# Resource Constrained Video Coding Systems

**Waqar Zia**

**Technische Universität München**
Lehrstuhl für Datenverarbeitung

# Resource Constrained Video Coding Systems

**Waqar Zia**

Vollständiger Abdruck der von der Fakultät für Elektrotechnik und Informationstechnik der Technischen Universität München zur Erlangung des akademischen Grades eines

**Doktor-Ingenieurs**

genehmigten Dissertation.

| | |
|---|---|
| Vorsitzender: | Univ.-Prof. Dr. sc. (ETH) Samarjit Chakraborty |
| Prüfer der Dissertation: | 1. Univ.-Prof. Dr.-Ing. Klaus Diepold |
| | 2. Univ.-Prof. Dr.-Ing. Eckehard Steinbach |

Die Dissertation wurde am 27.01.2012 bei der Technischen Universität eingereicht und durch die Fakultät für Elektrotechnik und Informationstechnik am 04.07.2012 angenommen.

# Abstract

This work provides a set of frameworks for computational resource management, applicable for a variety of resource-constrained video communication systems. Such existing systems do not employ computational resource optimizations because of the current lack of understanding of resource usage models for video codecs. This results in suboptimal performance, resource wastage and compatibility issues between communicating devices. The proposed frameworks have a special emphasis on integrability with practically deployed systems, low complexity and performance overhead, scalability, large dynamic range of operation, high accuracy, minimal invasiveness and minimal reliance on empirical performance tuning. The frameworks are integrable with a large variety of the existing rate-distortion optimization strategies. General design principles are also provided that can be applied to other emerging video communication applications and future video codecs as well. The real-time performance of the frameworks for a set of realistic video communication systems is evaluated. Quantitative results are provided, which demonstrate the resulting performance enhancements. The proposed frameworks not only provide optimization, but are also a step forward towards a major goal for video communication applications: to foster compatible communication between devices of varying processing capabilities.

# Zusammenfassung

Diese Arbeit bietet eine Reihe von Rahmenbedingungen für das Management von Rechnerressourcen, die für eine Vielfalt ressourcenbegrenzter Videokommunikationssysteme anwendbar sind. Solche bestehenden Systeme verwenden aufgrund des derzeit mangelnden Verständnisses für Ressourcenverbrauchsmodelle bei Video-Codecs keine Optimierung der Rechnerressourcen. Dies führt zu suboptimaler Leistung und Kompatibilitätsproblemen zwischen kommunizierenden Geräten. Die vorgeschlagenen Rahmenbedingungen legen einen besonderen Schwerpunkt auf geringe Komplexität und Leistungseinbußen, hohe Genauigkeit sowie geringe Invasivität. Die Rahmenbedingungen können in einer Vielzahl bestehender Rate-Distortion- Optimierungsstrategien integriert werden. Es sind quantitative Ergebnisse vorhanden, die die resultierenden Leistungsoptimierungen demonstrieren. Die vorgeschlagenen Rahmenbedingungen sind auch ein Schritt in Richtung eines der Hauptziele bei Videokommunikationsanwendungen: der Förderung einer kompatiblen Kommunikation zwischen Geräten mit unterschiedlichen Verarbeitungskapazitäten.

# Contents


**1 Introduction** **1**
- 1.1 State-of-the-Art Resource Optimization . . . 2
- 1.2 The Emerging Picture of Resource Optimized Video Communications 2
  - 1.2.1 The Changing Realm of Digital Multimedia . . . 3
  - 1.2.2 The Future of Resource Optimized Systems . . . 3
- 1.3 Our Contribution to Resource Optimized Video Communication Systems . . . 4

**2 Preliminaries** **5**
- 2.1 Resource Constrained Video Communication: Formulation . . . 6
  - 2.1.1 The Constrained Resources for Video Communication Systems 6
  - 2.1.2 Computational Resource Constraints of Implementation . . . 6
- 2.2 Video Codecs for Resource Constrained Communication Systems . . 8
  - 2.2.1 Distributed Video Source Coding . . . 8
  - 2.2.2 Block-Based Hybrid Video Codecs . . . 9
  - 2.2.3 Error Concealment for Block-based Video Codecs . . . 15
  - 2.2.4 Discussion . . . 16

**3 Resource Constrained Video Coding: Brief review** **17**
- 3.1 Encoder Optimizations . . . 17
  - 3.1.1 Joint Source-Channel Power Optimizations . . . 18
  - 3.1.2 Joint R-D-C Optimization . . . 20
  - 3.1.3 Motion Estimation Based Optimization . . . 21
  - 3.1.4 Mode Ranking . . . 21
  - 3.1.5 Encoder Resource Usage Prediction . . . 22
  - 3.1.6 Skip Mode Prediction . . . 22
  - 3.1.7 Computational Resource Management of a Video Encoder . . 23
  - 3.1.8 Variable Complexity Transform . . . 24
- 3.2 Decoder Optimizations . . . 24
  - 3.2.1 Decoder Resource Usage Modeling . . . 24
  - 3.2.2 Computational Resource Management by Quality Degradation 25
  - 3.2.3 RDC Optimizations For Streaming Applications Using Generic Complexity Metrics (GCM) . . . 26
- 3.3 Some comments on the reviewed work . . . 27

**4 Resource Constrained Video Coding Systems** **29**
- 4.1 Mobile Video Conversational Applications . . . 29

4.1.1 Error Robustness . . . 30
4.1.2 Performance Metrics . . . 32
4.1.3 Robust Wireless Video Communications . . . 33
4.1.4 Performance Evaluation . . . 37
4.1.5 Discussion of Results . . . 41
4.2 Telepresence Systems . . . 42
4.2.1 Computer Vision Techniques For Telepresence Systems . . . 43
4.2.2 Resource Constraints in TPTA Systems . . . 43
4.2.3 Resource Optimization in TPTA Systems . . . 44
4.2.4 Review of Video Quality Evaluation in TPTA Systems . . . 44
4.2.5 Evaluation Results and Discussion . . . 45
4.2.6 Discussion of Results . . . 48

**5 Computational Resource Optimized Video Codec 51**
5.1 System Classification . . . 51
5.1.1 Source Codec Configuration . . . 52
5.1.2 Channel Characteristics . . . 54
5.2 System-wide Timing Analysis . . . 54
5.2.1 Video Complexity Verifier . . . 56
5.3 Decoder Resource Usage Model . . . 59
5.3.1 Design Considerations . . . 61
5.3.2 Formulation of the Model . . . 62
5.3.3 Memory Usage Modeling . . . 66
5.3.4 Implementation Notes on Decoder Modeling . . . 67
5.4 Online Resource Optimization . . . 68
5.4.1 RDC Optimizations . . . 69
5.4.2 CD Mode Ranking . . . 71
5.4.3 System-wide Computational Resource Management . . . 72
5.4.4 Codec Behavior Under Lossy Channel Conditions . . . 79
5.5 Offline Resource Optimization . . . 80
5.5.1 GOP-based Resource Optimization . . . 81
5.5.2 Architectural Options for Optimization . . . 81
5.5.3 Design Considerations for GOP-based Optimization . . . 82

**6 Selected Performance Results 85**
6.1 Codec Software and Hardware Selection . . . 85
6.2 Confidence Level and Unknown Variance . . . 86
6.3 Decoder Resource Model Verification . . . 87
6.4 Online Optimizations . . . 89
6.4.1 Reference System . . . 89
6.4.2 3GPP PSC Application . . . 91
6.4.3 TPTA Applications . . . 102
6.5 Offline Optimizations . . . 106
6.5.1 Reference System . . . 106
6.5.2 Selected Performance Results . . . 107

**7 Conclusion and Outlook 109**

**A Evaluation Framework** **111**
A.1 3GPP conversational application . . . . . 111
A.1.1 Simulation Environment Components . . . . . 112
A.1.2 Simulation And Testing Environment . . . . . 113
A.2 TPTA Evaluation Methodology and Framework . . . . . 117
A.2.1 Performance Evaluation . . . . . 118

# Chapter 1

# Introduction

Global mobile video data traffic will increase by more than one hundred folds in the years between 2008 and 2013, to 1400 petabytes per month, according to a late 2009 estimate from Morgan Stanley Research [1]. Comparing this estimate with the actual growth to date shows that it was only moderately conservative [2]. Cisco expects that mobile video will make 66% of the 7000 petabytes per month of global mobile data traffic by 2015 [2]. But these numbers merely state the obvious; the evolved high-speed mobile network and the recent range of innovative smartphones are the catalysts for this expansion. Hence the aforementioned growth may not look too surprising now. What stays hidden behind these numbers is the tremendous processing power required to generate and consume this data round the clock. Surely mobile processors have come a long way to be able to achieve this amazing feat.

Mobile video communication applications have been an elusive business target; mobile video telephony has been promoted off and on in the past decade with limited success, but only because the conditions were not right for the anticipated growth. Mobile phones with high quality displays and cameras for video handling, as well as widely deployed LTE high speed networks for video transport have only started materializing in the past few years. Hence the science of video processing for mobile devices is more relevant than ever before.

Different aspects of mobile video processing and communication have been vigorously studied and are well-understood problems. The resources available for portable and mobile communication devices are ever so limited. The two prime resources of interest are the data transmission capacity and the computational resources. Incidentally, the limited data transmission capacity is not a unique constraint for mobile devices. It has come to be from systems much older than this; traditional terrestrial and satellite video communication have been adapting to this constraint since a long time. The principles developed there have been more or less adapted to the mobile video communication systems in a relatively straight-forward manner, and hence this optimization has a considerable head start. Today, the major chunk of this optimization is the well-addressed rate-distortion optimization problem; a wide range of deployed services are already benefiting from the proposed solutions.

The picture is a very different for the other major constraint for these systems; the

limited processing capabilities of mobile and portable communication devices. In today's deployments, handling of the computational resource constraint of a potable device is merely left to the implementation. But this is not without reason; a well-known system-wide deployable optimization for computationally resource constrained devices is not available.

## 1.1 State-of-the-Art Resource Optimization

During this work we have been involved in standardization for multimedia codecs for the next generation of hand-held mobile communications (within 3GPP technical specification group SA-4 for release 8 work). The standardization there can be considered as the state of the art for practically deployed systems. It is in these mobile communication systems where the problem of computational resource management is felt the most: it is desired to use the latest video codecs and good quality video, on portable devices that will have a vastly varying resource capabilities. It is surprising to know the solution that is being used in the standardization for the next generation of mobile video communication systems: use "the rule of the thumb."

It was decided in [3] to use the following set of coding options to limit the computational complexity for packet-switched conversational (PSC) applications:

- baseline profile for H.264/AVC [4],
- a single, most recent reference frame for inter-prediction,
- only 16x16, 8x16, 16x8, and 8x8 inter-prediction block modes.

Slightly more complex options were selected for multimedia broadcast-multicast (MBMS) services, since real-time coding is not an important consideration for these systems. There is no assurance whether any of these choices are optimal in any way.

However, the problem faced by the standardization community is not just the limited computational resources of the target devices, but also the limited understanding of the computational resource demands of video codecs on a specific hardware, its modeling, and prediction etc. Given all the constraints, the choices made above seem hardly surprising; it is a common practice not only in 3GPP but generally in other related video coding applications as well. These related applications have also been studied during our work, the details of which will be discussed in the following chapters.

## 1.2 The Emerging Picture of Resource Optimized Video Communications

Processing capabilities of mobile devices are ever changing; a few years ago, the mobile devices were not associated and known in the market for their processing capabilities. Today, a dual-core mobile processor [5] is considered a marketing and selling point for mobile phones. On the other hand, the hallmark of high quality

video codec has been H.264/AVC albeit its high computational complexity, but high efficiency video coding (HEVC) [6] is just around the corner claiming twice the visual quality at the same data rates at a much higher complexity cost.

The portable, mobile device industry is still fragmented because of simple hardware constraints. Just a minor change in screen size determines a different class of a device; a slightly larger screen than a *smartphone* makes a *tablet*. However, at the time of writing, Samsung has successfully demonstrated prototype of their foldable active-matrix organic (AMO) LED technology [7]. Industrial innovations like this are paving the way for convergence of the fragmented device categories.

### 1.2.1 The Changing Realm of Digital Multimedia

In the recent couple of years, there has been an explosion of online multimedia content storage and distribution; both in the commercial and the non-commercial domains. The contributing factors are ample availability of high speed internet access with data transfer rates in excess of 1 Mbps, sufficient for distribution of acceptable quality video content. This in conjunction with several factors e.g.

- high speed mobile all-IP access network of LTE (Release 8, 9) for portable, hand-held devices,
- modern mobile phone sets with WVGA (Wide Video Graphics Array, 800x480) or higher resolution, and
- newer adaptive multimedia streaming technologies (e.g. MPEG DASH [8], Apple HTTP Live Streaming [9], and WebM [10], etc.) to match the rapidly varying mobile network conditions,

have steered the same growth in the mobile telephony sector. These adaptive streaming technologies are HTTP based, easily deployable on the open internet. Hence same service can be offered in a scalable fashion to a mobile phone user as that offered to a DSL internet user, albeit at very different quality depending on the available network capacity and the end-user device capabilities. The intrinsic convergence serves to boost the growth in this sector. The resulting overall growth is fostering a renewed and shifted focus on video codecs. Specifically the shift has been related to an enhanced interest in royalty-free technologies, in order to make the life easier for the numerous small businesses and non-commercial users who drive the open internet based business growth. Although at the moment these efforts are premature, but are clearly gaining momentum. WebM based VP8 [11] video codec effort will be studied as an example in Chapter 2.

### 1.2.2 The Future of Resource Optimized Systems

Our vision for future of mobile video communication is of adaptive, resource aware devices that are able to achieve optimized video communication with other devices of vastly varying hardware and software capabilities. Codecs could be installed and optimized at the same time as the user requires them in a transparent fashion, and

are able to cater for the dynamic power constraints e.g. battery usage, thermal dissipation and signal strength.

## 1.3 Our Contribution to Resource Optimized Video Communication Systems

There exists a wide gap to bridge between the state-of-the-art and our vision of the future of video communication systems. We want to take a step towards this future by proposing a set of strategies that have relevance to practically deployed systems. A major concern in this regard is that video communication applications have an extensive number of variants, and newer emerging technologies change the shape of these applications dramatically.

Keeping this in view we will employ a pyramid-shaped approach for our proposals: on the bottom of the pyramid we will define design principles for resource optimization that can be shared by a wide variety of video communications systems, e.g. conversational, streaming, or multi-cast video communication systems. The preliminary basics for the shared technologies are discussed in Chapter 2. The state-of-the-art in the field of resource optimized video communications is reviewed in Chapter 3 and our contribution to establish a suitable reference system is briefly reviewed in Chapter 4. Our proposed design and optimization principles are specified in Chapter 5. Even with changed designs and newer technologies, the principles can be reused, e.g. this work mainly focuses on H.264/AVC video codec for evaluation, but the applicability to other existing and future codecs is also assessed in Chapter 2.

At the top of this pyramid, we pick a few systems and apply our optimization techniques to shed light on the performance enhancements. These systems are:

- mobile video conversational applications,
- video streaming applications and
- telepresence and teleaction (TPTA) systems.

For each of these systems we propose the optimization strategies on application-level, their interaction with the underlying transport mechanism, and deployment considerations in Chapters 5. Finally, the quantitative performance evaluation is presented in Chapter 6 before concluding this work.

# Chapter 2

# Preliminaries

Before delving into the details of system optimizations, let us review a few important concepts working at the heart of the targeted systems. Figure 2.1 shows an abstraction of the video transmission system in one direction (a similar reverse flow will compose a bidirectional communication). The captured video is source coded, followed by channel coding and transmission. At the receiver side, after reception and channel decoding the video data is decoded for presentation. This basic setup can result in a variety of system configurations, which will be classified in detail in Chapters 5. One of the major reasons for classification of different applications is the nature of channel; reliable or lossy channel models result in a huge impact on the system design and performance.

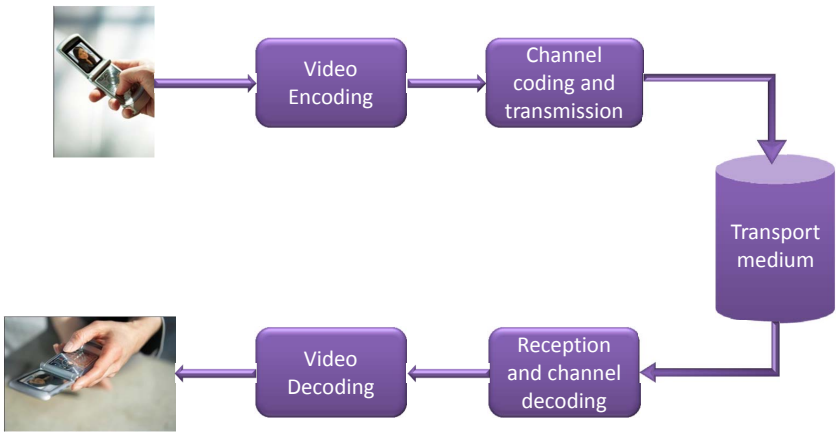



Figure 2.1: An abstraction for a video communication system

## 2.1 Resource Constrained Video Communication: Formulation

The underlying problem that we intend to address comes from the fact that video coding algorithms, like most other algorithms, have a non-trivial *computational complexity*. As a word of caution: *complexity* itself is a frequently overloaded term: e.g. *algorithmic complexity* is a similar term with a very different meaning: it is a metric of the complexity of an *object* or *content* such as text, and not to be confused with *computational complexity*.

An important consideration is the metric of complexity that can be used for assessment of any optimization in this regard. Surely, just specifying that video coding algorithms are solvable in polynomial-time, or even concluding that the complexity grows linearly with the number of pixels to be processed is of little help for system design purposes. We will use the physical *computational resource usage* as the metric of computational complexity, since this quantity is easily measurable by several metrics discussed in the following sections. Also, from a design point of view, provisioning of physical computational resources (hardware and software) and their usage is of main concern.

### 2.1.1 The Constrained Resources for Video Communication Systems

For video communication systems, two types of physical resources of interest are typically constrained: the channel transmission capacity and the computational resources of the hardware platform. The former resource constraint for video communication system has been well investigated, and well known concepts and results in this domain will be reused in this work. As a brief overview, since the information contained in real time video can vary dramatically based on the scene content, achieving a constant bitrate coding to match the allowed channel capacity is a challenge. If loss-less coding is employed, the resultant bitrate of coded video can vary anywhere up to the actual bitrate of raw video (assuming such a coding tool is unable to find and remove any redundancy in the content). Since the allowed buffering of coded video in most transmission systems is finite, the feasible solution in this regard comes from the realm of lossy video coding. This domain has been well established over the past several decades, especially in the last decade a lot of in-depth work has been carried out on the *rate-distortion optimizations*, see [12] and the references therein. Sections 4.1 and 4.2 in Chapter 4 will describe the transmission subsystem and conditions for the targeted communication systems in detail.

### 2.1.2 Computational Resource Constraints of Implementation

The main focus of this work is the computational resource constraint of the *implementation* (a combination of hardware and software). There can be several subtypes of computational resources in a hardware that can be considered as constrained.

1. **Processing resources:** this constraint is contributed by several factors, e.g. limited speed and processing capacity of the hardware, finite number of algorithmic units, their suboptimal usage by the software, thermal dissipation and power limitations, to name a few. These factors are the most pronounced for the case of portable and mobile consumer-grade devices, where processing power is at odds with limited size, overall weight and the manufacturing cost of the device. For some professional-grade systems, as those discussed in Section 4.2, manufacturing cost is not the main concern, but prolonged battery operation is still an important consideration.

   Every year the processing capability of hardware increases, but so do the processing requirements. A brief overview of the current and next generation video codecs in Section 2.2.2 will highlight this factor. This problem is quite similar in its importance to the channel capacity limitation issue, but is not as well addressed, and the reasons for this will be provided in detail in Chapter 3.

2. **Memory resources:** As with the computational hardware, physical memory is also an important resource. It seems however that for the time being, the growth of cheap and readily available physical memory has surpassed its usage requirements. At the moment, it is not uncommon for mid-range mobile phone sets with screens resolution suitable for video viewing (VGA/WVGA) to have in excess of 512 MB of physical memory. A raw YUV 4:2:0 frame buffer of WVGA resolution suitable for such displays occupies a mere 0.1% of this memory.

   At the same time, modern video codecs such as H.264/AVC apply coding algorithms recursively on raw video data to compress it. Immediately after the first recursion of coding, the residual data is already quite compressed. Hence the main overhead in terms of memory usage comes from storage of raw video buffers, and the possibility of multi-frame motion compensation, as will be discussed in Section 2.2.2 has the main impact on memory usage of the codec. An optimization strategy in this regard will be presented in Chapter 5.

3. **Memory data transfer capacity:** The transmission speed on internal busses of the hardware is finite, resulting in limitations on memory data transfer capacity. This limitation is applicable for the external physical memory or the cached memory of the processor. As noted already that modern video codecs are quite optimized in regards to compression, so memory data transfer capacity limitation may not be a prime concern, yet it is not negligible. In practical systems this data transfer capacity limitation will result in cache misses and CPU stalls, which costs in terms of lost processing power.

4. **Power:** Power constraint is reflected by the limited power available for devices with portable energy sources, e.g. battery operated devices, and thermal dissipation limits.

Incidently, the impact of first and third items is reflected as a combination on the CPU clock cycles (including stalled cycles) required by an algorithm to complete its processing. Hence CPU clock cycles spent on completion of an algorithm is a good metric for the usage of these resources.

The power constraint is also strongly related to the processing resources allocated to algorithms and hence can be indirectly managed by managing the limits on processing resources. This is also a suitable approach since power constraint does not need a very fast adaptation (e.g. on the scale of typical video frame duration, which is a few milliseconds), and can be controlled on a larger time scale, e.g. once every few seconds or even once every few minutes.

Hence in the following sections, whenever resource constraints are discussed, unless qualified, they imply the *processing resource constraints* (including the impact of memory data transfer capacity related CPU clock cycle consumption). Hence the term *complexity* will be used synonymously with *computational complexity*, and *resource* with *processing resource*, unless otherwise qualified.

## 2.2 Video Codecs for Resource Constrained Communication Systems

The most well-known video codecs are the hybrid block-based video coding standards from MPEG and ITU-T, e.g. H.264/AVC and MPEG-4 advanced simple profile (ASP) [13]. Distributed video codecs have also gained significant attention for resource constrained systems. In this section, an overview of these two main categories will be provided in relation with the target resource constrained system. Wavelets based video codecs are not considered here, since they are not best known for application on resource constrained devices.

### 2.2.1 Distributed Video Source Coding

Distributed video source coding (DVSC) techniques have been quite frequently proposed for shifting the computational complexity from the encoding end to the decoding end, as opposed to traditional block-based video coding, where the encoding is computationally more intensive. A detailed analysis of DVSC will not be provided here, and the reader is referred to [14, 15] and the references therein. We will jump directly to take a look at the performance results of the state-of-the-art DVSC.

There are no internationally standardized DVSC codecs, rather different institutes in the academia have proposed their own solutions. At the time of writing, the DISCOVER codec [16, 17] is best known for its performance in terms of compression and speed. At the transmitting end, a subset of the frames (called as key-frames) are coded in intra-only coding mode by using H.264/AVC, and the remaining set is coded in a distributed fashion. For the latter, 4x4 block sized discrete cosine transform (DCT) and quantization is performed.

Figure 2.2 is the plot of the benchmarks for this coded provided online at [17]. The bitrate is normalized to bits per pixel. As a reference, the uncompressed source video content for this test requires 12 bits per pixel (YUV 4:2:0 raw video data). The two curves show the performance comparison for the test sequence “Foreman”, the reference system is intra-only coded H.264/AVC. In this case intra-only H.264/AVC is

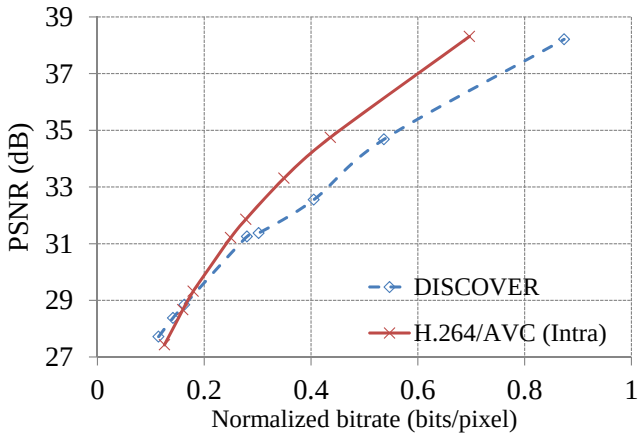



Figure 2.2: Performance evaluation [17] of DISCOVER DVSC for sequence "Foreman"

performing better than DISCOVER. On the other hand, using a frequency transform, quantization and channel coding already makes the computational complexity of DVSC quite close to that of intra-only coded H.264/AVC (the only remaining difference in terms of complexity is the intra-prediction employed by H.264/AVC, while post processing is not required at the encoding-end for intra-only coding of H.264/AVC) .

Another problem noted in the evaluations presented in [18] is that the performance of this codec flattens out below a peak signal-to-noise ratio (PSNR) of 40dB. Although this PSNR is more than sufficient for mobile video communication systems, we will see that it is not nearly sufficient for another system of interest presented in Section 4.2. For these reasons that it is concluded that in its current shape, DVSC is not best suited for application in the target systems.

### 2.2.2 Block-Based Hybrid Video Codecs

Hybrid video encoders based on block-based coding of video have been used extensively in the last four decades. All video codecs specified by international standardization development organizations (SDOs) like ISO/IEC and ITU-T are based on this technique. Each standardized codec introduced in this period achieved increasing compression efficiency compared to its predecessor (typically an increased compression efficiency of the order of 50% with each new codec). Several alternate technologies, e.g. wavelet-based video codecs, have been proposed to the SDOs during this time but none have proven successful so far. At the time of writing, H.264/AVC is an industry-wide accepted codec of choice, and the successful candidate technology as its successor (HEVC) also shares the same architecture [19]. H.264/AVC will be briefly touched in this section, for a detailed overview of its usage for video communication systems, see the description in [20] and the references therein.

This section will provide an overview of a few important standardized and non-

standardized video codecs, with an emphasis on H.264/AVC, since this is the codec used for most of the evaluation in this work. However, as mentioned previously, the principles developed here have a generic foundation and can be applied to similar block based video codecs.

### H.264/AVC

In the last decade, H.264/AVC has well-established itself as the de facto standard for robust and highly efficient video coding, to an extent that it hardly needs a formal introduction. A few important tools of this codec that have been used for robust video communication will be briefly touched upon here. For more details, see [21, 22] and the references therein.

One of the prime reasons of popularity of H.264/AVC for robust video communication is the simplicity and low complexity of the error resilience tools it provides. As a comparison, MPEG-4 ASP introduced sophisticated error-robustness tools like re-synchronization markers and Reversible Variable Length Coding (RVLC). These tools did not gain wide industrial acceptance because of the complexity involved in implementing them. For example, RVLC relies on a cross-layer design approach where the application layer (source decoder) tries to recover from the errors caused by transport layers. However, in typical communication systems this is hard to achieve; a packet that is detected as erroneous is discarded by lower layers and not given to the application layer to apply RVLC.

**Long-Term Memory Motion Compensation** Long-term memory motion compensated prediction (LTM-MCP) [23] and up to quarter-pel accurate motion compensation are two of the most important features introduced in H.264/AVC. LTM-MCP is considered to be a giant leap in the technology of hybrid video coding, similar to moving from intra-prediction to inter-prediction back in 1980s [23].

Compression and visual performance of inter-prediction process in earlier standards like ITU-T Rec. H.261 was considerably improved by adding fractional-pel interpolation in ISO/IEC MPEG-1 on the cost of some added complexity. Fractional-pel interpolation became an integral part of all the following standards like H.262 | MPEG-2 and H.263. However, all these standards allow only half-pel interpolation. Further interpolation using the same simple interpolation filters did not show any further improvements. Since the time of introduction of MPEG-1, silicon technology has progressed significantly and at present more complexity can be afforded to achieve better compression performance.

To this end several new inter-prediction options were introduced in H.263+, including Annex N: Reference Picture Selection Mode (RPS), which was later improved as LTM-MCP [23, 24] and is incorporated in H.264/AVC and MPEG-4 ASP.

