be represented at level C2. In order to not overly complicate the control algorithm, a permanent mapping of lower-level blocks

to higher-level blocks is established when the system is initialized. For example, the region covered by the single block in C2 relates directly to the blocks illustrated below it—the child blocks of the block a permanently mapped as the leftmost four blocks of C1.

The example in Figure 12 indicates the configuration for a single frame. Future frames for the same video sequence may be partitioned differently. For instance, a future frame may be sampled to highlight suspected saliency in the top left corner of the scene in addition the the bottom right as shown in Figure 13. The two regions of saliency are using high resolution, while the remaining areas of the frame are using the lowest resolution possible without violating the quadtree constraints. The details of adjusting region partitions from frame to frame are discussed in Section 4.2.4.

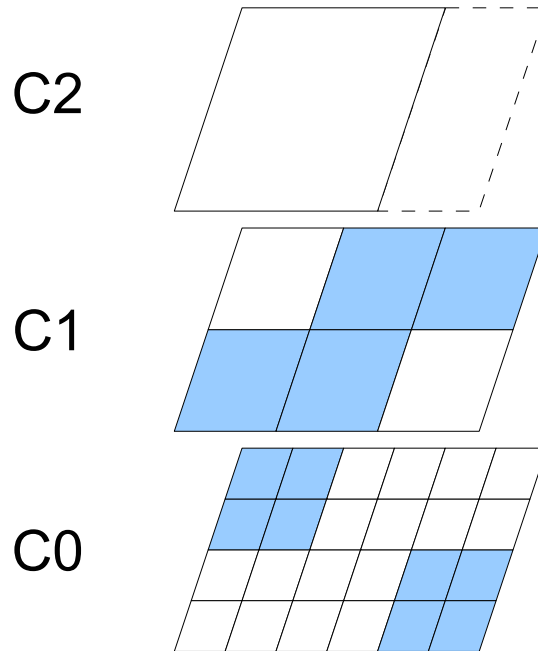


Figure 13: Hypothetical quadtree configuration—future frame

4.2.2 Adaptive Sampling

The proposed object detection and tracking system design uses two primary methods of sensing: full-resolution, and compressive. Full-resolution sampling is only applied

to the lowest level blocks, while compressive sampling is used for all higher level blocks in addition to possibly being used for leaf nodes. Considering the capability of sampling blocks using full resolution requires the extending of the representation from Figure 12 to that of Figure 14. The lowest level of the new representation represents the sampling of a small block of pixels using conventional, full resolution. Any block (or all) can use full-resolution sampling, provided it does not violate the previously discussed quadtree constraints and the block is not selected to be sampled compressively. The hypothetical example in Figure 14 shows a valid sampling configuration for a frame.

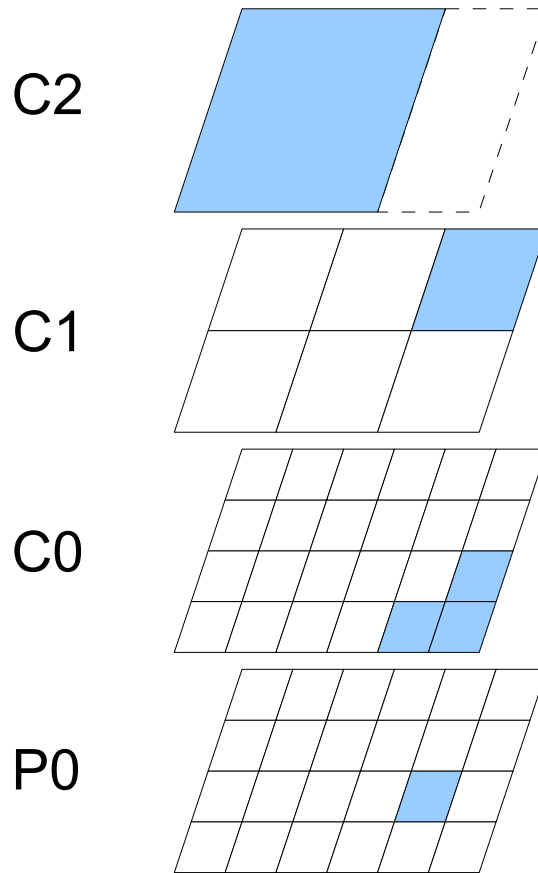


Figure 14: Hypothetical quadtree configuration with full-resolution layer

Full-resolution sampling is used when maximal detail is desired for a particular region. This is performed in a conventional manor where each pixel value is measured

independently of the others. Interpreting pixel values in a block as raster ordered vector, the sampling basis for every full-resolution block is the identity matrix, \mathbf{I} .

Compressive sampling [62] is used when less detail is desired for a particular region expected to contain predictable, background information. The amount of sampled information can vary by changing the block size used for a region as shown by the higher level blocks in Figure 14. Independent of the size of the block, the output of each CS block produces a uniform length vector of samples. The sampling basis used for each CS block is different, and blocks at different levels in the quadtree structure have the same number of rows. However, depending on the number of pixels covered by the block, the number of columns for different size blocks will vary. This produces greater compression as higher level nodes are used to sample frames.

The bases for all blocks represented in the quadtree structure are determined when the system is initialized. The basis for each of the lowest level CS blocks are created by selecting populating the basis coefficients with binary random variables with values of ± 1 :

$$\phi_{m,n} \sim 2 \left(\text{Bern} \left(\frac{1}{2} \right) - 1 \right) \quad (7)$$

The sampled values are defined as

$$\mathbf{y} = \mathbf{\Phi} \mathbf{x}, \quad (8)$$

and $\phi_{m,n}$ are the elements of $\mathbf{\Phi}$.

Bases for larger blocks are not selected randomly, but formed from the bases of the smaller block they cover. The columns of the basis for a larger block are reordered columns from the bases of smaller blocks covered by the larger block. A larger block basis is defined as

$$\mathbf{\Phi}_p = \left[\mathbf{p}_0 \quad \mathbf{p}_1 \quad \cdots \quad \mathbf{p}_{4N}, \right] \quad (9)$$

and the bases of the four largest blocks covered by the larger block (child blocks) are

defined as

$$\Phi_{c_n} = \begin{bmatrix} \mathbf{c}_{n_0} & \mathbf{c}_{n_1} & \cdots & \mathbf{c}_{n_N} \end{bmatrix} \quad (10)$$

where n indicates the raster order position of the child block relative to the parent block. The top left child block has $n = 0$, and the top right block has $n = 1$. The columns of Φ_p are defined in terms of the the child basis columns as

$$\begin{aligned} \mathbf{p}_i &= \mathbf{c}_{ab} \\ a &= \left\lfloor \frac{i \bmod 2N}{N} \right\rfloor + \left\lfloor \frac{i}{2N} \right\rfloor \\ b &= i \bmod N, \end{aligned} \quad (11)$$

Where N is the size of the child blocks in one dimension. The columns are reordered and combined such that the column ordering of each basis corresponds to the reordering of pixels when they are ordered by a parent block instead of a child block.

The reordering of the columns is necessary to ensure the following holds true:

$$\Phi_p \mathbf{x}_p = \Phi_{c_1} \mathbf{x}_{c_1} + \Phi_{c_2} \mathbf{x}_{c_2} + \Phi_{c_3} \mathbf{x}_{c_3} + \Phi_{c_4} \mathbf{x}_{c_4}, \quad (12)$$

where \mathbf{x}_p are pixel values associated with a parent node, \mathbf{x}_c are pixel values associated with child blocks, and Φ_{c_i} are the child block bases. The ordering of the basis columns corresponds to the raster ordering of pixels from the child blocks relative to the parent block. Performing this sort of reordered basis concatenation allows for parent sample vectors to be expressed as the sum of child sample vectors and, more importantly, allows for the inference between models described in the previous Section 4.2.3.

4.2.3 Background Modeling

Two types of models are used in the proposed design: pixel-level models and CS models. Pixel-level models are used in the conventional sense to obtain precise boundaries between foreground and background regions. CS models are used for established background regions that are temporally stable. If a CS modeled region is suspected or

expected to contain foreground objects, its modeling method may be updated to obtain greater resolution in the next sampled frame.

4.2.3.1 Full Resolution Modeling

Pixel-level modeling is performed using conventional methods described in [68]. Each pixel is assumed to be normally distributed and modeled as

$$x \sim N(\mu_x, \sigma_x^2). \quad (13)$$

The model parameters are updated with each frame using a predefined learning rate. The model parameters are updated by

$$\mu_x[n] = (1 - \alpha)\mu_x[n - 1] + \alpha x[n] \quad (14)$$

$$\sigma_x^2[n] = (1 - \alpha)\sigma_x^2[n - 1] + \alpha x^2[n]. \quad (15)$$

The learning rate is allowed to vary for the first frames of a sequence as to not bias the background with an initial value. The initial value of the learning rate should be 1, and the value will linearly decrease afterward, until some cutoff where the background is considered learned.

In order to prevent absorption of foreground objects into the background model, a weighted learning coefficient is used at the pixel level. The weighted learning rate is defined as

$$\alpha_w = w\alpha, \quad (16)$$

where

$$w = \begin{cases} 1 & \text{if } |x| < a\sigma \\ w_f & \text{otherwise} \end{cases} \quad (17)$$

If an observed pixel value is close to the expected range, model parameters will be updated normally. On the other hand, if the observed value is significantly far outside the expected range, the learning rate will be scaled down. This way, background

models are allowed to update over time, but are not significantly impacted when foreground objects appear. Figure 15 shows the coefficient values in comparison to the probability distribution function of each pixel, x . The nonlinear function allows for foreground objects to be filtered out of background models, while still allowing background models to adapt to changes in the background. Before the model statistics are known, w is kept at 1 to prevent interference with the initial learning process. Efficient adaptive learning was observed with $w_f \approx 0.1$ and $a \approx 3.5\sigma$.

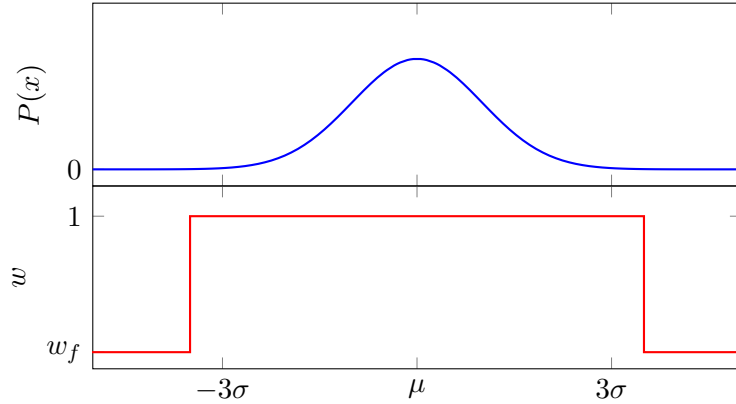


Figure 15: Weighted learning coefficient

4.2.3.2 Compressed Region Modeling

As described in Section 4.2.2, each CS block, regardless of its size, is compressively sampled to produce a fixed-length vector. The vector associated with a block, \mathbf{y} is used to model the behavior of the entire region covered by the block. From [7], background modeling for compressively sensed data can be performed by in the CS domain. This is performed by first defining the background for the block as the mean of sensed vectors over time:

$$\mathbf{b}[n] = \frac{1}{N} \sum_{i=n-N+1}^n \mathbf{y}[i] \quad (18)$$

The distance of each sample from the background is then computed as the log-squared-distance,

$$z = \ln(\|\mathbf{y} - \bar{\mathbf{y}}\|_2^2). \quad (19)$$

The distance, z , can be modeled as a Gaussian random variable, provided the vector length of \mathbf{y} is of sufficient length (greater than 30), allowing the the central limit theorem to be applicable. The distance value can, therefore, be used to model the background for a region of pixels using a single variable and two parameters:

$$z \sim N(\mu_z, \sigma_z^2). \quad (20)$$

The practical implementation of CS models requires maintaining five parameters—two more than the three previously discussed. The first required parameter is the background mean from which distances will be measured. The background mean is computed as

$$\mathbf{b}[n] = (1 - \alpha)\mathbf{b}[n - 1] + \alpha\mathbf{y}[n], \quad (21)$$

where α is the learning rate configured for the system. The mean and variance of the log-square-distance of the samples relative to the background mean needs to be updated with each frame as well. The square-distance mean is updated as

$$\mu_z[n] = (1 - \alpha)\mu_z[n - 1] + \alpha z[n], \quad (22)$$

and the variance is updated as

$$\sigma_z^2[n] = (1 - \alpha)\sigma_z^2[n - 1] + \sigma z[n]. \quad (23)$$

Additionally, the parameters for the distribution of square-distance values are maintained The square-distance values are defined as

$$w = \|\mathbf{y} - \mathbf{b}\|_2^2. \quad (24)$$

The mean for the distribution is updated as

$$\mu_w[n] = (1 - \alpha)\mu_w[n - 1] + \alpha w[n], \quad (25)$$

and the variance is updated as

$$\sigma_w^2[n] = (1 - \alpha)\sigma_w^2[n - 1] + \sigma w[n]. \quad (26)$$

The extra model parameters maintained for w are necessary for inference of parameters among models, which is further explained in Section 4.2.3.4.

4.2.3.3 *Foreground Partitioning*

Partitioning of foreground regions is only performed for blocks sampled at full resolution—any region not sampled at full resolution is assumed to be background. To avoid expensive reconstruction operations associated with CS, blocks that are currently sampled compressively must be sampled conventionally before partitioning can be performed for the region covered by the block. Pixel-level partitioning is straightforward—if the current, observed value for a pixel lies within three standard deviations of the pixel mean, it is considered background, and any other value is considered foreground:

$$f = \begin{cases} 0 & \text{if } |x - \mu_x| < c\sigma \\ 1 & \text{otherwise,} \end{cases} \quad (27)$$

where c is a constant determined ahead of time, usually 3.

4.2.3.4 *Model Inference*

The inference of higher order models from lower order models is essential to the proposed design. To efficiently represent a region with different background models at different points in time requires the ability to switch between models without relearning model parameters, which would be prohibitively expensive. Switching is allowed by inferring model parameters when switching from a higher resolution to a lower resolution. Higher resolution model parameters are stored and used again when switching from a lower resolution model to a higher resolution model. In general, only one promotion or demotion operation is allowed per active block in a frame, restricting how fast the resolution can be modulated. This allows for the reuse of stored information in higher resolution models and is expanded on in Section 4.2.4.

If a model for a region is being activated after a lower resolution model was being

used, nothing is inferred from the previously active, low resolution model. Instead, the stored state of the model is used.

The CS model is based on the distance of sensed vectors in the CS domain from the mean of the vectors across time. This is represented in terms of the sensed vectors as

$$z = \ln \|\mathbf{y} - \mu_{\mathbf{y}}\|_2^2. \quad (28)$$

However, inference needs to come from pixel-level parameters. Given $\Phi \mathbf{x} = \mathbf{y}$, (28) can be approximated as

$$z \approx \ln(M_{\Phi} (\|\mathbf{x} - \mu_{\mathbf{x}}\|_2^2)), \quad (29)$$

where M_{Φ} is the number of rows in the sampling basis to be used for the block. The distribution of $\mathbf{x} - \mu_{\mathbf{x}}$ is the same as $\mu_{\mathbf{x}}$ with zero mean:

$$\mathbf{x} - \mu_{\mathbf{x}} \sim N(0, \sigma_x^2), \quad (30)$$

and its square is the gamma distribution

$$\mathbf{w} = (\mathbf{x} - \mu_{\mathbf{x}})^2 \quad (31)$$

$$\sim \Gamma\left(k = \frac{1}{2}, \theta = 2\sigma_x^2\right). \quad (32)$$

The approximate mean and variance of \mathbf{w} in terms of parameters of x is therefore

$$\mu_w = \mathbf{k}_w \theta_w \quad (33)$$

$$= \sigma_x^2 \quad (34)$$

$$\sigma_w^2 = \mathbf{k}_w \theta_w^2 \quad (35)$$

$$= 2\sigma_x^4. \quad (36)$$

Summing the squared values produces a squared distance measurement

$$d = \sum^n w_n \quad (37)$$

$$d = M_{\Phi} (\|\mathbf{x} - \mu_{\mathbf{x}}\|_2^2) \quad (38)$$

with mean and variance that depends on the distribution of \mathbf{w} and therefore \mathbf{x}

$$\mu_d = M_\Phi \sum_n \sigma_{x_n} \quad (39)$$

$$\sigma_d^2 = 2QM_\Phi^2 \sum_n \sigma_{x_n}^4, \quad (40)$$

where Q represents the expected covariance among elements of \mathbf{w} . This value was empirically observed over test sequences and approximated as $Q \approx 5$. The distribution of d can be estimated as

$$d \sim \Gamma(k_d, \theta_d) \quad (41)$$

$$k_d = \frac{(\sum_n \sigma_{x_n}^2)^2}{2Q \sum_n \sigma_{x_n}^2} \quad (42)$$

$$\theta_d = \frac{2QM_\Phi \sum_n \sigma_{x_n}^4}{\sum_n \sigma_{x_n}^2}. \quad (43)$$

Finally, taking the natural log of d yields z with the following estimated distribution in terms of the observed parameters of \mathbf{x} :

$$z \sim N(\mu_z, \sigma_z^2) \quad (44)$$

$$\mu_z = \psi(k_d) + \ln(\theta_d) \quad (45)$$

$$= \psi\left(\frac{(\sum_n \sigma_{x_n}^2)^2}{2Q \sum_n \sigma_{x_n}^2}\right) + \ln\left(\frac{2QM_\Phi \sum_n \sigma_{x_n}^4}{\sum_n \sigma_{x_n}^2}\right) \quad (46)$$

$$\sigma_z^2 = \psi_1(k_d) \quad (47)$$

$$= \psi_1\left(\frac{(\sum_n \sigma_{x_n}^2)^2}{2Q \sum_n \sigma_{x_n}^2}\right), \quad (48)$$

where $\psi(\cdot)$ is the digamma function and $\psi_1(\cdot)$ is the trigamma function. Using these approximated parameters for z , the learned parameters for the pixels of a block can be easily translated into CS parameters, allowing for immediate translation and use of a new background model in the CS domain, without the need to perform additional learning, or hold and reprocess prior samples.