Digital video coding has been considered as a computationally intensive task even since the time of H.261 and MPEG-1 video coding standards. On the other hand memory requirements for video coding have become less of an issue compared to

available hardware, hence whatever extra free space is available can be used for Long-term memory motion compensation [24].

Despite the popularity of LTM-MCP and fractional-pel interpolation, their interdependence to achieve compression gain has been scantily addressed in literature. Wiegend et al. [24] presented an analysis for a few macroblocks in a single frame based on bilinear half-pel interpolation. In [25, 26], analysis of the aliasing distortion introduced by fractional-pel interpolated motion compensation is presented. The dominant source of spectral distortion is reported to be the digitization of video.

In [27], we have presented the study of the comparative impact of various phenomenon leading to the gains provided by LTM-MCP. These factors are:

- **Repetition of Image scene content:** Repetition of Image scene content can provide compression gain but it is limited in practical systems because of small amount of memory used for long-term prediction and the relative speed of motion with respect to search range used for motion estimation.
- **Noise triggered random matches:** Noise Triggered Random Matches are possible for scene content with considerable input noise.
- **Spectral Distortions:** Spectral Distortions impact the processed video by the following means:
  - **Analog to digital conversion of video:** This step involves processing of the video signal by a hardware to capture analog video, e.g. comb filter video decoder [28], and a high speed digital-to-analog encoder e.g. [29]. In [27] we have studied and found the overall aliasing component to be in the vicinity of -100dB.
  - **Downscaling:** Depending on the application, the acquired video data may be downscaled, and this again requires using a downsampling filer. As studied in [27], the aliasing component introduced by a typical such filter is approximately -6.3 dB at higher frequencies (at 80% of the Nyquist frequency).
  - **Fractional-pel interpolation:** H.264/AVC performs half- and quarter-pel interpolation using a 6-tap FIR filter and a bilinear filter, respectively, with aliasing components of -8dB and -4.7dB, respectively at high frequencies. These filters have to be applied recursively for some interpolation configurations, as discussed in [27].

  By analyzing the numbers above, it can be seen that a significant spectral distortion is introduced in the digital video signal, and one of the prominent contributors is the fractional-pel interpolator because of their simple implementation to limit computational complexity. With multiple temporally preceding reference frames made available by LTM-MCP, a match can be found in one of these frames where the amount of this filtering required is lesser compared to the match found in other frames. Since natural scene content contains ample smooth motion, the probability of this event is quite high.

As can be seen by the above discussion, the most frequently occurring phenomenon

contributing to the gain of LTM-MCP is related to its synergy with finding block matches that have a reduced amount of spectral distortion, as compared to other marginalized phenomenon such as the noise-triggered random matches. Hence, the major portion of the increased compression achieved by LTM-MCP comes from this phenomenon, and the results we presented in [27] clearly demonstrate this.

In addition to providing enhanced compression, LTM-MCP is an important tool for robust video communication. It provides multiple reference frames for inter-prediction, increasing the chances of finding a reference region not effected by channel losses. The proposed technique based on this principle is described in Section 4.1.

**Slice Structure coding** Slice structure coding is one of the most simple yet effective tools to achieve efficient and robust video communication over lossy channels. The network-layer phenomenon where slices help foster robustness is related to the re-packetization and segmentation at various protocol layers of the communication system. As the coded video data is passed from the application layer to the layers below, it is re-packetized and eventually segmented to match the packet size on the physical layer of transmission. The packet sizes on the lower layers depend on the type of the communication network, but are typically in the vicinity of a few hundred bytes. On the other hand a coded video frame has a size typically of the order of kilobytes. Due to this disparity, re-segmentation of coded video frame becomes a necessity.

Losses on physical channel however result in loss of integral number of packets on the physical layer. If for example, a single packet on the physical layer is lost, which actually constitutes a small fraction of the data of the entire coded video frame, the entire frame may be lost if there is no re-synchronization provided within the coded frame data. Also, if temporal and/or spatial prediction is done from this lost region, the decoded frame will be distorted. Hence slices in general provide exactly these two missing features:

- Re-synchronization points within a coded video frame; a decoder can synchronize decoding from the start of a slice.
- Provide the possibility of disabled prediction across slice boundaries to limit spatio-temporal error propagation.

The concept of slices has been present even in the older codecs, for example as a group of blocks (GOB) in H.261 and H.263, albeit with a limited functionality; an entire row of macroblocks had to be coded as a single GOB. H.264/AVC comes with the most advanced and flexible form of this tool that allows placing slice boundary at an arbitrary macroblock boundary within a frame. Figure 2.3 shows the sketch of such a slice structure coding in H.264/AVC.

This gives the flexibility to select almost any desired slice size. Hence this tool can be very effectively used to match the packet sizes throughout the protocol stack to minimize the need of re-packetization. In an event when channel losses occur, slice sizes matched to the packet sizes used by lower layers result in a similar amount of data lost across the layers. Hence a single packet of a few hundred bytes lost on the physical layer will not cause an entire frame of probably several kilobytes to be

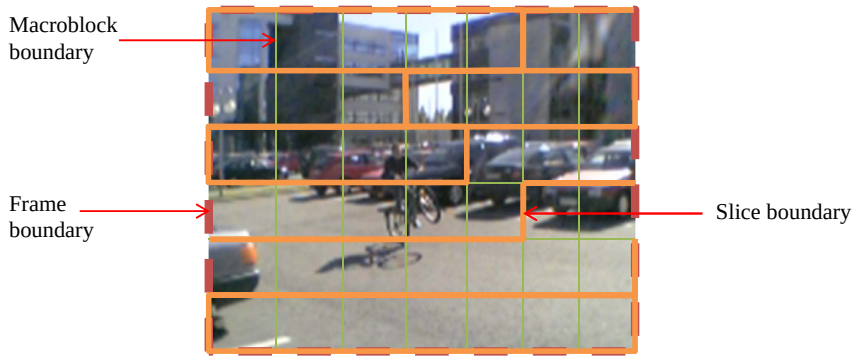



Figure 2.3: A sketch of slice structure frame coding for H.264/AVC

lost. Since re-synchronization can be done on slice boundaries, it results in a robust system design. It should be noted however that for a fixed number of bytes per slice, a different fraction of the raw video frame will be contained in one slice, depending upon the redundancy in that portion of the frame.

Off course, partitioning the frame spatially reduces the coding efficiency by increasing packetization overhead and reduced potential of removal of spatial redundancy; this results in a decrease of compression performance. We will present detailed results for performance tuning using slice structure coding in Chapter 4.

### HEVC

The Joint Collaborative Team on Video Coding (JCT-VC) formed jointly by ITU-T VCEG and ISO/IEC MPEG is working on the standardization of the next generation of video codec beyond H.264/AVC, referred to as HEVC [6]. Call for Proposals (CfP) was made in January 2010 and the responses to the call were received in April 2010.

The considered technology (see [30, 31, 19] and references therein) is still based on very similar principles of hybrid block-based video coding. In the following, a high-level overview of the basic coding strategies for HEVC is presented that make the comparison easier. Since at the time of writing the details of the specification are not finalized, hence only the underlying principles will be highlighted here.

As with H.264/AVC, a network abstraction layer (NAL) is separated from the coding layer. A treeblock concept is used to code a frame, not very different from the macroblock concept of H.264/AVC. In terms of tools for error robustness, possibility of slice structure coding is provided, where a slice is partitioned by treeblocks. A slice provides the similar functionality of disabling the prediction across slice boundaries, hence making a slice more or less independently decodable unit.

A set of intra-prediction modes is available in various directions, currently 36 different intra-prediction modes are possible in the working draft, with 33 of them providing the possibility of intra-prediction in different angular directions using the spatially neighboring samples. The intra-chroma prediction can have two different possibilities, one alternative offers the possibility of using intra-luma as the predic-

tor for intra chroma, and 5 different modes are available in this case. Otherwise, 4 different prediction modes are available.

For inter-prediction, up to a quarter-pel interpolation is possible for luma, using an 8 tap spatial FIR filter. For chroma, one-eighth-pel interpolation can be done, and a 4 tap FIR filter is used for all the interpolation configuration. As with H.264/AVC, there is a possibility of weighted inter-prediction. Similarly, multiple reference frames can be used for inter-prediction.

The transform block sizes range from 4x4 to 64x64 sized blocks, using an integer-transform (i.e. a reversible frequency transform on 32-bit machines, a similar principle as that used for H.264/AVC). The scanning of transform block coefficients is done by a traditional zigzag scanning variants.

Similar to H.264/AVC, HEVC comes with the possibility of in-loop post-processing filtering in the shape of a deblocking filter and an adaptive loop filter. Although, a distinction is made between a prediction-boundary and a transform-block boundary for deblocking. This filter is applied sequentially to vertical and horizontal boundaries, respective, with varying boundary strengths based on the coded block modes and pixels to be filtered.

The entropy coding scheme gives the same two options as for H.264/AVC: a context adaptive variable length coding (CAVLC), that offers a lower-computational complexity alternative, as well as context-adaptive binary arithmetic coding (CABAC).

For the next some time, the proposed technologies will studied further and the specification text will be formalized. Still, from a high-level overview it is easy to see that the codec is based on same design principles as for the previous video codecs such as H.264/AVC, and the advancement is incremental, not disruptive.

### VP8

VP8 [11] is an open source video codec supported by a consortium of companies, in an effort to move towards a royalty-free codec. It is easy to observe from an overview of VP8 that its architecture and working is quite similar to that of block-based hybrid video codecs such as H.264/AVC. The block transforms are all 4x4 DCT based (both for luma and chroma components). It comes with a set of intra- and inter-prediction modes not very dissimilar to that of H.264/AVC, though the options are much more limited. LTM-MCP is limited to three frames. Up to a quarter-pel luma interpolation and one-eighth pel chroma interpolation can be performed, and the block sizes for inter-prediction within an macroblock can be arbitrary.

Hence the resource management techniques developed for for H.264/AVC can be applied in a straight-forward manner to VP8. The subjective performance comparison of VP8 [32] reports it to be quite close to H.264/AVC, but still lagging to that of the upcoming HEVC codec.

### 2.2.3 Error Concealment for Block-based Video Codecs

In spite of the robustness tools available for block based video codecs, including application level FEC, residual errors will remain depending upon the level of protection used as allowed by the application delay and general QoS requirements. When the received data is erroneous, it has a significant effect on the quality of the decoded video; not only instantaneously but also on the following video in decoding order, owing to spatio-temporal error propagation. This phenomenon occurs because of spatial and temporal predictors used in block-based video codecs to remove the redundancy, and the detailed analysis of this will be presented in Chapter 4.

The typical standardized video codecs only specify the behavior of the decoder in a loss-less scenario, and a tight synchronization is assumed between the reconstruction at the encoder and at the decoder. When this assumption is not fulfilled anymore, e.g. in the case of data lost on channel, the reaction of the decoder is not specified. However, as discussed previously, most robust video codecs come with error-robustness tools that can be exploited to react and address this situation.

In the wake of the reception of an erroneous packet, he first reaction has to come from the video decoder to minimize its impact as much as possible. Hence generally a process referred to as error-concealment is invoked. This process is not standardized and has been a topic of research, with hundreds of publications available (see the detailed overview we provided in [33] and the references therein). Here we briefly provide the overview of the categories of the techniques. Mainly, these techniques are based on:

1. spatial prediction,
2. temporal prediction, and
3. hybrid prediction.

Spatial prediction techniques uses the correctly received and already concealed neighboring pixels of the frame to reconstruct a lost region of the image. As expected, these techniques can benefit a lot from Flexible Macroblock Ordering (FMO) tool provided in H.264/AVC. These techniques, e.g. in [34] are typically based on the smoothness constraint of the image and use weighted filtering to reconstruct the lost region. As expected, the result of such technique will result in significant blur, and model based refinements have been proposed in e.g. in [35] based on image texture statistics.

Temporal prediction techniques use the previously correctly received or concealed frames to reconstruct the lost regions. Hence they are based on the smoothness of motion constraint of natural scene content. The most simple and widely used variant is the previous frame concealment (PFC) which copies the collocated macroblock in the temporally preceding frame to the lost region. Although this technique will not be effected by the blur usually caused by spatial prediction based techniques, but any significant motion in the frame with respect to the temporally preceding frame will result in significant artifacts. Still this technique is widely used because of it suitability for resource constrained devices, since such a simple copying has the computational complexity comparable to one of the most simple coding modes

known to block based hybrid video coding: the *skip* inter-prediction mode. More complex variants based on prediction of the lost motion vector have been proposed (e.g. [36]) to alleviate the problems associated with PFC.

The most sophisticated approach in this regard is that of hybrid error concealment that uses both spatial and temporal prediction. A simple variant as proposed by [37] is to use spatial prediction for intra coded images and temporal prediction otherwise.

Hence it can be seen that the error concealment has been enhanced at the cost of increased computational complexity. Which technique is most suitable depends on the intended application and its susceptibility to channel losses.

### 2.2.4 Discussion

In spite of existence of several theoretically disruptive technologies like wavelets based coding and DVSC, block-based video coding techniques have been the front runner for quite some time now, and seem to stay this way for next some time. A brief look at the upcoming technologies such as HEVC makes this quite clear. Hence it seems feasible that the analysis presented in this work is not just limited for the selected codec, i.e. H.264/AVC, but the general optimization principles discussed in the following chapters can be used as a basis of optimization for the upcoming video codecs as well.

# Chapter 3

# Resource Constrained Video Coding: Brief review

In this chapter, the recent studies in the field of resource constrained video coding will be reviewed. The techniques reviewed here try to address resource optimization problem for diverse applications and they might focus on different video codecs. But they all have one common denominator: they try to manage and optimally operate a part of a video coding subsystem within some given, constrained computational resources. Studies that solely focus on computational complexity reduction are not presented here, since these are mostly implementation dependent and can be integrated into a given resource optimization technique.

As discussed in Chapter 2, a comprehensive picture of how the computational complexity of evolving video codecs is changing in comparison with the evolving hardware computational resources over time is essentially missing. While the latter is approximately doubling every year, the former is an unknown. However experts in field of video coding know the answer as a rule of thumb: as the codecs and the hardware on which they run are evolving, the problem of computational resource management of video coding is becoming increasingly important. A survey of literature review in the following also reaches the same conclusion.

## 3.1 Encoder Optimizations

These techniques involve the optimization of the resource consumption of the video encoder to optimize the rate distortion performance in some specific way. The approaches are categorized in the following sections from cross-layer optimization approaches to algorithms that focus on a single coding module e.g. the motion estimation.

### 3.1.1 Joint Source-Channel Power Optimizations

The target of such optimization is cross-layer power distribution between the source coding and channel coding modules of a wireless video transmission system, such that the end-to-end distortion is minimized.

#### Early Contributions

This approach has been investigated as early as in [38] for motion-JPEG based codec. In other earlier works on joint source coding and transmission power allocation such as [39, 40, 41, 42], the energy consumption for source coding was simply ignored. MPEG-1/MPEG-2 based codec used in [41] had been treated in such fashion, and in such systems transmission power has been a main resource to be optimized.

Relatively later in [43, 44, 45, 46], the computational resource usage of motion estimation module of a hybrid video codec was included in the optimizations, and will be touched later in Section 3.1.3.

In a set of related work by Xiaoan Lu et al. [47, 48, 49, 50, 51], the source coding power consumption is related to solely the percentage of intra-coded macroblocks as a fraction of total number of coded macroblocks. This relation is used for source-channel power distribution. In essence, this approach also considers the motion estimation to be the exclusive source of video encoder computational complexity. This approximation may work well for older video codecs like H.263, which had a handful of coding options (the evaluations is indeed provided exclusively for this codec).

#### Dynamic Voltage Scaling (DVS):

In a special category of hardware capable of DVS, the clock frequency of a processor can be dynamically adjusted to manage the power consumed by it. Hence a joint power allocation can be realized easily by controlling the processor clock frequency. A lot of work on joint source channel power optimization has been done based on such hardware, some of it is reviewed here briefly.

**Based on Implementation Profiling:** In [52], an implementation of the H.264/AVC reference software running on Intel® PXA270 processor is profiled to measure the amount of basic instructions used by the major encoding modules. The modules with the largest computational demand have been identified for this H.264/AVC encoder implementation. The encoding computational complexity is controlled exclusively by configuring the motion estimation algorithms. The complexity is related to processor clock frequency and hence the power requirement of source coding is controlled using DVS.

**Joint Power-Rate-Distortion Analysis:** Joint Power-rate-distortion analysis has been done in a number of related works by Zhihai et al. [53, 44, 54, 55, 56, 57].

The focus is to add the power constraint to the existing rate-distortion constrained minimization problem already explored in great details for video coding systems. The encoding complexity is related to the number of sum of absolute difference (SAD) calculations, non-zero blocks and number of bits generated. The encoder computational complexity is controlled by limiting the number of SAD calculations.

In the most recent of these related studies [57], the distortion $D(R, P)$ at a given rate $R$ and normalized power consumption $P$ is approximated as

$$D(R, P) = \sigma^2 2^{-\lambda R P^{1/\gamma}} \tag{3.1}$$

where $\sigma^2$ represents the variance of encoded picture, and $\gamma$ is a constant empirically determined for a given hardware. As with the rest of similar approaches to make this a joint minimization problem discussed later, the crucial step is the selection of the $\lambda$ parameter in the Lagrangian minimization. In [57], the $\lambda$ is empirically approximated as:

$$\lambda = C_0 + C_1 \times 2^{C_2(\zeta \cdot 256 + \frac{1}{M}\sum_{i=1}^{M}[m_x(i)-\overline{m}_x]^2 + [m_y(i)-\overline{m}_y]^2)} \tag{3.2}$$

where $C_0$, $C_1$, $C_2$ and 256 are empirical constants, $\zeta$ is the ratio of intra-coded macroblocks in a frame, $M$ is the number of macroblocks in the frame, $[m_x(i), m_y(i)]$ is the motion vector of $i^{\text{th}}$ macroblock and $[\overline{m}_x, \overline{m}_y]$ is the mean motion vector of the frame. It is evident that the relation is based on an empirical mathematical relation between motion statistics and the power-distortion tradeoff based on a set of evaluated video content.

**Some Comments on Joint Source-Channel Power Optimizations**

While the approach seems a theoretically intriguing choice of joint and cross-layer optimizations, it has several pitfalls in practice. Source coding and channel coding aspects should be left separate according to the Shannon separation principal, this is how designs can be best optimized [58].

The rate-distortion constrained optimization for video codecs is already quite complex problem with extensive work done on it. Introducing another constraint as power makes it even more complex. At the same time, combining the power distribution of multiple layers is difficult to achieve at best in practical systems, these modules may very well be separate hardware components. Hence it is best suited to solve the source coding computational complexity problem exclusively on source coding layer. A good approach in this regard can be where an "outer-control loop" manages the power distribution between source coding and transmission modules, based on some slowly varying parameters, such as the amount of energy available from a portable power source. For example, in the usage scenario in [57] shown in Figure 3.1, the average transmission power can be used to select the optimum operating point that is adjusted at some suitable discrete points in time.

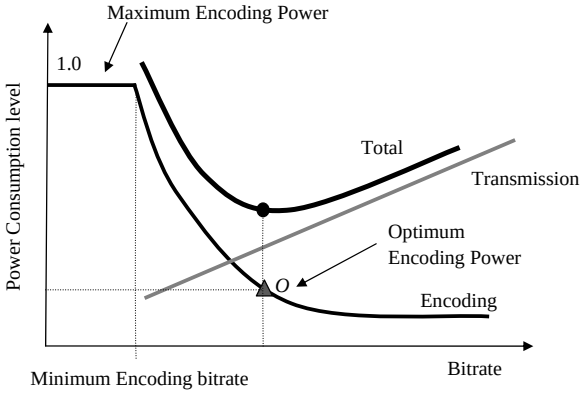



Figure 3.1: Energy tradeoff between video encoding and wireless data transmission [57].

### 3.1.2 Joint R-D-C Optimization

In the domain of video codec resource management, the most frequently made choice is to go for a joint R-D-C optimization. The reason is simple: the rate-distortion constrained minimization is already a well-understood problem for video coding. Encoding computational resource is yet another resource bound by the processing resources at the terminal, and hence the attempt is to add this constraint into the existing optimization problem to achieve a joint optimization of rate, distortion and computational resource usage. Similar approaches to optimize based on the decoding end complexity will be discussed in Section 3.2.3.

#### Analytic R-D-C Convex Hull:

R. Vanam et al. in [59] have used the encoder coding options to achieve the optimization by determining a D-C convex hull. The coding options are sorted in order of MSE, and an optimal coding point is chosen on the D-C curve. Unfortunately, it is not feasible to determine precisely which coding option will achieve a lower MSE without actually coding an image data first, and iterative encoding algorithm is also proposed.

In [60, 61, 62] an empirical 3D curve fitting technique is employed. In [60, 61], Evgeny Kaminsky et al. have related the computational complexity $\mathcal{C}$ to distortion D as

$$\mathcal{C}(R, D) = \log_{\beta}(A \cdot \alpha^{-R}/D) \tag{3.3}$$

Where $A$, $\alpha$, and $\beta$ are empirically determined constants tuned by using a few training video test sequences. A similar approach in [63] has also used a training set of sequence to determine the D-C convex hull and use it to control the computational resource usage of the video encoder.

The feasibility and theoretical basis of adding the complexity constraint to the existing RD constrained-minimization problem will be investigated in more detail in Section 5.4.1.

### 3.1.3 Motion Estimation Based Optimization

In these studies, the dominant contributor to the encoder complexity has been attributed as the motion estimation of hybrid video coding, e.g. [52, 43, 44, 45, 49]. Similar approach has been taken even as recently as [64]. Hence to manage the encoding complexity dynamically to a desired level, basic motion estimation parameters such as the search window or amount of SAD calculations are controlled.

It is understandable why so many studies consider motion estimation as playing the dominant role in the overall computational complexity; while this is a reasonable assumption for simple motion estimation techniques like full search, it does not hold well for other highly optimized motion estimation techniques like [65, 66, 67, 68, 69, 70]. For the implementations that use these highly optimized motion estimation algorithms, other encoding modules (e.g. block transforms, intra-and inter-compensation, entropy coding, etc.) are also of significance for this optimization problem in general [71].

### 3.1.4 Mode Ranking

In several studies, the authors strive to select and use a subset of coding modes to achieve computational resource savings. This can be a part of other computational resource management techniques, for example used by [59] as discussed before in conjunction with analytic RDC convex hull.

#### Complementary Macroblock Sets

In [72, 73, 62], Tiago A. da Fonseca et al. have developed a technique where a frame to be coded is spilt into two complementary sets of macroblocks $S$ and $S'$. All coding modes are used for coding the macroblocks in set $S$, while only $D$ dominant modes within $S$ are used to code the macroblocks in $S'$. The ratio between macroblocks assigned to $S$ and $S'$ is empirically, 10% of all the macroblocks in the image go to $S$. The actually location of macroblocks in the image are chosen pseudo-randomly. The number of dominant modes actually explored in $S'$ depends on the computational resources available, and this can be adjusted dynamically to achieve the targeted computational resources.

This technique relies on the semi-stationary statistics of optimal coding modes within a single frame, which might not hold well for a variety of scene content.

**Discussion on Coding Mode Ranking**

To predict which coding mode will work optimally for a given macroblock by using computations that are significantly smaller than the actual coding is a difficult task at best. The more simple a technique is used for this purpose, the more crude the results are. For example, in [74] the coding gain achieved by different coding tools within H.264/AVC is sorted simply according to the average performance based on the measured results on a number of test sequences. Such average statistics will give suboptimal results on a local scale, e.g. on a frame or a sub-frame level. These aspects will be investigated in more detail in Section 5.4.2.

### 3.1.5 Encoder Resource Usage Prediction

In [75, 76] Yuri V. Ivanov et al. have proposed a method of optimizing the encoding end computational resource usage by first predicting the computational resource usage of a complete frame. This is done by first predicting the proportion of macroblocks that fall under a particular class amongst a set of 5 different classes. The classes essentially describe the modes used for coding the macroblocks. The prediction is done by using the mode decisions done for the suitable neighboring macroblocks.

Next, based on this predicted distribution, the resource usage of coding a complete frame is predicted. If this prediction goes beyond target resource budget, more complex modes are excluded.

As shown in [76], the predicted resource usage may have in excess of 30% peak error, with average error between 5 to 10%.

### 3.1.6 Skip Mode Prediction

In such techniques as proposed in [77, 78, 79, 80], the resource usage is controlled by effectively selecting what proportion of the macroblocks in a picture can be coded in skip mode. Skip mode has the least computational overhead at the encoding end for typical implementations.

In [77, 78] C.S. Kannangara et al. have used a threshold lagrangian RD cost, determined based on statistical parameters of the image being coded, e.g. the MSE between the current and the previous frame, and the available computational resources. If the lagrangian cost of coding the macroblock is above this threshold, the macroblock is skipped. Hence the selected threshold actually controls the computational resource usage of coding a frame. The costs of the coded macroblock $J_{code}$, and the skipped macroblocks $J_{skip}$ are expressed as:

$$\begin{aligned} J_{code} &= D_{code} + \lambda_r \cdot R_{code} + \lambda_c \cdot \mathcal{C}_{code} \\ J_{skip} &= D_{skip} \end{aligned}$$

The decision whether or not to skip a macroblock is expressed by the relation

$$D_{code} + \lambda_r \cdot R_{code} + \lambda_c \geq D_{skip} \tag{3.4}$$

Hence Equation 3.4 becomes the basis for the resource management.

One aspect that is quite peculiar in Equation 3.4 is that it is assumed that $\mathcal{C}_{code} = 1$, i.e. the coding computational resource usage of all the coded macroblocks is the same. Theoretical and practical observations indicate on the contrary in a vast number of studies. Just as one example: for H.264/AVC, the motion compensation cost for a block that is compensated by using a full-pel configuration requires just a copy operation in memory, while for the motion compensation for a block that is compensated by using a half-pel, quarter-pel configuration, each pixel is derived by applying two six-tap FIR filters, followed by a bilinear filtering and clipping, in addition to the copying. As expected, in [27] it was shown that for a specific implementation of H.264/AVC, the computation resource cost of the latter mode is approximately 10 times more than the former mode. Hence the assumption of constant block complexity will have a huge impact on any later optimization that is based on Equation 3.4.

### 3.1.7 Computational Resource Management of a Video Encoder

In [81, 82], Li Su et al. have provided the most comprehensive work to date on computational resource management of an H.264/AVC encoder. It addresses both the motion estimation computational resource management as well as coding complexity management.

Like [83, 77] it also usees the concept of virtual resource buffers but in a more flexible way: the buffer fullness of the virtual resource buffer is used as a control input to the encoder resource controller that uses a proportional-only control mechanism.

At the motion estimation stage, several alternate complexity configurations execution paths are enabled. The selected path depends on the available computational resources. The paths are listed in the following in decreasing order of complexity:

1. Full motion search
2. Full fractional-pel search and reduced full-pel search
3. Only full-pel search
4. Reduced full-pel search only
5. No motion search, setting motion vectors to zero

At the macroblock coding stage, the available coding modes are divided into three distinct sets of modes: SKIP mode, Inter16 mode (16x16 to 8x16), Inter8 modes (8x8 to 4x4) and Intra-coding modes. Based on the available computational resources, only a few sets are used. Which of the sets to select is yet again the mode-ranking problem discussed before. In this work, the ranking is done by using modes of the immediate spatial and temporal neighbors as predictors.

This work will be studied in more detail in Chapter 6 and used as a reference for evaluation of the proposed framework.

### 3.1.8 Variable Complexity Transform

In [71] Richardson et al. have proposed a variable complexity DCT algorithm. The complexity is adjusted to match the available computational resources. It is shown to work for implementation where DCT is a significant contributor of complexity, shown to be approximately 30% for an implementation of MPEG-4 ASP.

## 3.2 Decoder Optimizations

The techniques presented in this section focus on the resource management of the video decoder under constrained computational resources. The resource usage modeling of the video decoder is an important part of such optimization, hence it is discussed before the optimization techniques.

### 3.2.1 Decoder Resource Usage Modeling

The target of these techniques is to model the video decoder or part of it, to be able to predict the computational resource usage that will entail to decode a specific portion of the compressed data. Such models do not themselves constitute a complete resource management technique, but are rather an important cornerstone of the technique.

#### Video Complexity Verifier (VCV)

At the decoding end, the concerned video coding standardization bodies have tried to develop a video complexity verifier (VCV) mechanism [84, 85, 86] with a varying degree of success. The target is to develop a model that can be used to profile various video streams based on the expected computational resource usage at the decoding end, following the lines similar to the HRD approach of [87]. These models are based on video coding statistics such as the bitrate. Hence the VCV can be useful as a part of an over-all computational resource management scheme by predicting the computational resource usage at the decoder. However, mostly such attempts have gained little traction and in the standardization community referred to as "implementation issues."

A detailed characterization of the algorithmic modules of an H.264/AVC decoder in context of typical scene content is provided in [88, 89].

**Computational Resource Usage Model for SVC**

A rather simple computational resource usage model for SVC codec is proposed in [90] that assumes a constant computational overhead for all macroblocks within a given frame; the overhead is differentiated only based on the type of picture coding (intra-coded or I-picture, inter-predicted or P picture and bidirectionally predicted or B picture).

By using this approximate model, it can be approximated to some extent what scalable coding configuration on group of pictures (GOP) level is suitable for a decoder.

**Computational Resource Usage Model for Individual Decoding Modules:**

In a series of related works [91, 92, 93, 94], Szu-Wei Lee et al. have developed the resource usage models for various coding modules of H.264/AVC. These models have been developed separately for motion compensation and CAVLC decoding modules.

In [91] the computational resource usage model of H.264/AVC motion compensation module is modeled as a function of:

- the number of data cache misses,
- the number of applied horizontal and vertical interpolation filters and
- the number of motion vectors per MB.

Based on these variables the predicted computational resource usage of the motion compensation process is predicted with a worst case error of 10% when tested for a variety of test content.

Likewise, in [93, 94], the computational resource usage of H.264/AVC CAVLC decoding module is modeled as a function of 5 variables of the H.264/AVC bitstream:

- the number of non-skipped MBs,
- the number of CAVLC executions,
- the number of trailing ones,
- the number of remaining non-zero coefficients, and
- the number of run executions.

In an anticipated future work by the authors, these models will be eventually combined for resource usage prediction of the decoder.

### 3.2.2 Computational Resource Management by Quality Degradation

In [95] T. Lan et al. have developed a basic decoding resource usage model for MPEG-1/2 video decoders, that achieves a significantly accurate decoding resource

usage prediction. The model is based on four parameters including coed block pattern, motion vector count and magnitude, and macroblock type. In [96] a computational resource usage model of MPEG-4 ASP based on 7 coding parameters is proposed.

The predicted resource usage is utilized to select one of several decoding routines that trade complexity with precision. The underlying framework of computational resource scalable algorithms, also applicable for video encoders, has been discussed in [97]. An example of possible tradeoff at the decoder between performance and resource usage is as follows: in codecs like MPEG-1/2 and MPEG-4 ASP, perfect reconstruction was not possible by the inverse transform implemented with 32 bit integer arithmetic, and some non-compliant inverse transform routines could be even faster, but would result in an increased amount of reconstruction error. This however is not applicable for H.264/AVC which allows for perfect reconstruction for error-free case.

### 3.2.3 RDC Optimizations For Streaming Applications Using Generic Complexity Metrics (GCM)

In a set of comprehensive investigations by Mihaela van der Schaar et al. in [98, 99], the decoding computational resource usage of a video decoder is modeled using GCMs. Although as described in [98], the model is demonstrated for wavelets based JPEG-2000, it can be possibly extended to DCT-based hybrid video codecs. The resource usage modeling approach used in these investigations is different from those presented in Section 3.2.1. The generic resource usage model is based on:

- the percentage of decoded nonzero transform coefficients,
- the percentage of decoded motion vectors out of the maximum number of possible motion vectors.

The GCM is then translated into implementation dependent real complexity metrics (RCM). In [98] the model is used for bitstream adaptation at the transmitting end to optimize the overall streaming application.

As shown in Figure 3.2, the proposed resource usage model is used to select the representation of video to stream to the client based on its available computational resources. The bitstream adaptation is expressed as the optimal $j^*(i)$, $\lambda_r^*$, and $\lambda_c^*$ expressed as

$$\{j^*(i), \lambda_r^*, \lambda_c^*\} \forall b_i = \arg \min_{j(i),\lambda_r,\lambda_c} \left\{ \sum_{i=1}^{N} \left( D_i^{j(i)} + \lambda_r \cdot R_i^{j(i)} + \lambda_c \cdot \mathcal{C}_i^{j(i)} \right) \right\} : R_{GOP} \leq R_{max} \text{ and } \mathcal{C}_{GOP} \leq \mathcal{C}_{max} \quad (3.5)$$

for the access unit $b_i$ with $N$ access units per GOP, $j(i)$ indicates the adaptation point, $R_{GOP}$ and $\mathcal{C}_{GOP}$ are the cumulative bitrate and resource usage of a GOP

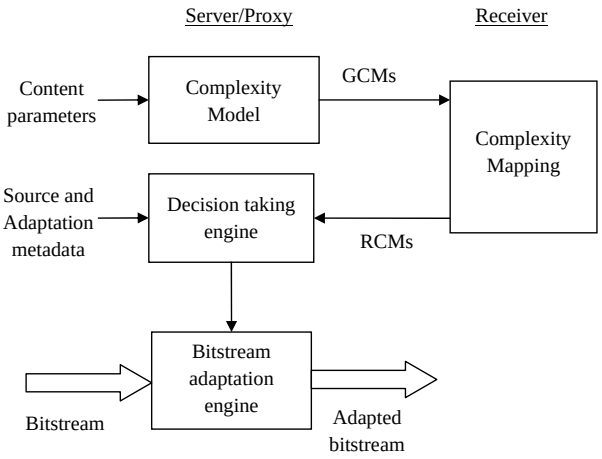



Figure 3.2: Decoder resource usage model being used for bitstream adaptation [98]

respectively, while $R_{max}$ and $\mathcal{C}_{max}$ express the maximum allowable rate and resource usage on a GOP basis respectively.

This approach is one of the most comprehensive solution of the video decoder resource optimization problem because it tries to address the problem of decoding resource management at the transmitting end and uses a model based approach for this purpose. The selected approach to integrate the technique in a streaming service however requires per access unit based communication between client and the server. The overhead might not be desirable in a streaming application framework in regards to scalability.

## 3.3 Some comments on the reviewed work

As commented earlier, the studies presented in the preceding sections attempt to optimize performance of various modules within the video coding subsystem of an application. A few studies have been explicitly marked that comprehensively address a broader part of the video coding subsystem. What is essentially lacking in the picture is a study that tries to optimize the end-to-end video coding subsystem. This problem will be revisited in Chapter 5 with our proposed framework.

By observing the chronological order of the studies presented in the preceding section, it becomes clear that the problem of computational resource management is becoming increasingly important, with more and more work being done to solve parts of this problem with each following year.

# Chapter 4

# Resource Constrained Video Coding Systems

The computational resource management problem exists for any application that contains a video coding subsystem, when a part of that subsystem operates on a computationally constrained hardware. The type of the video coding subsystem depends on the intended application in terms of the video codec used, codec settings of quality, buffering parameters, delivery format, error protection and recovery, etc. Eventually, the permutations of the parameters result in an extensive number of variants of the video codec subsystem. In Chapter 5 we will describe a categorization of these variants. The issue of computational resource constraint effects most of such video codec applications.

As introduced before, the target of this work is to formalize a solution of computational resource management in a generalized way for a range of such variants, instead of proving a precise solution for one variant that has little applicability for the other. A few of the systems however will be used as the basis of detailed experimental evaluation and verification. The details of those system configurations as well as the optimizations we have proposed for these reference systems will be provided in the following sections of this chapter.

## 4.1 Mobile Video Conversational Applications

Video conversational applications for hand-held devices in UMTS-like networks represent a culmination of bidirectional, live and mostly point to point (only two terminals conversing with one another) service. However, the principles discussed here can also be directly extended to a live multicast/broadcast based application.

The application is characterized by the following features:

- Low end-to-end delay: as understood for a conversation application, the end-to-end delay has to be minimized as much as possible. For example, as specified in 3GPP technical specification [100], it should be of the order of 100ms.

- Portable, consumer-grade devices: A mobile conversational application is targeted for consumer hand-held or portable devices that come with the typical constraints of limited computational resources and power as discussed in the preceding sections. As with any communication application, available transmission capacity is a constraint. But this constraint is also cumulated by the other constraint of limited processing power of the devices.
- Error prone communication: With the current technology, specifically for wireless communications, reliability is never a given. The problem is compounded by the low delay requirements: re-transmission based strategies have little room in these application since they might introduce an arbitrary amount of delay. Error protection like application level forward error control (FEC) is always a possibility, but channel capacity and complexity constraints limit the scope. In spite of all the efforts, there will always be residual error rate, and a robust system design must take this into consideration.

In the following section, we describe an existing and well known form of such application: 3GPP Conversational Packet Switched Video Services.

3GPP mobile video telephony services are enabled by conversational packet switched multimedia services [101] which are based on IP multimedia subsystem (IMS). In such a service, a hand-held transceiver is connected via bidirectional high speed packet access (HSPA) link to base station, which connects to the core network. The remote terminal may also be a hand-held device. Bidirectional transmission of compressed H.264/AVC packetized video data is done via RTP along with RTCP control information, the protocol stack is shown in Figure 4.1. Upon reception, the lower protocol layers detect and discard corrupted data packets.

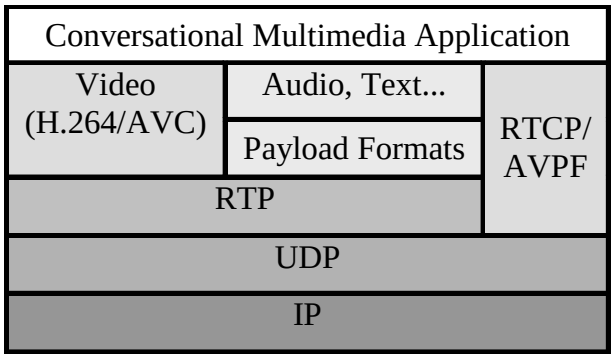



Figure 4.1: Protocol stack of 3GPP Conversational Packet Switched Video Services

### 4.1.1 Error Robustness

Mobile communication channels suffer from frequent fading that results in bursty losses. As mentioned previously, channel coding is an expensive option, since it requires even more channel capacity. Regardless of channel protection, losses are bound to happen in mobile environments. Figure 4.2 shows one instance when a

typical video content that has not been optimized to handle channel errors is subject to losses.

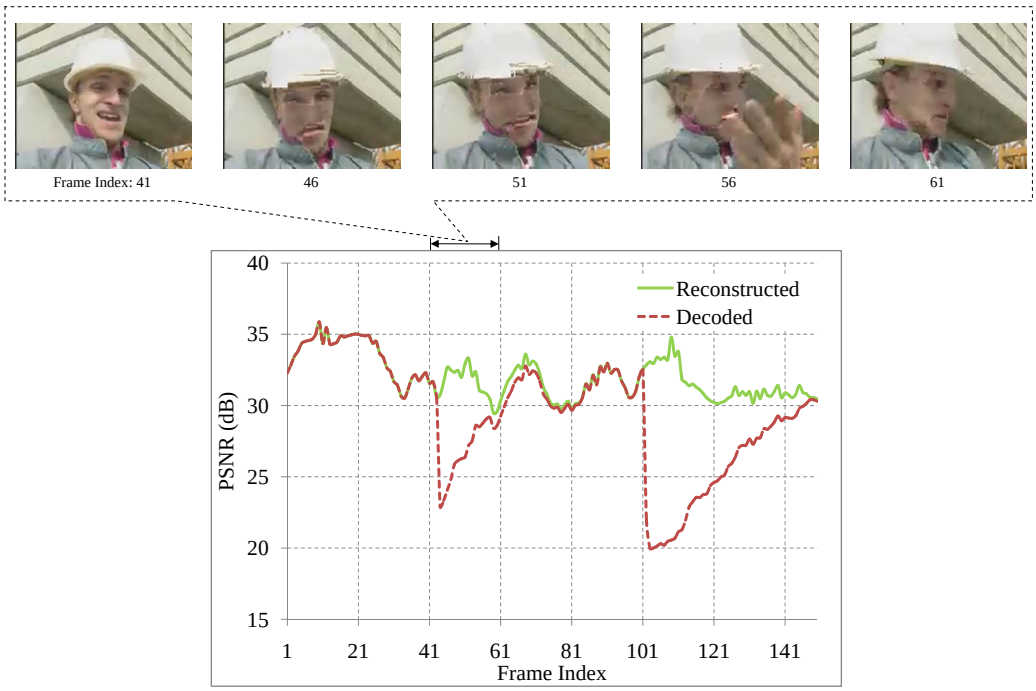



Figure 4.2: Instantaneous impact of an error on video quality

The results are shown for 3G packet-switched conversational application, and the application level parameters are specified in detail by 3GPP [102]. This application is characterized by its stringent low-delay and low resource usage requirements, since the processing has to be done in real time on hand-held devices. As a result, the maximum allowed buffering at the encoder is limited to 250 ms and to limit buffering overheads, complete intra-frame updates are not used. A simple pseudo-random intra MB refresh technique is used, with 5% MBs of every frame coded in intra mode, as proposed by earlier error protection strategies based on pseudo-random intra updates as discussed in [103, 104, 105]. Only the most recent frame is used for motion compensation to limit the computational resource use. No other mechanism is provided for error robustness, e.g. slice-structure coding is not used.

It can be seen from Figure 4.2 that with a system that is not well tuned, a single radio-link control (RLC) packet data unit (PDU) loss at frame number 45, devastating distortion is observed not only for that frame but for the following frames as well. The peak signal-to-noise ratio (PSNR) of the luma (Y) picture component also expresses this, with a drop of several dBs. This is as expected from H.264/AVC that achieves very high compression efficiency by meticulously removing spatio-temporal redundancies.

However, synchronization is assumed to exist between the encoder and the decoder, i.e. it is assumed that the decoder will have the same reconstructed data to use for prediction as the encoder has. With a loss of data at the decoding end, this

assumption is violated. This results in an instantaneous drop of quality. However, even in subsequent frames, this lost region may be used for prediction at the encoder. As a result, when the decoder also uses this lost region for prediction, and as this process is repeated recursively, it results in the error propagation via the predictors. Hence, a loss of video data will result not only in distortions in the immediately effected frame but also the subsequent frames owing to this spatio-temporal error propagation.

The difference between reconstructed and decoded signals reduces subsequently because of the effects of saturation and breaking of prediction chain, e.g. with intra-coded blocks. Hence the residual error rate is a significant problem for video communication on existing mobile communication systems, and only robust, *loss aware* coding can handle the error propagation appropriately. At the same time, real time video coding along with speech processing etc. on a hand-held device leaves little room for employing error-resilience techniques that increase the computational resource use.

Before delving into the details of the robust system design, lets discuss the appropriate performance metrics.

### 4.1.2 Performance Metrics

#### Objective Metrics

A typical assessment metric is the PSNR, which for 8-bit precise signal with an MSE $e$ is defined as

$$\mathrm{PSNR}(e) \triangleq 10 \log_{10} \left( \frac{255^2}{e} \right). \tag{4.1}$$

Another metric used is percentage of degraded video quality (PDVD), defined as:

$$\begin{aligned} \mathrm{PDVD} &= \frac{\sum_{i=1}^{N} f(\hat{d}_i, \tilde{d}_i)}{N} \%, \\ f(\hat{d}_i, \tilde{d}_i) &= 1 \text{ if } (\hat{d}_i - \tilde{d}_i) > 2 \text{ dB}, 0 \text{ otherwise} \end{aligned} \tag{4.2}$$

whereby $\hat{d}_i$ is the reconstructed PSNR of $i^{th}$ frame, and $\tilde{d}_i$ is the decoded PSNR of the corresponding frame for a sequence with $N$ frames. This metric represents what fraction of decoded video has considerable distortion added because of losses, and hence tends to decouple this distortion from the compression losses. The threshold of this is selected as 2 dB.

#### Subjective metrics

Objective quality metrics such as PSNR can not give a reliable and accurate measurement of the visual quality of a video that was first encoded at low bit rate and

then transmitted over a lossy channel. A number of different objective video quality metrics has been proposed (see e.g. [106] for an overview). But none of those metrics has been verified for H.264/AVC in a satisfactory way. In addition for large majority of the so far proposed metrics no independent and reliable verification exist, that shows a high correlation between the estimated quality of the metrics and the results of subjective tests. For this reason subjective testing still is the only reliable methodology to precisely measure the picture quality of a distorted video.

The subjective tests are conducted for a presented data by subjects who not active in the field of video coding (non-experts). They are screened for visual accuracy and color blindness and trained for the task of evaluating the video. The setup of the test room as well as the test procedure and the processing of the results followed the recommendations given in [107, 108].

For the target system, the aim of the test is to rate the ability of the proposed schemes to correct errors that are introduced by the lossy channel, and a continuous quality evaluation was performed. In a continuous quality evaluation subjects evaluate the quality in real time using a slider (the slider is located on the screen next to the displayed video and is steered by using the mouse input) with a continuous scale. To ensure that only the degradations introduced by the channel are evaluated and not the degradations introduced by coding at low bit rate, the video under test is placed side by side to the same video without the channel degradations and the subjects were asked to rate the amount of additional degradation. The position of the slider is recorded every 50 ms and like this a quality degradation over time graph is obtained. To check if the test subjects were able to reproduce their own results each test case is evaluated a second time during the whole test.

### 4.1.3 Robust Wireless Video Communications

In this section we will provide a brief review how we enabled a robust video coding subsystem in wireless mobile applications such as discussed in Section 4.1.

#### Slice-Based Coding

As discussed in Section 2.2.2, slice structure coding limits the impact of a loss to a part of a coded frame, rather than an entire frame. The effects of loss of the entire frame were shown in Figure 4.2. We now observe the impact of slice size on error resilience of this application.

To obtain the results, we use a Radio Access Bearer (RAB) that supports transmission of 128 kbps, with a radio frame size of 320 bytes. We compare three different channel configurations: loss-less, with moderate RLC-PDU loss rate of 0.5%, and finally one with a higher loss rate of 1.5%. The content used in this example is the Quarter Common Intermediate Format (QCIF) sized test sequence “Stunt” [109] at 15 frame per second. The target bitrate of the video encoder is matched to the throughput of channel while taking into account packetization overheads of the protocol stack.

Figure 4.3 shows a comparison of the effects of various slice sizes at different channel loss rates. A point on the curves represents the average the Y PSNR, of several unique channel realizations at the parameterized loss rate in order to achieve higher statistical significance. The effects of reduced compression efficiency resulting from using smaller slice sizes is visible on the error-free curve by the reducing PSNR for smaller slice sizes. However, at a loss rate of 0.5%, the drawbacks of using larger slice sizes become obvious. The advantage of using slice sizes smaller than 350 bytes do not sufficiently compensate for their overhead. However, increasing slice size beyond this results in a the drop of PSNR. This is because of a greater portion of a frame affected by a lost RLC PDU. The performance degradation is much more drastic for a loss rate of 1.5%, shown by a significant drop of PSNR for larger slice sizes.

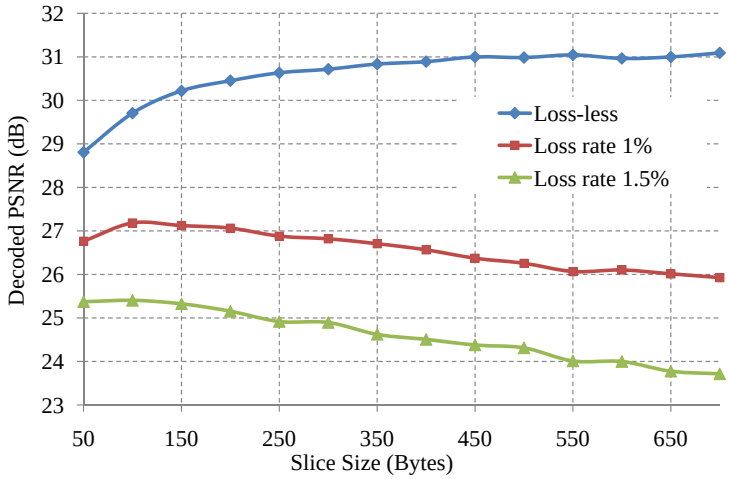



Figure 4.3: Performance comparison with varying slice sizes, with RLC-PDU loss as a parameter.

To configure the system in an error robust configuration, appropriate slice size should be selected. Typically it is close to half the radio frame size (RFS), but too small a slice size can result in considerable degradation of quality because of added restrictions on the prediction syntax. The RFS for the experiment in Figure 4.3 was set to 320 bytes. A suitable slice size observable from graph is 200 bytes.

### Intra-Coding

As discussed before, a complete intra-frame update is infeasible for the target system because of stringent channel capacity and delay requirements, hence several gradual intra-update mechanisms have been proposed, e.g. by using a pseudo-random intra-coded macroblock updates [103, 104, 105].

Another technique investigated here is temporal subsequences with instantaneous RIR tuning (SSIT), which is expressed as

$$\rho = \alpha \cdot \beta^{s-s'} \tag{4.3}$$

where on receiving the feedback of a lost packet, the RIR rate $\rho$ is instantaneously increased to a peak value $\alpha$ and is then reduced with each frame according to $\beta$. Here, the latest loss report is received while encoding frame $s'$ while the current frame being encoded is $s$. $\alpha$ and $\beta$ are tuned experimentally, and the selected value of both is 0.5. This technique expedites error recovery compared to RIR, while avoiding the buffering overheads of transmitting a complete intra frame.

### Interactive-Error Control

As discussed in the preceding sections, H.264/AVC is the state-of-the-art codec of choice for robust video communication. Several of its error resilience tools e.g. slice structure coding and flexible macroblock (MB) ordering (FMO) [110] etc. provide some robustness in an open-loop video communication system. For a more robust system, long-term memory (LTM) motion compensated prediction (MCP) along with average statistical information of channel knowledge has been employed in selecting optimal mode decisions, for example in [111, 23].

The stringent delay requirement of the target application also provides a window of opportunity. It is known that feedback-based error resilience techniques perform better for smaller delays [112]. In addition to this, a bidirectional communication link with possibility of control traffic makes feedback based techniques an ideal choice.

Further feedback based techniques in conjunction with accelerated retroactive decoding (ARD) have been investigated in [113, 114, 115]. In [116], proxy-based RPS is employed along with temporal sub-sequences for conversational applications. However, these techniques are unsuitable to be directly applied to the target system because of the complexity and delay constraints, as discussed in the following section.

A computational complexity constrained interactive error control (IEC) technique based on simple and efficient interactive error tracking (IET) has been devised and is introduced in the following. A realistic simulation of the system is done and several objective and subjective assessments will be provided that will help identify the most robust system configuration.

The IEC techniques elaborated here work on packet loss report from the receiver, which is translated to lost reference regions at the encoder by using IET described below.

**Error Tracking:** An abstract depiction of spatio-temporal error propagation is shown in Figure 4.4. In this example, a packet transmitted by the encoder at time $t$-3$T$ is lost, and the loss report is received at time $t$.

The encoder keeps a record of the recent packets it has transmitted and the corresponding reference area in each such packet. Hence the lost packet number is translated into the corresponding lost reference region of frame $t$-3$T$.

After this, error tracking is applied. We propose a technique in which lost area in a reference frame is assumed to grow at a rate equal to the motion vector search range

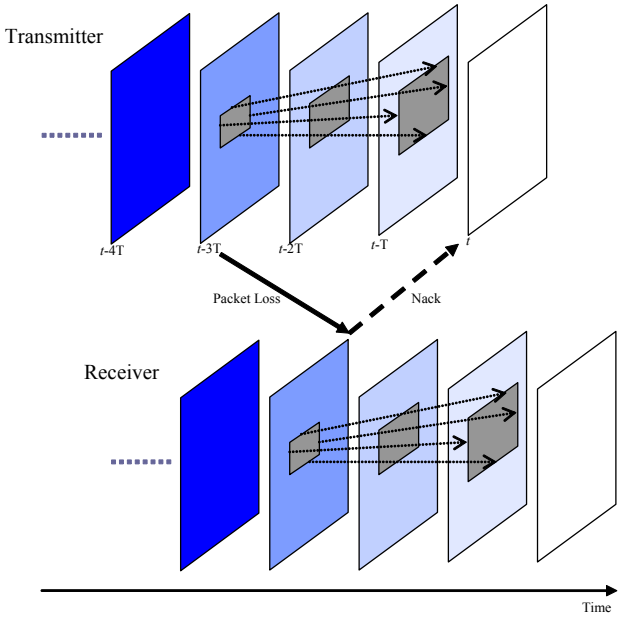



Figure 4.4: IEC with error tracking. The video frame rate is $1/T$

plus 2 pixels in each temporally predicted frame. The additional 2 pixels serve to compensate the effects of H.264/AVC fractional-pel interpolation. Simulation results will show the suitability of this technique. The shaded regions in Figure 4.4 are lost.

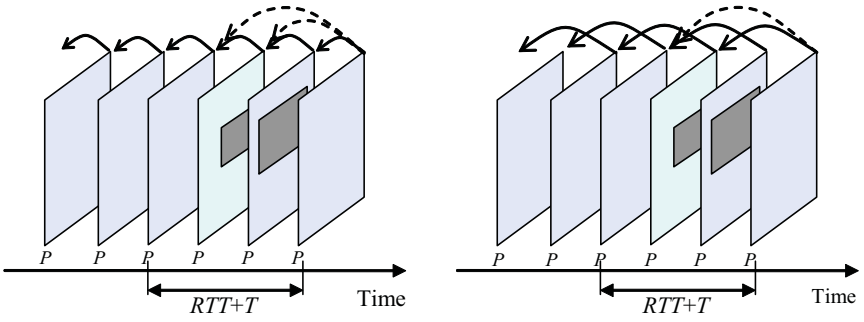
**IEC with Error Tracking:** For the proposed system, for all the blocks $b$ in a given access unit (AU), the rate ($r$) distortion ($d$) minimization problem is stated as:

$$\forall_b \quad m_b^* = \arg\min_{m \in \mathcal{O}} (d_{b,m} + \lambda_{\mathcal{O}} r_{b,m}) \tag{4.4}$$

The minimization is done for the usable option set $\mathcal{O}$ with a lagrangian multiplier $\lambda_{\mathcal{O}}$. Since long term memory (LTM) motion compensated prediction (MCP) takes a considerable part of the total resources [117], the option set is restricted to only one frame for MCP process. Two complexity constrained IEC techniques will be employed for the study, as shown in Figure 4.5.

In this system, the average round-trip-time ($\overline{RTT}$) of the feedback messages is measured dynamically. For both the configurations, reference frames up to $\overline{RTT}+T$ are kept in the reference frame buffer.

For the configuration in Figure 4.5(a), the inter-coding option set is limited to the most recent reference frame. The distortion $d$ corresponding to inter-prediction from lost regions is indeterminate as such, and hence such modes are *invalid*. If none of the inter-coding modes within $\mathcal{O}$ is valid, only then a modified option set $\hat{\mathcal{O}}$ is used by recursively including temporally older reference frames one by one until at least




(a) Most-recent frame MCP (b) Sub-sequences

Figure 4.5: Proposed IEC configurations.

one valid inter-coding mode is found. Intra-coding modes are unaffected by this process.

For the second configuration as depicted in Figure 4.5(b), the inter-coding option set is limited to the reference frame that temporally precedes the frame being encoded by $\overline{RTT}$. The modification of option set is done in the same way as for IEC1, except that there is only one additional, temporally preceding, reference frame available in case of errors. This technique will be referred to as IEC2. The additional robustness by using reference frames $\overline{RTT}$ away from the current frame is that the loss report typically arrives before the reference is used for MCP and the probability of error propagation is reduced as such. However, temporally older reference frames reduce the compression efficiency, hence the cost-benefit analysis will be provided by this work.

**Effect on the computational resource usage of the encoder:** The number of modes in $\hat{\mathcal{O}}$ are less than or equal to the number of modes for error-free case $\mathcal{O}$. Since the implementation does not require actual distortion calculations for invalid modes, this technique, referred to as IEC1, ensures that the computational resource usage stays strictly inbound in the case of error reports.