Changing the modeling and sampling method from a high CS resolution to a low CS resolution presents similar challenges as moving from pixel-level modeling

to CS modeling. High resolution parameters must be translated to low resolution parameters without resampling data or recomputing statistics from saved values. The key to making this possible is the use of subspaces described in Section 4.2.2. Using the proposed hierarchical sampling method, the following relationship between parent node and child nodes exists:

$$\mathbf{y}_p = \sum_{c \in C} \mathbf{y}_c \quad (49)$$

This implies that the modeled distance metric of a parent model can be described in terms of its children as

$$z_p = \ln d_p^2 \quad (50)$$

$$= \ln \sum_{c \in C} d_c^2, \quad (51)$$

where C is the set of four child nodes for p . Using a Taylor series, the mean for a parent metric can be estimated in terms of its children as

$$\mu_{z_p} \approx \ln \mu_{d_p^2} - \frac{\sigma_{d_p^2}^2}{2\mu_{d_p^2}^2} \quad (52)$$

$$\approx \ln \sum_{c \in C} \mu_{d_c^2} - \frac{\sum_{c \in C} \sigma_{d_c^2}^2}{2 \left(\sum_{c \in C} \mu_{d_c^2} \right)^2}. \quad (53)$$

The variance for the parent metric can be approximated as

$$\sigma_{z_p}^2 \approx \frac{\sigma_{d_p^2}^2}{\mu_{d_p^2}^2} \quad (54)$$

$$\approx \frac{\sum_{c \in C} \sigma_{d_c^2}^2}{\left(\sum_{c \in C} \mu_{d_c^2} \right)^2}. \quad (55)$$

The inference of these parameters requires the modest overhead of updating statistics for d^2 for each active CS model. These parameters can then be used to approximate the parameters of low resolution models.

The distance measurement parameters can also be inferred from the CS models of child nodes. The parameters in terms of child CS nodes are

$$\mu_z \approx \ln \sum_{c \in C} \mu_{z_c} - \frac{\sum_{c \in C} \sigma_{z_c}^2}{2 \left(\sum_{c \in C} \mu_{z_c} \right)^2} \quad (56)$$

$$\sigma_z^2 = \frac{\sum_{c \in C} \sigma_{z_c}^2}{\left(\sum_{c \in C} \mu_{z_c} \right)^2}, \quad (57)$$

where C is the set of child nodes relative to the current node. This inference is a result of proper basis selection, described in the following section.

Following the assumption that each pixel is normally distributed, the distance of a pixel from its mean value is modeled as

$$x \sim N(0, \sigma_x^2). \quad (58)$$

If the pixel values are squared, the distribution of the resulting values can be represented with a gamma distribution:

$$y \sim \Gamma \left(\frac{1}{2}, 2\sigma_y^2 \right) \quad (59)$$

4.2.4 Adaptive Control

The adaptive control system consists of three systems shown in Figure 11: Throughput Control, Sensing Control, and DVFS Control. Throughput control is performed by evaluating each active block region and determining if more or less detail may be needed based on current sample values, background models, and object tracking predictions. The Throughput Control system produces a list of active blocks to be used for the next frame, which is used as input for the Sensing Control and DVFS Control. The Sensing Control selects the bases to use for sampling the next frame, and DVFS Control predicts the computational workload of the next frame.

The functionality of the Throughput Control system is described in Algorithm 1. When the system is initialized, the active block list, B , is comprised explicitly of all full-resolution blocks. After the full-resolution blocks have acquired a background model (40 frames), the procedure in 1 is called for each frame. Any portion of the scene that overlaps with a predicted object location is will be immediately sampled at full resolution during the next frame (lines 3–8). If the current samples deviate beyond a predefined threshold, $t_d\sigma$, for the associated block model, resolution is increased, and the children of the current block are activated for the next frame in place of the current block (lines 10–14). Similarly, if the deviation of the current samples from the block model is below a given threshold, $t_p\sigma$, for a set of sibling blocks, the parent block of those siblings is activated in their place for the next frame.

Algorithm 1 Throughput control algorithm

```

1: procedure UPDATEACTIVEBLOCKS( $\Sigma, L$ )
2:   for all  $b \in B, \sigma \in \Sigma$  do                                     ▷  $B$ : Set of active blocks
3:     for all  $l \in L$  do                                             ▷  $L$ : Predicted object locations
4:       if  $l$  overlaps  $b$  then
5:         FULLDEMOTE( $b$ )
6:          $fullDemote \leftarrow \text{true}$ 
7:       end if
8:     end for
9:     if not  $fullDemote$  then
10:      if  $v_b > t_d\sigma_b$  then
11:        for all  $c \in C$  do                                           ▷  $C$ : Child set for  $b$ 
12:          Add  $c$  to  $B$ 
13:        end for
14:        Remove  $b$  from  $B$ 
15:      else if  $v_s > t_p\sigma_s \quad \forall s \in S$  then                   ▷  $S$ : Sibling set for  $b$ 
16:        PROMOTE( $S$ )                                                 ▷ Infer model parameters from  $S$ 
17:        for all  $s \in S$  do
18:          Remove  $s$  from  $B$ 
19:        end for
20:      end if
21:    end if
22:  end for
23:  return  $B$ 
24: end procedure

```

To ensure stable operation of the control system, a dampening method is used. Promotion from four child blocks to a single parent block requires *approximating* model parameters of the parent block. To prevent potential oscillation due to inaccuracies of parameter inference, a recently promoted block is held as active for a predefined number of frames. Empirical results indicate that low values on the order of 2 sufficed to stabilize the system. Recently demoted blocks do not infer model parameters, thus, hold time is required.

Additionally, to allow the system to react quickly to objects being tracked, regions that are expected to contain tracked objects in the next frame are automatically promoted to full-resolution, bypassing the normal process of demoting one level per frame. The normal demotion process is restricted to one level per frame because spatial resolution is increased with each demotion, allowing the system to more closely target the object location with each frame. Full promotion is possible when specific object coordinates are known—overlapping regions with the predicted object are set to full-resolution, while all other nodes are only demoted low enough to not violate the quadtree constraint.

Each active region has an associated sampling basis that is determined when the system is initialized.

The amount of data captured and processed in any given frame depends on the information from the frames that came before it, not the information contained in the frame itself. The workload for each frame can be predicted immediately before the frame is to be sampled and processed. Since the system is actively scaling its own throughput ahead of time, accurate predictions of computational power can be output to a DVFS controller to scale the object detection system within small margins.

Computational complexity for a given architecture is predicted by observing nominal processing times and calculating the per-block processing time and overhead time for full-resolution and compressed blocks. The prediction coefficients are defined as

$$\begin{bmatrix} \mathbf{x}_m & \mathbf{x}_q \end{bmatrix} = \mathbf{A}^+ \begin{bmatrix} \mathbf{b}_m & \mathbf{b}_q \end{bmatrix}, \quad (60)$$

where \mathbf{b}_m and \mathbf{b}_q are column vectors containing model update time per frame and overhead update time per frame, respectively. The matrix, \mathbf{A} , is comprised of two column vectors—the first containing the associated number of pixel-level blocks and the number of compressive blocks, respectively. The column vector, \mathbf{x}_m , represents the predicted model update time rates for pixel-level and CS blocks, while the column vector \mathbf{x}_q represents the predicted overhead rates for pixel-level and CS blocks. With \mathbf{x}_m and \mathbf{x}_q calculated for a selected architecture, a computational complexity prediction can be made for each adaptively processed frame, one frame in advance. These predictions can, in turn, be used to aggressively scale hardware voltage and frequency for maximal power savings.

4.3 Results

To validate the proposed design, surveillance video sequences from the PETS 2015 data set [43] were processed using proposed architecture as well as the conventional pixel-level Gaussian model architecture. Function-level processing time was measured and compared for all test sequences. Additionally, quality comparisons were performed to evaluate the accuracy of the proposed architecture relative to the conventional method.

To function properly, the purpose design need to remain sensitive to new foreground activity even while operating with reduced data. Figure 16 shows the area of the each frame being sampled with full resolution (with the expectation of finding foreground objects) relative to the total area of the scene. This is essentially the percentage of the scene that is expected to contain foreground objects. Also, the figure shows the percentage of pixels in error relative to the ground truth obtained using conventional pixel-level gaussian models for each frame. An increase in error

signifies undetected foreground area. In the presented example, each small increase in error is immediately followed by an increase in foreground sampling, indicating the control system has detected the general regions containing new objects and has begun increasing throughput to obtain more detailed information about those regions. Once the system has adapted to the unexpected changes, the error decreases. In this way, the system is able to continuously scale its throughput relative to the complexity of the input signal. After the initial learning phase and subsequent down-scaling, the area drops to less than 1.6% of the scene area and only increases when foreground activity begins. During foreground activity, full resolution processing does not exceed 34% due to the effort being spatially constrained to regions with expected foreground activity.

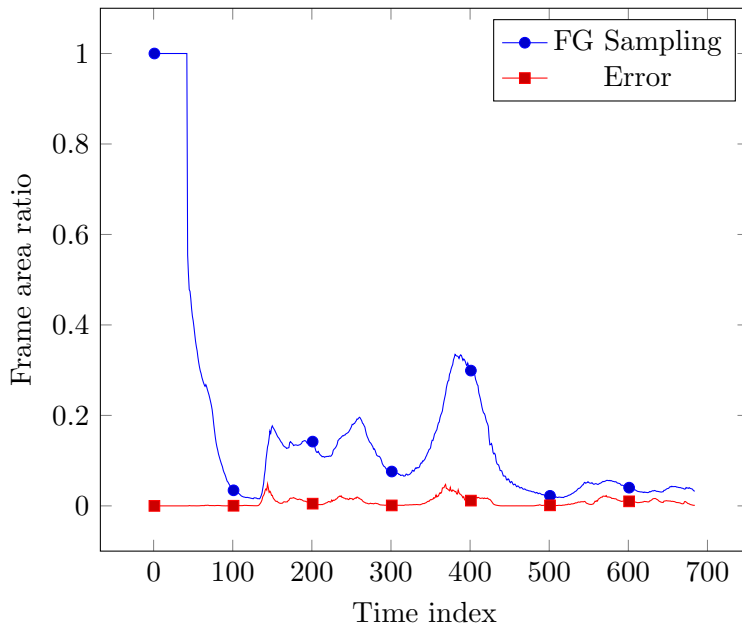


Figure 16: System scaling response and performance for a single test video sequence

In addition to the previous test sequence, 15 other sequences from the PETS data set were tested. All test sequences contained static regions at various times that allowed the adaptive architecture to scale throughput below conventional levels. The average throughput for each test sequence was measured for the proposed system

design and the conventional design. The results are shown in Figure 17. In each test sequence, the conventional throughput is determined strictly by the size of the frame. In contrast, the throughput of the adaptive system design is influenced by frame size, but is also largely dependent on the complexity of the signal. Average throughput reductions of at least 50% were achieved for each test sequence.

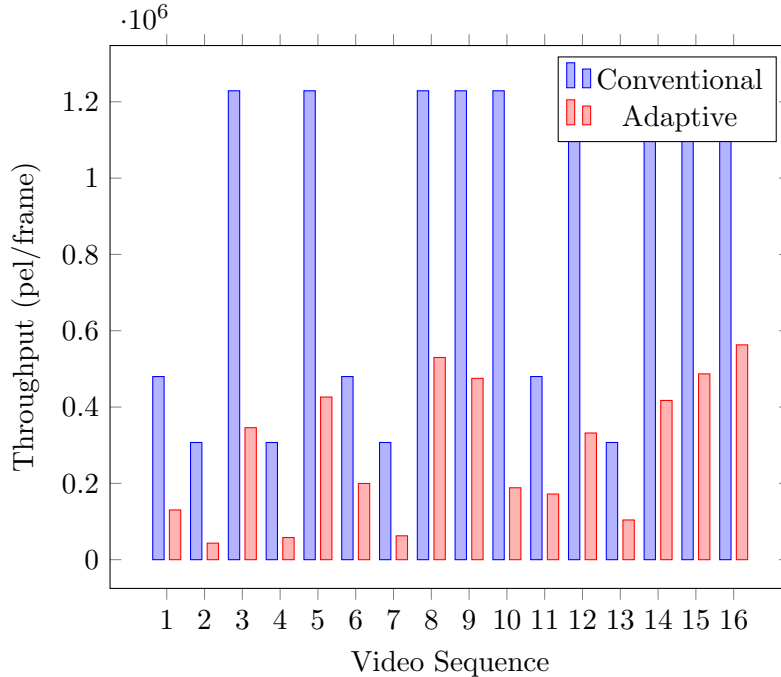


Figure 17: Average throughput for each test sequence measured in pixels per frame

Reduced throughput alone does not justify the proposed design. The additional modeling methods and control functionality add additional computational complexity to the proposed design, relative to the conventional design. To measure potential savings, each test sequence was processed using the proposed design and the conventional design while measuring time spent modeling updating models and measuring additional overhead associated with the proposed control system. These measurements are shown in Figure 18. For each video sequence, conventional processing times are proportional to throughput, which is determined by frame size. Adaptive processing time is also proportional to throughput, however, with a greater coefficient, compared to conventional processing. This is due to increased complexity of updating the CS

models compared to pixel-level models. However, the computational savings due to reduction in the total number of models to maintain allows for a net reduction in computational effort for the proposed design relative to the conventional design.

The overhead processing time indicated for the adaptive design is the additional overhead for the control system in the proposed design. This overhead scales with the number of active CS regions and is relatively small compared to the computational effort required for maintaining background models. The overhead effort does not significantly impact the net computational savings of the proposed design.

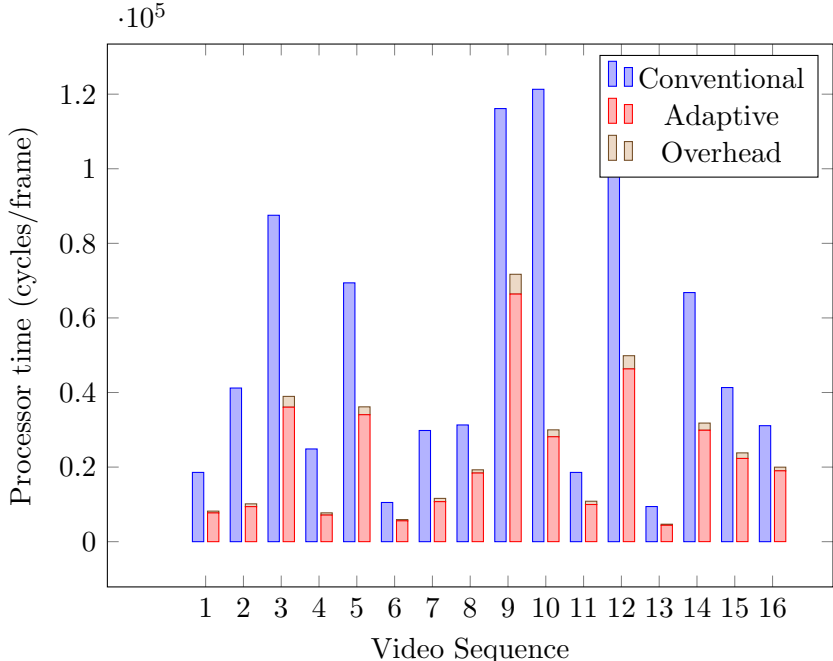


Figure 18: Processor time vs frame index

To justify the reduced throughput and complexity of the proposed design, the system output needs to be validated. Validation is achieved comparing the object detection output of the proposed design against the conventional design. The amount of foreground activity in the output of each design for sequence 12 is shown in Figure 19. The results indicate close tracking of the adaptive, proposed design with the results achieved using full resolution. These results demonstrate the ability of the proposed system to quickly respond to the appearance of foreground objects and

adjust sampling to acquire the necessary detail to identify the objects.