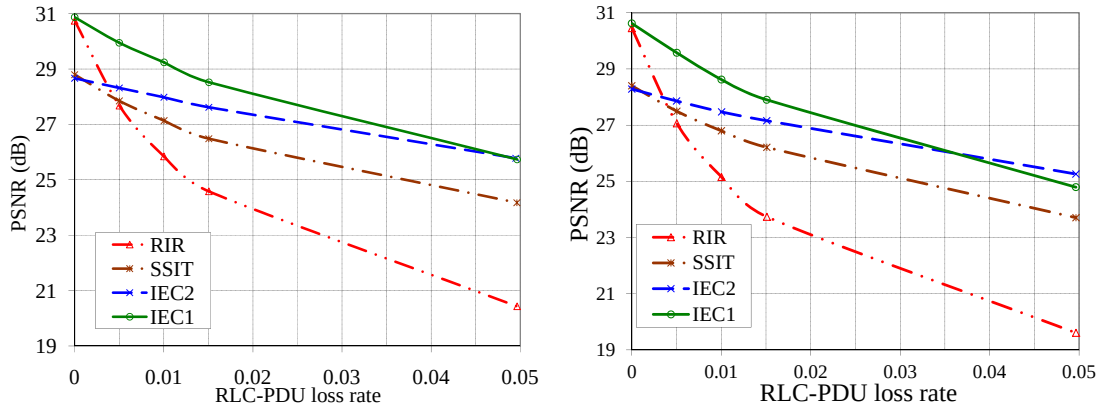
It should also be noted that the loss report handling and IET are packet-based processes. For the practical system configuration as investigated later, there are typically less than 10 packets per frame. Hence the computational overhead of such processing is negligible compared to the rest of the codec complexity.

### 4.1.4 Performance Evaluation

The simulation environment used for generating the results has been documented in Section A.1. The video sequences have been selected by the video adhoc group within 3GPP [118]. The results are reported for QCIF sized sequences “Party” at 15 frames per second (fps). A 3GPP channel simulator [118] with realistic loss patterns is employed. The radio access bearer supports 128 kbps. In order to achieve better statistical significance, each test is repeated with 128 different channel realizations,

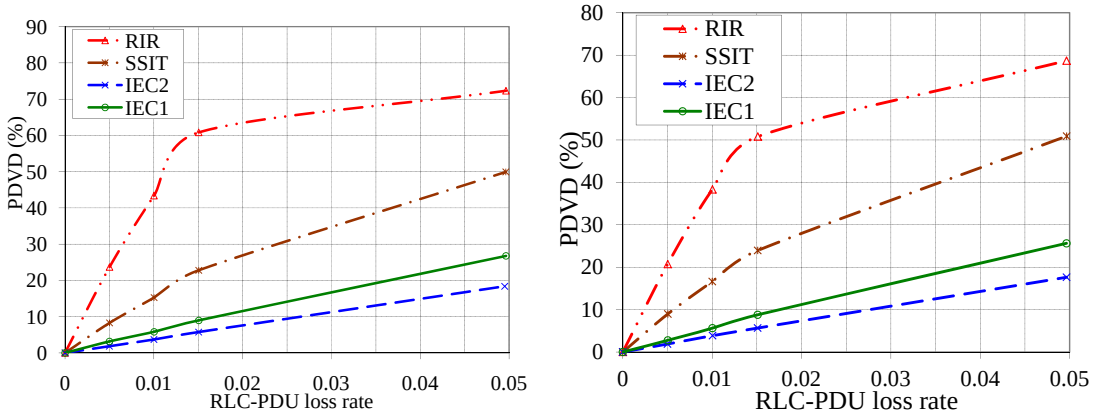
and the readings averaged. In a realistic depiction, the feedback traffic is multiplexed along with normal video traffic, which is exposed to a RLC-PDU loss rate of 0.5%.

### Results based on objective metrics




(a) Test sequence "Party" (b) Test sequence "Stunt"

Figure 4.6: Average results based on PSNR vs. RLC-PDU loss rate performance




(a) Test sequence "Party" (b) Test sequence "Stunt"

Figure 4.7: Average results based on PDVD vs. RLC-PDU loss rate performance

Figure 4.6(a) show the decoded PSNR vs. RLC-PDU loss results for sequence "Party." The measured $\overline{RTT}$ for the experiments was less than 130 ms and hence 3 reference frames were used. RIR rate was set to 5% of the total MBs in a frame. It can be seen that RIR technique performs worst. Using temporal sub-sequences along with instantaneous RIR tuning performs reasonably better for lossy channel conditions. IEC2 gives better performance, however the best configuration is IEC1. It is evident that the quality loss incurred by using sub-sequences, because of temporally older references, can not be compensated by the error robustness it adds for all practical loss scenarios. IEC1 shows an improvement of 4 dB at a loss rate of 1.5%.

A very similar trend has been seen to be followed by other test sequences, e.g. "Stunt", shown in Figure 4.6(b) and Figure 4.7(b).

Figure 4.7(a) also shows that the proposed configurations give a significantly robust system with only about 20% of the video affected under the worst case losses of 5%. The feedback overhead only amounts to a maximum of 0.8% of a 128 kbps channel at a loss rate of 5%.

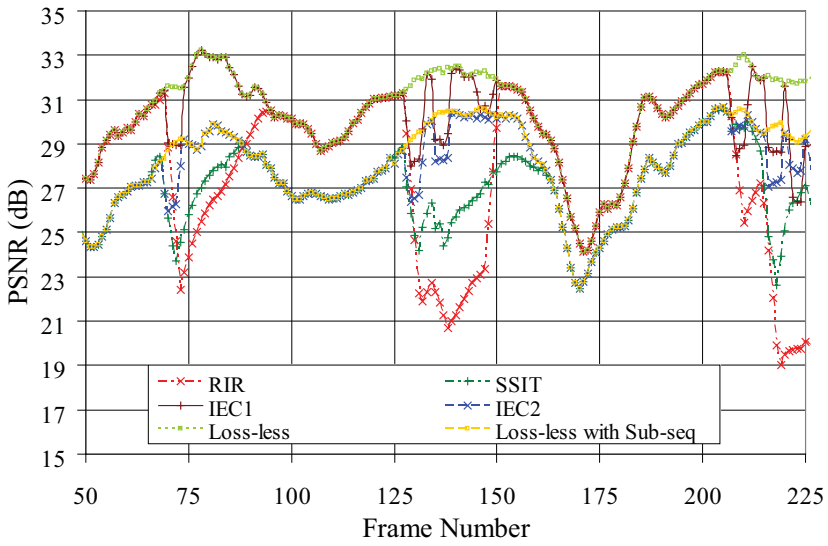



Figure 4.8: PSNR variation within sequence "Stunt"

Figure 4.8 shows the variation of PSNR for the sequence "Stunt" for one channel realization. In addition to the results for all the techniques at an RLC-PDU loss rate of 1.5%, error-free results with *and* without using sub-sequences are plotted. Only a selected portion of sequence is plotted for better viewing. IEC techniques show very fast recovery from the losses. The results show as expected from theoretical basis, IEC based techniques recover from the effect of loss as soon as the feedback is received, compared to the gradual intra-update based techniques.

### Results based on subjective tests

Figure 4.9 shows the instantaneous visual comparison of frame number 220 from Figure 4.8. In this case, the subjective results follow closely to the objective analysis.

However, the objective results can be misleading in many other cases. To evaluate this, detailed subjective test were conducted for the presented data by 16 students not active in the field of video coding (non-experts). The setup has been specified in 4.1.2. All subjects except one could very well reproduce their own results, the results from the latter were removed from the final results. In addition to detailed quality versus time results, the mean quality over time for a give test case is also given for an easy, direct comparison.

As an example, the results for the sequence "Party" are plotted in Figure 4.10, for

(a) RIR (b) SSIT

(c) IEC2 (d) IEC1

Figure 4.9: Subjective comparison for the investigated techniques.

more detailed analysis the reader is referred to the results we presented in [119]. The x-axis is the playback time of the sequence, and y-axis is the continuous quality scale from 0 to 100, the maximum value representing no observed visual degradation compared to the reference video. The plotting is done by removing an initial transient period required by testing subjects to adapt to the displayed video. In addition, on the extreme right, *mean observed quality* is also plotted for a rough but quick comparison. The number along with labeling of the curves represents the RLC-PDU loss rate.

It can be seen from the results that IEC1 and IEC2 show superior performance than SSIT and RIR at high loss rate of 5%. However, in this range, IEC1 is only marginally superior compared to IEC2. At moderate loss rates of 1.5%, IEC1 performs much superior to IEC2. At this loss rate, the reported performance of IEC2 is no more better than the simple RIR. This is contributed by the relatively large camera panning motion in the sequence resulting in further degradation of reference signals. Another interesting observation is that for large RLC-PDU losses of 5%, SSIT performs almost as worse as RIR. For moderate losses of 1.5%, SSIT performs even worse than RIR.

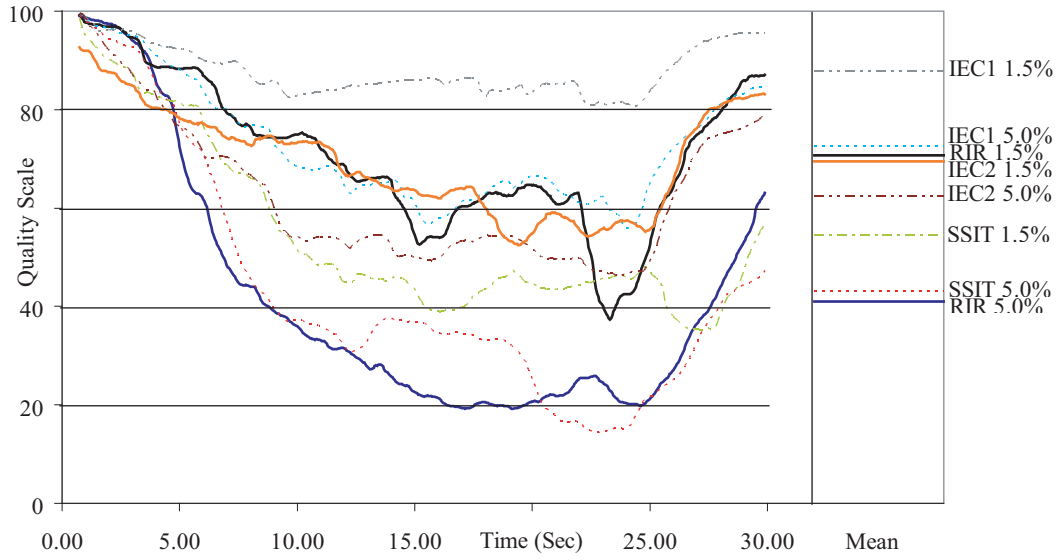



Figure 4.10: Subjective test results for sequence "Party"

### 4.1.5 Discussion of Results

As seen in the results, the visual quality of error protection by the use of subsequences only is not sufficient and for some sequences is even below the quality that can be achieved by using very simple random intra-MB refresh (RIR) technique. The popular concept of using temporal subsequences to achieve added error robustness severely degrades the subjective performance because of flicker artifacts. Several results show that users would rather prefer no IEC than using subsequence based SSIT, although a large objective performance enhancement is achieved. Also, even

without any channel losses, the prediction degradation of subsequence based IEC2 considerably degrades the performance and it is concluded that for the given system, subsequence based error control techniques are not suitable.

This subjective result is in contrast to what PSNR values have suggested in Figure 4.6(a). While for IEC1 PSNR values can at least capture the tendency of the visual quality, the visual quality for SSIT for this sequence is clearly below the visual quality for RIR.

This demonstrates that for such investigations, PSNR values can be severely misleading. The reason for this poor correlation between PSNR values and visual quality for the technique of subsequences is rapid fluctuations in video quality because of individual subsequences being affected by loss in an un-even, independent fashion. This fluctuation is reported as an annoying flicker artifact by subjects evaluating the video sequences. However, metrics like the PSNR cannot show the impact of such artifacts.

In spite of this, subjective tests are manyfold time-consuming and expensive. Hence for system design, such tests should be planned carefully at intermediate levels and less frequently compared to objective evaluation.

Overall, IEC1 provides a robust system, and a considerably smaller performance margin between different loss scenarios enhances the QoS. Hence the IEC technique using most recent reference frame is best performing solution for the target system and it improves the overall performance of the system by 4 dB for a moderate RLC-PDU loss rate of 1.5%, with more advantage for higher losses. On top of that, it has a negligible impact on the computational complexity of the system.

## 4.2 Telepresence Systems

Telepresence is the experience of being immersed in a location distant from one's physical location, and a wide variety of application scenarios is possible to achieve this goal. A common denominator in these systems is a multi-view (typically stereo) camera system that transmits stereo video to a remote observation site.

In most application scenarios, the channel capacity is limited between the sites and hence source coding of transmitted data is necessary. Typical choice for source coding of video is standardized lossy video coding techniques like H.264/AVC or MPEG-4 ASP.

At the observation site, an observer is immersed in the remote scene using the received video and a generated virtual 3D environment. Applications where the observer can physically interact back with the transmission site are called as telepresence and teleaction (TPTA) systems. A *teleoperator* carries the stereo camera system and the observer is referred to as human operator. The remainder of this section will talk about TPTA systems, telepresence system being a subset of that. Such a scenario is relevant to a broad range of applications, including remote education, entertainment, advertising, nuclear industry, outer space etc, e.g. see [120]

for an overview. Interestingly this diverse field of applications also comes with the challenges of diverse demands and constraints.

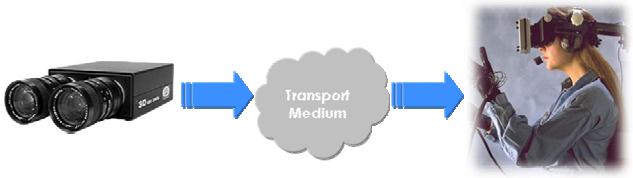



Figure 4.11: A depiction of a Telepresence scenario.

### 4.2.1 Computer Vision Techniques For Telepresence Systems

In addition to monitoring in real-time, another main use of the received video at the observation site is to virtually immerse the observer in the remote scene. This is done by using the received video and a generated virtual 3D environment. The generation of virtual 3D environment at the human operator site requires *stereo matching* [121, 122] to be performed on the received video. Most well known and suitable stereo matching algorithm for TPTA are real-time algorithms based on dynamic programming (DP) [123, 124] and belief propagation (BP) [125, 126, 127].

### 4.2.2 Resource Constraints in TPTA Systems

Evidently, even the real-time stereo matching algorithms require a significant amount of computational resources. On the other hand in many TPTA application scenarios, the teleoperator site is severely constrained in terms of available computational resources, e.g.

- Frequently the teleoperator is a mobile entity with self-contained constrained portable power source.
- The teleoperator might have to survive in a hostile environment for extended periods of time without power replenishing, e.g. in nuclear or space environments.
- Some solutions focus on the extension of an existing mono-view system to a 3D system. In order to reuse such system the upgrading is done only at receiving end [128]. Likewise the multiple video sources can be decentralized, e.g. 2 robots each with a mono view camera. Stereo matching in this case can only be performed remotely.
- The teleoperator might be an inexpensive unit with limited processing power e.g. a hand-held mobile device with stereo cameras used for personal communication or distant learning, etc.

In order to enhance the operational time of the teleoperator and reduce the computational demands, it is prudent to perform stereo matching away from the teleoper-

ator, e.g. at an intermediate proxy or receiving end where the resources are not so constrained.

### 4.2.3 Resource Optimization in TPTA Systems

The main computational resource optimization task at the teleoperator site is to address the real-time video coding. In spite of the fact that in telepresence system, the target video quality might be significantly higher than, e.g. in consumer-end mobile video conversational applications, there are quite a few similarities:

- **Low-end to end delay:** As for a conversational application, minimization of end-to-end delay is critical in teleoperator systems. Hence the principals of stringent buffering requirements also exist here.
- **Error robustness:** As with low delay conversational applications, the task of designing an error robust system is challenging. Although here, with more computational resources available at the decoder end, techniques based on ARD might also be feasible.
- A bi-directional communication link is present for teleaction.
- As mentioned before, computational resources are constrained at the teleoperator site.

### 4.2.4 Review of Video Quality Evaluation in TPTA Systems

Performing stereo matching at the receiver site on the data that has been subject to lossy source coding is currently not well understood. Much of the work relevant to the transmission of multi-view content in conjunction with stereo matching has been done in the area of 3D TV. However, this application has much different constraints than TPTA. In contrast to TPTA, in this application the resources at the transmission end are higher compared to the receiving end. Therefore, the typical research focus is based on performing stereo matching at the transmitting end, e.g. [129, 130, 131]. In [128], the authors do focus on performing stereo matching at the receiving end. The target of their study however, is the evaluation of multi-view video compression alone and does not focus on the impact on stereo matching.

This issue is also important since standard video codecs like MPEG-4 ASP or H.264/AVC are designed for human psychovisual (HPV) model; optimization for computer vision algorithms like stereo matching is not a design criteria for them. Likewise, the subjective and objective metrics discussed in Section 4.1.2 are also not directly applicable.

This is evident from Equation 4.4, which is equally applicable in this application. The main target is to determine the relation between the cost function of RD minimization ($d$) and that of stereo matching algorithms. Since both minimizations have different targets and constraints, it is evident that the two are not linearly related.

**Performance Metrics**

Quantitative evaluation is required for both the received video sequences and the estimated disparity maps. To evaluate the quality of the estimated disparity maps from the compressed images, the percentage of bad pixels (PBP) of the reconstructed disparity map is used. For all the regions of an $M \times N$ sized image it is defined as

$$\begin{aligned} \text{PBP}_{\text{ALL}} &= \frac{\sum_{x=1}^{M}\sum_{y=1}^{N} f(d_{x,y}, \hat{d}_{x,y})}{M \times N}\%, \\ f(d_{x,y}, \hat{d}_{x,y}) &= 1 \text{ if } |d_{x,y} - \hat{d}_{x,y}| > C, \text{ else } 0, \end{aligned} \tag{4.5}$$

where $\hat{d}_{x,y}$ is the disparity at a pixel location in the reconstructed disparity map from the compressed image and $d_{x,y}$ is the corresponding disparity in the reference disparity map. The explanation of the reference disparity map will be given in the following section. $C$ is the pre-defined threshold. In a similar fashion, $\text{PBP}_{\text{NON OCCL}}$ and $\text{PBP}_{\text{DISCNT}}$ are calculated exclusively for the non occluded and discontinuous regions of the image respectively, to give further insight into the performance.

### 4.2.5 Evaluation Results and Discussion

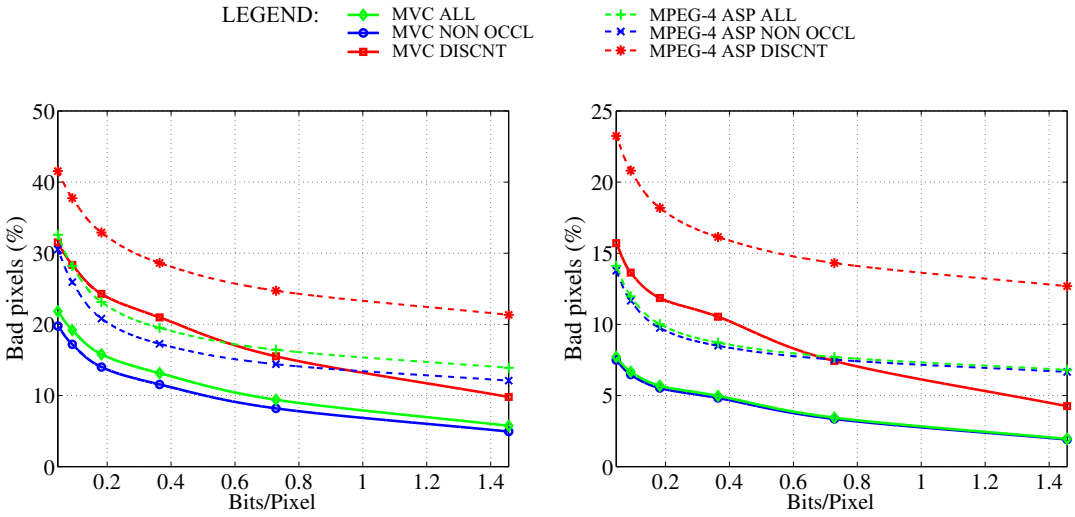
The framework used for the generating the results is described in detail in Section A.2. The most suitable test content for the target system comes in the form of multi-view video sequences used for the evaluation of H.264/AVC Annex H (MVC). Natural scene content is best suited for most application scenarios, and it typically contain significant noise. For multi-view sequences, the results for two views close to the center of camera grid are shown here.

A variety of stereo matching techniques have been evaluated. The results for the techniques based on the real-time DP algorithm of [124] that employs a high quality adaptive weighting approach described in [132] to compute the matching costs and is relatively more complex. This is a scanline based technique since the disparity map is computed separately for each image scanline. Such techniques are less computationally intensive than global techniques, say based on belief propagation (BP) optimization principle [125].

A single solution to RD minimization for an access unit is not a sufficient measure to highlight the dependence between the two cost functions. In order to achieve higher statistical significance, several solutions are achieved by starting the minimization at different blocks within an access unit and then applying it to all blocks in a loop-around fashion. Each different iteration gives a unique solution and a corresponding value of PBP. At least 128 such readings are then averaged to get the final value in order to have a higher statistical significance.

From a system design point of view, the PBP plot versus the transmission bitrate is the most meaningful, since the channel capacity is the actual resource of concern. However, given the rate distortion (RD) curves, the transmission bitrate can directly be transformed into PSNR.

**Results**




(a) "Vassar" with BP

(b) "Vassar" with CDP

Figure 4.12: Performance results for test sequence "Vassar"

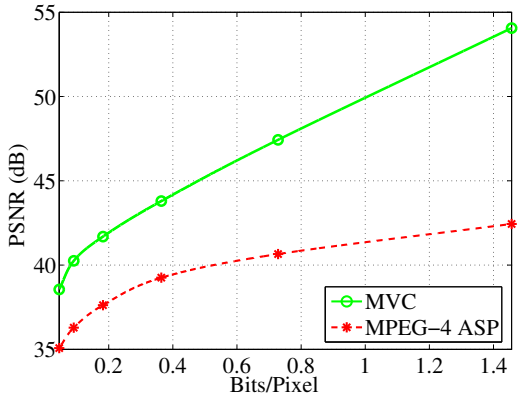



Figure 4.13: Rate-distortion performance comparison of MPEG-4 ASP and MVC for "Vassar"

Figure 4.2.5 shows the the performance for views 3 and 4 of the VGA sized test sequence "Vassar," shot at 25 fps. PBP (shown as "Bad pixels (%)") is plotted against transmission bitrate. $\text{PBP}_{\text{ALL}}$, $\text{PBP}_{\text{NON OCCL}}$ and $\text{PBP}_{\text{DISCNT}}$ are shown as "ALL," "NON OCCL," and "DISCNT" on the graphs, respectively. RD characteristics are given in Figure 4.13. It should be noted that the transmission of uncompressed video content requires 24 bits/pixel. The parameter $C$ for calculating PBP is set to 0, since a small change in the disparity might lead to a large error in the interpreted depth. The results for H.264/AVC Annex H are represented by "MVC". Only baseline profile is used to limit the resource usage, with a single temporal reference and a single frame as inter-view reference.

The main result of interest is the curve "ALL," since it gives the overall performance

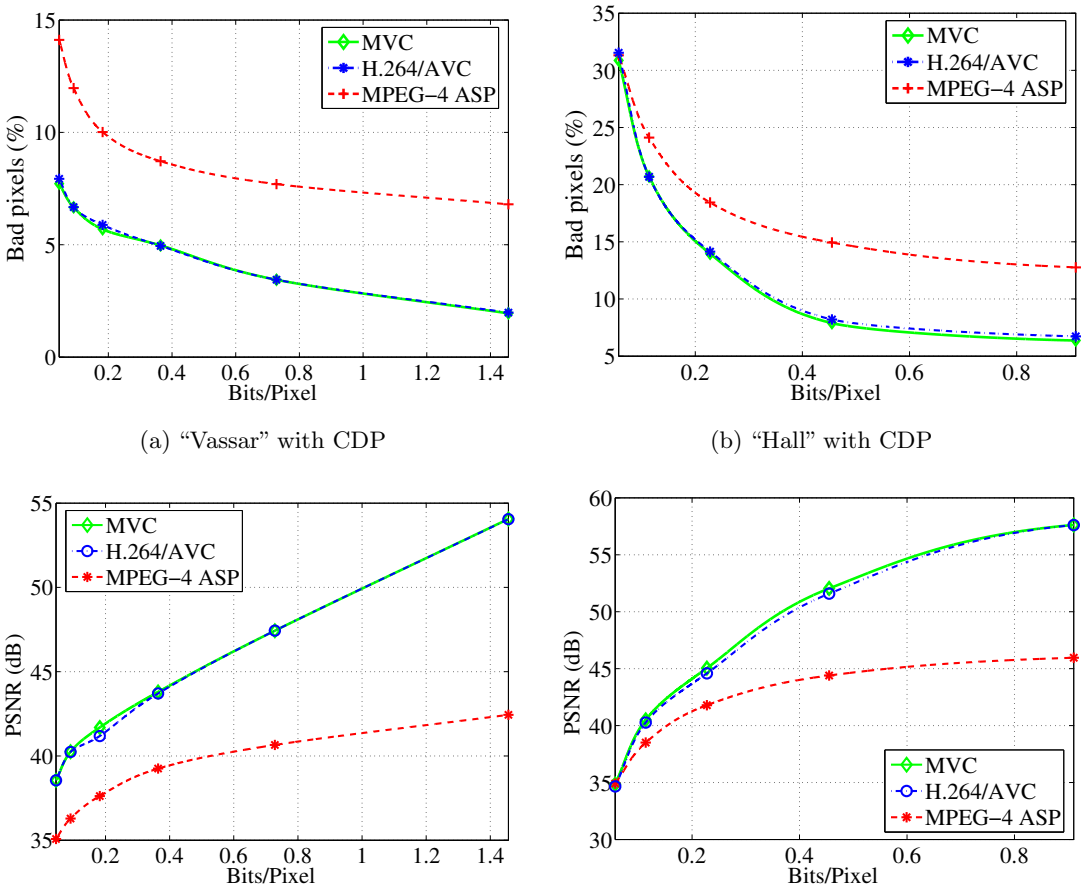
of all the regions of the image. It can be observed that for BP, even at the highest transmission rate 14% of pixels are still erroneous with MPEG-4 ASP, in contrast to 5% for MVC. The performance is still not converging to its best even at high data rates. CDP is the most robust technique amongst several stereo-matching techniques that were evaluated, and with MVC it converges to less than 3% bad pixels above 0.8 bit/pixel (a compression factor of 30 times). The best performance of CDP with MPEG-4 ASP shows 7% bad pixels.

It has been demonstrated in [133] that MVC achieves a better and robust performance across the board. This is contributed by the RD curves shown in Figure 4.13, where MVC achieves a gain of more than 5dB compared to MPEG-4 ASP in the vicinity of 0.8 bit/pixel. Typical PBP losses are in the vicinity of 5% and 3% at around 1 bit/pixel when MVC is used in conjunction with BP and CDP respectively. It can also evident that MVC shows extra performance gain for discontinuous regions of the image (plotted as "DISCNT") as compared with MPEG-4 ASP. This is as expected, since compared to the traditional 8x8 transform size in MPEG-4 ASP, MVC is equipped with a 4x4 transform size that reduces the *ringing artifacts*, especially around object boundaries and discontinuities [21]. For more detailed results the reader is referred to our work in [133].

It is worthwhile to comment on how much is actually gained by Annex H of H.264/AVC. Figure 4.14 shows the performance of the 3 codecs in comparison: H.264/AVC Annex H (shown as MVC), H.264/AVC without Annex H (shown as H.264/AVC) and MPEG-4 ASP. Only "ALL" regions are plotted to avoid a clutter. From the performance results and RD curves of "Vassar" it can be seen clearly that there is no observable performance difference between MVC and H.264/AVC. There is only a marginal advantage for the sequence "Hall". It is as expected since the only time spatial prediction provides a higher compression gain is when temporal predictor is considerably bad. This is possible when the frame rate is significantly low, as for the sequence "Hall," which has a framerate of 5 fps.