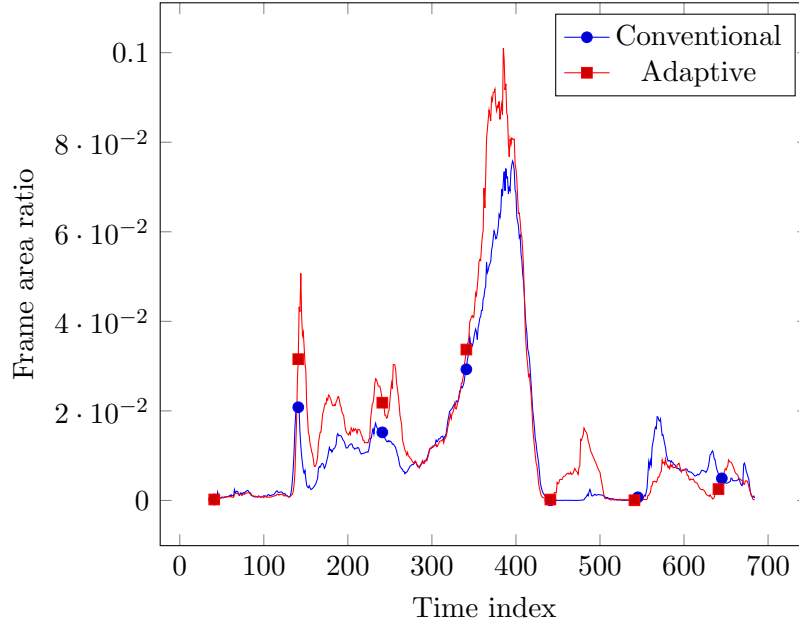


Figure 19: Object detection area comparison

As implied by the example in Figure 16, a certain amount of error may be induced by the preposed design. The computational savings are irrelevant if they are produced at the expense of excessive quality degradation. Small differences in foreground/background segmentation can be tolerated, but in general, the segmentation masks must agree to produce accurate object detection and tracking results. To quantify the error associated with the proposed design, the output mask of the adaptive design was compared to the output of the conventional design for each of the sample video sequences. The differences between the generated foreground/background masks in terms of mean-squared-error are shown in Figure 20. For each video sequence, the error in the proposed design is minimal, indicating quick adaptation by the control system. The results indicate median error for all sequences is below 1% and 75th percentile for all sequences is less than 2%.

After observing processing times and throughput metrics for the test video sequences, prediction coefficients were calculated using 60. The coefficients were used

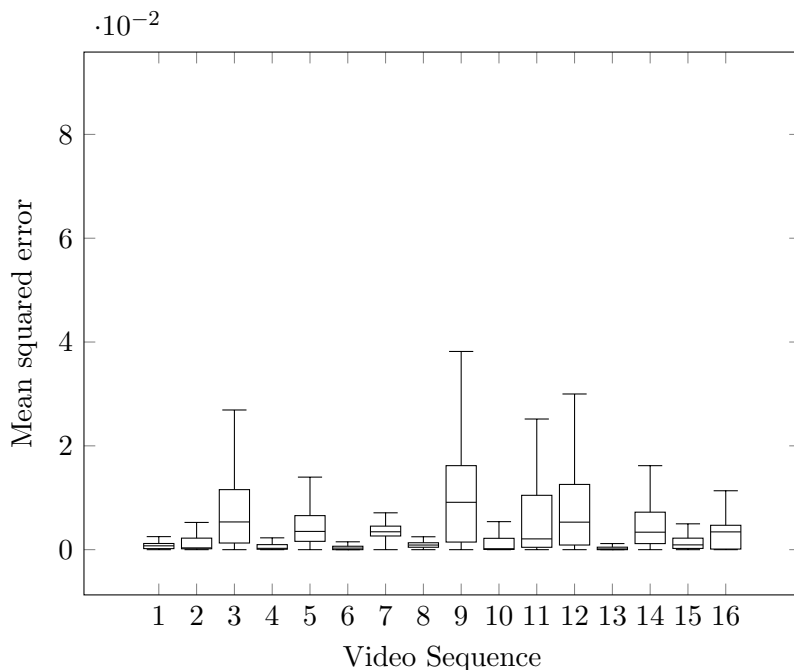


Figure 20: Mean squared error of the segmentation mask relative to conventional processing

to produce processing requirements predictions for test sequence 12. The prediction results are shown along side the observed processor time values in Figure 21. The results indicate close tracking of predicted computational complexity with actual complexity. The presented results were calculated using a general purpose CPU. More specialized hardware is likely to have increased prediction accuracy. Based on the architecture used to perform the tests, model update rates are estimated as 8.9354 and 9.2505 cycles per block for pixel-level blocks and compressive blocks, respectively. The overhead rates are calculated using 60 as 0.1609 and 0.9727 cycles per block for pixel-level blocks and compressive blocks, respectively. These rates are used to generate the prediction in Figure 21.

The example shown in Figure 22c demonstrates the ability of the proposed design to focus processing power on foreground regions and scale down throughput and processing for background regions. The larger, blue blocks indicate large regions being sampled compressively and using a single model for each block. The yellow blocks

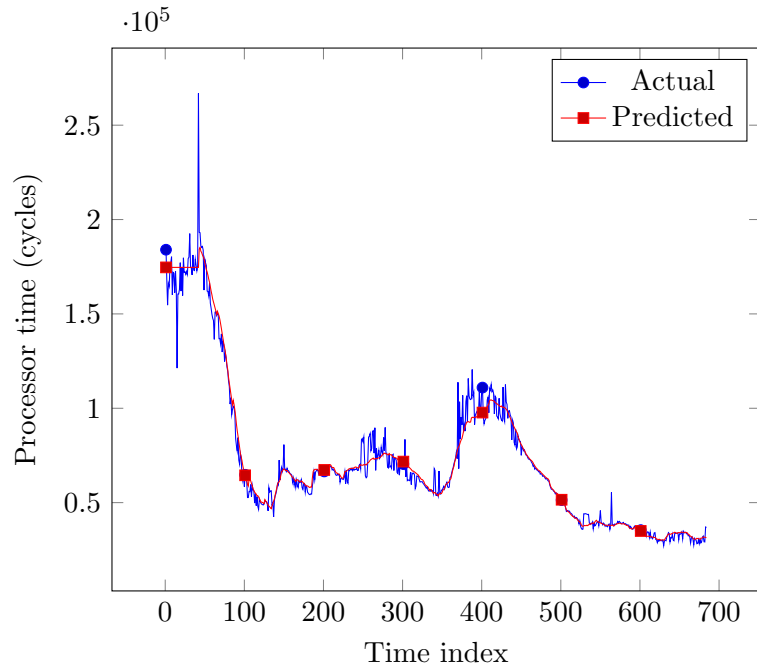


Figure 21: Predicted computational complexity compared to observed computational complexity

concentrated around the foreground object indicate high resolution sampling using independent pixel-level models to obtain high-resolution segmentation boundaries. Tracking information along with model information is used to continually update the resolution and modeling method for regions as objects pass through.

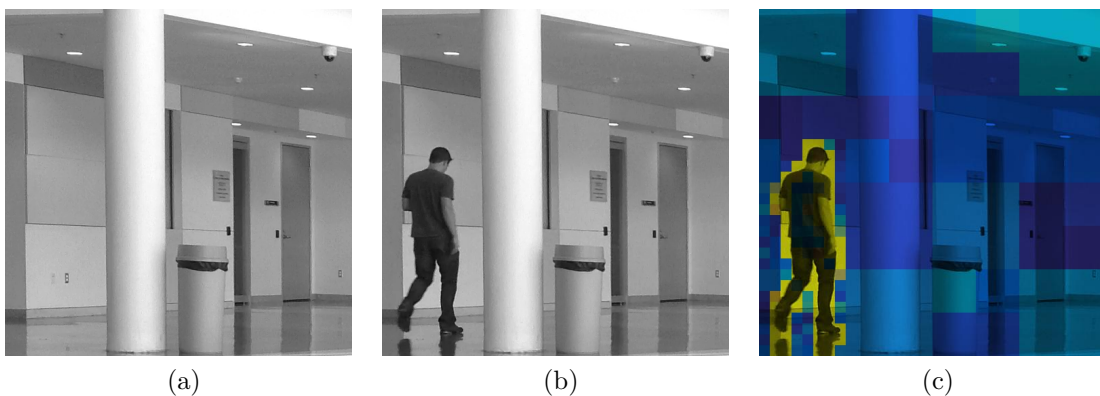


Figure 22: Adaptive object detection example showing background model (a), sample scene with object present (b), and quadtree detection structure superimposed on sample scene (c). Colors on the yellow end of the spectrum indicate high probability of an object being present while colors at the blue end of the spectrum indicate the opposite.

4.4 *Conclusion*

By using multi-resolution input frames in the proposed design, throughput for the entire system is made variable. Collecting high resolution information about foreground regions and lower information about background regions is shown to be sufficient to produce proper object tracking results.

Using features relevant to the high-level application (object detection and tracking) is demonstrated to be a good method for controlling the multi-resolution throughput. Rather than collecting maximal information to be processed and filtered in each frame, this research shows that capturing an amount of information proportional with the expected need is sufficient to appropriately influence future decisions on capturing more or less information. The selected feedback feature, region stability, is not only successful at distinguishing between foreground and background, but determining when more or less detail about regions is appropriate.

Constraining the multi-resolution frame configurations with a quad-tree structure reduces the complexity of the control algorithm, while allowing for a large dynamic range in terms of overall throughput. Additionally, it offers a hierarchical method for refining the boundaries of regions of interest.

The proposed model inference methods allow for fast, efficient transitions between resolution configurations. This is essential to allow the system to respond to quickly changing foreground regions. Alternatively, relearning model parameters each time the resolution configuration is changed for a given region would create enough delay to make the system impractical.

Up front selection of resolution configurations for each frame prior to processing allows for efficient prediction of computational demand for each frame. Prior work has focused on performing analyses of full-resolution information to predict roughly how much computational effort may required, which requires continual monitoring and updating of the processing of the full resolution information. In contrast, the

proposed method effectively bounds the required computational effort for each frame and adjusts subsequent frames as necessary. As a result, precise predictions of computational effort are made and can be used for precise voltage and frequency scaling.

Combining variable throughput, a control system aligned with the goals of the object tracking system, and accurate complexity predictions allows great power savings at the expense of a modest lag in detection time while the system adjusts to new, unexpected foreground regions. Due to the non-linear relationship between power and DVFS, the reductions in predictable computational effort can be magnified in terms of power savings.

The experiments conducted in this research used single Gaussian distributions as base models. Future work should explore the possibility of using multi-modal Gaussian models at both the pixel-level and for compressive models.

As previously stated, there are many methods for performing object detection and tracking. The adaptive methods presented in this research are not applicable strictly to background modeling approach studied. Other, more complex methods that require more computational complexity, in general, may benefit from scaled throughput as well. Further work is warranted for evaluating the principles of the proposed system design to other object tracking methods to direct the attention and computational effort of the object detection and tracking systems to salient regions.

CHAPTER V

ERROR DETECTION AND MITIGATION IN VIDEO ENCODING HARDWARE

5.1 *Introduction*

The novel contribution of this research is a video encoder design methodology referred to as parallel independent signature processing (PISP). PISP is a method of performing error detection and mitigation of error effects by creating a compact signature for each sampled MB, and processing each signature in parallel with the MB it was created from. Figure 23 demonstrates this principle by generalizing a video encoder as a series of systems (T_i) operating on the sampled MB, \mathbf{b}_s , to yield the reconstructed MB, $\hat{\mathbf{b}}_s$. The systems shown in Figure 23 can represent any system or collection of systems from Figure 1. For instance, T_1 might represent the prediction system; T_2 might represent the transform system (or perhaps one of its subsystems); and finally, the reconstructed MB, $\hat{\mathbf{b}}_s$ is produced. The signature for the sampled MB, \mathbf{s}_s , is generated by the signature generating matrix, \mathbf{G} . As the MB is processed by the MB processing systems, T_i , the signature is processed by analogous systems, U_i . The practicality of the PISP design is subject to the selection of systems, U_i , such that the following equation holds:

$$\mathbf{G}\mathbf{b}_i = \mathbf{s}_i, \forall i \tag{61}$$

If the necessary signature processing systems exist under the constraint in (61), an incorrectly computed result for any of the MB processing systems will produce a final reconstructed MB that does not match the corresponding, reconstructed signature. Errors can, therefore, be detected by comparing $\mathbf{G}\hat{\mathbf{b}}_s$ to $\hat{\mathbf{s}}_s$. If the two results match,

the MB and signature were processed correctly with high probability. Otherwise, an incorrect result was produced by one of the MB or signature processing systems.

Essential to proper operation of the encoder and decoder is the state of the reference MBs, $\hat{\mathbf{b}}_s$. If the decoder memory differs from the encoder memory, an effect called *decoder drift* occurs. For this reason, the exact MB reconstruction performed on the decoder is also performed in the encoder (along with quantization error), as indicated by the shaded systems in Figure 1. If decoder drift occurs, artifacts caused by memory differences can propagate from one MB to another. If an erroneous MB is used to predict other MBs, the resulting, encoded MBs may also be in error. Thus, errors can be propagated from one MB to others.

Properly functioning encoders and decoders are designed to prevent decoder drift. However, soft errors may allow a decoder drift to occur. Specifically, errors occurring in any of the shaded systems in Figure 1 can cause a drift, allowing errors to propagate. If an operation by a shaded system is performed differently in the encoder than its partner system in the decoder, the values recorded in memories of the encoder and decoder for an MB will be different. Any future prediction made from this incoherent MB will create more unmatched MBs. In this way, errors are propagated and the memory of the decoder drifts away from the memory of the encoder. It is, therefore, imperative to develop protection for all vulnerable systems of the video encoder, especially when using unreliable hardware. Section 5.2 describes the proposed method for protecting the vulnerable encoding systems through error detection and mitigation.

5.2 *PISP Design Details*

In general, the length of the signature for a given MB is expected to be much smaller than the length of the MB vector. Therefore, the generating matrix, \mathbf{G} , reduces dimensionality, making the generalized design in Figure 23 impractical. To compensate

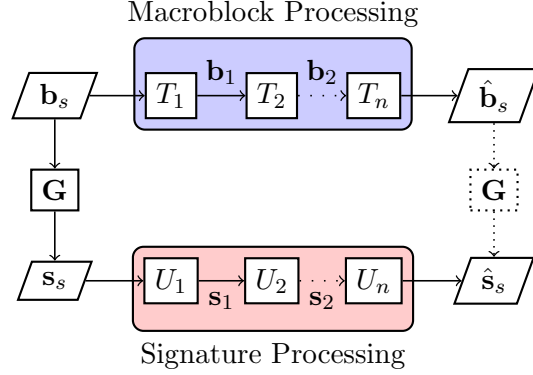


Figure 23: Ideal PISP theoretical overview with linear systems. Signature processing produces a signature matching the output of MB processing.

for this, a differential signature system, \mathbf{D} , is introduced in Figure 24. A particular differential signature system, \mathbf{D}_i , computes a differential signature vector, \mathbf{s}_{d_i} , that represents the difference between \mathbf{s}_i and \mathbf{s}_{i-1} . The computation is performed as a function of the input to the corresponding MB processing system, \mathbf{T}_i . The differential signature is defined as

$$\begin{aligned}
 \mathbf{s}_{d_i} &= \mathbf{s}_i - \mathbf{s}_{i-1} \\
 &= \mathbf{G}(\mathbf{b}_i - \mathbf{b}_{i-1}) \pmod{2^l},
 \end{aligned} \tag{62}$$

however, the computation must be performed a priori, relative to the MB processing system, T_i —the output of the system cannot be observed directly. To enable this, each MB processing system is modeled as linear transform (when possible), which is indicated by the notation change in Figure 24. This assumption allows (62) to be redefined as

$$\begin{aligned}
 \mathbf{s}_{d_i} &= \mathbf{G}\mathbf{b}_i - \mathbf{G}\mathbf{b}_{i-1} \\
 &= \mathbf{G}\mathbf{T}_i\mathbf{b}_{i-1} - \mathbf{G}\mathbf{b}_{i-1} \\
 &= \mathbf{G}(\mathbf{T}_i - \mathbf{I})\mathbf{b}_{i-1} \\
 &= \mathbf{D}_i\mathbf{b}_{i-1} \pmod{2^l},
 \end{aligned} \tag{63}$$

where l is the bit depth of the the pixels samples, and the differential signature systems are defined as

$$\mathbf{D}_i = \mathbf{G}(\mathbf{T}_i - \mathbf{I}). \quad (64)$$

As indicated by Figure 24, the previously defined signature processing systems, U_i , are reduced to a summation with a differential signature. The output of each system is defined as,

$$\begin{aligned} \mathbf{s}_i &= \mathbf{s}_{i-1} + \mathbf{s}_{d_i} \\ &= \mathbf{s}_{i-1} + \mathbf{D}_i \mathbf{b}_{i-1} \pmod{2^l}. \end{aligned} \quad (65)$$

If each MB processing system can be modeled as a linear transform, independent signature processing can be realized as shown in Figure 24. Furthermore, once a generating matrix has been selected, each differential signature system matrix can be precomputed once, reducing the amount of overhead added by the proposed design.

Not all systems of a video encoder are linear. These systems cannot be incorporated into the proposed method in Figure 24 directly. Instead, the nonlinear operations in these systems (such as rounding in the quantization system) are modeled using additive noise which can be directly measured and used to adjust differential signatures. The use of this method is detailed further in Section 5.5.

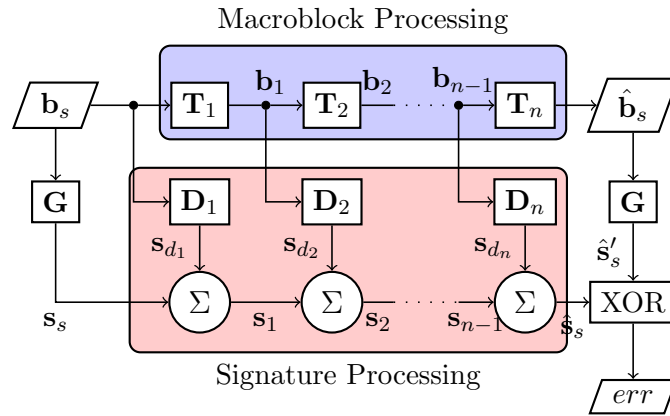


Figure 24: Realizable PISP Overview. Differential systems, \mathbf{D}_i , are used to provide missing information to signature processing system.

The signature generating matrix, \mathbf{G} , can represent any linear block code (LBC). The generated code word functions as a signature for the data, so the length of the code should be much smaller than the length of a MB vector. Code parameters can be changed to provide more or less robust error detection capability. To minimize computational overhead, the generating matrix needs to be sparse. Additionally, modular arithmetic is used to maintain a tight signature space and reduce computational overhead for signatures. The modulo is 2^l , where l is the bit depth of the MB samples.

Although there are many generating matrices that will satisfy the design constraints, for practical reasons, a pseudo-random, sparse, binary generating matrix is used with the design. Each element of the matrix is assigned a value of 1 or 0 with some preference to 0 while ensuring each column of the matrix contains at least one non-zero value. The use of binary values avoids expensive multiply operations when computing signatures and when computing differential signatures (explained later in this section).

MBs are typically represented as multiple matrices, each representing a single color plane. However, in this research, each matrix is converted to a raster ordered vector of the matrix elements. Each color plane is appended in order. The H.264 standard may use different prediction modes for different planes. As a result, the matrix representing the linear transform for an MB processing system is generally sparse. A general MB processing system modeled as a linear transform will take the form

$$\mathbf{T}_i = \begin{bmatrix} \mathbf{T}_{i_Y} & 0 & 0 \\ 0 & \mathbf{T}_{i_{Cb}} & 0 \\ 0 & 0 & \mathbf{T}_{i_{Cr}} \end{bmatrix}, \quad (66)$$

where each of the matrices along the diagonal performs a linear transform on a single color plane indicated by subscript (luma, blue chrominance, and red chrominance).

Throughout this research, when a transform is described, only the primary color plane will be addressed to avoid being overly verbose. The same transform can easily be scaled and applied to other color planes.

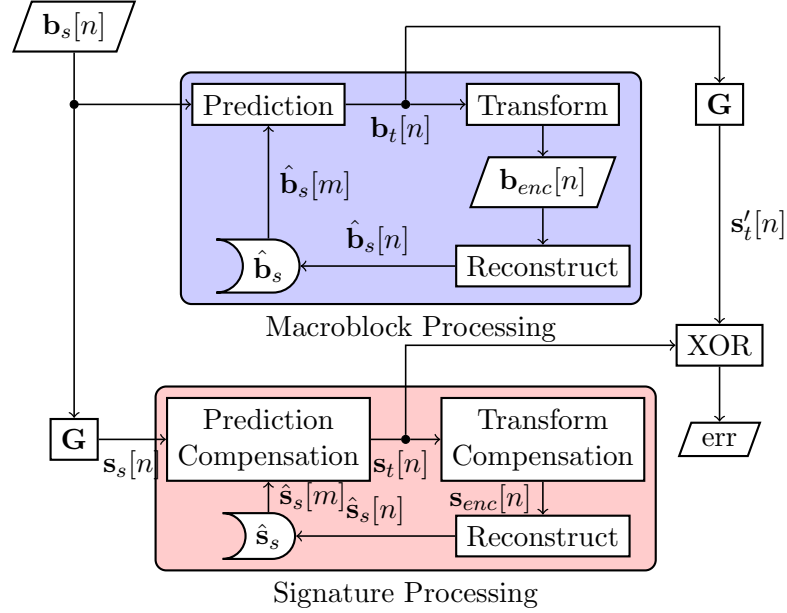


Figure 25: Proposed video encoder design

The application of the generalized PISP design to a conventional MCHVE yields the design shown in Figure 25. The MB processing systems and signature processing systems are analogous to the generalized counterparts in Figure 23. The development of the necessary differential systems for the specific design is discussed in later sections. The prediction and transform systems contain all processing elements depicted in Figure 1. Signature processing is performed in parallel with MB processing. As each MB is encoded, reconstructed, and stored in memory, a corresponding signature is generated, processed, and stored. Any time a stored MB is used as a prediction reference to produce a residual MB, $\mathbf{b}_t[n]$, the corresponding, stored signature will be used to generate the prediction signature, $\mathbf{s}_t[n]$. For each processed MB, the residual signature is checked for coherence with the associated, residual MB. Performing the

comparison after prediction allows for validation of proper processing by the prediction system. Performing validation at any other point may not detect errors occurring within the prediction systems (MC and IP) as the same, potentially erroneous, prediction is added back during reconstruction (shown in Figure 1), obscuring any potential errors that may exist in an encoded MB. The validation signature generated directly from the residual MB, $\mathbf{s}'_i[n]$, and the residual signature, $\mathbf{s}_i[n]$, are compared. If the signatures match, correct processing up to this point is highly likely. Otherwise, non-matching signatures indicate that a processing error has occurred and mitigating action should be initiated.

In the following subsections, the specifics of the proposed encoder design are presented. Section 5.3 validates the ability of the proposed design to detect errors. Section 5.4 analyzes the prediction system of the conventional encoder design and develops the necessary prediction signature processing system of the proposed encoder design. Section 5.5 analyzes both the transform and decoding systems of the conventional encoder design and develops the transform signature processing system of the proposed encoder design. Finally, error detection and mitigation methods are presented in Section 5.6.

5.3 PISP Design Validation

General validation of the proposed method is performed by evaluating the effects of additive error from the system, \mathbf{T}_{i-1} , in Figure 24. For the design to be practical, a validation signature generated directly from a particular MB must be incoherent with the corresponding signature calculated by the signature processing system when the MB vector is in error:

$$\mathbf{G}\mathbf{b}_i \neq \mathbf{s}_i \quad \text{if } \mathbf{b}_i \text{ is erroneous.} \quad (67)$$

Assuming an error is made by system \mathbf{T}_{i-1} , let the erroneous output from the system be defined as

$$\tilde{\mathbf{b}}_{i-1} = \mathbf{b}_{i-1} + \mathbf{e}, \quad (68)$$

where \mathbf{e} is additive error and \mathbf{b}_{i-1} is the correct result. The next MB processing system, \mathbf{T}_i , uses the incorrect result producing,

$$\begin{aligned} \tilde{\mathbf{b}}_i &= \mathbf{T}_i \tilde{\mathbf{b}}_{i-1} \\ &= \mathbf{T}_i (\mathbf{b}_{i-1} + \mathbf{e}). \end{aligned} \quad (69)$$

The corresponding signature computed using the erroneous MB is

$$\begin{aligned} \tilde{\mathbf{s}}_i &= \mathbf{s}_{i-1} + \mathbf{D}_i (\mathbf{b}_{i-1} + \mathbf{e}) \\ &= \mathbf{s}_{i-1} + \mathbf{D}_i \mathbf{b}_{i-1} + \mathbf{D}_i \mathbf{e} \\ &= \mathbf{s}_i + \mathbf{D}_i \mathbf{e} \\ &= \mathbf{s}_i + \mathbf{G} (\mathbf{T}_i - \mathbf{I}) \mathbf{e} \\ &= \mathbf{s}_i + \mathbf{G} \mathbf{T}_i \mathbf{e} - \mathbf{G} \mathbf{e} \pmod{2^l}. \end{aligned} \quad (70)$$

Direct evaluation of the MB result yields the following validation signature:

$$\begin{aligned} \tilde{\mathbf{s}}'_i &= \mathbf{G} \tilde{\mathbf{b}}_i \\ &= \mathbf{G} \mathbf{T}_i \mathbf{b}_{i-1} + \mathbf{G} \mathbf{T}_i \mathbf{e} \\ &= \mathbf{s}_i + \mathbf{G} \mathbf{T}_i \mathbf{e} \pmod{2^l}. \end{aligned} \quad (71)$$

A comparison of (70) and (71) reveals the extra term $-\mathbf{G} \mathbf{e}$ in the former, indicating that although an error in the MB processing path will affect both the MB and signature processing systems, the signature will be incoherent with the MB as long as the error is not in the null space of the signature generating matrix. Therefore, the signature will diverge from its counterpart MB whenever errors are present in the MB signal, with high probability. The divergence can be effectively used for error detection.

5.4 Prediction Signature Processing

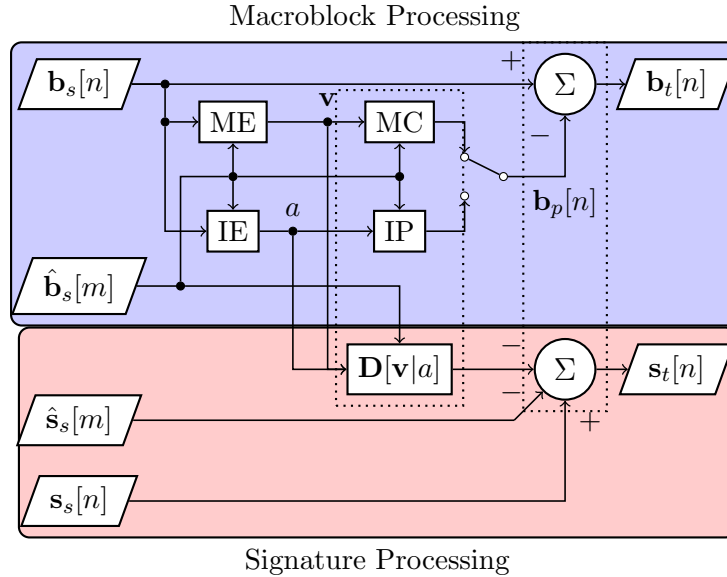


Figure 26: Prediction Signature Processing Model Overview

Prediction signature processing is performed by modeling the entire prediction system as two, sequential, linear transforms, emphasized in Figure 26. The first modeled system consists of the MC or IP system, depending on the type of prediction being used. The second modeled system is the subtraction operation where the predicted MB is subtracted from the sampled one. An analogous signature processing system is shown in Figure 26 for both MB processing systems. The ME and IE systems analyze the sampled MB to select the appropriate prediction mode and communicate the selected mode to the MC or IP system, respectively. Differing modes of prediction imply differing linear transforms to be used by the MC/IP system. Likewise, the selected mode (\mathbf{v} or a) is communicated to the differential signature system so an appropriate linear transform can be selected. The transforms, therefore, depend on the prediction mode, and the generating matrix, \mathbf{G} . Once \mathbf{G} is selected, each \mathbf{D} can be precomputed and cached to conserve computational effort. Each \mathbf{D} matrix is of the same dimensions as \mathbf{G} , therefore, storing a number of the transforms is feasible.

The residual MB is defined as

$$\begin{aligned}\mathbf{b}_t[n] &= \mathbf{b}_s[n] - \mathbf{b}_p[n] \\ &= \mathbf{b}_s[n] - \mathbf{T}\hat{\mathbf{b}}_s[m],\end{aligned}\tag{72}$$

where \mathbf{T} is the (linear transform modeled) prediction matrix.

The first modeled system (MC/IP) operates strictly on the previously encoded MB, $\hat{\mathbf{b}}_s[m]$, given the selected prediction mode. The resulting prediction is defined as

$$\mathbf{b}_p[n] = \mathbf{T}\hat{\mathbf{b}}_s[m],\tag{73}$$

where \mathbf{T} is the prediction transform used by either the MC system or the IP system. The value of \mathbf{T} varies depending on the selected prediction mode. The associated, processed signature is defined as

$$\mathbf{s}_p[n] = \hat{\mathbf{s}}_s[m] + \mathbf{D}\hat{\mathbf{b}}_s[m] \pmod{2^l},\tag{74}$$

where $\hat{\mathbf{s}}_s[m]$ is the signature associated with the stored MB, $\hat{\mathbf{b}}_s[m]$, and \mathbf{D} is the differential signature system associated with the selected MB prediction transform. The derivation of \mathbf{T} and \mathbf{D} are described in the following subsections.

The second modeled system in Figure 26 is a subtraction operation that operates on the prediction MB and the sampled MB. The resulting residual MB is defined as

$$\mathbf{b}_t[n] = \mathbf{b}_s[n] - \mathbf{b}_p[n].\tag{75}$$

Because the subtraction operation is a linear operator, a differential signature transform is not needed. Instead, the same linear operator can be applied to the corresponding signatures. The residual signature is defined as

$$\begin{aligned}\mathbf{s}_t[n] &= \mathbf{s}_s[n] - \mathbf{s}_p[n] \\ &= \mathbf{s}_s[n] - \hat{\mathbf{s}}_s[m] - \mathbf{D}\hat{\mathbf{b}}_s[m] \pmod{2^l},\end{aligned}\tag{76}$$

which is the implementation shown in Figure 26.

Although the prediction signature processing definition in (76) works well for many simple prediction modes that base predictions on a single MB, there are also prediction modes that use information from multiple stored MBs to create a single prediction. To accommodate these prediction modes, (72) is redefined as

$$\mathbf{b}_t[n] = \mathbf{b}_s[n] - \sum_m \mathbf{T}[m - n] \hat{\mathbf{b}}_s[m], \quad (77)$$

allowing the residual MB to be described as a sum of multiple predictions. Valid values for m depend on the type of prediction being used and are described in detail in the following subsections. Similarly, (76) is redefined as

$$\mathbf{s}_t[n] = \mathbf{s}_s[n] - \sum_m \hat{\mathbf{s}}_s[m] - \sum_m \mathbf{D}[m - n] \hat{\mathbf{b}}_s[m] \pmod{2^l}. \quad (78)$$

This allows for more complex prediction modes to be used, such as an MC prediction that spans multiple (up to 4) MBs. The details of the prediction transformations for intra and inter prediction are described in detail in Section 5.4.1 and Section 5.4.2, respectively.

5.4.1 Intra Prediction System Design

The H.264 intra prediction system, as a practical example, creates a prediction for an MB using only previously encoded, neighboring MBs immediately above, left, and diagonally above and left of the MB currently being encoded. Within these MBs available to be used for prediction, only the pixels bordering the MB being encoded are used to generate a prediction. Operations performed on these pixel values are generally either a replication or averaging operation—both can be easily modeled as linear transforms. The prediction modes examined (as defined by [1]) are horizontal projection, vertical projection, and DC (averaging). Other prediction methods are available in the H.264 standard but have not been explicitly referenced for conciseness.