It can be observed that a compression factor of 30 or less in general should be selected to avoid a large performance degradation. Also, although global stereo matching methods like BP are more complex than the scan-line based techniques, CDP shows the best results here for natural scene content that has significant noise. If MPEG-4 ASP is chosen owing to its lower complexity as compared to MVC, CDP should be used as a candidate stereo matching algorithm to avoid large performance degradation. CDP shows the best results in conjunction with MVC; at a compression factor of 30, the results for CDP along with MVC converge to a mere 3% bad pixels. The resulting system configuration is extremely robust; considerable complexity of stereo matching can be removed from the resource constrained transmission end while having minimal performance degradation and a significant transmission data reduction of 30 times. MVC also enables enhanced performance around spatial discontinuities of scene content, owing to its small transform block size.

It is also observed that for a delay constrained stereo video application, MVC does not provide any dramatic gain over traditional H.264/AVC. The gains come only when the target frame rate is so low that the quality of temporal predictor is significantly degraded in comparison to the spatial predictor.




(a) "Vassar" with CDP

(b) "Hall" with CDP

(c) "Vassar" RD

(d) "Hall" RD

Figure 4.14: Comparisons for H.264/AVC and MVC.

### 4.2.6 Discussion of Results

Following observations are of interest:

1. Experts in fields of video coding would recognize a video coded at 35-40 dB as high-quality content. The objective tests for the target system reveal that for this system, this is not nearly sufficient. For a robust system configuration, the required PSNR is in the vicinity of 50 dB.

2. Subjective tests for this system are not very helpful. At this quality, content might be indistinguishable by test subjects, while the objective evaluation reveal even the indiscernible quality difference impact stereo matching significantly.

3. Objective results provide sufficient guidance to select suitable components and operating points for the target system.

4. MVC does not provide any considerable performance gain over traditional H.264/AVC for the target system at normal NTSC/PAL frame rates.

Comparison of the findings in Section 4.1 and Section 4.2 brings us to a very important conclusion: there is no single optimal option in the domain of quality metrics and evaluation for video, they have to be carefully selected based on the target application and the evaluation has to be formulated carefully to meet the application demands.

# Chapter 5

# Computational Resource Optimized Video Codec

In Chapter 3, various approaches to address a computationally constrained system were reviewed. It is evident that a wide majority of literature in this field tries to mange the computational resource usage of the video encoder only. This is contributed by the factor that the motion estimation required at the encoding end makes the task of encoding many folds more complex than the task of decoding. Yet some other studies focused on solving the computational resource management problem at the decoder end. However it was observed that a comprehensive technique that has the potential to address the problem for a variety of application scenarios is missing. Many of the investigations are in fact presented as an information theoretic point-of-view: they optimize a part of the source codec without the context of the target application and the deployment problem that needs to be addressed.

In the following section we propose a classification of the various video source coding configurations that are targeted for computational resource optimization. This is followed by the approach that we propose to address a variety of these configurations. As discussed before, instead of enabling a highly optimized application where the design principles have little relevance to another variant, a generalized solution is proposed that can address a variety of systems.

## 5.1 System Classification

As highlighted in Chapter 4, video coding applications come in various shapes. These variants have different resource optimization considerations. Figure 5.1 shows an abstraction of the generalized system architecture that can encompass these various applications, where $n = \{1, 2, \cdots, N\}$. Various sets of hops, proxies and heterogeneous networks may be a transparent part of the overall system. In the following we classify the main parameters of interest that distinguish these application variants in terms of the applicability of a resource optimization strategy.

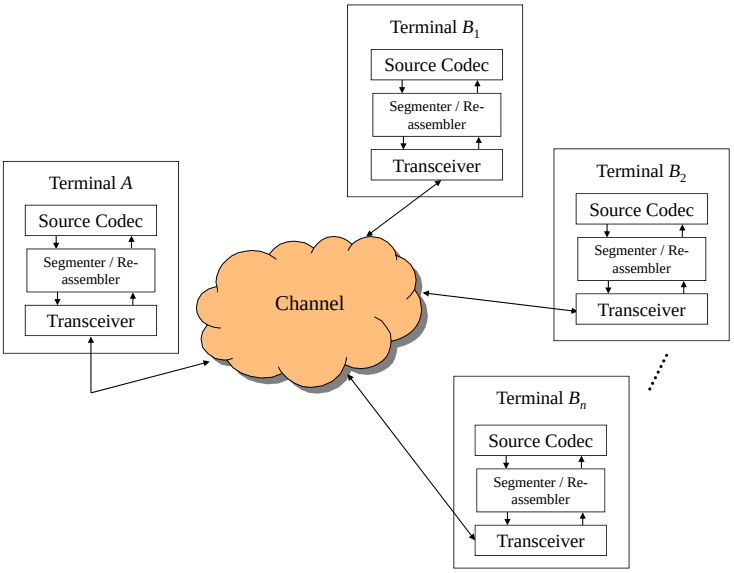



Figure 5.1: An abstraction of a generic video coding application

### 5.1.1 Source Codec Configuration

The most important parameter that impacts the resource optimization strategy is the configuration of the video source codec used by the application.

#### Type of Adaptation

On the basis of the type of adaptation possible, we distinguish between *online* and *offline optimization* configurations.

**Online Optimized Configuration:** Online optimization in this context is when optimization for resource usage is done in real-time, while generating the coded video content, e.g. on a frame or even on a sub-frame level. From an architectural point-of-view, such applications allow for resource optimizations embedded *within* the video encoder module. Hence a resource optimized video conversational application and live multicast are example applications of online optimization.

Note that in this context, even an application that encodes a previously saved content in real time may be suited for an online optimization, e.g. a streaming application that has to transcode the video data for each session. Real-time transcoding is however typically avoided in applications because these are not commercially scalable with the number of clients.

**Offline Optimized Configuration:** Offline optimized configuration in this context are used in applications where real-time resource usage optimizations are not possible; in terms of architecture: when the optimization is not possible within the video encoder.

The optimizations might still be possible for the receivers after the segmentation of the coded content on a GOP level or even less frequently. However, these optimizations are quite different from online optimization, as in this case the suitable content is merely selected post encoding. It is evident that this variant offers lesser optimization flexibility.

A typical example would be an adaptive streaming application, where at the output of segmentation at terminal $A$, a set of representations with different coding parameters are available. An optimization at this level can be selection of a suitable representation for a given client based on the available resources. The encoder or sets of encoders might be encoding live content or the content might be pre-encoded.

### Codec Topology

On the basis of codec topology, the video source codec configurations are distinguished between *point-to-point* and *point-to-multipoint* topologies.

In a *point-to-point* codec topology a sending terminal or sender delivers a unique video content to a receiving terminal (receiver, for short notation) over the channel. An example of this topology is in a video conversational application, where a person is communicating with only one other person. Hence in Figure 5.1, $N = 1$, i.e. only two terminals are communicating with each other: $A$ and $B_1$. The source coding module at both the terminals $A$ and $B_1$ perform both real-time video encoding as well as decoding simultaneously. After the network abstraction at the source coding level, there is little need for segmentation of the content in order to minimize the end-to-end delay. Both the terminals are connected via bidirectional communication link.

In a *point-to-multipoint* topology, a sender is sending the *same* video content to multiple receivers over the channel. In this case for Figure 5.1, $N \geq 1$.

An example of this configuration is in a (possibly live) multimedia multicast application over packet switched networks, e.g 3GPP Multimedia Broadcast Multicast (MBMS) applications [134, 135]. Terminal $B_n$ will only employ a real-time video decoder, while for live content terminal $A$ will employ a real-time video encoder.

From a resource-optimization framework perspective, point-to-point configuration offers a greater potential for optimization as compared to point-to-multipoint optimization; for the latter several terminals share the same video representation while each may have a different resource budget. Hence for point-to-multipoint topology, some terminals may perform suboptimally.

It should however be noted that the typical adaptive streaming applications like MPEG DASH also fall under point-to-point topology: although in this case for Figure 5.1, $N \geq 1$, the sender can share a unique video representation with a given receiving terminal, independently of another receiver. Hence this configuration can be decomposed into $N$ unique point-to-point configurations.

#### Direction of Communication

Here we distinguish between *unidirectional* and *bidirectional* communications. Bidirectional communications offer the best possibility of adaptation; both sender-end and receiver-end resource optimizations are feasible. Even for unidirectional communications, there is a distinction between *receiver-aware* and *receiver-agnostic* applications. The former has a possibility of a backward data flow when the receiver joins a session, in the form of a handshake protocol etc. This allows for some information exchange necessary for system-wide optimizations. Receiver-agnostic applications on the other hand do not offer this possibility, and hence offer the least flexibility for an optimization framework.

### 5.1.2 Channel Characteristics

Several channel characteristics significantly impact any video coding application, e.g. capacity, end-to-end delay, and error susceptibility. The first two attributes often together determine the third: typically low delay and severely capacity constrained channels are error prone, as discussed before in Chapter 4. However, in terms of the impact on resource optimization framework, error susceptibility is of prime interest. Specifically, the optimization framework has to cater for the impact of channel losses on dynamic resource utilization by the terminals.

## 5.2 System-wide Timing Analysis

As the video data flows from the system from the uncompressed end to the decoded end via the encoder, transport medium and the decoder, several timing constraints are to be observed. Both the ends, i.e. the uncompressed video data generation or video capturing and the decoded video display are assumed to work synchronously, in a periodic fashion. It implies that capturing frame interval, and the decoded frame rendering interval is the same, i.e. $T^{cap} = T^{rend} = T$ (substituting $T$ in place of a time interval $\Delta t$ for simplification of notation). This also implies identical raw data generation and consumption rate, respectively. For the sake of simplicity and without loss of generality, temporal sub-sampling and/or spatial scaling are considered to be external to this setup.

A crucial timing analysis is typically formulated with the assumption of constant bitrate transmission on the channel. Hence in Figure 5.2, the bitrate at the input and the output interface of the channel is constant, denoted by $R$. Since even with the most sophisticated rate-distortion minimization algorithms a strictly constant bitrate coding cannot be guarantee (bar using bitstream stuffing), the output of the encoder and the input of the decoder work at a variable bitrate as shown by the bold interfaces in Figure 5.2. To compensate for this discrepancy of a variable bitrate interfaces mixed with constant bitrate in a single chain, buffering ($B^{HRD}$) must be employed as shown in Figure 5.2

The timing and the data flow of this buffered system is addressed in detail for

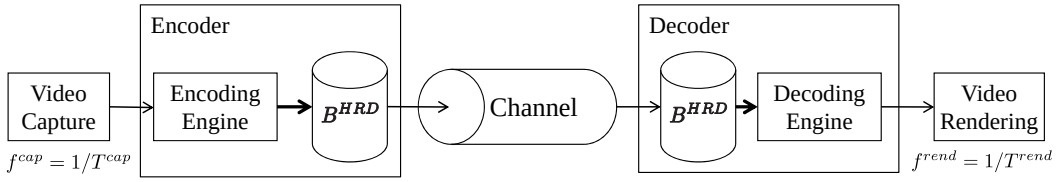



Figure 5.2: Hypothetical reference decoder (HRD) data flow and the buffering

standardized codecs by a mechanism referred to as video buffering verifier (VBV) or recently for H.264/AVC, a part of the hypothetical reference decoder (HRD).

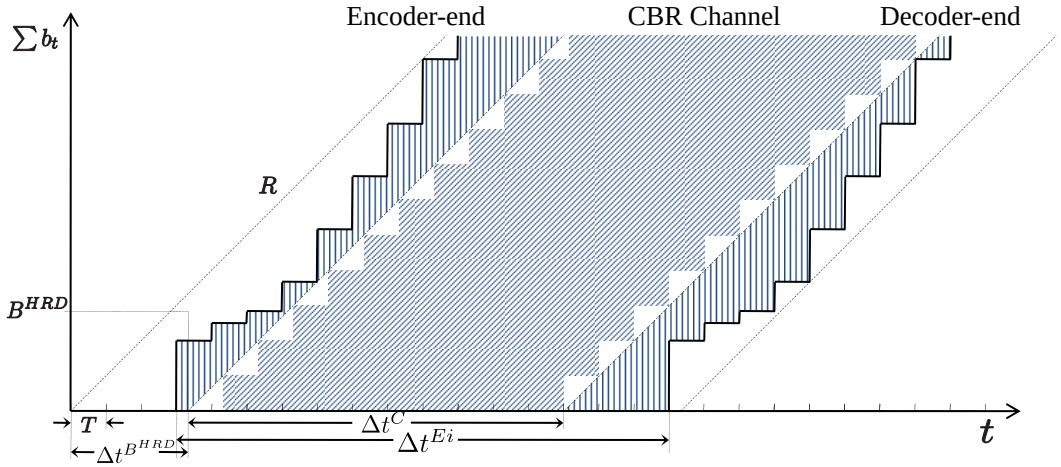



Figure 5.3: The timelines and intervals of different processes in a traditional hypothetical reference decoder (HRD) model

As an example Figure 5.3 shows the timing analysis typically done for the VBV formulation, as in [84]. The ticks on the time axis $t$ have a period $T$, and are time-aligned to the frame capturing *and* rendering times. For simplicity and without loss of generality, the composition delay $\Delta t^c$ is assumed to be zero, hence the composition time is equal to the decode time, as in a forward-only inter-predicted video. The amount of bits processed by the system at any time is denoted by $b_t$, the end-to-end network-related delay is denoted by $\Delta t^C$, the buffer size is $B^{HRD}$, the corresponding maximum buffering delay is $\Delta t^{B^{HRD}}$, hence in this model the ideal end-to-end delay $\Delta t^{Ei}$ is given by

$$\Delta t^{Ei} = \Delta t^C + \Delta t^{B^{HRD}}. \tag{5.1}$$

The important assumption in this system is that the system (both the encoder and the decoder) is equipped with infinite resources and hence the encoding and decoding steps are temporally atomic. This assumption has quite pervasive implications as we see in the following.

### 5.2.1 Video Complexity Verifier

In any realistic system, the encoding and the decoding steps are not atomic and for encoding/decoding each frame in Figure 5.2, each terminal requires a variable time duration. Hence the thick lined boxes of Figure 5.4 represent the modules that require a variable delay in processing the information. As a result, the input interface of the encoder and the output interface of the decoder will require additional buffering to cater for this variable delay. At the same time, this delay does not have a direct relation to the variation of the coded bitrate (compensated by $B^{HRD}$), hence a buffer is added immediately at the output of the encoder and the input of the decoder. In order to reuse the existing VBV models, these buffers are treated separately from the HRD. They can however be combined when required in the final model.

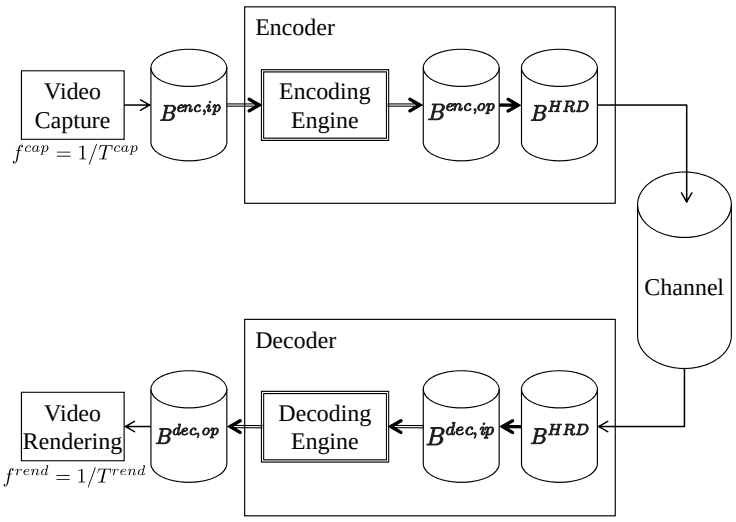



Figure 5.4: An abstraction of generic video complexity verifier (VCV)

If a maximum timing jitter of $\max_p(\Delta T_p^{enc})$ and $\max_p(\Delta T_p^{dec})$ is allowed at the encoder and the decoder, respectively, where $p$ denotes a frame index, the size of pre-encoder and post decoder buffer in number of frames is given by

$$\begin{array}{rcl} B^{enc,ip} & = & \text{ceil}\left(\left(\max_p\left(\Delta T_p^{enc}\right)\right)/T\right) \\ B^{dec,op} & = & \text{ceil}\left(\left(\max_p\left(\Delta T_p^{dec}\right)\right)/T\right) \end{array} \tag{5.2}$$

respectively. Both the now required additional post-encoder and pre-decoder bitstream buffers must have a size of $B^{HRD}$, since theoretically a VBV compliant system can generate an amount of data up to $B^{HRD}$ in an infinitesimal amount of time. The end-to-end delay $\Delta t^E$ of the system is increased to

$$\Delta t^E = \Delta t^{Ei} + T \cdot \left(B^{enc,ip} + B^{dec,op}\right). \tag{5.3}$$

where $\Delta t^{Ei}$ is given by Equation 5.1.

Figure 5.5 extends the timing analysis of the HRD to include the computational complexity aspects as well, according to the architecture in Figure 5.4. As in other

literature, we refer to this timing formulation as the video complexity verifier (VCV). It is worthwhile to note from the diagrams that the VCV has a few things in common with the VBV, but quite a few differences as well.

- The VBV at the encoder end is frequently compared to a leaky bucket analogy; a buffer with variable inflow and a constant outflow. At the decoder end this analogy for VBV is "inverted". The VCV however at any terminal is based on a variable computational demand by video data fed at periodic intervals of time, and processed by essentially constant application of computational resources. While the VBV timings at the encoder are mirrored at the decoding end, there is essentially little similarity or relation in the shapes of the VCV timing diagrams. The VCV timing at each terminal depends on the implementation as well, in addition to the coded content.
- VCV is a virtualization of the system; corresponding to a variable resource demand and constant resource supply are the video frames (coded or un-coded) flowing within the system. Hence unlike the VBV buffer fullness the VCV buffer fullness does not directly indicate the actual video frame buffer fullness, rather the fullness of a "virtual resource usage buffer."

With the encoding and decoding being non-atomic processes, the step boundaries in the VBV indicative of the addition and removal of bits at the encoding and the decoding end respectively no longer be necessarily aligned to the frame capturing/rendering times. However, we keep this alignment in the generalized timing analysis, keeping the VBV timings intact in order to build the VCV model on top of the established VBV model.

To analyze the end-to-end timing of a frame, let the frame capture time, encoding start time, encoding completion time, encoded data output time, decoder input time, decoding start time, decoding completion time and rendering time be denoted by $t_p^{cap}$, $t_p^{enc,st}$, $t_p^{enc,end}$, $t_p^{enc,op}$, $t_p^{dec,ip}$, $t_p^{dec,st}$, $t_p^{dec,end}$, and $t_p^{rend}$, respectively. For the system, the capturing and rendering are considered atomic processes without loss of generality, since otherwise they will simply add a constant delay in the overall model. In this analysis, $t_p^{cap}$, $t_p^{enc,op}$, $t_p^{dec,ip}$, and $t_p^{rend}$ are time-aligned to timing interval $T$. Here for $n \in \{enc, dec\}$ signifying the encoder and a decoder, respectively,

$$0 \leq \sum_{t=t_i}^{t_f} \mathcal{C}_t^n - \mathcal{H}^n.(t_f - t_i) \leq \mathcal{C}^{n,max}, \forall t_i \leq t < t_f \tag{5.4}$$

where $\mathcal{H}^n$ are the steady-state rate of availability of the computational resources at a terminal $n$ in the given period, and

$$\sum_{t=t_p^{n,st}}^{t_p^{n,end}} \mathcal{C}_t^n = \Delta\mathcal{C}_p^n = \Delta t_p^n.\mathcal{H}^n. \tag{5.5}$$

$\mathcal{C}_p^{n,max} = B^n.\mathcal{H}^n.T$, where depending on the terminal, $B^n$ is the pre-encoder buffer $B^{enc,ip}$ or the post-decoder buffer $B^{dec,op}$.

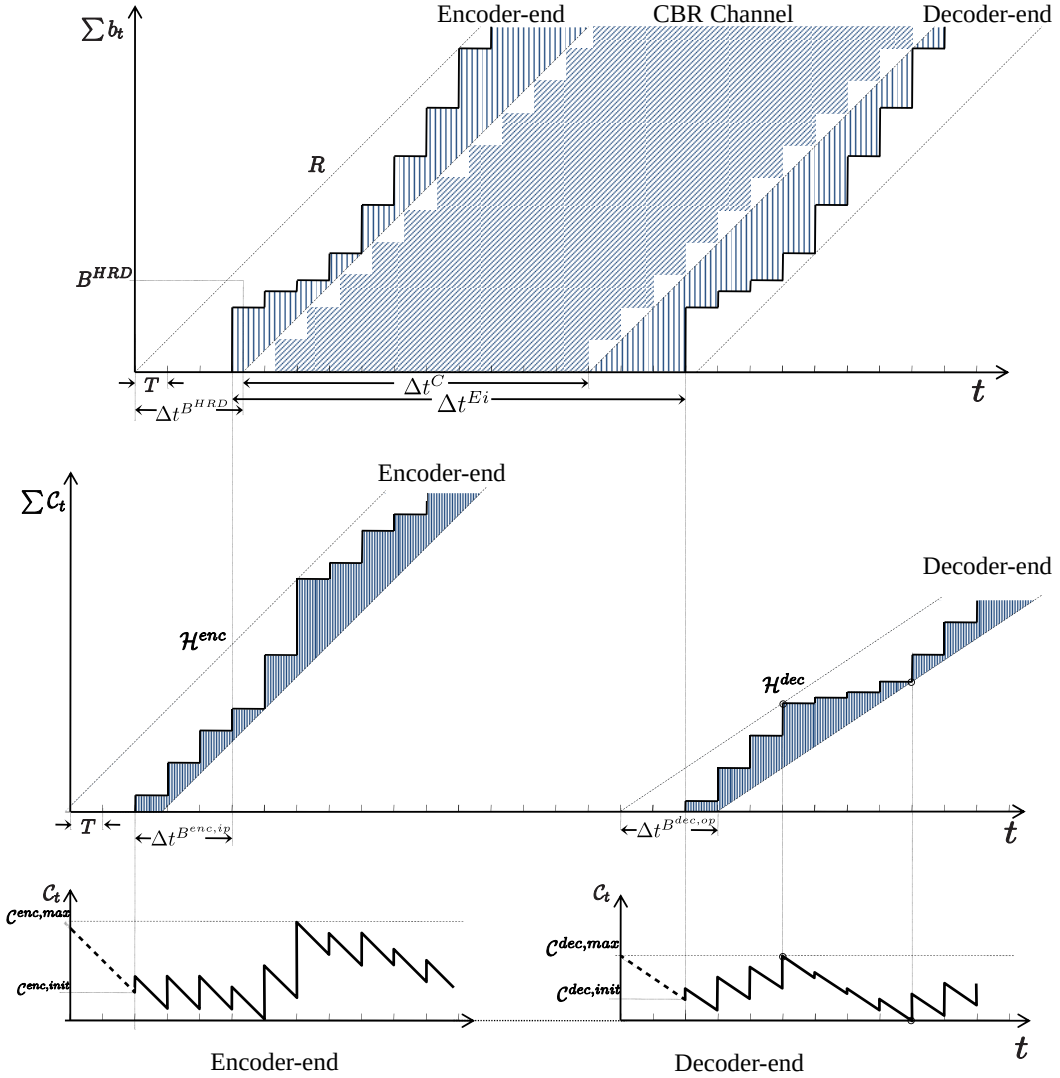



Figure 5.5: Timelines of various processes for VCV

Here $t$ is discretized to a factor of the GCD of $\mathcal{H}^n$ and $T$. The bounds of Equation 5.4 relate to the virtual resource usage buffer (and *not* the actual frame buffers), the lower-bound in Equation 5.4 guarantees no buffer under-runs, while the upper-bound prevents overflows. Figure 5.5 shows the VCV timing in the context of VBV. As can be seen, the computational resource demand schedule is aligned with the capturing schedule $t_p^{cap}$ at the encoding end. At the decoding end, the computational resource demand schedule is aligned with the decoder input schedule $t_p^{dec,ip}$. $\mathcal{H}^{enc}$ and $\mathcal{H}^{dec}$ are intentionally shown different, the comparison signifies little parallels with VBV; the slope and the shape of the computational resource demands at the terminals are as unique as the implementation. The virtual computational resource usage buffer status is also shown at the bottom of Figure 5.5. As expected, unlike for VBV the maxima and minima of the buffers at the terminals may not be aligned with the each other in general.

Since in Figure 5.5 the decoder schedule is aligned to the the decoding input and not the rendering process, only the impact on the encoding-end delay is visible, that increases by $t^{B^{enc,ip}}$. Figure 5.6 explores the time-lines in a bit more detail. Initially, the start of the encoding schedule $t_p^{enc,st}$ leads the encoder output schedule $t_p^{enc,op}$ by $\Delta t^{\mathcal{C}^{enc,init}} = \mathcal{C}^{enc,init}/\mathcal{H}^{enc}$. Likewise the start of the decoding schedule $t_p^{dec,st}$ leads the rendering schedule $t_p^{rend}$ by $\Delta t^{\mathcal{C}^{dec,init}} = \mathcal{C}^{dec,init}/\mathcal{H}^{dec}$.

It is also worthwhile to notice the status of the actual pre- and post-encoder/decoder buffers. A yet-unprocessed frame has to reside in the buffer, and the usage of raw-frame buffers is an important consideration because of the size of uncompressed video. The bottom part of Figure 5.6 shows the status of these buffers. At any point in time, a single frame is present inside the encoder or the decoder, which is the non-shaded portion between the shaded parts. The remaining data is split in the pre- and post- encoder/decoder buffers. The initial encoding start time is given as $t^{enc,st^{init}} = t^{cap^{init}} + \Delta t^{\mathcal{C}^{enc,init}}$. At any point in time, $t_p^{enc,st} = t_{p-1}^{enc,end}$. Similar observations can be made for the decoder.

## 5.3 Decoder Resource Usage Model

A crucial step involved in computational resource management is to model the computational resource usage. Especially, the decoder resource usage modeling is an essential cornerstone for online optimization framework, as will be described in Section 5.4. The existing techniques for modeling the computational resource usage of video codecs require fully invasive measurements of the codec's algorithmic modules. To achieve this, in-depth knowledge of the algorithmic modules is required along with a complete access and understanding of the implementation. In most cases this is not possible, e.g. complete access and understanding of the implementation is commonly not given to a system designer, or the user installs a codec on an unknown hardware. This is a severe disadvantage to the science of computational resource management.

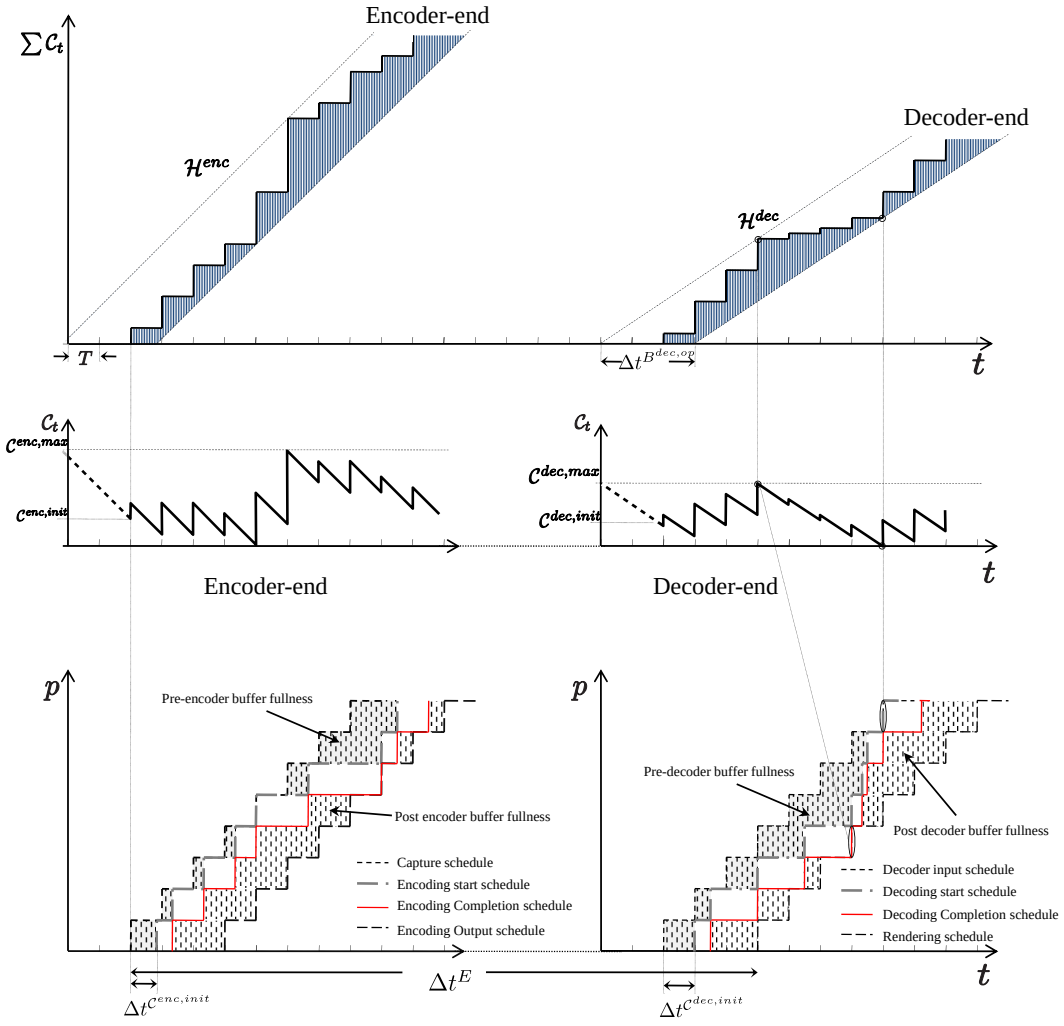



Figure 5.6: Status of the frame buffers used for VCV

### 5.3.1 Design Considerations

Following are some important design considerations for the decoder resource usage model:

#### Decoder Model Prediction Accuracy

The model has to be able to predict the actual computational resource usage of a decoder with sufficient accuracy. Since no model can provide an error-free prediction, one already has to provide means to address the prediction error $\epsilon$, in order to provide prediction error robustness.