Horizontal Prediction The horizontal intra prediction mode utilizes the rightmost column of pixels from the reference block immediately to the left of the MB currently

being encoded. These pixels are projected to fill each row of the prediction with the same value. In terms of a linear transformation, the elements of \mathbf{T} can be expressed as

$$t_{k,l}[w] = \begin{cases} \delta \left[\left[\frac{k}{N_B} \right] N_B - l \right] & \text{if } w = -1 \\ 0 & \text{otherwise,} \end{cases} \quad (79)$$

where k and l are the row and column indexes for the matrix, and N_B is the horizontal dimension of an MB.

As a hypothetical example, consider the horizontal prediction transform for an MB of size $L = 2$ and a single color plane. The hypothetical MB to be transformed is represented by the raster ordered vector,

$$\hat{\mathbf{b}}_s[n-1] = \begin{bmatrix} 208 & 32 & 231 & 233 \end{bmatrix}^T. \quad (80)$$

From (79), the prediction transformation for the 2×2 MB is

$$\mathbf{T}[-1] = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \quad (81)$$

The horizontally predicted MB is

$$\begin{aligned} \mathbf{b}_p[n] &= \mathbf{T}[-1] \hat{\mathbf{b}}_s[n-1] \\ &= \begin{bmatrix} 32 & 32 & 233 & 233 \end{bmatrix}^T. \end{aligned} \quad (82)$$

Assuming the pseudo-randomly generated binary generating matrix,

$$\mathbf{G} = \begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 \end{bmatrix}, \quad (83)$$

the stored signature associated with the stored MB (assuming no prior errors) is

$$\begin{aligned} \hat{\mathbf{s}}_s[n-1] &= \mathbf{G} \hat{\mathbf{b}}_s[n-1] \pmod{2^8} \\ &= \begin{bmatrix} 240 & 183 \end{bmatrix}^T. \end{aligned} \quad (84)$$

From (64), the differential signature transform associated with the horizontal MB prediction transform is

$$\begin{aligned} \mathbf{D}[-1] &= \mathbf{G}(\mathbf{T}[-1] - \mathbf{I}) \\ &= \begin{bmatrix} 0 & 0 & -1 & 1 \\ -1 & 1 & -1 & 1 \end{bmatrix}. \end{aligned} \quad (85)$$

Using (85), the signature associated with the predicted MB is computed as

$$\begin{aligned} \mathbf{s}_p[n] &= \hat{\mathbf{s}}_s[n-1] + \mathbf{s}_d[n] \pmod{2^8} \\ &= \hat{\mathbf{s}}_s[n-1] + \mathbf{D}[-1]\hat{\mathbf{b}}_s[n-1] \pmod{2^8} \\ &= \begin{bmatrix} 242 & 9 \end{bmatrix}^\top. \end{aligned} \quad (86)$$

The signature can be verified by directly evaluating the signature of the calculated prediction and comparing it to the calculated signature:

$$\begin{aligned} \mathbf{s}'_p[n] &= \mathbf{G}\mathbf{b}_p[n] \\ &= \begin{bmatrix} 242 & 9 \end{bmatrix}^\top \\ &= \mathbf{s}_p[n], \end{aligned} \quad (87)$$

This confirms (with high probability) the prediction was computed without error.

Vertical Prediction Vertical prediction for data processing is very similar to horizontal intra prediction, projecting the bottom row of pixel values down from the MB immediately above the current MB, and can be modeled as

$$t_{k,l}[w] = \begin{cases} \delta[l-1-M_B N_B + M_B - ((k-1) \bmod M_B)] & \text{if } w = -N_F \\ 0 & \text{otherwise,} \end{cases} \quad (88)$$

where M_B is the vertical dimension of an MB measured in pixels, and N_F is the frame width measured in MBs. The prediction transform for a hypothetical 2×2 MB with

$w = -N_F$ is

$$\mathbf{T} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \quad (89)$$

DC Prediction The DC intra prediction mode averages the values of the immediately neighboring MBs above and to the left of the MB being encoded. The average value is used as the prediction value for all pixels in the predicted MB. The prediction transform can be expressed as

$$t_{k,l}[w] = \begin{cases} \frac{1}{cM_B} \delta[(l - M_B + 1) \bmod M_B] & \text{if } w = -1 \\ \frac{1}{cM_B} \delta \left[\left\lfloor \left\lfloor \frac{l - M_B^2 + M_B}{M_B} \right\rfloor \right\rfloor \right] & \text{if } w = -N_F \\ 0 & \text{otherwise,} \end{cases} \quad (90)$$

where c is the number of MBs available for prediction (0, 1, or 2). If $w = -1$, the reference MB is immediately to the left of the current MB. If $w = -N_F$, the reference MB is immediately above the current MB. If the current MB being encoded does not have an MB to its top and/or left, c is evaluated as 1 or 0. For example, if the MB currently being encoded is the second MB in the frame, only the MB immediately to the left is available for DC prediction, yielding

$$t_{k,l}[-1] = \frac{1}{2} \delta[(l - 1) \bmod 2]$$

$$\mathbf{T}[-1] = \frac{1}{2} \begin{bmatrix} 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 \end{bmatrix}, \quad (91)$$

for an MB with dimensions of 2×2 . All other cases for this example would produce a zero matrix. From (91), it is shown that multiplication by the hypothetical transform

will produce a vector of identical elements, equal to the mean of the second and fourth elements in the input vector. These two positions correspond with the rightmost column of an MB in matrix format. Using (90), it is possible to model the DC prediction mode as a linear transform, and an associated signature transform is created by using (64).

Other Prediction Modes There are a total of nine intra prediction modes defined for H.264 [1]. The other modes are similar to the three previous examples—each predicted pixel value is computed as a linear combination of stored pixel values. Therefore, for each intra prediction mode, there exists a prediction transformation matrix. This, in turn, guarantees the existence of the differential signature transformation.

5.4.2 Inter Prediction System Design

Like intra prediction, the goal of inter prediction is to produce a prediction of the current sampled MB, $\mathbf{b}_s[n]$, based on a previously encoded MB(s), $\hat{\mathbf{b}}_s[m]$. The reference MBs for inter prediction are selected from a previously encoded frame, rather than the current frame. The prediction is a group of contiguous pixels covering the same area as one MB. The group of pixels may come from up to four different MBs in a previous frame as shown in Figure 27. The spatial relation of nine MBs are shown, with the center MB occupying the same relative frame position as the current MB being encoded. The prediction, $\mathbf{b}_p[n]$ depends on the selected frame used for prediction and the MV, \mathbf{v} . In the figure, the indices for $\hat{\mathbf{b}}_s$ represent the vertical and horizontal position of the reference MB, relative to the current MB, measured in MBs.

The inter prediction computation can be interpreted as a sum of shifted reference MBs, and can, therefore, be implemented as a linear transformation. For example, in Figure 27, to produce the upper left section of $\mathbf{b}_p[n]$, the contents of $\hat{\mathbf{b}}_s[0,0]$ should be masked to preserve only the data shaded in blue. The masked information should then be shifted up and left by $-\mathbf{v}$. Summing the masked, shifted MBs yields the

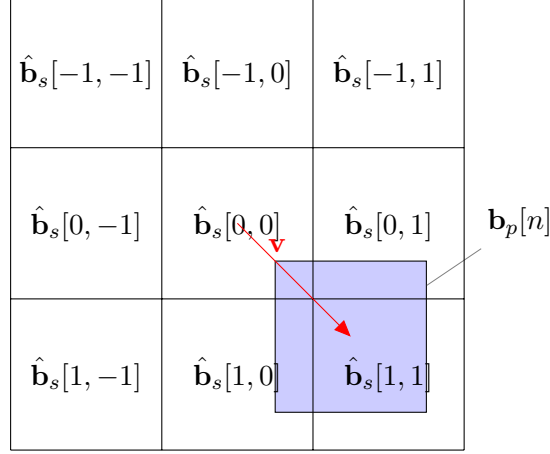


Figure 27: Inter prediction signature processing with coordinates in terms of \mathbf{u} for simplicity.

prediction,

$$\mathbf{b}_p[n] = \sum_m \mathbf{T}[\mathbf{u}, \mathbf{v}] \hat{\mathbf{b}}_s[m], \quad (92)$$

where

$$\begin{aligned} u_1 &= m \bmod N_F - n \bmod N_F \\ u_2 &= \left\lfloor \frac{m \bmod M_F N_F - n \bmod M_F N_F}{N_F} \right\rfloor. \end{aligned} \quad (93)$$

In the preceding equation, M_F is the height of each frame in the video sequence measured in MBs. The frame coordinates of the reference MB relative to the MB currently being processed are represented by \mathbf{u} . The inter prediction transform depends on the MV, \mathbf{v} , in addition to the relative position of each reference MB. The general linear prediction transform is defined as

$$\mathbf{T}[\mathbf{u}, \mathbf{v}] = \mathbf{S}[M_B u_1 - v_1, N_B u_2 - v_2], \quad (94)$$

where \mathbf{S} is the shifting and masking matrix which is defined as

$$\mathbf{S}[y, x] = \text{sft}(\text{repd}(\text{sft}(\mathbf{I}_{M_B}, x), M_B), y M_B). \quad (95)$$

The function, $\text{sft}(\mathbf{A}, b)$ shifts the rows of some matrix \mathbf{A} down by b , and \mathbf{I}_{M_B} is the identity matrix of size M_B . Empty rows are filled with zeros. The function, $\text{repd}(\mathbf{A}, b)$ creates a block diagonal matrix of \mathbf{A} repeated b times.

The function of the inter prediction transform is to shift the pixel values of each contributing reference MB into non-overlapping regions, such that the sum of all transformed reference MBs is $\mathbf{b}_p[n]$. For all reference MBs in Figure 27 (including the ones outside the range shown) that are not partially overlapped by the prediction MB, the transform is zero and need not be computed.

As a hypothetical example, the inter prediction transform for a 2×2 MB with frame size 7×8 (in MBs) and MV, $\mathbf{v} = [1 \ 1]^T$ is presented. The index of the MB being encoded is $n = 65$ while the reference MB under consideration is $m = 18$. The relative location of the reference MB is directly below and right of the MB being encoded. The relative reference MB coordinates are $\mathbf{u} = [1 \ 1]^T$, and the transform for this case is

$$\begin{aligned}
\mathbf{T} [[1 \ 1]^T, [1 \ 1]^T] &= \mathbf{S} [M_B - 1, N_B - 1] \\
&= \mathbf{S} [1, 1] \\
&= \text{sft}(\text{repd}(\text{sft}(I_2, 1), 2), 2) \\
&= \text{sft}(\text{repd}(\begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix}, 2), 2) \\
&= \text{sft}\left(\begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}, 2\right) \\
&= \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}. \tag{96}
\end{aligned}$$

If the reference MB at index 18 is $\hat{\mathbf{B}}_s[18] = \begin{bmatrix} 208 & 33 \\ 231 & 233 \end{bmatrix}$, the associated prediction contribution is

$$\begin{aligned}
\tilde{\mathbf{b}}_p[65] &= \mathbf{T} [[1 \ 1]^T, [1 \ 1]^T] \hat{\mathbf{b}}_s[18] \\
&= [0 \ 0 \ 0 \ 208]^T \\
\tilde{\mathbf{B}}_p[65] &= \begin{bmatrix} 0 & 0 \\ 0 & 208 \end{bmatrix}. \tag{97}
\end{aligned}$$

The upper left pixel in the reference MB is selected and shifted down and right. When summed with the other relevant predictions as shown in (92), the complete prediction MB is produced.

5.5 Transform Signature Processing

The transform system does not operate on previously encoded information. Therefore, a notation change is introduced in this section for simplicity. The index for the MBs and signatures are dropped—the index of n is implied.

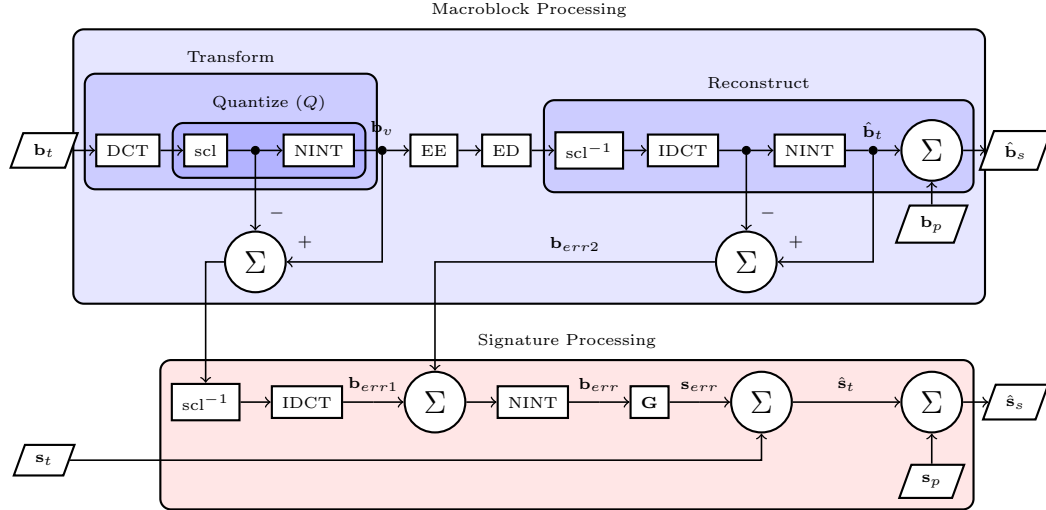


Figure 28: Transform signature processing design. Two sources of rounding error are captured, inverse transformed, and added to the residual signature to form the reconstructed residual signature.

Rather than attempting to perform signature processing for every subsystem within the transform and reconstruction systems, the difference between a residual MB, \mathbf{b}_t , and its reconstructed version, $\hat{\mathbf{b}}_t$ is observed. The transform performed by the DCT system is lossless—its inverse counterpart reconstructs the original information exactly. However, the quantization operation creates quantization error that causes the reconstructed signal to differ slightly from the original signal, and cannot be modeled as a linear transform. The proposed signature processing design for the transform system (and part of the reconstruction system) shown in Figure 28 determines the quantization error from the MB transform and decoding operation and adds the signature associated with the error to \mathbf{s}_t to produce $\hat{\mathbf{s}}_t$. The previously computed \mathbf{s}_p can then be added to produce the reconstructed signature, $\hat{\mathbf{s}}_s$. Essentially, the quantization error is captured and added into the signature processing system.

There are two sources of quantization error. Both are committed to the MB data with a rounding operation shown by the nearest integer (nint) operation in Figure 28. Without the rounding operation, there is no data loss. Therefore, a reconstructed MB can be defined in terms of quantization error by

$$\begin{aligned}\hat{\mathbf{b}}_s &= \hat{\mathbf{b}}_t + \mathbf{b}_p \\ &= \mathbf{b}_t + \mathbf{b}_{err1} + \mathbf{b}_{err2} + \mathbf{b}_p,\end{aligned}\tag{98}$$

where \mathbf{b}_{err1} and \mathbf{b}_{err2} are the fractional rounding errors added during the transform and reconstruction process shown in Figure 28. Since adding the error to the MB signal is a linear operation, the signature generated from the quantization error can be directly added to the residual signature:

$$\begin{aligned}\hat{\mathbf{s}}_s &= \mathbf{s}_t + \mathbf{s}_{err} + \mathbf{s}_p \\ &= \mathbf{s}_t + \mathbf{G} [\mathbf{b}_{err1} + \mathbf{b}_{err2}] + \mathbf{s}_p \pmod{2^l}.\end{aligned}\tag{99}$$

The prediction signature, \mathbf{s}_p , is calculated using (74).

The same quantization error that defines the difference between \mathbf{b}_t and $\hat{\mathbf{b}}_t$ also defines the difference between \mathbf{b}_s and $\hat{\mathbf{b}}_s$:

$$\hat{\mathbf{b}}_s = \mathbf{b}_s + \mathbf{b}_{err} \pmod{2^l}.\tag{100}$$

Thus, it may seem practical to validate the entire encoding system by adding the quantization signature error \mathbf{s}_{err} to each originally generated signature, \mathbf{s}_s , to validate each reconstructed MB. However, such a design leaves the MB prediction systems unprotected. Since the computed prediction is first subtracted from the sampled MB and later added during reconstruction, erroneous predictions would be masked and not detected. For this reason, the prediction system must be modeled as indicated in Figure 26, but the transform system can be simplified as shown in Figure 28, eliminating the need to model the nonlinear systems exactly.

5.6 Error Detection and Mitigation

Error detection in the proposed encoder design is performed by comparing each MB to its corresponding signature after prediction has been performed as shown in Figure 25. The comparison point is selected to ensure protection of all systems in the video encoder, including the prediction systems. After prediction has completed for a given MB, a validation signature is created by multiplying the residual MB by the generating matrix:

$$\mathbf{s}'_t[n] = \mathbf{G}\mathbf{b}_t[n] \pmod{2^l}. \quad (101)$$

Error detection is calculated by comparing the validation signature to the residual signature using an XOR operation. The error detection indicator is defined as

$$err[n] = \mathbf{s}_t[n] \oplus \mathbf{s}'_t[n]. \quad (102)$$

An error is indicated if $err \neq 0$. Due to the cyclical nature of data in the video encoder, if an error is detected, there are multiple potential sources of the error. The error could be induced by the prediction system while computing the current MB, or the error could be a result of incorrectly processing any of the MBs used as reference for the prediction. All potential sources of the detected error must be addressed to prevent the effects of the error from propagating.

Detected errors are mitigated by preventing MBs containing errors from being used as a reference for predicting any future MBs. More conventional thinking may motivate the mitigation strategy of reprocessing the MBs possibly containing errors. However, as error rates increase, reprocessing becomes impractical as it demands many times the typical amount of processing for the same amount of information. Instead, reference MBs and the MB containing the detected error are marked as unavailable for prediction. Reference MBs are marked as unavailable because a detected error may be a result of prediction operations on the current MB or transform operations on the reference MBs. As a result, the effects of detected errors are quarantined

and not allowed to propagate. Artifacts present in the decoded video sequence will be transient and will be limited to a single MB. If there are no available references from which to predict an MB, a reference-free prediction mode is selected by the encoder. In the case of H.264, this can be achieved by beginning a new slice with the MB in question and selecting the DC intra prediction mode.

Although all the MB and signature processing systems can be implemented with unreliable hardware, the XOR comparison and control system need to be implemented on reliable hardware—reliable hardware is not required for the processing of signature, in general. There are many ways of making the control system reliable, including hardening the circuit or using separate hardware. Despite the need for reliable hardware, the overwhelming majority of operations performed for video encoding can be performed on unreliable hardware. Guaranteeing the reliability of the comparator and control system is a much more manageable task compared to the entire video encoder (which is the current solution).

5.7 Results

5.7.1 Experimental Setup

To evaluate the effectiveness of the proposed PISP design, software based simulations were performed so that error rates could be controlled. The H.264/AVC JM Reference Software [71] was augmented to implement PISP functionality and error injection capability. Error modeling was performed by randomly flipping bits at the output of each data operation (multiply, add, etc.) with a controlled BER. The BER used in all experiments refers to the probability of a given bit of the output of an add operation. For all experiments, the same tests are performed on the proposed encoder design as well as the conventional design as a baseline.

Both the implemented PISP and conventional video encoders produce an H.264

protocol compliant bit-stream. The encoders can perform both intra and inter prediction, however, the intra prediction modes were limited to those listed in Section 5.4.1. The inter prediction range was limited to maximum MV component magnitude of 16 with a maximum of 5 reference frames. Sub-pixel motion estimation was not supported, and the fast full method for ME was selected due to potential unexpected behavior of more complex algorithms in the presence of errors. Experiments on the conventional encoder were conducted with a variable IDR period. Additionally, experiments for both encoder designs varied BER and QP values. The set of restricted prediction modes results in a higher than typical bit-rate, but provides an accurate baseline for comparison between conventional and PISP encoding.

For all PISP design experiments performed, the signature generating matrix, \mathbf{G} , was a pseudo-random binary matrix with four rows, producing a 4-dimensional vector with each element consisting of 8 bits. The probability of each matrix element being equal to 1 is 0.33, with the constraint that each column of \mathbf{G} must contain at least a single 1 value. Dimensionality can be increased or decreased to modify the likelihood of error detection, however, since recovery is not a goal of the proposed design, the 4-byte signature suffices to provide adequate error detection capability for a single MB of data (384 bytes).

Six well known video test sequences were used as inputs for all experiments performed: city, container, crew, foreman, harbour, and soccer. All of the videos were limited to 300 frames in CIF, *YV12* format. All individual color samples were 8-bit integers. All experiments were performed on each of the test video sequences independently, and the statistical results of the combined experiments are presented.

The BER in all experiments was varied from 10^{-11} to 10^{-6} while processing the defined benchmark video sequences. This allows for a broad coverage of operating conditions ranging from practically no errors, to errors so frequent the hardware is

practically inoperable. Error rates greater than 10^{-6} produced video output impractical for use.

5.7.2 Experimental Results

The performance of the proposed design is first evaluated against the conventional H.264 design by examining the quality of a decoded signal with a fixed QP of 16 and a varying bit error rate. The results in Figure 29 indicate nominal performance for the conventional encoder design when BER is very low. As the BER increases, the quality of conventionally encoded sequences degrades quickly. The quality of the PISP design also degrades as the BER increases, but the rate of degradation is preferable to that of the conventional design. Reducing the IDR period of the conventional encoder produces a better quality signal, though this comes at a higher cost in terms of compression (discussed later).

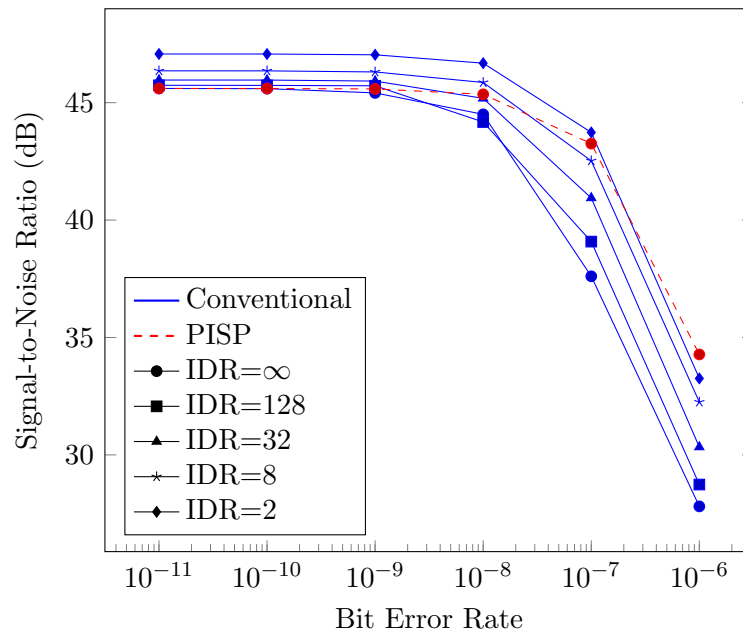


Figure 29: Mean signal quality (relative to original image) of PISP method and conventional method for various IDR periods

To analyze the performance of the proposed design under various QP settings, the video sequences were encoded using the conventional encoder with a fixed IDR period,

while varying the QP setting. The same experiments were performed on the PISP encoder, and the results are shown in Figure 30. The quality of both conventionally and PISP encoded sequences decreases with lesser QP values or greater BER. But in each case, the quality of the PISP encoded sequences are less affected by greater BER values.

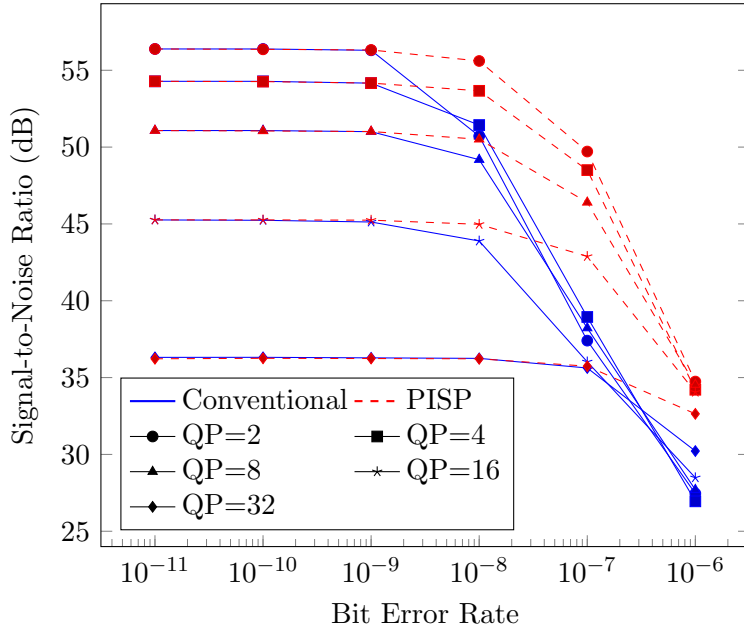


Figure 30: Mean quality of various QP values for PISP and conventionally encoded frames with an IDR period of ∞

To evaluate the power performance of the proposed design, the average amount of time required to encode each individual frame was measured. The results, shown in Figure 31, indicate the trend of more power being required for longer IDR periods, as this requires for more inter encoded frames. Each conventional encoding method tends to maintain a relatively constant workload with respect to BER. However, the PISP encoder uses less power with increasing BER. This is because there are fewer valid references from which to perform inter prediction. The PISP encoder does not spend energy performing ME for invalid MBs. Under nominal circumstances ($BER \leq 10^{-11}$), the computational overhead is negligible relative to the higher IDR periods. For much higher BERs, the computation time of the PISP method drops

below the fastest featured conventional method.

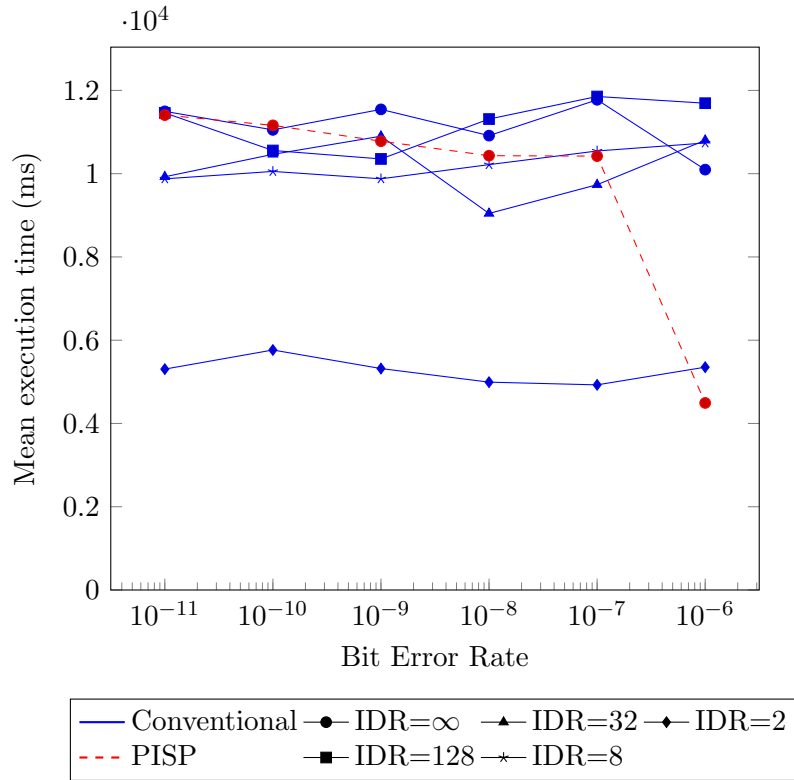


Figure 31: Mean execution time of PISP method and conventional method for various IDR periods

The relative quality improvements produced by the proposed design are offset by decreased compression performance. Figure 32 shows the bit-rate performance (per frame) of the proposed and conventional encoders. Under near-nominal conditions, the PISP design increases the bit-rate by 2.7% compared to the conventional design, while having little impact on quality. However, with a BER of 10^{-7} , bit-rate performance remains relatively unchanged while an increase in quality of 5.7 dB is observed. Even at the very high BER of 10^{-6} with a 5.5% increase over conventional (infinite IDR period) bit-rate, the PISP design increases mean signal quality by 6.5 dB.

Mean quality measurements do not demonstrate the change of video quality over

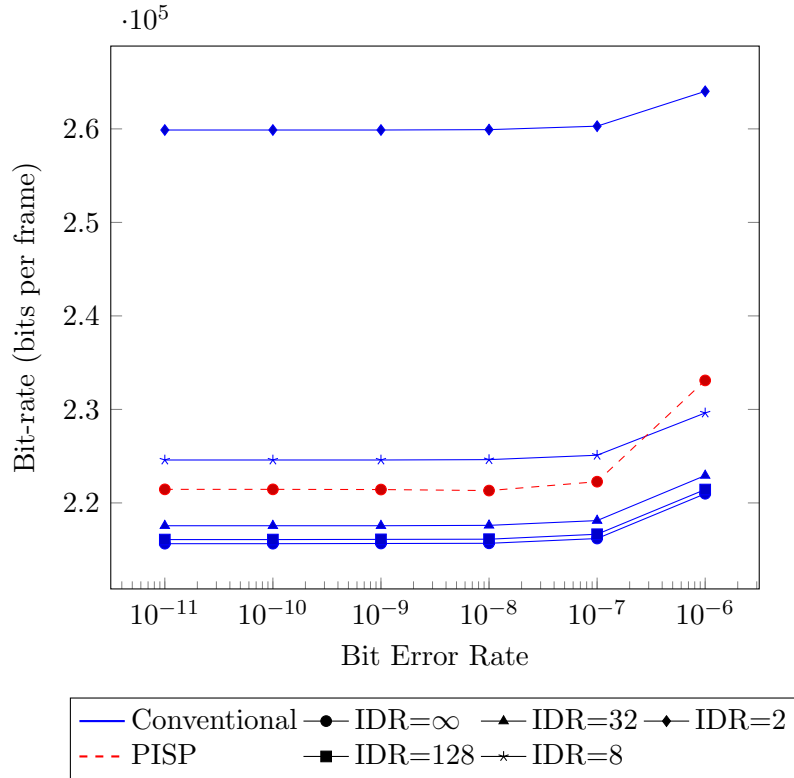


Figure 32: Mean bit-rate of PISP method and conventional method for various IDR periods

time. In Figure 33, the quality outcome of the experiments is shown as a function of time. Without the forced memory refresh, the conventionally encoded sequences continually accumulated errors resulting in a continually decreasing quality for $BER > 10^{-9}$. Although the BER determined the mean quality of the PISP encoded sequences, the quality remained more constant over time, and did not degrade as the conventionally encoded sequences did.

The effect of quality degradation over time can be observed in Figure 34. Using the foreman sequence as a benchmark, the images in Figs. 34 (a–e) and Figs. 34 (f–j) represent regularly-spaced, decoded frames that were encoded conventionally and with PISP, respectively. The accumulation of artifacts is evident in the conventionally encoded frames, while the quality of the PISP encoded frames remains relatively consistent. The artifacts observed in the PISP encoded frames are due to errors

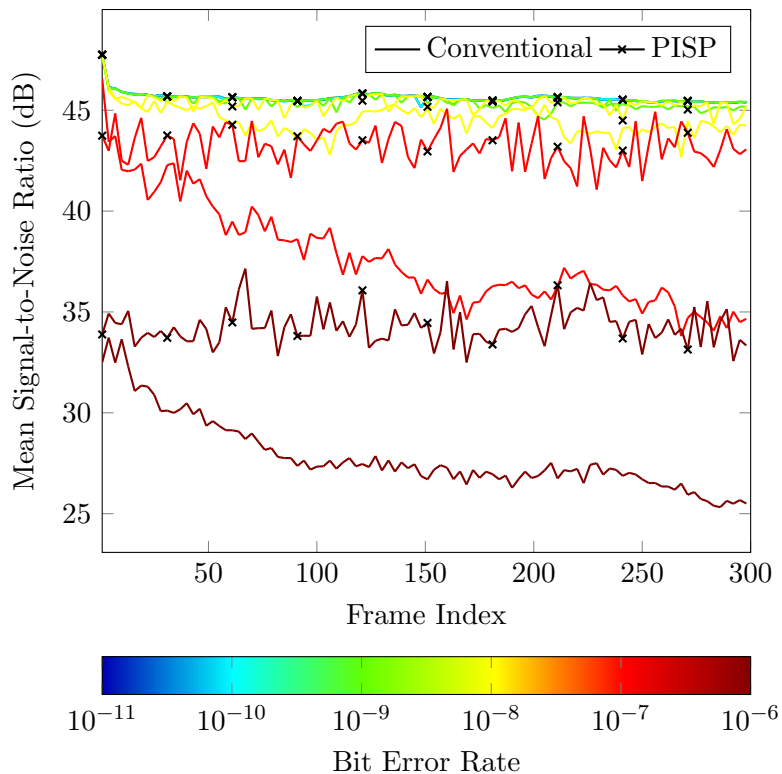


Figure 33: Mean quality of aggregated video sequences with varying BER as a function of time with a fixed QP of 16 and a IDR period of ∞ .

induced while processing the currently displayed frame only. The artifacts observed in the conventionally encoded frames are a result of not only errors induced while processing the displayed frame, but error induced in prior frames as well.