1. Providing a safety factor in model usage based on a comprehensive knowledge of the observed average error $\bar{\epsilon}$ as well as the maximum (peak) prediction error $\epsilon^{max}$ values for the model. The target values used in optimizations should cater for these values.

2. A feedback channel to report the long-term drift between the predicted and actual values, the reports can be on a GOP level. This can allow the optimization algorithm to readjust its optimization parameters.

3. Extra post-decoding buffers to cache the frames to be presented. But this buffering is absolutely limited because of:

   (a) Memory constraints, since uncompressed video data requires significant memory space.

   (b) More buffering eases the peak load computational requirements but will also adversely effect the end to end delay and jitter, which is undesirable in most scenarios.

4. Optimized coding to cater for any eventual catastrophic events because of prediction errors, e.g. providing decoder refresh frames in a periodic fashion.

The types of the error metrics of concern are:

1. The maximum (peak) prediction error $\epsilon^{max}$ for a given frame. If $\epsilon^{max} \gg 0$ (i.e. model prediction too optimistic) and prediction error robustness is not sufficiently provided to handle this, it will result in visual quality degradation in the form of a form of a jitter.

   On the other hand, if $\epsilon^{max} \ll 0$ (i.e. model prediction too pessimistic), it will result in visual quality degradation because of sub-optimal coded content.

2. Average prediction error $\bar{\epsilon}$ over a period of time. This parameter is more critical since average prediction error translates into a prediction drift between the sending and the receiving terminal. If $\bar{\epsilon} > 0$ and prediction error robustness is not sufficiently provided to handle this, it will result in failure to maintain the decoding and presentation time-line till such a time when a decoder refresh is provided. This will result in a catastrophic degradation.

A negative prediction error ($\bar{\epsilon} < 0$) will result in a degraded video quality and suboptimal resource usage. However, the effects are not as drastic as for $\bar{\epsilon} > 0$.

**Relevance to applications**

Although there can be several approach variations to solve the computational resource management, the approaches that have good potential to be integrated in current as well as future applications should be considered. The following factors contribute to this:

**Ease of determining the model:** Techniques based on an intrusive analysis of an implementation have little practical relevance. With smart portable devices in today's world it is not possible to predict an implementation at design-time. For example, a user installs a newly available video streaming client software on one of the several possible compatible devices. Even with a single-OS, single-hardware based solutions, it is expensive to do detailed codec analysis. Hence techniques based on fully intrusive measurements of the codec's algorithmic modules will gain little adaptation.

**Low communication and storage overhead:** The model should have a compact notation so that it can be conveniently communicated to the sending end with minimal overhead. There should be little, if any, need of bidirectional communication otherwise for optimizations.

**Low computational complexity:** The usage of model in real-time should have very little computational overhead; there is little use of a technique for computational resource management that increases the computational overhead significantly.

### 5.3.2 Formulation of the Model

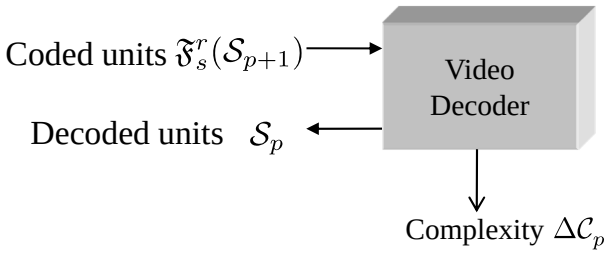



Figure 5.7: A high-level abstraction of the video decoder

Figure 5.7 represents an abstraction of the digital video decoder. Let $\mathfrak{F}^r(\mathcal{S}_p)$ denote the coded or compressed version of the $p^{\text{th}}$ video frame $\mathcal{S}_p$ at a decode time $t$. Here $\mathbf{E} \subset \mathbb{R}^{\mathbf{r}}$ identifies the space spanned by the encoding process of the target video

codec such that $\mathfrak{F}^{r\dagger}\mathfrak{F}^{r}(\mathcal{S}_p) = \mathcal{S}_p$, where $\mathfrak{F}$ is the encoding process identified by $r$. $r$ is in turn identified by the information embedded in $\mathfrak{F}^{r}(\mathcal{S}_p)$ (note that for older codecs before H.264/AVC, the reconstructed version is only approximately equal to the original $\mathcal{S}_p$ for practical implementations). The impact of losses will be considered later, here lossless delivery is assumed.

Let $\mathfrak{F}^{r}(\mathcal{S}_p)$ be split by a reversible process into $S$ data packets and fed to the decoded in correct order, as shown in Figure 5.7. As a result at time $t + \Delta t_p^c$ the decoded frame is output from the decoder, where $\Delta t_p^c$ is the composition offset of the said video frame. At the said time $t$, corresponding decoding processing $\mathfrak{F}^{r\dagger}$ consume computational resources provided by the platform on which this decoder implementation runs.

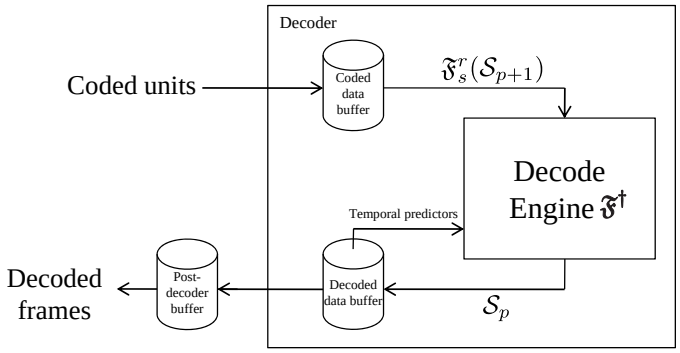



Figure 5.8: Flow of the data in the buffered model

The timing model at the compressed interface are specified by standardized video decoders, as discussed in Section 5.2. The coded picture is removed from the buffer at the decoder input at a time instant $t$, as shown in Figure 5.8, to be decoded into $\mathcal{S}_p$. For a real implementation $\mathcal{S}_p$ will be made available in the decoded buffer at a time $t + \Delta t_p$, where $\Delta t_p$ is the result of consumption of computational resources in the decode block. It is this quantity that is needed to be predicted by a decoder model.

$$\Delta \mathcal{C}_p = \Delta t_p.\mathcal{H} = f(\mathfrak{F}^{r}(\mathcal{S}_p)) = \sum_{s=1}^{S} \{f(\mathfrak{F}_s^{r}(\mathcal{S}_p))\}\,. \tag{5.6}$$

Equation 5.6 essentially relates the computational resource usage $\mathcal{C}_p$ to be strictly dependent on the coded data $\mathfrak{F}^{r}(\mathcal{S}_p)$, since this is the sole input of the decoder, and for standardized decoders, the control decisions are independent on previous decoded output. On the right hand side, this is related to the packetized data, since this is the accessible data at the input of the decoder. Hence the model needed is of the mapping function $f(.)$ in Equation 5.6.

The proposed model is defined by an implementation-independent model $\mathbf{\Psi}$ and an implementation transform $\mathbf{\Theta}$ as

$$\Delta \mathcal{C}_p = \langle \mathbf{\Theta}, \mathbf{\Psi}^{r} \rangle + \epsilon,\ \epsilon = \langle \mathbf{\Theta}', \mathbf{\Phi}^{u} \rangle. \tag{5.7}$$

Hence the implementation-independent model $\mathbf{\Psi}$ is $r$ dimensioned, in the same subspace $\mathbf{E}$, and is uniquely determined by $\mathfrak{F}^r$. The error term $\epsilon$ is in turn defined by another implementation transform $\mathbf{\Theta}'$. $\mathbf{\Phi}^u$ however is correlated to $\mathfrak{F}^r$, but $\mathbf{U} \nsubseteq \mathbf{E}$, where $\mathbf{U}$ is the space spanned by $\mathbf{\Phi}^u$. Note that this model essentially decouples the decoder computational resource usage from the data to be decoded.

In order to understand how the model relates to the architecture of the *decode engine* module of Figure 5.8, it is expanded in Figure 5.9. The codec control module extracts essential information from $\mathfrak{F}^r(\mathcal{S}_p)$ to identify $r$, so that $\mathcal{S}_p$ could be reconstructed. Along its path $\mathfrak{F}^r(\mathcal{S}_p)$ is processed by several groups of decoding algorithms, e.g. $a_l$, $l = 1, 2, ..., A$ is one such group. As an example, in an H.264/AVC decoder, full-pel, half-pel and quarter-pel interpolation filters are a set of algorithms that make the group, which is referred to as motion compensation. Which algorithm of a given group will act upon data is decided by the used coding options, and the thin lines leading from the codec control to individual algorithmic modules provide this control. The thick lines leading to $\Delta\mathcal{C}_p$ represent the computational resource usage of each of these algorithms.

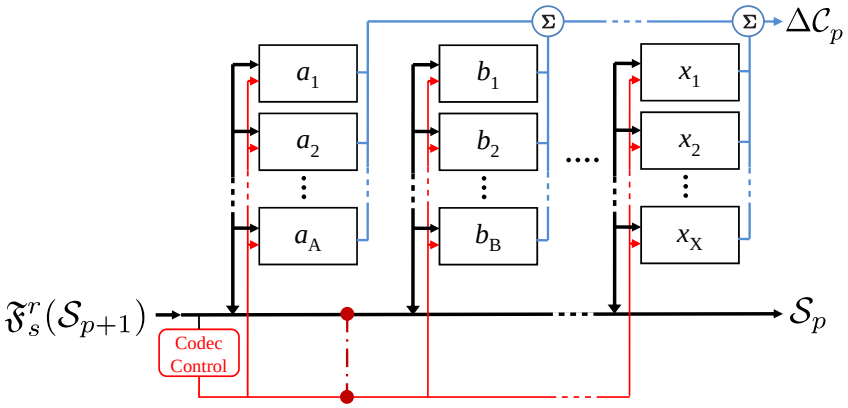



Figure 5.9: An architectural abstraction of the decoder

The computational resource usage arising from the application of algorithms on the coded data based upon the codec control decisions result in the major part of computational overhead, and this is represented by the first term on the right hand side of Equation 5.7 ($\langle \mathbf{\Theta}, \mathbf{\Psi}^r \rangle$). This term is the major contributor of the computational resource usage. All the codec control decisions are observable at the encoder by a simple implementation.

However, in general there are a few algorithmic manipulations done by a standard compliant decoder that are not easily measurable at the encoder. An example of such algorithms for an H.264/AVC decoder is the management of the motion vector pertaining to a block of pixels pointing outside the bounded area of the reference image. Such computational overhead is represented by the second term on the right hand side of Equation 5.7. In Figure 5.8, this is shown by the dashed connecter between the data-flow lines and control lines.

### Low Complexity Approximation

In order to approximate the second term on the right hand side of Equation 5.7 with reasonable computational overhead, we approximate the resource usage to vary linearly within a range of an activity factor $\sigma$ expressed as

$$\Delta\mathcal{C}_p \approx \Delta\hat{\mathcal{C}}_p = \langle \mathbf{\Theta}_l, \mathbf{\Psi}^r \rangle, \mathcal{L}_{l-1} < \sigma \leq \mathcal{L}_l. \tag{5.8}$$

$$\sigma = \sqrt{\frac{1}{M}\sum_{b=1}^{M}\left((V_b^x)^2 + (V_b^y)^2\right)} \tag{5.9}$$

Equation 5.9 expresses the selected activity factor $\sigma$ as the mean squared horizontal and vertical motion vectors $V^x$ and $V^y$ for all the blocks $M$ in a frame. The number of pieces $l = 1, 2, ..., L$ for this piecewise linear model and the thresholds $\mathcal{L}_l$ are tuned empirically. Here $\mathcal{L}_0 = 0$ and $\mathcal{L}_L \to \infty$. The selected activity factor is particularly low cost in terms of computational resources needed to estimate it, which is of importance since this factor needs calculation in real-time.

### Determining Implementation Transform

The implementation-independent model $\mathbf{\Psi}$ is fixed for a given codec. From Equation 5.8, the implementation transform $\mathbf{\Theta}_l$ for a given activity factor $l$ is derived as

$$\mathbf{\Theta}_l = \Psi^{l\dagger}\vec{\hat{\mathcal{C}}}, \tag{5.10}$$

which signifies that all the rows of matrix $\Psi$ contain vectors with same activity factor $l$ and has rank $r$. The vector $\vec{\hat{\mathcal{C}}}$ contains the corresponding known $\Delta\hat{\mathcal{C}}_p$ for each of the rows. The implementation notes for determining $\vec{\hat{\mathcal{C}}}$ are provided in the following sections.

### Implementation-Independent Model for H.264/AVC

When applied to H.264/AVC base-line decoding of video, $r$ corresponding to the number of the coding modules to be modeled is determined as follows. There are five groups of algorithmic modules, namely intra-prediction, inter-prediction, reconstruction, deblocking and entropy encoding.

1. Intra prediction consists of 24 different coding algorithms:
   - 9 different I_4x4 prediction modes, as specified in Section 8.3.1.1 of [4],
   - 8 different I_8x8 prediction modes, as specified in Section 8.3.2.1 of [4],
   - 4 different I_16x16 prediction modes, as specified in Section 8.3.3 of [4],

- and 4 intra-chroma prediction modes, as specified in Section 8.3.4 of [4].

2. Inter-prediction consists of 72 different algorithms: the 8 unique prediction modes: P_SKIP, P_16x16, P_16x8,..., P_4x4, as specified in Section 7.4.5 of [4] (including the 4 sub macroblock modes as specified in Section 7.4.5.2 of [4]). Each of these 8 unique prediction modes can have 9 unique interpolation configurations: full-pel, half-pel and quarter-pel interpolation in both horizontal and vertical directions, as specified in Section 8.4.2.2.1 of [4], resulting in 9 interpolation permutations.

3. Reconstruction consists of four different modes: reconstruction for normal blocks, reconstruction for blocks with a single DC coefficient, reconstruction of empty blocks and finally, an implicit copy mode, which is effectively a P_SKIP mode with zero motion vectors.

4. Deblocking filter comes with 10 different filter algorithms for varying boundary strengths: 6 different filtering options for luma and 4 for chroma, as specified in Section 8.7 of [4].

5. The entropy coding for baseline H.264/AVC is modeled by 7 algorithmic modules, each of these modules depend on: number of coefficients, number of DC coefficients, number of trailing ones, number of trailing ones for DC blocks, number of coefficient token for VLC, number of coefficient token for FLC, and the number of bits written, respectively, as specified in Section 9.2 of [4].

In total this makes $r = 118$ for the modeling of the baseline decoding. Hence the size of data required to express the model is just $118 \cdot L \cdot R$ bytes, where $R$ is the number of bits used to represent a single entry of the model.

It will be later shown by quantitative evaluations in Chapter 6 that for most platforms $L = 2$ is able to provide with a fairly accurate model sufficient for most applications. With a typical application using 32 bit arithmetic, the total amount of raw data needed to express the model is just 944 Bytes.

### 5.3.3 Memory Usage Modeling

As discussed in Chapter 2, the dominant contributor of memory usage is LTM-MCP. Different implementations however store raw video frame data in different fashion, resulting in differing memory usage increase per increase of reference frame. This depends on the precision and bit-depth per sample, and the basic memory unit used to store it. Typically 8 bit representation of pixels is used for normal-fidelity video and one byte of memory is used to store it. The main difference comes from the fact that several implementations save an up-scaled version of the raw video buffer to avoid repeated interpolation process. Hence for H.264/AVC where a quarter-pel interpolation of luma is allowed, the luma buffer is upsampled 4 times and stored. This process however is more common at the encoding end, where such up-sampling can help the motion estimation process. In general, if $LTM$ bytes are required to store a reference frame, then the memory requirement of $\mathcal{J}$ reference frames is approximated as $\mathcal{J} \cdot LTM$.

### 5.3.4 Implementation Notes on Decoder Modeling

The implementation steps required to be performed to determine the decoder model are as follows:

1. A coded video bitstream of natural scene content and known coding options $\Psi^l$ is fed to the decoder.
2. For each sample of decoded data output $\mathcal{S}_p$, the corresponding $\Delta\mathcal{C}_p = \Delta t_p.\mathcal{H}$ is measured.
   (a) A platform-dependent accurate cycle counting mechanism must be used for this instrumentation.
   (b) The error in the instrumentation step is platform dependent. A post-processing may be performed to reduce this error. For example:
      i. A Monte Carlo method is employed by performing multiple iterations of steps 1 and 2. The number of iterations to be performed is commented upon in Section 6.2. Typically, 64 iterations were found to provide sufficiently accurate results for most of the platforms.
      ii. Data is reconstructed by using the first principal component (using PCA).
3. As a result of instrumentation we will have $l$ linear regressions to be solved, solution is given by Equation 5.10.
4. In order to assess the memory usage $LTM$ of a single added reference frame, a comparison of the codec's run-time memory usage is done between $\mathcal{J} = 1$ and $\mathcal{J} = 2$. The difference of the decoder's run-time memory provides with the estimate of $LTM$. It should be noted however that for this comparisons, just an increased number of stored but unused short-term reference frame is necessary and sufficient; it is not required to perform motion compensation from multiple frames. Hence, by using two anchor streams with different number of stored reference frames and actually using a single, most recent reference frame (as described for IEC-1 in Chapter 4), the memory instrumentation analysis will be orthogonal to the computational resource usage instrumentation above. Hence this measurement can be taken along with the computational resource usage instrumentation described before, and a separate instrumentation is not required.

   Based on the results and the maximum allowed memory budget at the terminal, a client can select an appropriate maximum number of reference frames, $\mathcal{J}^{max}$.

Hence the required process to be carried out to determine the decoder model is of a black-box modeling that can be fully automated. There are two typical scenarios to achieve this: if the implementation is known at the device-manufacturing stage, only a script has to be run that feeds the implementation with a set of video content with known coding options, several iterations are made and corresponding slice or frame level accurate times are measured. This data is then processed offline to determine the model.

On the other hand, if the implementation is not known at the design step, e.g. as is typical for programmable devices as smart-phones or PCs, where user can install a codec at will to result in a unique implementation, the model can be determined at install time by this fully automated process.

A quantitative analysis of the performance and accuracy of the proposed decoder model will be presented in Chapter 6.

## 5.4 Online Resource Optimization

For this category of optimization, at any instant of time, the optimization algorithm is capable of impacting the current and subsequent decisions for encoding mode selection and motion estimation. Hence this adaptation can be performed on any sample level; down to a single frame, or a part of the frame. Since this optimization targets to impact the encoding decisions directly, it is performed at the sending (encoding) end.

One important problem arises because of this optimization configuration: in a vast majority of applications, the decoders are resource constrained. The question arises: how can the sending end achieve optimizations of computational resource usage at the other end of the network? In existing systems, even the lowest delay feedback channels cannot be used to solve this problem, all it can achieve is to provide some periodic feedback of past statistics. Hence it is pivotal that a computational resource usage model of the decoder is made available at the sending end to approximate the computational resource usage incurred at the receiving end.

This has an important implication: this receiver optimization relies on a receiver-aware application, e.g. there should be a hand-shake mechanism as a part of session-joining protocol, or some alternate means where the sender has the possibility to learn the model of the receiver. The case when this is not possible is discussed in Section 5.5. The sending end will then use this model in conjunction with the resource management mechanism to optimize the end-to-end system.

On the other hand, the observation of computational resource usage of the encoder for online optimizations at the sender end is achievable by real time instrumentation.

Hence in this class of optimization techniques, the computational resource usage model developed in Section 5.3 is made available at the sending end, along with the current steady-state resources $\mathcal{H}$ and memory usage constraint $\mathcal{J}^{max}$ for optimization. Hence this optimization will integrate the computational resource optimization problem of all the resource-constrained terminals joining the service.

To summarize the above, this optimization is characterized by:

1. Online encoding or re-encoding of video.
2. Resource optimization performed at the encoding end.
3. If resource optimization of decoding end is to be performed, the sending end has to be aware of the real time resource usage of the decoder by using the

computational resource usage model of the decoder. Hence the sending end has to be "receiver aware."

### 5.4.1 RDC Optimizations

Although any optimization for computational resources might be categorized as RDC optimization since both the RD and computational resource optimizations are being performed, here RDC optimization only refers to the *joint* RDC optimizations.

We have seen the parallels between the VBV and VCV in Section 5.2.1: the available computational resources per unit time are limited to some constant $\mathcal{H}^n$ for a terminal $n$. The resultant analysis draws parallel from the constrained bitrate coding.

The rate-distortion minimization problem for H.264/AVC has for each block a coding option set $\mathcal{O}$ that consists of all the combinations of quantization parameters, coding modes and motion vector configurations for inter-prediction. As a result, the set of complete coding configurations of all the blocks is infeasibly large for practical computational purposes. Hence, the problem is simplified to reduce the complexity. The typical approach as described in [21] is to split the overall optimization problem into individual independently solvable problems:

1. Since the distortion for a given mode and content is an asymptotically increasing function with an increasing quantization parameter, the quantization parameter is removed from the optimization and selected by other mechanisms e.g. based on [136]. Hence the dimension of the problem is reduced significantly.

2. The optimization of motion vectors is treated separately from the other coding modes. These two optimizations are performed separately as independent lagrangian minimization problems.

The second step of optimizations based on Lagrangian minimizations is described in detail in [12], and its theoretical basis is covered in detail in [137]. While it may seem very intuitive to apply a similar approach to the optimization of computational resource usage as well, as discussed in Section 3.1.2, we must compare the theoretical basis to see if this is indeed a feasible approach.

#### RD Versus CD

For a constrained optimization problem as the one shown in Equation 4.4, the cost function $J_{b,m}$ is given as

$$J_{b,m} = (d_{b,m} + \lambda_{\mathcal{O}} r_{b,m}) \tag{5.11}$$

The costs are obtained my measuring or reliably estimating distortion for a given block coding mode. The minimal cost is determined by differentiating against the

reconstruction distortion and setting to zero:

$$\frac{\mathrm{d}\, J_{b,m}}{\mathrm{d}\, r_{b,m}} = \frac{\mathrm{d}\, d_{b,m}}{\mathrm{d}\, r_{b,m}} + \lambda_{\mathcal{O}} = 0 \tag{5.12}$$

which gives

$$\lambda_{\mathcal{O}} = -\frac{\mathrm{d}\, d_{b,m}}{\mathrm{d}\, r_{b,m}}. \tag{5.13}$$

Although $\mathrm{d}\, d_{b,m}/\mathrm{d}\, r_{b,m}$ can be measured or estimated (although at a cost of computational resource usage) for each optimization, the crucial factor in this optimization is the lagrangian multiplier $\lambda_{\mathcal{O}}$. This parameter has to be based on stationary or at least semi-stationary statistics and has to hold for a range of the optimizations for Equation 5.11.

Understandably, the lagrangian multiplier is equal to the negative slope of the R-D curve and hence determine how much distortion reduction can be optimally traded for a corresponding rate for a given optimization point (Equation 5.11) of the R-D curve. Interestingly, this slope has a very well defined and direct link to the quantization parameter in hybrid video codec; at a given operating point at the R-D curve, for a given mode, it is the indeed the quantizer that trades rate and distortion. As a result an empirical curve-fitting becomes feasible to achieve a stationary and stable solution, and in [137] it was determined to be $\lambda_{\mathcal{O}'} = 0.85 Q^{H.263^2}$ for an H.263 and MPEG-4 ASP and in [12] it was determined to be $\lambda_{\mathcal{O}'} = 0.85 \cdot 2^{(Q^{H.264}-12)/3}$ for H.264/AVC, where $\mathcal{O}'$ is the remaining option set when the motion vector selection has been split already from the optimization problem.

By the above understanding, it is worthwhile to see if the same approach of lagrangian constrained optimization can be used for CD tradeoff. A direct extension to the cost would be

$$J_{b,m} = \left(d_{b,m} + \lambda^r_{\mathcal{O}} r_{b,m} + \lambda^c_{\mathcal{O}} \mathcal{C}_{b,m}\right). \tag{5.14}$$

One obvious problem with direct extension to Equation 5.14 is that a reliable estimate of distortion is required. In real system making such reliable estimate for this optimization or the optimization of Equation 5.11 has no rate overhead, but it will potentially have a considerable computational resource overhead. Hence an estimation of cost in Equation 5.14 will in turn increase the cost itself.

Another question is hinged on one crucial aspect: what is the suitable stationary choice of $\lambda^c_{\mathcal{O}}$ frequently introduced in Chapter 3. Similar to Equation 5.13, for a specific rate it is

$$\lambda^c_{\mathcal{O}} = -\frac{\partial d_{b,m}}{\partial \mathcal{C}_{b,m}}. \tag{5.15}$$

Unfortunately however, quite unlike the quantity expressed in Equation 5.13 which was directly related to the quantization parameter, the quantity expressed in Equation 5.15 is not dependent on any simple underlying phenomenon to establish a valid solution for a range of optimization of Equation 5.14. It rather depends on the sample being coded, the type of redundancy it has, and that a particular coding tool is designed to remove the particular type of redundancy or not. For example, for SD scene content, 8x8 or 4x4 inter-prediction modes might be more suitable than 16x16, while for HD scene content it may be the other way around. Same goes for different parts of a single frame where the texture is different. This strong content dependence and the potential for large overhead makes a similar approach of constrained minimization highly unsuitable for computational resource optimization problem.

### 5.4.2 CD Mode Ranking

As discussed in the preceding section, it is difficult to achieve a joint RDC optimization, mainly because of the lack of available statistically (semi-)stationary CD relationship. Stable and reliable statistical relations are generally based on the actual, underlying physical phenomenon in the system being optimized, as shown in the preceding section for RD optimizations. However, it is still feasible to have a stationary CD *ranking* of individual coding modes.

Mode ranking is the approach for optimal grading or ranking of the coding modes based on the stationary CD performance statistics. Since this is based on stationary statistics, the real-time performance overhead is also minimal. Two types of considerations have to be made: distortions in system with reliable communication link versus lossy communication medium.

With a reliable mode ranking available, the approach that can be used to address the resource management problem consists of a two step process:

1. Based on the available resources, dynamically and adaptively select a suitable subset $\hat{\mathcal{O}}$ of the coding options $\mathcal{O}$ in Equation 5.11, based on the developed mode ranking.

2. Perform the constrained minimization as usual with only rate-distortion tradeoff.

After developing the basis of mode ranking in this section, the dynamic adaptive algorithm will be discussed in Section 5.4.3

#### Ranking of Inter-prediction Techniques Based on Interpolation Configuration

A mode ranking scheme based on stationary statistics (that do not vary significantly with scene-content) is introduced by ranking the fractional-pel interpolated modes in conjunction with Long-term memory motion compensated prediction (LTM-MCP) tool of H.264/AVC.