5.8 Conclusion

Traditionally, video encoding hardware is designed for the worst case to ensure predictable operation. Parallel independent signature processing makes it possible to perform reliable video encoding on unreliable hardware. By detecting errors as they occur in the encoding hardware, the quality of the signal is allowed to gracefully degrade, rather than catastrophically failing with increasing probability of computational error. The proposed PISP design is capable of reducing the error rate in an encoded video sequence by preventing errors from propagating from one MB to another. Although some of the error mitigation methods designed in the H.264 standard

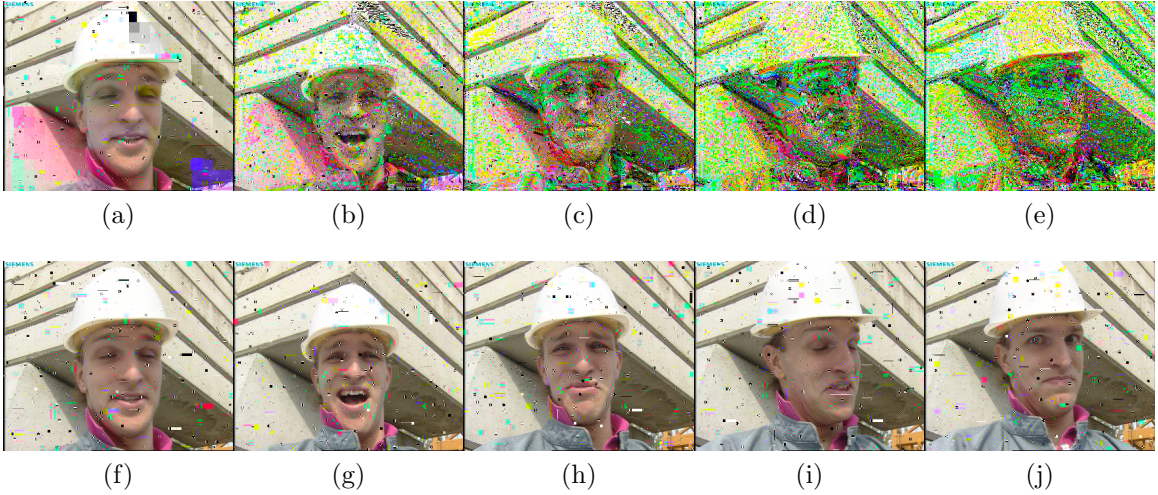


Figure 34: Qualitative comparison of conventional inter prediction encoding (a–e) and PISP inter prediction encoding (f–j) subjected to $BER = 10^{-6}$

are capable of increasing video quality and help to prevent decoder drift, there is an increased cost in terms of compression. The proposed method allows the encoder to adapt without making assumptions about the reliability of the hardware.

The proposed design is not without increased costs. Additional operations are required to compute the signatures used for error detection. However, the extra cost relative to the whole is negligible. If the reliability of the hardware is in doubt, the proposed design can compensate for mistakes if transient artifacts in the decoded signal can be tolerated. It should also be noted that longer execution times may be observed in Figure 31. This is due to the large number of random numbers that needed to be generated for error injection. This represents a constant factor across experiments.

The goal of this work was to demonstrate the advantages of a generalized design that can be applied to any MCHVE. Due to the complex nature of video encoders, only a subset of the prediction methods could be performed. Future goals include obtaining more precise results by including more prediction methods, and evaluating a more recent encoding standard, such as HEVC. Since differential signature transforms

can be precomputed and stored once a generating matrix is selected, the demand on memory—in terms of space and reliability—is an necessary area of future exploration.

In addition to allowing for the use of unreliable hardware, experimental results show that PISP can effectively allow video encoding hardware to better perform in error-rich environments. High-radiation environments that may cause unrecoverable artifacts in an encoded signal can be mitigated, allowing for a higher quality signal.

Errors encountered during processing do not have to be a product solely of the environment the hardware is asked to operate in. If the hardware is asked to run at a lower than ideal voltage, the same error scenario is possible. PISP allows operating at these levels to be feasible. Often, it may be desirable to preserve battery life at the expense of quality. Long term surveillance is an example of this.

Conventional video encoding implements error concealment by intra encoding frames at regular intervals. The shorter the interval, the fewer frames a potential error may be able to propagate through. If the source of errors is primarily in the video encoding hardware (as opposed to the channel), PISP can be implemented to increase the interval between intra encoded frames, allowing for greater compression.

The presented research was initially begun before the release of the latest standard from the ITU-T Video Coding Experts Group, the HEVC standard [69]. The general principle of the proposed design is transferable to the HEVC standard. Signature difference transforms can be computed for various prediction methods, and quantization error can be captured from the transform coding system to process signatures. However, it is not clear how best to segment the regions of each frame to be associated with individual signatures. Additional work needs to be done to determine if associating signatures with fixed regions across frames is best, or perhaps associating signatures with the variable sized coding blocks. Additionally, it needs to be determined whether or not the range in coding block size will lead to an increased overhead cost of maintaining signatures that will make the design impractical.

The proposed design has the potential to extend error correction ability through the channel to the decoder. Depending on the type of configuration, stored signatures may be packaged in the bit stream along with the MB, giving the decoder an extra tool for detecting errors and potentially requesting retransmission of information or invoking error concealment methods. However, this would likely necessitate changes in the H.264 standard. The proposed PISP design is implemented entirely on the encoder, and is therefore, compliant with the H.264 standard.

CHAPTER VI

CONCLUSION

6.1 Conclusions

The increasingly large volume of information associated with video processing presents a challenging problem, especially for energy constrained embedded systems. Traditional methods of sampling as much information as possible and using digital systems to refine the collected information down to its useful meaningful content are increasingly difficult to implement in energy and power constrained systems. The presented research demonstrates adaptive methods of dealing with video data that extend beyond the typical design abstraction layers. This allows the hardware and algorithm to more closely adapt exerted effort based on the information content of the signal relative to the high-level application goals.

The presented research demonstrates the ability of video processing systems to reduce power consumption by focusing processing effort on regions of saliency. The definition of a salient region varies from one high-level application to the next, but if a metric is found that can adequately describe the quality of the signal or saliency, it may be used to control sampling and throughput of the signal being processed.

The research shows that there is a trade-off between adaptable throughput and processed signal quality. There tends to not be just one acceptable output for a processed video signal, but a set of acceptable signals that have different associated qualities. The research shows that the quality of the signal can be considered when determining how to scale the signal throughput. As long as the scaled down signal contains enough information about the impact of the missing information in the quality metric for the processed signal, the effects of over-scaling the signal are made

temporary. In this way, a certain amount of quality is continually being traded for for reduced throughput and power. The amount of information that must be processed to maintain the quality level is then dependent on the the information in the original video signal, and how it aligns to interpreted quality of the processed signal.

Inevitably, while adaptively processing a video sequence, a point in time will be reached where abrupt changes in the signal prevent the system from correctly predicting the correct amount of information to gather and process for a given frame. The designs presented in this research are created in such a way that the impact of these situations is minimized. In the case of video encoding, the retained lower frequency samples help to maximize the signal quality, while it cannot reach its goal. In the case of object detection, a brief lag manifests while the system localizes the unexpected changes. In both cases, the consequences of incorrect predictions produce a controlled, temporary decrease in quality. Many applications are tolerant to these transient quality shifts.

With the proper sensors, the proposed throughput reductions translate directly to dynamic power savings. This is despite the additional computational load created by by the throughput control systems. Even with the additional overhead, significant power savings are demonstrated due to the reduced average throughput. The video processing example is able to reduce dynamic power by downsampling regions that are expected to contain less information, before the primary signal processing operations occur. The object detection processing reduction is performed by switching to different types of models that can cover regions of pixels. In both cases, the overall signal bandwidth is reduced in a very controlled manor.

Despite the scaling of input signals being presented to the video processing hardware, significant changes in the hardware are not necessary. In both examples provided in the research, when the input signal is scaled down, it is done in a way that preserves basic data structures. The described video encoder still processes a block

of pixels with fixed size, however, it is the region that the block is associated with that varies. The processing hardware does not need to know explicitly about this difference. Similarly, the object detection and tracking system hardware processed specific data structures: block-based background models. Although a second model type is introduced, the system operates on each block independently. The blocks may represent different varying regions, but they are still updated and processed the same. Only the control algorithm needs to be aware of the block placement.

In addition to the dynamic power savings possible through scaling throughput, the unique method of predicting the information that needs to be sampled ahead of time allows for even greater potential power savings through DVFS. Instead of trying to predict relatively long-term trends in processing workload and scale voltage and frequency for long periods of time, the same metrics being used to predict how much throughput will be needed can be used to determine how much computational effort will be required for the relatively small time span associated with processing a single frame. The processing throughput scales very closely with the number of fundamental units being processed by the hardware, and the number of units are influenced by the proposed control system.

Additional power savings may still be possible with continued device scaling. However, with continued device scaling comes increased susceptibility to errors. Hence, the importance of the presented research in error tolerance. The research demonstrates the ability to detect errors made by the encoding hardware, though detection is done in software. Due to the high reuse of information for compression in video encoding, the presented research demonstrates a low cost methods for detection and mitigation of hardware errors that does not require reprocessing of data. Overall, this design method can help extend power savings by allowing continued device scaling with reduced consequences.

6.2 Future Work

Future work includes extending the object detection and tracking research to include multi-modal models and color information rather than grayscale alone. The extension to each is fairly straight forward, but the power savings rate may be different.

The presented research on object detection and tracking is based on fixed camera operation. Although there is still a popular need for fixed camera surveillance, there is an increasing demand for moving camera surveillance, including the need to perform object detection and tracking. Extending the presented research to moving cameras is a logical next step.

The error detection research is specific to MCHVEs and specifically uses the H.264 codec for testing and verification. Future work in this area should include details on applying the method specifically to the H.265 protocol. Additionally, the application of the error detection method to other video processing applications should be researched. Error mitigation schemes will likely differ, but translation of the detection method may be expanded.

The application of the proposed throughput reduction methods may also be applicable to machine learning. Given the relatively low-dimensional space that represents the area of concern for many machine learning objectives and the extremely high-dimensional space of sampled video signals, there is potential for the application of the throughput reduction methods to machine learning.

The presented research depends on the ability acquire image information sensed in non-traditional ways. In order to fully implement the proposed systems, new image sensors with fine control over sensing modes would be ideal. However, due to the cost of such new devices, their design and manufacturing may not be feasible. Therefore, more research is warranted in the area of low-power transformation of full-resolution data to a compressed representation. Transformation into the proposed CS bases described in this research may be possible using simple filter banks as the

transformation would only require fixed-point add operations.

REFERENCES

- [1] “Advanced video coding for generic audiovisual services,” Feb. 2014.
- [2] “High efficiency video coding,” Apr. 2015.
- [3] ARAKIDA, H., TAKAHASHI, M., TSUBOI, Y., NISHIKAWA, T., YAMAMOTO, H., FUJIYOSHI, T., KITASHO, Y., UEDA, Y., WATANABE, M., FUJITA, T., TERAZAWA, T., OHMORI, K., KOANA, M., NAKAMURA, H., WATANABE, E., ANDO, H., AIKAWA, T., and FURUYAMA, T., “A 160 mW, 80 nA standby, MPEG-4 audiovisual LSI with 16 Mb embedded DRAM and a 5 GOPS adaptive post filter,” in *Solid-State Circuits Conference, 2003. Digest of Technical Papers. ISSCC. 2003 IEEE International*, pp. 42–476 vol.1, 2003.
- [4] AUSTIN, T., BLAAUW, D., MAHLKE, S., MUDGE, T., CHAKRABARTI, C., and WOLF, W., “Mobile supercomputers,” *Computer*, vol. 37, pp. 81–83, May 2004.
- [5] BAY, H., ESS, A., TUYTELAARS, T., and VAN GOOL, L., “Speeded-Up Robust Features (SURF),” *Computer Vision and Image Understanding*, vol. 110, pp. 346–359, June 2008.
- [6] CANDÈS, E. and WAKIN, M., “An Introduction To Compressive Sampling,” *IEEE Signal Processing Magazine*, vol. 25, no. 2, pp. 21–30, 2008.
- [7] CEVHER, V., SANKARANARAYANAN, A., DUARTE, M. F., REDDY, D., BARANIUK, R. G., CHELLAPPA, R., FORSYTH, D., TORR, P., and ZISSERMAN, A., “Compressive Sensing for Background Subtraction,” in *Computer Vision – ECCV 2008*, no. 5303 in Lecture Notes in Computer Science, pp. 155–168, Springer Berlin Heidelberg, 2008.
- [8] CHANG, I. J., MOHAPATRA, D., and ROY, K., “A Priority-Based 6T/8T Hybrid SRAM Architecture for Aggressive Voltage Scaling in Video Applications,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 21, no. 2, pp. 101–112, 2011.
- [9] CHEN, T.-C., HUANG, Y.-W., and CHEN, L.-G., “Fully utilized and reusable architecture for fractional motion estimation of H.264/AVC,” in *IEEE International Conference on Acoustics, Speech, and Signal Processing, 2004. Proceedings. (ICASSP '04)*, vol. 5, pp. V–9–12 vol.5, May 2004.
- [10] CHEN, T.-C., LIAN, C.-J., and CHEN, L.-G., “Hardware architecture design of an H.264/AVC video codec,” in *Asia and South Pacific Conference on Design Automation, 2006*, pp. 8 pp.–, Jan. 2006.

- [11] CHEN, Y.-H., CHEN, T.-C., TSAI, C.-Y., TSAI, S.-F., and CHEN, L.-G., “Algorithm and Architecture Design of Power-Oriented H.264/AVC Baseline Profile Encoder for Portable Devices,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 19, pp. 1118–1128, Aug. 2009.
- [12] CHEONG, H.-Y., CHONG, I., and ORTEGA, A., “Computation Error Tolerance in Motion Estimation Algorithms,” in *2006 IEEE International Conference on Image Processing*, pp. 3289–3292, 2006.
- [13] CHIEN, S.-Y., HUANG, Y.-W., CHEN, C.-Y., CHEN, H. H., and CHEN, L.-G., “Hardware architecture design of video compression for multimedia communication systems,” *IEEE Communications Magazine*, vol. 43, pp. 122–131, Aug. 2005.
- [14] CHIPPA, V., MOHAPATRA, D., ROY, K., CHAKRADHAR, S., and RAGHUNATHAN, A., “Scalable Effort Hardware Design,” *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 22, no. 9, pp. 2004–2016, 2014.
- [15] CHIU, L.-C., CHANG, T.-S., CHEN, J.-Y., and CHANG, N.-C., “Fast SIFT Design for Real-Time Visual Feature Extraction,” *IEEE Transactions on Image Processing*, vol. 22, no. 8, pp. 3158–3167, 2013.
- [16] CORREA, G., ASSUNCAO, P., AGOSTINI, L., and DA SILVA CRUZ, L., “Complexity control of high efficiency video encoders for power-constrained devices,” *IEEE Transactions on Consumer Electronics*, vol. 57, no. 4, pp. 1866–1874, 2011.
- [17] DADKHAH, M., DEEN, M., and SHIRANI, S., “CMOS Image Sensor With Area-Efficient Block-Based Compressive Sensing,” *IEEE Sensors Journal*, vol. 15, no. 7, pp. 3699–3710, 2015.
- [18] DATAR, M., IMMORLICA, N., INDYK, P., and MIRROKNI, V. S., “Locality-sensitive Hashing Scheme Based on P-stable Distributions,” in *Proceedings of the Twentieth Annual Symposium on Computational Geometry*, SCG ’04, (New York, NY, USA), pp. 253–262, ACM, 2004.
- [19] DERPANIS, K., LEUNG, E., and SIZINTSEV, M., “Fast Scale-Space Feature Representations by Generalized Integral Images,” in *IEEE International Conference on Image Processing, 2007. ICIP 2007*, vol. 4, pp. IV – 521–IV – 524, Sept. 2007.
- [20] DHOOT, C., CHAU, L.-P., CHOWDHURY, S., and MOONEY, V., “Low Power Motion Estimation Based on Probabilistic Computing,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 24, no. 1, pp. 1–14, 2014.
- [21] DONOHO, D., “Compressed sensing,” *IEEE Transactions on Information Theory*, vol. 52, no. 4, pp. 1289–1306, 2006.
- [22] EBRAHIMI, T. and HORNE, C., “MPEG-4 natural video coding – An overview,” *Signal Processing: Image Communication*, vol. 15, pp. 365–385, Jan. 2000.