Based on the analysis in Section 2.2.2 and our work in [27], we have been able to rank the 6 unique fractional interpolation configurations in descending order of their CD tradeoff $\partial d_{b,m}/\partial \mathcal{C}_{b,m}$: FP-FP, FP-HP, FP-QP, HP-HP, QP-QP, and HP-QP, respectively. With LTM-MCP, there is possibility of finding a block match in one of the previous frames where the interpolation configuration introduces a lower distortion. This occurs in practice whenever there is smooth camera or object motion, which is common in natural scene content. This fact results in the dominant portion of prediction gain for LTM-MCP. The converse also holds: the probability of finding a block match in a temporally farther frame is higher only at an interpolation configuration which introduces lesser distortion as compared to the block match in one of the newer reference frames.

Hence the inter-prediction modes should be ranked as above with each added farther temporal reference for inter-prediction. This mode-ranking technique is applicable to most scene content and hence the ranking is also stationary.

#### Ranking of Modes for Robust Video Coding

The preceding section provides ranking only for a subset of the available coding modes. For the remaining modes, we use a second criteria: ranking the modes based on their average performance statistics for a variety of scene content coded in a variety of transmission conditions. Obviously this greedy technique may not be optimal for local optimizations, but it will provide us performance enhancement when considered over a longer-term.

In regards to robust video communication, typically, the more neighboring samples are used to generate a predictor in a given mode, the more that coding mode is prone to spatio-temporal error propagation. This process has been elaborated in earlier presented results, e.g. of Section 4.1.3 and Section 4.1.3. Interestingly, the number of samples used to generate a predictor has a direct impact on the computational cost of a coding mode, hence it is no wonder that the ranking provided in the preceding section is also ascending with ascending number of pixels used for inter-prediction. Hence if such a ranking is used for optimizations, it will also foster robust video coding for lossy transmission systems.

For the remaining coding modes, statistics rank intra-prediction modes have a higher $\partial d_{b,m}/\partial \mathcal{C}_{b,m}$ then inter prediction, since although the latter results in reduced distortion but has a much higher computation resource usage cost of motion estimation and compensation. The complete ranking used for H.264/AVC will be discussed in the following sections.

### 5.4.3 System-wide Computational Resource Management

To achieve system-wide optimization, different components devised in the preceding sections are put together. The proposed framework for an optimized video delivery setup is described by the following steps:

1. As introduced in proceeding sections, a fixed, implementation dependent compact model is determined for a given receiving terminal. As this is fixed for a given implementation, it needs only to be determined once per implementation. Here implementation refers to a unique hardware-software combination.

2. As highlighted before, for most high quality, real-time adaptations, even a low-delay feedback channel is not sufficient for the purpose of feedback of $\mathcal{C}^n$ of a terminal: a typical NTSC video system might operate close to 30 fps or a frame coding period of around 33 mSec. Even if the computational resource optimization is done once every frame, a feedback channel with maximum delay of 33 mSec is difficult to ascertain for most applications (a good solution should do multiple optimizations per frame). Also the optimization should not mandate a bidirectional communication link, except for a bidirectional handshake information exchange for live case.

   Hence to estimate the $\mathcal{C}^n$ at the receiver, the sender has to utilize the resource usage model of the receiver. The encoding end is made aware of this model of the terminals connected to it, as shown in a high-level abstraction in Figure 5.10. Here an example of a given receiver joining an optimized video communication session is shown. Upon the request to join this optimized service, the sending end optionally queries the receiver that wishes to join about its video decoder model. Two pieces of information are needed to be communicated to the sending end to enable it to do optimizations:

   - As discussed in Section 5.3.2, the computational resource usage model of the receiver that consists of $\mathbf{\Theta}_l, L$, and $\mathcal{L}_l, l \in \{2, ..., L-1\}$. The memory usage model needs just the reporting of $\mathcal{J}^{n,max}$.
   - The steady state availability of resources for this terminal $\mathcal{H}^n$.

   Obviously this is just one abstraction. A service can be envisioned that allows updating of the $\mathcal{H}^n$ based on dynamic resource situation, e.g. power level available at the terminal.

3. The sender then uses this model for optimized video delivery for the rest of the session. During the encoding process the sender monitors the coding options being utilized during the transmission of video data. As discussed before, as an example for H.264/AVC, a 118 component baseline model is required, this means monitoring the corresponding 118 coding modes. This gives the implementation-independent $\mathbf{\Psi}^r$ for a given frame $p$. If $\mathbf{\Psi}^r$ belongs to activity factor $l$, then the approximated computational resource usage incurred is given in Equation 5.9. The number of reference frames that the encoder should use for inter-prediction is given by $\mathcal{J}^{max} = \min_n(\mathcal{J}^{n,max})$.

### Sub-frame Prediction

Since current state-of-the-art video codecs like H.264/AVC have several coding modules that still work on a traditional frame-based level (e.g. the deblocking filter, applied after all slices of the frame are decoded and implementation specific features like the distortion estimates at the encoding end and frame up-sampling etc.) some

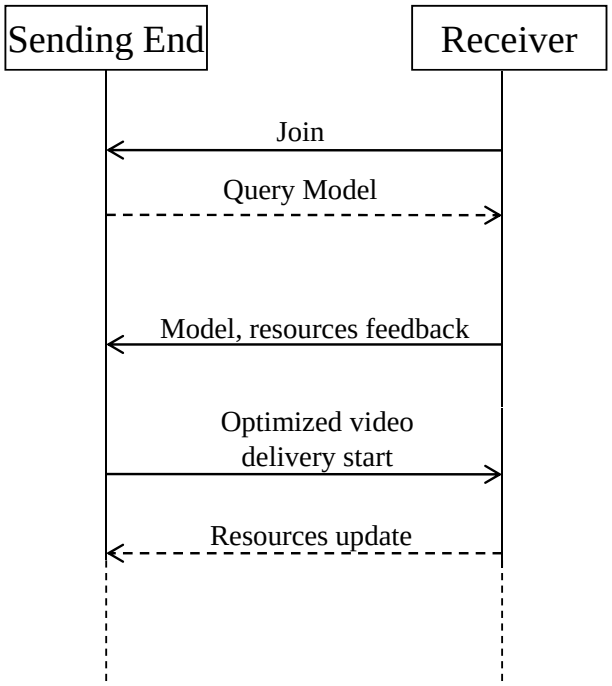



Figure 5.10: An abstraction of session setup for optimized delivery

additional processing is required for sub-frame based computational resource optimization. For this purpose, linear prediction of the model components is done on sub-frame level to predict the resource usage of the overall frame, i.e. $\Delta\mathcal{C}_p = 1/\rho \cdot \Delta\mathcal{C}_p^{\rho}$, for a fraction $\rho$ of the $p^{\text{th}}$ frame processed and the corresponding computational resource usage $\Delta\mathcal{C}_p^{\rho}$.

Since the sender can monitor its own computational resource usage, the use of the video encoding model is not required at the sender.

Now since all the required components of the model are available at the encoding end, computational resource optimization can be performed.

### Complexity Quantization

As discussed in Section 5.4.1, the best computational resource optimization approach that is feasible for practical systems is to select a subset $\hat{\mathcal{O}}$ of the coding options to trade complexity for distortion and then use the conventional RD optimizations. Since the typical paradigm of lossy coding trades rate for distortion by exploiting a quantization parameter, we introduce a similar the notion of computational complexity quantizer $Q^{\mathcal{C}}$. This subset is denoted as $\mathcal{O}^{Q\mathcal{C}}$.

The purpose of this mechanism is to introduce coding options to a set of allowed coding options $\hat{\mathcal{O}}$, in this example by starting with intra-only coding, corresponding

to the maximum, value of this quantizer $Q^{\mathcal{C},max}$, and then adding incrementally complex inter-coding options. The following provides a solution for H.264/AVC but this can be extended to other hybrid video codecs.

**An Example Complexity Quantizer for H.264/AVC:** The options are added based on the mode ranking approach described in Section 5.4.2. For $Q^{\mathcal{C}} \geq Q^{\mathcal{C},max} - 9$, full-pel P_SKIP and P_16x16 coding is allowed for a fraction of macroblocks given by: $1 - 1.3e^{-0.36(Q^{\mathcal{C},max} - Q^{\mathcal{C}} + 1)}$, the remaining macroblocks are Intra-coded. From each further reduction in $Q^{\mathcal{C}}$, following steps are taken:

1. Reference frame index $j$ is set to 1, $j = 1$ is the most recent reference.
2. With each decrement of $Q^{\mathcal{C}}$, interpolation configurations FP-FP, FP-HP, FP-QP, HP-HP, QP-QP, and HP-QP respectively are added to $\hat{\mathcal{O}}$ for $P_j$_SKIP and $P_j$_16x16 inter-coding modes.
3. With each further decrement of $Q^{\mathcal{C}}$, inter coding options $P_j$_8x16, $P_j$_16x8, $P_j$_8x8, $P_j$_8x4, $P_j$_4x8, and $P_j$_4x4 respectively are added to $\hat{\mathcal{O}}$.
4. Increment $j$, repeat from step 2.

$Q^{\mathcal{C},max} - 20$ represents the highest possible computational resource usage for a decoder in this configuration, since increasing number of reference frames only increase the memory requirements but not the computational resource overhead. However, for the encoder, further increase in computational resource usage is possible by incrementing $j$ as above, since each added reference frame increases the motion estimation effort required. $Q^{\mathcal{C},max}$ depends on $\mathcal{J}^{max}$, given as $Q^{\mathcal{C},max} = 12\mathcal{J}^{max} + 9$. $Q^{\mathcal{C}}$ is initialized at $Q^{\mathcal{C},max}/2$.

This example is be no means the only possible solution; the dynamic range of this quantizer can still be extended beyond by adding simpler options at $Q^{\mathcal{C},max}$ (like limiting the intra-prediction modes to a subset of all intra-coding modes provided by H.264/AVC) or at the other end by adding more complex options at $Q^{\mathcal{C},min}$ than the H.264/AVC baseline encoding (e.g. bidirectional predicted frames, CABAC, etc.).

### $Q^{\mathcal{C}}$ Controller

A key problem in this regard is to achieve stable control of the system. If $Q^{\mathcal{C}}$ is the control parameter of interest and the computational resource usage is the main output of interest, then what is the nature of the system it controls?

The main input of video codec is the digital video itself, another important input is the bitrate quantization parameter. However, the main concern is the former, since statistics of real time video input are an unknown. On the other hand, in order to maintain a reasonable computational resource usage bounds, existing video coding techniques do not make real-time statistical analysis of the input video content.

As discussed in Section 3.1.5, some authors have tried to make a real-time estimate of the computational resources. However, before going into this analysis, let us briefly leap forward to take a look into how the resource demand varies in a real video

coding system. Figure 5.11 is shown to get an overview of the system behavior of a video codec performing on a ultra-low voltage CPU, coding the sequence "Foreman" at 64 kbps. The complete simulation description of such realistic scenario will be treated later in Chapter 6; for now the only aspect of interest is how the system behaves for a slight shift in the initial conditions.

Figure 5.11 plots the real-time computational resource usage as it varies with each input frame of video. The "original" curve is the performance of the system with $Q^{\mathcal{C}} = Q^{\mathcal{C},max} - 16$. In the curve "altered," $Q^{\mathcal{C}}$ is increased by a single step; $Q^{\mathcal{C}} = Q^{\mathcal{C},max} - 15$ for a single frame index $p = 28$, and is reverted back to $Q^{\mathcal{C}} = Q^{\mathcal{C},max} - 16$ from the next frame. Hence the input differs from the original for only one frame. The comparison of the two curves immediately expose a highly non-linear, chaotic system [138]; the more the time passes by, the more the output differs from the original output.

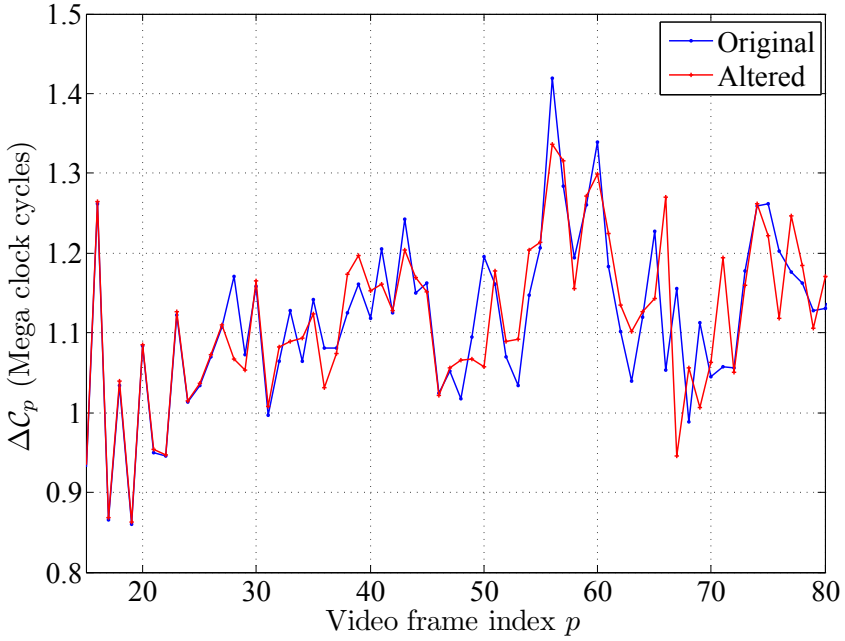



Figure 5.11: A computational resource demand snapshot, showing a chaotic system

As discussed in [138], nonlinear systems have more complex behavior than linear systems, and their analysis is much more difficult. This is so because nonlinear equations cannot in general be solved analytically. Secondly, powerful mathematical transforms (like Laplace and Fourier) do not apply to nonlinear systems. "Hence there are no systematic tools for predicting the behavior of nonlinear systems." [138] Rather, an inventory of analysis and design tools e.g. adaptive control and feedback linearization etc. are used.

At the same time, the lack of a suitable model for real-time video content make this non-linear system time varying as well. Recently, intelligent feedback based control systems have been seen to perform very well for systems that are difficult to model [139]. It is proposed to use the previous output of the system for modeling

the system and this approach will be followed here.

**Design considerations for the controller:** The target of the controller is to achieve $\Delta\mathcal{C}_p^n$ of a terminal as close as possible to $\mathcal{H}^n.T$. However since a single video content is coded and then decoded by various terminals (set aside occasional channel errors), a single control variable $Q^{\mathcal{C}}$ is controlling all the processes. Yet the output from each terminal, $\Delta\mathcal{C}_p^n$ is unique. Hence it is not possible to achieve a globally optimal configuration for all the connected terminals in general. To determine the targets of this optimization, a few considerations have to be made.

- Overshoot of $\Delta\mathcal{C}_p^n$ beyond $\mathcal{H}^n.T$ at a terminal will result in jitter of a video frame encoding or decoding, and its accumulation results in an added delay. Each terminal has an allowed maximum delay quota owing to the buffering $B^{enc,ip}$ for the encoder and $B^{dec,op}$ at the decoder, as introduced in Section 5.2.1. If the upper bound on $\Delta\mathcal{C}_p^{n,max} = B^n.\mathcal{H}^n.T$ is overshot, or equivalently, the upper bound defined in Equation 5.4 is overshot, this will result in failure of a terminal to maintain its presentation timeline.
- On the other hand too little $\Delta\mathcal{C}_p^n$ compared to $\mathcal{H}^nT$ results in wastage of resources at the terminal.

It can be seen from the above considerations that while an undershoot will result in quality degradation, an overshoot of the VCV buffer bounds will result in breakdown of service for a terminal, and hence is a hard limit.

**Controller Design:** The controller used here is shown in Figure 5.12. To achieve the control with higher degree of precision, subframe level control is performed at intervals of $\Delta t^{sf} < T$; $\rho = \Delta t^{sf}/T$. To formulate the control, a change of notation is required however. In Figure 5.5 and the related formulation so far have used a simplified notion of $\mathcal{C}_t^n$, where the terminal $n$ incurs the resource demand of an entire frame $\Delta\mathcal{C}_p^n$ in an atomic fashion; as soon as the frame is made available for coding or decoding. This notation is sufficient to understand the frame-level processes, since the capturing and rendering deadlines are typically frame based. In reality, the resource demand is incurred in a continuous fashion, as sub-frame level video data is coded or decoded. We denote this more realistic resource demand variation as $\tilde{\mathcal{C}}_t^n$. The prediction of sub-frame computational resource demand $\Delta\mathcal{C}_p^\rho$ has already been described in Section 5.4.3.

At each sub-frame level optimization, the change in the control parameter $\Delta Q^{\mathcal{C}_n}$ at any time instant $t^{cur}$ is determined as

$$\Delta Q^{\mathcal{C}_n} = \mathcal{K}_1 \left( \frac{\sum_{t^{cur}} \tilde{\mathcal{C}}_t^n - \mathcal{H}^n t^{cur}}{\mathcal{C}^{n,max}} \right) + \mathcal{K}_2 \left( \frac{\sum_{\Delta t^{sf}} \tilde{\mathcal{C}}_t^n}{\mathcal{H}^n \Delta t^{sf}} - 1 \right) \tag{5.16}$$

Were $\mathcal{K}_1$ and $\mathcal{K}_2$ are constants selected empirically, and $\Delta t^{sf}$ is the time elapsed since the last time $Q^{\mathcal{C}}$ was updated by $\Delta Q^{\mathcal{C}}$. As the term in Equation 5.16, $\sum_{t_{cur}} \mathcal{C}_t^n - \mathcal{H}^n t_{cur}$ tends to $\mathcal{C}^{n,max}$, the first factor $\left(\sum_{t^{cur}} \tilde{\mathcal{C}}_t^n - \mathcal{H}^n t^{cur}\right)/\mathcal{C}^{n,max}$ tends to 1.

Hence $\mathcal{K}_1$ is selected so that $Q^{\mathcal{C}}$ is increased appropriately to avoid the violation of the limits of Equation 5.4. The normalization makes implementation independent selection of $\mathcal{K}_1$ possible.

The factor $\mathcal{K}_2$ ensures that the resource demand $\sum_{\Delta t^{sf}} \tilde{\mathcal{C}}_t^n$ matched the resource supply $\Delta t^{sf} \mathcal{H}^n$ as much as possible. In case of an exact match, the second term is reduced to zero. The overshoot and underrun of resource usage makes this term greater or less than zero, respectively. Here as well the normalization is performed to ensures implementation independent selection $\mathcal{K}_2$ possible. The actual selected values of $\mathcal{K}_1$ and $\mathcal{K}_2$ will be provided in Chapter 6.

There is a single control parameter, $Q^{\mathcal{C}}$, applied to all the processes (the encoding at the sending terminal and decoding at several receiving terminals) each generally resulting in a unique output, i.e. the $\Delta \mathcal{C}_p^n$ at each terminal. The controller will suggest a quantizer for all of the terminals considered for optimization $\hat{Q}^{\mathcal{C}_n,\text{new}} = \hat{Q}_n^{\mathcal{C},old} + \Delta Q^{\mathcal{C}_n}$, which is fed to the post-processing block in Figure 5.12. This block performs the following operations:

- To begin with, largest quantizer $\max_N(\hat{Q}^{\mathcal{C}_n,\text{new}})$ is chosen as the initial candidate $\tilde{Q}^{\mathcal{C},\text{new}}$.
- The control parameter to be used is the given as:

  $Q^{\mathcal{C}} = \text{CLIP}(\text{ROUND}(\tilde{Q}^{\mathcal{C},\text{new}}), 1, Q^{\mathcal{C},\max})$.

For streaming applications where pre-coded segments of media are made available, the same principle as above can be used by considering $\rho > T$ to be the segment duration.

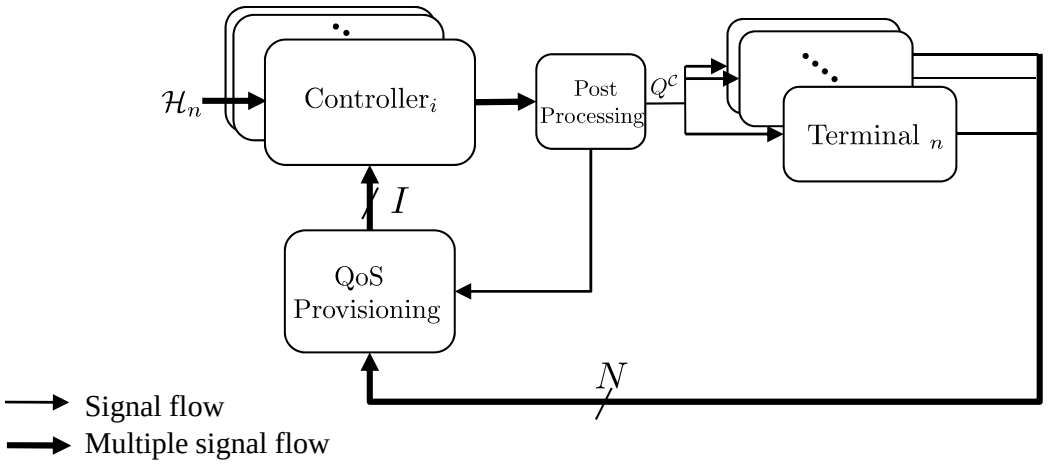



Figure 5.12: The block diagram of the control system

**QoS Provisioning:** As discussed before, a single control parameter will be available for optimizing an entire connected service. At the same time, the upper bounds of computational resource usage at each terminal has to be strictly maintained. When these two considerations are combined, this results in an undesirable effect:

the terminal with least available resources dictates and effectively limits the overall QoS.

Therefore, to let the service perform amicably, a parameter $\kappa = I/N$ is introduced at the controller. This parameter determines which $I$ of the total $N$ terminals must be optimized for. The sending terminal is an exception: it has always to be optimized for. The QoS provisioning block shown in Figure 5.12 achieves this by performing the following steps in the given sequence:

1. The running average of quantizer $Q^{\mathcal{C},av}$ over one second period is maintained at the QoS provisioning block as the metric of quality of the connected service.
2. A variable $\text{RES}_n$ is maintained for each terminal, initialized to zero. If $Q^{\mathcal{C},av} \geq Q^{\mathcal{C},TH}$ (a predetermined threshold value), $\text{RES}_{nc}$ is incremented after each frame-level optimization, where $nc$ is the most resource constrained terminal: $nc = \arg(\max_N(\hat{Q}^{\mathcal{C}_n}))$. This information is provided to the QoS module via the connection from the post-processing module shown in Figure 5.12.
3. If $I/N > \kappa$, and $\text{RES}_n$ for a terminal $n$ is greater than a threshold, the terminal is removed from further optimizations.
4. Such a terminal $n$ removed from optimization can be reconsidered after a transmission of an IDR frame, with its $\text{RES}_n$ reset to zero.

The QoS admission control policy can appropriately adjust $\kappa$ till the quality metric $Q^{\mathcal{C},av}$ reaches a desired threshold.

### 5.4.4 Codec Behavior Under Lossy Channel Conditions

So far the model developed in Section 5.3 assumes a loss-less reception of coded video data. Likewise, the encoder behavior for sender end optimizations is also described for a loss-less scenario. The entire system is however effected by losses, and the impact on both the encoder and the decoder is summarized in the following.

#### Effect on Resource Usage at the Decoding End

The decoder, upon detection of an erroneous bitstream initiates the error-concealment process. As already described in detail in Section 2, this is not a standardized process and a plethora of techniques exist to fulfil this purpose. It has already been a recommendation that the error concealment techniques should not be significantly more complex than the actual decoding modes which are lost. For example, in our experiments PFC has been used. The technique has two advantages: it is easier to compare with other results since it is widely used, and secondly its complexity is in comparable to the P_SKIP mode, which in general is considered to be the most simple coding mode, as already discussed in Chapter 3.

For delivery of video over lossy channels, a robust system configuration has been proposed in Chapter 4. For such a system, the sender-end is made aware of the losses. Upon reception of such a message, if the concealment technique used follows

the above guidelines, no action is needed to update the computational resource usage estimate of the decoder made at the sender end, as long as the concealment is less complex than decoding itself. This will be corrected by the periodic updates by the decoder as discussed in Section 5.4.3. Till such an update is received, the only side-effect of this will be a probable resource buffer underrun.

#### Effect on Resource Usage at the Encoding End

Similar to the decoding-end concealment with a strong focus on computational resource constrained devices, Chapter 4 has also discussed the proposed resource constrained error-recovery techniques at the encoding end. Specifically, IEC-1 strives to recover from errors without significantly increasing the computational resource demand.

However, to enable the prediction from a frame not impacted by losses, the quantizer in Section 5.4.3 is modified as follows. The start index $j$ of reference frame is initialized with the most recent reference frame with reference regions available for prediction (a lost reference region is already known at the encoder by the error-tracking mechanism).

The resultant change in the computational resource usage is well modeled by the decoder model introduced in Section 5.3.2, and hence the remainder of computational resource optimization can follow as usual in the case of channel losses.

## 5.5 Offline Resource Optimization

For this category of optimization, the optimization algorithm may only select one of several possible pre-encoded versions or *representations* of the same source content. With a pre-determined set of coded representations, only the decoding end resources are targeted for optimization.

Since hybrid video codecs provide with the possibility of hierarchical spatio-temporal predictions, this adaptation can only switch to independently-decodable samples. For example, an IDR frame of H.264/AVC in a given representation. Likewise, for scalable/multi-view coded content, a higher layer/view is dependent on lower layer(s)/view(s) for decoding, respectively, and cannot be decoded independently. This adaptation is typically referred to as Group Of Pictures (GOP) based adaptation.

Since online optimizations as discussed in Section 5.4 can adapt on a frame or even sub-frame level, they can achieve a superior performance. However, three important pre-requisites were also defined in Section 5.4, if any of them is not fulfilled, offline resource optimization is the only available option. A wide range of applications fall under this category:

1. Online encoding is hardly scalable for commercial applications: the computational resource demand grows linearly with number of unique representations to be generated. Hence even if the source content is live, a scalable application

only generates a handful of representations for adaptation for all the clients. Hence online optimization strategy proposed in Section 5.4 is not usable directly.

2. Receiver-awareness is also a potential bottleneck for scalable applications. As before, the online optimization strategy proposed in Section 5.4 is not usable since it relies on a decoding-end model available at the encoding end for resource optimization of the decoder.

Two such very important classes of well-known applications are the traditional terrestrial and satellite video broadcast, and the more recent adaptive HTTP streaming technologies like DASH and Apple HLS etc. In these applications, adaptation is achievable by the means of a set of alternate representations made available to the client to choose from. These existing systems only target channel capacity optimizations however.

### 5.5.1 GOP-based Resource Optimization

A GOP-based adaptation is performed by selecting a GOP from the representation $x$ in a subset $\hat{\mathcal{X}}$ of all the representations $\mathcal{X}$, such that:

$$x = \arg\min_{x \in \mathcal{X}} \left( \sum_{t=t_i}^{t_f^{GOP}} \mathcal{C}_t^x - \mathcal{H} \cdot (t_f^{GOP} - t_i) - \mathcal{C}^{max} \right) \Bigg|_{\sum_{t=t_i}^{t_f^{GOP}} \mathcal{C}_t^x - \mathcal{H} \cdot (t_f^{GOP} - t_i) \leq \mathcal{C}^{max}} \tag{5.17}$$

$$\Delta \mathcal{C}_p^x = \langle \mathbf{\Theta}_l, \mathbf{\Psi}^{r,x} \rangle, \mathcal{L}_{l-1} < \sigma \leq \mathcal{L}_l, \tag{5.18}$$

This subset $\hat{\mathcal{X}}$ consists of representations that are short-listed by an outer optimization loop, e.g. channel capacity optimization, if it exists. Equation 5.18 uses the formulation of Equation 5.8 for prediction of resource usage. Equation 5.17 selects the GOP from a representation such that highest resource buffer fullness is achieved without overflow. This optimization targets to maximize the resource utilization. $t_i$ is the time of availability of the first frame intended to be presented in the decoder buffer.

### 5.5.2 Architectural Options for Optimization

In general, offline optimizations can be performed either at the sending-end for each of the receivers to be optimized or at each of the receiver themselves (unlike online optimizations of Section 5.4 which are performed exclusively on the sending end). Each possibility has its own implication, as discussed in the following.

#### Sending-end Optimizations

In this configuration, the receiver communicate their respective computational resource model to the sending end in a similar fashion as described for online optimiza-

tions in Section 5.4.3. Using these models, the sending end can select the appropriate representations to stream to individual receiver based on the optimization criteria specified in Section 5.5.1.