- [23] GALLANT, M., COTE, G., and KOSENTINI, F., “An efficient computation-constrained block-based motion estimation algorithm for low bit rate video coding,” *IEEE Transactions on Image Processing*, vol. 8, pp. 1816–1823, Dec. 1999.
- [24] GHANBARI, M., “The cross-search algorithm for motion estimation,” *IEEE Transactions on Communications*, vol. 38, pp. 950–953, July 1990.
- [25] GOLDSTEIN, J., PLAT, J. C., and BURGESS, C. J. C., “Redundant Bit Vectors for Quickly Searching High-Dimensional Regions,” in *Deterministic and Statistical Methods in Machine Learning* (WINKLER, J., NIRANJAN, M., and LAWRENCE, N., eds.), no. 3635 in Lecture Notes in Computer Science, pp. 137–158, Springer Berlin Heidelberg, 2005.
- [26] GRABNER, M., GRABNER, H., and BISCHOF, H., “Fast Approximated SIFT,” in *Computer Vision – ACCV 2006* (NARAYANAN, P., NAYAR, S., and SHUM, H.-Y., eds.), vol. 3851 of *Lecture Notes in Computer Science*, pp. 918–927, Springer Berlin Heidelberg, 2006.
- [27] HASHIMOTO, T., KUROMARU, S., MATSUO, M., YASUO, K., MORI-IWA, T., ISHIDA, K., KAJITA, S., OHASHI, M., TOUJIMA, M., NAKAMURA, T., HAMADA, M., YONEZAWA, T., KONDO, T., HASHIMOTO, K., SUGISAWA, Y., OTSUKI, H., ARITA, M., NAKAJIMA, H., FUJIMOTO, H., MICHİYAMA, J., IIZUKA, Y., KOMORI, H., NAKATANI, S., TOIDA, H., TAKAHASHI, T., ITO, H., and YUKITAKE, T., “A 90 mW MPEG4 video codec LSI with the capability for core profile,” in *Solid-State Circuits Conference, 2001. Digest of Technical Papers. ISSCC. 2001 IEEE International*, pp. 140–141, 2001.
- [28] HE, Z., CHENG, W., and CHEN, X., “Energy Minimization of Portable Video Communication Devices Based on Power-Rate-Distortion Optimization,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 18, no. 5, pp. 596–608, 2008.
- [29] HE, Z.-L., TSUI, C.-Y., CHAN, K.-K., and LIOU, M., “Low-power VLSI design for motion estimation using adaptive pixel truncation,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 10, no. 5, pp. 669–678, 2000.
- [30] HEGDE, R. and SHANBHAG, N., “Soft digital signal processing,” *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 9, pp. 813–823, Dec. 2001.
- [31] HEGDE, R. and SHANBHAG, N., “A voltage overscaled low-power digital filter IC,” *IEEE Journal of Solid-State Circuits*, vol. 39, no. 2, pp. 388–391, 2004.
- [32] HUANG, Y.-W., CHEN, T.-C., TSAI, C.-H., CHEN, C.-Y., CHEN, T.-W., CHEN, C.-S., SHEN, C.-F., MA, S.-Y., WANG, T.-C., HSIEH, B.-Y., FANG, H.-C., and CHEN, L.-G., “A 1.3TOPS H.264/AVC single-chip encoder for HDTV applications,” in *Solid-State Circuits Conference, 2005. Digest of Technical Papers. ISSCC. 2005 IEEE International*, pp. 128–588 Vol. 1, Feb. 2005.

- [33] HUANG, Y.-W., HSIEH, B.-Y., CHEN, T.-C., and CHEN, L.-G., “Analysis, fast algorithm, and VLSI architecture design for H.264/AVC intra frame coder,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 15, no. 3, pp. 378–401, 2005.
- [34] JAIN, J. and JAIN, A., “Displacement Measurement and Its Application in Interframe Image Coding,” *IEEE Transactions on Communications*, vol. 29, pp. 1799–1808, Dec. 1981.
- [35] JAVAID, H., SHAFIQUE, M., HENKEL, J., and PARAMESWARAN, S., “Energy-Efficient Adaptive Pipelined MPSoCs for Multimedia Applications,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 33, pp. 663–676, May 2014.
- [36] JUN, D. and JONES, D., “Cascading Signal-Model Complexity for Energy-Aware Detection,” *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 3, pp. 65–74, Mar. 2013.
- [37] KIM, H. and PARK, I.-C., “High-performance and low-power memory-interface architecture for video processing applications,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 11, no. 11, pp. 1160–1170, 2001.
- [38] KIM, M., HWANG, I., and CHAE, S.-I., “A fast VLSI architecture for full-search variable block size motion estimation in MPEG-4 AVC/H.264,” in *Proceedings of the ASP-DAC 2005. Asia and South Pacific Design Automation Conference, 2005.*, vol. 1, pp. 631–634 Vol. 1, Jan. 2005.
- [39] KOGA, T., IINUMA, K., HIRANO, A., IJIMA, Y., and ISHIGURO, T., “Motion-compensated interframe coding for video conferencing,” in *Proceedings of the National Telecommunications Conference*, p. G5.3, 1981.
- [40] KUHN, P., *Algorithms, Complexity Analysis and VLSI Architectures for MPEG-4 Motion Estimation*. Springer US, 1999.
- [41] KUO, H.-C., WU, L.-C., HUANG, H.-T., HSU, S.-T., and LIN, Y.-L., “A Low-Power High-Performance H.264/AVC Intra-Frame Encoder for 1080pHD Video,” *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 19, pp. 925–938, June 2011.
- [42] LEE, Y.-M. and LIN, Y., “Zero-Block Mode Decision Algorithm for H.264/AVC,” *IEEE Transactions on Image Processing*, vol. 18, pp. 524–533, Mar. 2009.
- [43] LI, L., NAWAZ, T., and FERRYMAN, J., “PETS 2015: Datasets and challenge,” in *2015 12th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, pp. 1–6, Aug. 2015.

- [44] LI, R., ZENG, B., and LIOU, M., “A new three-step search algorithm for block motion estimation,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 4, pp. 438–442, Aug. 1994.
- [45] LI, X., WIEN, M., and OHM, J.-R., “Rate-Complexity-Distortion Optimization for Hybrid Video Coding,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 21, no. 7, pp. 957–970, 2011.
- [46] LIN, W., PANUSOPONE, K., BAYLON, D., and SUN, M.-T., “A Computation Control Motion Estimation Method for Complexity-Scalable Video Coding,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 20, no. 11, pp. 1533–1543, 2010.
- [47] LIU, L.-K. and FEIG, E., “A block-based gradient descent search algorithm for block motion estimation in video coding,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 6, pp. 419–422, Aug. 1996.
- [48] LOWE, D. G., “Distinctive Image Features from Scale-Invariant Keypoints,” *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.
- [49] MALVAR, H., HALLAPURO, A., KARCZEWICZ, M., and KEROFISKY, L., “Low-complexity transform and quantization in H.264/AVC,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, pp. 598–603, July 2003.
- [50] MOHAPATRA, D., CHIPPA, V., RAGHUNATHAN, A., and ROY, K., “Design of voltage-scalable meta-functions for approximate computing,” in *Design, Automation Test in Europe Conference Exhibition (DATE), 2011*, pp. 1–6, 2011.
- [51] MOHAPATRA, D., KARAKONSTANTIS, G., and ROY, K., “Significance Driven Computation: A Voltage-scalable, Variation-aware, Quality-tuning Motion Estimator,” in *Proceedings of the 14th ACM/IEEE International Symposium on Low Power Electronics and Design, ISLPED ’09*, (New York, NY, USA), pp. 195–200, ACM, 2009.
- [52] MUTHUKARUPPAN, T., JAVAID, H., MITRA, T., and PARAMESWARAN, S., “Energy-aware synthesis of application specific MPSoCs,” in *2013 IEEE 31st International Conference on Computer Design (ICCD)*, pp. 62–69, 2013.
- [53] NAKAYAMA, H., YOSHITAKE, T., KOMAZAKI, H., WATANABE, Y., ARAKI, H., MORIOKA, K., LI, J., PEILIN, L., LEE, S., KUBOSAWA, H., and OTOBE, Y., “An MPEG-4 video LSI with an error-resilient codec core based on a fast motion estimation algorithm,” in *Solid-State Circuits Conference, 2002. Digest of Technical Papers. ISSCC. 2002 IEEE International*, vol. 1, pp. 368–474 vol.1, 2002.
- [54] NARAYANAN, S., SARTORI, J., KUMAR, R., and JONES, D., “Scalable stochastic processors,” in *Design, Automation Test in Europe Conference Exhibition (DATE), 2010*, pp. 335–338, Mar. 2010.

- [55] NASCIMENTO, J. and MARQUES, J., “Performance evaluation of object detection algorithms for video surveillance,” *IEEE Transactions on Multimedia*, vol. 8, pp. 761–774, Aug. 2006.
- [56] NG, K.-H., PO, L.-M., WONG, K.-M., TING, C.-W., and CHEUNG, K.-W., “A Search Patterns Switching Algorithm for Block Motion Estimation,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 19, pp. 753–759, May 2009.
- [57] NISHIKAWA, T., TAKAHASHI, M., HAMADA, M., TAKAYANAGI, T., ARAKIDA, H., MACHIDA, N., YAMAMOTO, H., FUJIYOSHI, T., MAISUMOTO, Y., YAMAGISHI, O., SAMATA, T., ASANO, A., TERAZAWA, T., OHMORI, K., SHIRAKURA, J., WATANABE, Y., NAKAMURA, H., MINAMI, S., KURODA, T., and FURUYAMA, T., “A 60 MHz 240 mW MPEG-4 video-phone LSI with 16 Mb embedded DRAM,” in *Solid-State Circuits Conference, 2000. Digest of Technical Papers. ISSCC. 2000 IEEE International*, pp. 230–231, 2000.
- [58] NISTER, D. and STEWENIUS, H., “Scalable Recognition with a Vocabulary Tree,” in *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2, pp. 2161–2168, 2006.
- [59] OMOHUNDRO, S. M., *Five Balltree Construction Algorithms*. 1989.
- [60] PO, L.-M. and MA, W.-C., “A novel four-step search algorithm for fast block motion estimation,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 6, pp. 313–317, June 1996.
- [61] ROBUCCI, R., GRAY, J., CHIU, L. K., ROMBERG, J., and HASLER, P., “Compressive Sensing on a CMOS Separable-Transform Image Sensor,” *Proceedings of the IEEE*, vol. 98, no. 6, pp. 1089–1101, 2010.
- [62] ROMBERG, J., “Imaging via Compressive Sampling,” *IEEE Signal Processing Magazine*, vol. 25, no. 2, pp. 14–20, 2008.
- [63] SHAFIQUE, M., BAUER, L., and HENKEL, J., “enBudget: A Run-Time Adaptive Predictive Energy-Budgeting scheme for energy-aware Motion Estimation in H.264/MPEG-4 AVC video encoder,” in *Design, Automation Test in Europe Conference Exhibition (DATE), 2010*, pp. 1725–1730, 2010.
- [64] SHAFIQUE, M., MOLKENTHIN, B., and HENKEL, J., “An HVS-based Adaptive Computational Complexity Reduction Scheme for H.264/AVC video encoder using Prognostic Early Mode Exclusion,” in *Design, Automation Test in Europe Conference Exhibition (DATE), 2010*, pp. 1713–1718, 2010.
- [65] SHEN, L., ZHANG, Z., and LIU, Z., “Adaptive Inter-Mode Decision for HEVC Jointly Utilizing Inter-Level and Spatiotemporal Correlations,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 24, no. 10, pp. 1709–1722, 2014.

- [66] SRINIVASAN, R. and RAO, K., “Predictive Coding Based on Efficient Motion Estimation,” *IEEE Transactions on Communications*, vol. 33, pp. 888–896, Aug. 1985.
- [67] STAUFFER, C. and GRIMSON, W., “Adaptive background mixture models for real-time tracking,” in *Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference On.*, vol. 2, p. 252 Vol. 2, 1999.
- [68] STAUFFER, C. and GRIMSON, W., “Learning patterns of activity using real-time tracking,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, pp. 747–757, Aug. 2000.
- [69] SULLIVAN, G., OHM, J., HAN, W.-J., and WIEGAND, T., “Overview of the High Efficiency Video Coding (HEVC) Standard,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, pp. 1649–1668, Dec. 2012.
- [70] TAKAHASHI, M., HAMADA, M., NISHIKAWA, T., ARAKIDA, H., TSUBOI, Y., FUJITA, T., HATORI, F., MITA, S., SUZUKI, K., CHIBA, A., TERAZAWA, T., SANO, F., WATANABE, Y., MOMOSE, H., USAMI, K., IGARASHI, M., ISHIKAWA, T., KANAZAWA, M., KURODA, T., and FURUYAMA, T., “A 60 mW MPEG4 video codec using clustered voltage scaling with variable supply-voltage scheme,” in *Solid-State Circuits Conference, 1998. Digest of Technical Papers. 1998 IEEE International*, pp. 36–37, 1998.
- [71] TOURAPIS, A., “H.264/AVC Reference Software - JM 18.6,” 2009.
- [72] TSAI, J.-J. and HANG, H.-M., “On the Design of Pattern-Based Block Motion Estimation Algorithms,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 20, pp. 136–143, Jan. 2010.
- [73] TSENG, P.-C., CHANG, Y., HUANG, Y.-W., FANG, H.-C., HUANG, C.-T., and CHEN, L.-G., “Advances in Hardware Architectures for Image and Video Coding - A Survey,” *Proceedings of the IEEE*, vol. 93, pp. 184–197, Jan. 2005.
- [74] VAN DER SCHAAF, A. and VAN HATEREN, J. H., “Modelling the power spectra of natural images: Statistics and information,” *Vision Research*, vol. 36, pp. 2759–2770, Sept. 1996.
- [75] VARATKAR, G. V. and SHANBHAG, N. R., “Energy-efficient Motion Estimation using Error-Tolerance,” in *ISLPED’06 Proceedings of the 2006 International Symposium on Low Power Electronics and Design*, pp. 113–118, Oct. 2006.
- [76] VARATKAR, G. V. and SHANBHAG, N. R., “Variation-Tolerant Motion Estimation Architecture,” in *2007 IEEE Workshop on Signal Processing Systems*, pp. 126–131, Oct. 2007.
- [77] VARATKAR, G. and SHANBHAG, N., “Error-Resilient Motion Estimation Architecture,” *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 16, pp. 1399–1412, Oct. 2008.

- [78] WANG, H., KWONG, S., and KOK, C.-W., “An Efficient Mode Decision Algorithm for H.264/AVC Encoding Optimization,” *IEEE Transactions on Multimedia*, vol. 9, pp. 882–888, June 2007.
- [79] WIEGAND, T., SULLIVAN, G., BJONTEGAARD, G., and LUTHRA, A., “Overview of the H.264/AVC video coding standard,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, pp. 560–576, July 2003.
- [80] YAO, L., FENG, H., ZHU, Y., JIANG, Z., ZHAO, D., and FENG, W., “An architecture of optimised SIFT feature detection for an FPGA implementation of an image matcher,” in *International Conference on Field-Programmable Technology, 2009. FPT 2009*, pp. 30–37, Dec. 2009.
- [81] YAP, S. Y. and MCCANNY, J. V., “A VLSI architecture for variable block size video motion estimation,” *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 51, pp. 384–389, July 2004.
- [82] YILMAZ, A., JAVED, O., and SHAH, M., “Object Tracking: A Survey,” *ACM Comput. Surv.*, vol. 38, no. 4, 2006.
- [83] YU, A., MARTIN, G., and PARK, H., “Fast Inter-Mode Selection in the H.264/AVC Standard Using a Hierarchical Decision Process,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 18, pp. 186–195, Feb. 2008.
- [84] ZENG, H., MA, K.-K., and CAI, C., “Fast Mode Decision for Multiview Video Coding Using Mode Correlation,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 21, pp. 1659–1666, Nov. 2011.
- [85] ZHU, S. and MA, K.-K., “A new diamond search algorithm for fast block-matching motion estimation,” *IEEE Transactions on Image Processing*, vol. 9, pp. 287–290, Feb. 2000.