The advantage of this strategy is that the model stays fixed for a given decoder implementation and as a result the communication overhead is minimized. The drawback however is that, as noted already, such receiver awareness may prove a bottleneck for scalability, e.g. commercial applications targeting hundreds of thousands of clients. Secondly, some widely deployed applications such as the traditional TV broadcast are inherently designed to be receiver-agnostic.

#### Receiving-end Optimizations

In this case, the computational resource usage model $\mathbf{\Theta}_l$ is still needed at the receiver, albeit for different reasons. The receiver should be able to predict the expected computational resource usage of a given representation *before* attempting to decode it, so as to compare it with its computational resources and make a well-informed selection. In order to make use of this model, the receiver has to have the knowledge of $\mathbf{\Psi}^r$, the associated activity factor $l$, and the number of reference frames used $\mathcal{J}$. Hence the overhead of this adaptation is:

1. Real-time data transfer overhead of $\mathbf{\Psi}^r$, $l$ and $\mathcal{J}$ from sending-end to the receiver. Considering the maximum range of each of the components specified in Section 5.3.2, which depends on the resolution of the content used, the overhead data rates are calculated and listed in Table 5.1. The overhead is considerable for QCIF case, when compared to the typical data rates used for such video content. However, these are raw data rates, and compression of this data can be considered if required (e.g. by using run-length coding).

| Content type | Frame rate | Signaling overhead per frame (bytes) | Bitrate overhead (kbps) |
|---|---|---|---|
| HD | 30 | 233 | 56 |
| SD | 30 | 203 | 49 |
| QCIF | 15 | 144 | 17 |

Table 5.1: Signaling overhead for receiver-end optimization

2. Equation 5.8 is used on run-time to predict the computational resource usage at the receiver on a GOP level.

### 5.5.3 Design Considerations for GOP-based Optimization

How the representations are provided and selection (adaptation) can be performed for the GOP-based resource optimization algorithm introduced in Section 5.5.1 actually depends on what selection options are provided by the application.

In the case of a typical TV broadcast, there is no option provided of selecting between representations, hence SVC is the only available option for adaptation. In this case,

a broadcasted channel provides a scalable video content, each layer can be considered as a representation. Hence the client with more resources can decode a higher layer and vice versa. Since the framework of such applications offers limited capabilities for signaling of the content model information above, it has to be embedded within the content, e.g. the private data fields within the MPEG-2 TS [140]. In such adaptation, the higher layers not decoded are simply discarded at the client side, and no channel capacity is saved.

For adaptive streaming applications like DASH, the server has a possibility of providing a set $\mathcal{X}$ of independent representations for adaptation. The model information can be provided as a part of "index segments" in DASH, which can be accessed independently of the content, and hence the selection can be performed before actually accessing and downloading the media. This approach results in channel capacity saving.

These optimizations rely on an end-to-end delay of at least one GOP interval. Since end-to-end delay is not a major constraint in such systems, most often if not always a reliable channel is available. Hence error robustness is not a design consideration for these optimizations.

# Chapter 6

# Selected Performance Results

In Chapter 5, we have presented a comprehensive analysis of a generic computational complexity constrained video communication framework in the beginning. Video communication applications were categorized based on the underlying service topology and type of adaptation, followed by a comprehensive timing analysis. A decoder computational resource usage model was proposed and analyzed. Finally optimization frameworks were proposed for online and offline encoding configurations..

In this chapter the proposed framework is evaluated quantitatively along with the definition of the reference system for the sake of comparison. Various system configurations of interest are simulated and evaluated.

## 6.1 Codec Software and Hardware Selection

In order to have a reliable verification, several well-known H.264/AVC software codecs were evaluated on a wide range of hardware platforms.

- The most suitable option in this regard is using MVC capable H.264/AVC codec by Nokia [141]. This software is designed to be implemented on mobile platforms for real-time video encoding and decoding.
- x264 video encoder [142] is written for fast processing on desktop computers as well as portable laptops etc. for generating content mainly for storage and local viewing. The corresponding video decoder typically used is FFmpeg [143], which is also designed for similar purpose.
- As a reference, all techniques were also verified on JM software [144], although it is a reference software and not intended for any real application.

The above codeces have been modeled on a variety of realistic hardware platforms:

- Hardware architecture (HA) 1: Qualcomm's MSM7200/7201ATM Chipset Solution [145] consisting of a 528 MHz ARM11$^{TM}$ applications processor. The optimizations was verified on a mobile phone set [146]. JM reference software

was not evaluated on this platform since it is not designed to work on such portable devices.

- HA2: An ultra-low-voltage Intel® Atom™ Processor N270 [147].
- HA3: AMD Opteron™ [148], based on a conventional desktop CPU. Simulations based on non-constrained computational resources were typically done on this platform.

## 6.2 Confidence Level and Unknown Variance

A typical problem faced in assessing the systems similar to the target system is the impact of several underlying randomly varying phenomenon on the observations. In the target system, if the resource management is operating on a lossy channel, then these phenomenon include the variations in channel characteristics, the non-deterministic nature of the resource constrained hardware, and last but not the least: the varying statistics of the video source content. It is difficult to know the exact distribution of these phenomenon before hand, and for real-time video source content, this is next to impossible.

A similar problem discussed in [149] deals with the varying channel characteristics. To formulate the problem, the probability that the mean metric of interest $\overline{QM_i}$ after $I$ iterations of the experiment is away from the true mean $\overline{QM}$ of that metric by the confidence interval $\epsilon$ is given as

$$\mathbb{P}\left(\left|\overline{QM} - \overline{QM_i}\right| > \epsilon\right) = 1 - \beta,\ i = 1, ..., I. \tag{6.1}$$

The bound $\beta$ is the confidence level, which is lower bounded by Chebyshev inequality [150] as $\beta \geq 1 - (1/\epsilon I)\sigma^2_{\overline{QM}}$. As described in [149] citing work of [151], this is simplified by considering $\sigma^2_{\overline{QM}} \approx \sigma^2_{\overline{QM_i}}$.

However the main problem for generating the results here is to be able to predict a suitable number of iterations $I$ such that $\overline{QM_i}$ has a high confidence level. The above formulation does little help in this case, since it is not possible to arrive at even the approximation $\sigma^2_{\overline{QM_i}}$ without a priori knowledge of $I$. The approach used here in the simulations is the following: initially set $I = C1$ and then $I$ is increased dynamically. The simulations are further iterated till $\left|\overline{QM_j} - \overline{QM_i}\right| < TH1,\ i = 1, ..., I,\ j = 1, ..., J,\ J = I - C1$. Typical tolerance value $TH1$ is 3% to ensure the results have converged suitably. $C1$ has to be set large enough to avoid a false positive generated by a local minima of $\overline{QM_i}$ since as a result simulations will be terminated prematurely. Here it is set to 5. With these settings, in all the results presented here, $I$ always stays under 128 iterations. Similarly, 64 iterations were found to be sufficient for determining the implementation transform of the decoder model.

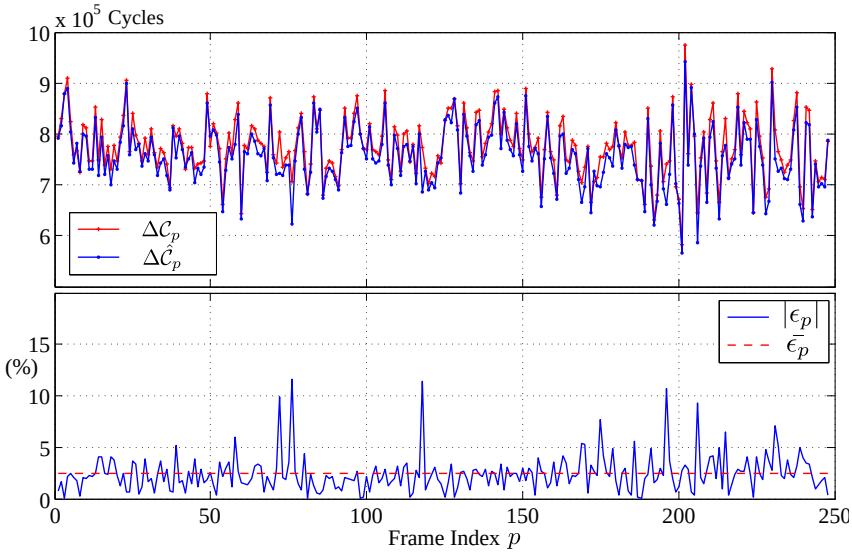



Figure 6.1: Performance of the simplest decoder model ($L = 1$) for sequence "Foreman"

## 6.3 Decoder Resource Model Verification

In this section, we will provide a quantitative evaluation of the decoder resource usage model proposed in Section 5.3.2 to shed light on its accuracy and robustness. The "training set" in this case is the natural and readily available scene content that provide the known coding options $\Psi^l$ and measured computational resource demand $\vec{\mathcal{C}}$ (following the guidelines of Section 5.3.4) to generate the model $\mathbf{\Theta}_l$ as given in Equation 5.10.

We have used the sequence "Bar" and "Stunt" for this purpose. The former contains significant static scene content with relatively high background texture, while the latter consists of hectic motion throughout the content. Hence combined, the two provide a good mix of the activity factor $\sigma$. The coding conditions used were 64 kbps with most-recent reference frame used for inter-prediction. At first, a linear model approximation is evaluated as the simplest approximation ($L = 1$) and its performance is assessed.

Figure 6.1 assesses the performance of the simplest approximation with $L = 1$, which implies a linear model. This evaluation is done on HA1. The model is then applied to predict the computational resource usage ($\Delta\hat{\mathcal{C}}_p$) of "Foreman" and the instantaneous result is plotted along with absolute error magnitude $|\epsilon_p|$ (in percents). It can be seen that the prediction follows the actual resource usage $\Delta\mathcal{C}_p$ to some extend, but $|\epsilon|$ frequently peaks significantly above 5%, with a mean value of approximately 2.5%.

For the second set of evaluation, the same configuration was used as above, with $L = 2$, and $\mathcal{L}_1$ is empirically tuned to 1.1, hence essentially splitting the static and very low-motion scenes from all the other content. The performance is shown in Figure 6.2 along with $|\epsilon_p|$, $\epsilon_p^{max}$, $\bar{\epsilon_p}$ (mean error), and $\sum_p \epsilon_p$ (cumulated error). It can be seen that this approximation is sufficiently accurate. This has been verified

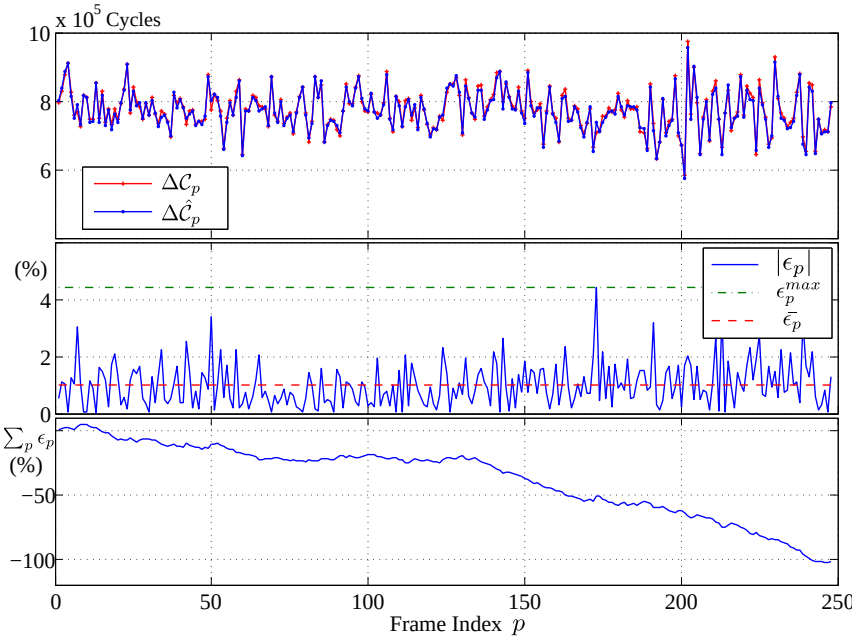



Figure 6.2: Decoder model with $L = 2$, its instantaneous, average, peak and cumulative error performance for "Foreman"

for a variety of content. The peak error magnitude $\epsilon_p^{max}$ stays well below 5% and $\bar{\epsilon_p}$ is in the vicinity of 1%.

Another result using same modeling parameters is tested on HA2 for sequence "Party" coded at 128 kbps showing a similar and stable result in Figure 6.3. In this case the "training set" was coded at half the bitrate (64 kbps) of the verification scenario. Verification has been done for up to 4 times the image resolution and bitrate as used for "training set." Hence this modeling configuration is considered sufficiently accurate to be reliably used across the range of implementations and coding parameters.

It may be noted however, that although average and peak error is within well manageable region, the accumulated error is still of concern for the framework proposed in Chapter 5, since the sending end relies on this estimate. As shown in Figure 6.2, $\sum_p \epsilon_p$ grows to 100% over a period of 200 frames. Hence as already discussed in Section 5.3.1, this problem must be catered for. The two possible approaches (that can even be used in conjunction) to address this are:

- A periodic update from the decoder about the actual status of its resource buffer fullness, and the sending end synchronizes its estimate to rid of this accumulation.

- A factor of safety is ensured such that $\mathcal{H}.T > \Delta\hat{\mathcal{C}}_p + \bar{\epsilon_p}$. If this condition is fulfilled, there will be no steady-state accumulation of error and the problem of drift is avoided altogether.

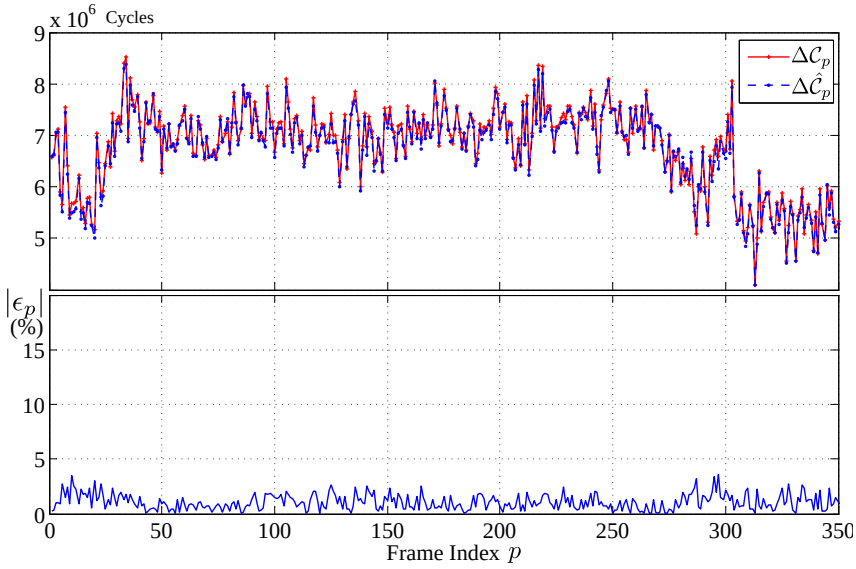



Figure 6.3: Verification of the model on HA2 for sequence "Party"

## 6.4 Online Optimizations

In this section we investigate the performance of the optimizations for online encoding applications. Two target systems will be investigated: a mobile video conversational application and TPTA system.

$\mathcal{K}_1$ and $\mathcal{K}_2$ used for the optimizations defined in Section 5.4.3 (Equation 5.16) are found empirically as 5.6 and 7.3, respectively. As described in Section 5.4.3, these values are independent of the implementation (both software codec and the hardware), and are fixed for a given video coding standard.

### 6.4.1 Reference System

The foremost question for evaluation of the proposed system of Section 5.4 is which reference system to use for quantitative performance evaluation and comparisons? The proposed system is abbreviated as PROP. For most results we use two reference systems:

#### Fixed Computational Resource Usage Configuration

This technique is an enhanced form of what is being deployed in current and next generation of mobile video communication applications e.g. in [3], as introduced in Section 1.1. The enhancement done is that for coding a video sequence at a given terminal $n$ is

$$\mathcal{C}^{n,max} = \max_p (\Delta\mathcal{C}_p^n). \tag{6.2}$$

This will give the best performance achievable by this technique. Practically however, it is quite difficult to implement Equation 6.2 for online-encoding systems since it relies on the resource usage required for coding video frames in future. Instead, $\max_p(\Delta\mathcal{C}_p^n)$ is determined from a set of training sequences (sequences encoded using the same parameters as for encoding the stream being evaluated, e.g. the bitrate). We will abbreviate this technique as FCC. As a reference this is still well suited, since it will reflect the optimal performance bounds achieved by existing systems.

**Resource Management Using Complexity Allocation Management (CAM)**

This technique, abbreviated here as CCAM, is based on the Complexity Allocation Management technique proposed in [81, 82]. This technique has a two-level resource management framework: one on frame level and the other on a macroblock level.

**Frame-level CAM:** The overall technique begins with the prediction of $\Delta\mathcal{C}_p^n$ denoted as $\Delta\hat{\mathcal{C}}_p^n$. The prediction is made from a recently coded frame with similar optimization, bounded by a minima and maxima (those seem to converge to 0 and $\mathcal{C}^{n,max}$ respectively very quickly). $\Delta\hat{\mathcal{C}}_p^n$ is the frame level computational resource usage target.

The frame-level optimizations are achieved by selecting one of five different ME options, as enlisted in Section 3.1.7, each with a different level of computational complexity. Mode ranking is done based on RD performance in the previous frame.

**Macroblock-level CAM:** After coding a fraction $\rho$ of a frame, the computational resource budget estimate for the remaining frame is given by

$$\Delta\hat{\mathcal{C}}_p^{n^{1-\rho}} = \Delta\hat{\mathcal{C}}_p^n - \Delta\mathcal{C}_p^{n^\rho} \tag{6.3}$$

which is distributed evenly in all the remaining macroblocks. Based on this estimate, a subset of coding modes is selected, as detailed in [82]. Hence this technique also is virtually using the notion of complexity quantization, yet splitting it into two separate parts is not very optimal; there is little use of performing motion estimation for inter-prediction modes that are ruled out for the coding step.

Another problem in both the frame-level and macroblock CAM of this technique is that it uses close temporal and spatial neighbors as predictors. This results in an undue statistical bias in mode decisions till the predictors are reset at GOP level.

**Extensions:** This technique has been extended here for this work to also manage computational resource usage of a video decoder following the same principles used above for video encoder at frame and macroblock levels. At the frame level, the 5 different motion estimation paths detailed in Section 3.1.7 can also be used to manage the resources at the decoder as well, since they impact on motion-compensation

(inter-prediction) process, and this process is typically the most computational-intensive at the decoding end. For example, removing the fractional-pel search at level-3 removes the requirement of fractional-pel motion compensation. Setting motion vectors to zero at level-5 simplified the motion compensation significantly. Also, the QoS provisioning concept introduced in Section 5.4.3 is extended to this technique so that it can be used to manage resource usage of an entire video coding system.

### 6.4.2 3GPP PSC Application

As discussed in Section 4.1, and Section 5.1.1, video conversational applications on portable devices are the most challenging in terms of robust performance in wake of losses and limited computational resources. The assessment of these systems is presented here first.

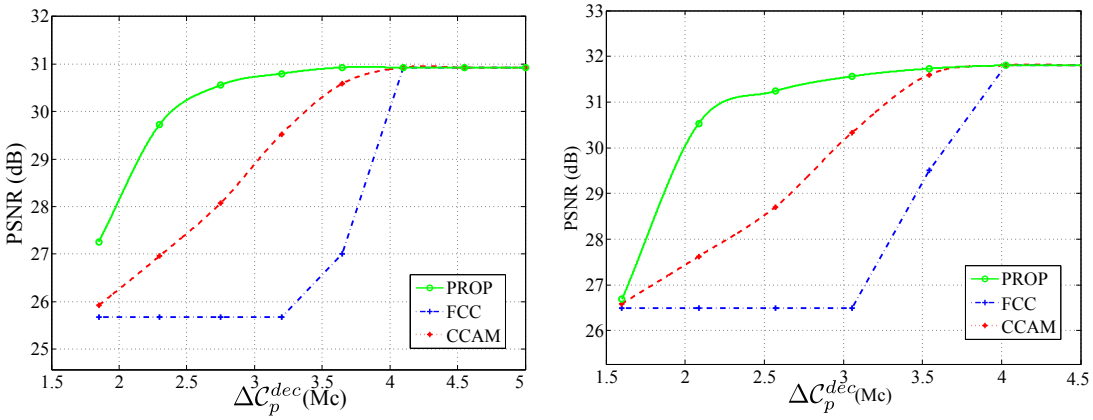
The details of the target system have been introduced already in detail in Section 4.1. Typically in terms of the codec topologies introduces in Section 5.1.1, it falls under point-to-point topology. Still video-conference type of applications can be envisioned as an extension, which will fall under point-to-multipoint topology. For the optimizations, the receiver(s) will feedback their computational resource usage model to the sending end as discussed before in Section 5.4.3.

#### Simulation framework

An overview of the simulation framework can be found in Annex. A. The codec configuration has been discussed in detail in Section 5.4.3. QCIF sized video test sequences "Stunt" [109] has been used at 15 fps, while "Foreman" has been used at 30 fps due to its popularity and hence easy comparisons with established benchmarks. Although extensive tests have been conducted on other sequences as well, similar results have been inferred from them as well. Here for the sake of compactness, the results from these two sequences are presented.

"Stunt" is a scene shot with a typical portable camera unit and represents hectic motion throughout. The motion is so intense in most of its parts that the ME suffers significantly since good inter predictors are hard to find. The resource varies quite dramatically on various platforms tested for both encoding and decoding purposes. "Foreman" on the other hand represents a sequence shot it outdoors with potentially very good inter-prediction candidates in general. Specially the last part of the sequence represents a pseudo-static scene and this typically results in very small coded error and very small resource demand on the terminals. Hence in the following results, the last 200 frames of this sequence are shown since it shows the entire dynamic range of motion and stationary characteristics, and it is this part that is found to be the most challenging for the control algorithms.

The content used for the training purpose for FCC are the video test sequences used in [3] and QCIF sized "Foreman".




(a) Sequence "Stunt" (b) Sequence "Foreman"

Figure 6.4: Computationally resource constrained decoder average performance

The guidelines of robust system configuration in Section 4.1.3 are followed. The channel coding overheads are given due consideration. As a result while a RAB with effective bitrate of 128 kbps is used, the effective bitrate allocated for video is 92% of this (approx. 118 kbps). This bitrate is quite quite suitable for the selected test content with the given coding parameters. Unless specified, $B^{enc,ip} = B^{dec,op} = 2$.

Since there are a considerable number of parameters involved, it is more prudent to see the results by initially fixing one set of parameters and then going into more details later. Hence in the first part the lossless performance is assessed in detail. With the understanding of the performance of the techniques in the loss-less scenario, we will proceed to add the effects of channel losses and its analysis.

### Performance in loss-less configuration

To begin with, the performance of the system is analyzed in the absence of channel losses. These results will also be applicable in the case when a reliable communication medium is used, although in conversational applications, this luxury is not possible because of the strict end-to-end delay requirements.

Since for the target communication system the bitrate is strongly coupled to the transmission capacity offered by the RAB, the average results for PSNR are provided against $\Delta\mathcal{C}_p^n$ for each of the techniques. At first, the computational resources at only the decoding end are limited on HA1, while the encoding-end is configured to have non-constrained computational resources. The results are shown in Figure 6.4. On an initial view the results for both the sequences look similar; approximately 4 mega clock cycles (Mc) per frame (denoted as $\Delta\mathcal{C}_p^{dec}$) is a sufficient amount of resources for achieving $Q^{\mathcal{C}} = 1$ throughout coding the sequences. Beyond this, there is no further improvement observable in any on the techniques.

On the other extreme, in the vicinity of $\Delta\mathcal{C}_p^{dec} = 1.8$ Mc, only $Q^{\mathcal{C}} = Q^{\mathcal{C},max}$ is able to maintain the upper bound defined in Equation 5.4, and the controllers are no longer to maintain real-time coding below this value of $\Delta\mathcal{C}_p^{dec}$, denoted as $\Delta\mathcal{C}_p^{dec,min}$.

Apparently since the quantization mechanism is different in PROP as compared to CCAM, PROP can sill control on $\Delta\mathcal{C}_p^{dec}$ slightly lower than $\Delta\mathcal{C}_p^{dec} = 1.8$ Mc. However, for "Foreman" $\Delta\mathcal{C}_p^{dec,min}$ is slightly lower for all the techniques, around 1.6 Mc.

Hence for this system, the optimization potential lies in the dynamic range of computational resources $\Delta\mathcal{C}_p^{dec,min} \leq \Delta\mathcal{C}_p^{dec} \leq \Delta\mathcal{C}_p^{dec,max}$. As discussed in Section 5.4.3, the codec used for this evaluation provides the possibility to extend this dynamic range almost arbitrarily (e.g. by using LTM-MCP) if the resources are available. The purpose of these simulation is to highlight the principle and the potential for optimization.

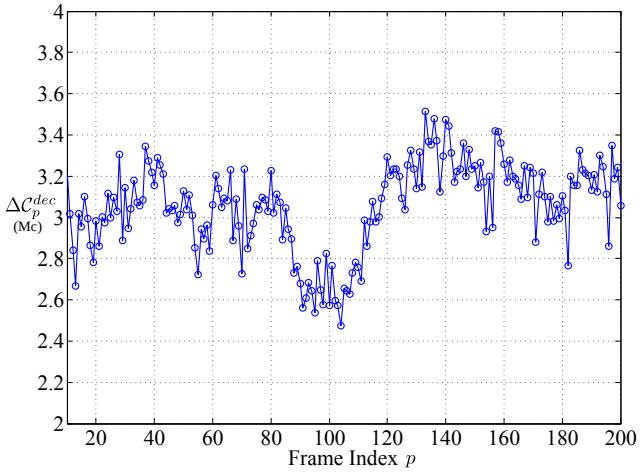
The range of PSNR variation for "Foreman" however is much more compared to "Stunt" in the same period of resource variation; "Foreman" is able to achieve much higher PSNR at peak resource availability due to smooth motion in the content compared to "Stunt"; in the latter, at times good inter-prediction matches are infrequent with the available motion search range. At the same time due to higher texture, at very low resource configuration where mostly low complexity intra-coding modes are more frequent, the PSNR is lower than that of "Stunt".

It can be seen that FCC performance is the worst in the range of interest, while PROP outperforms CCAM by a margin of up to approximately 3dB.

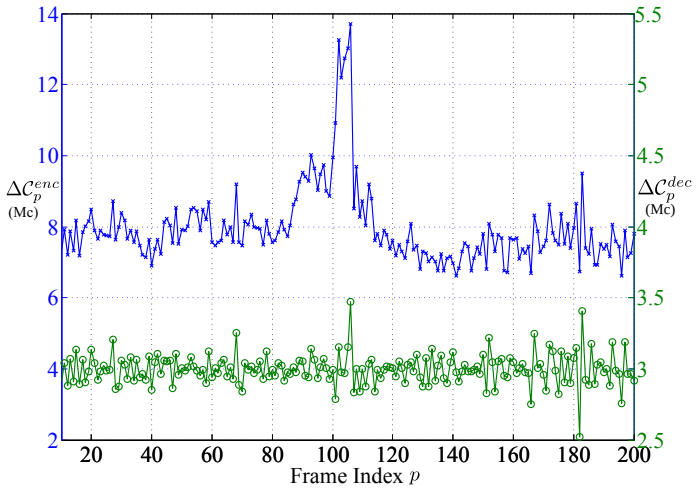
It is easily understandable why would FCC perform the worst; its PSNR performance will be at best equal to PROP for the sample with maximal resource demand within the entire set of training sequence, while the other techniques adapt to the content at even sub-frame level. For the other techniques it is worthwhile to take a closer look at the instantaneous optimization results to understand the working and identify the reasons for the performance difference.

Figure 6.5(a) shows the plot of the instantaneous resource usage by a decoder with unconstrained resources. It can be seen that on this specific hardware platform the resources usage can vary between 2.6 Mc and 3.4 Mc. Correspondingly, by using PROP as shown in Figure 6.5(b), the resource usage is optimized for a budget of 3 Mc. This figure also shows the trend of unconstrained resource usage by the encoder in the same session; although the two terminals are operating on different platforms and hence the two curves in this figure are on separate Y-axis, not meant to be compared with each other. It is the shape of the curves that is of interest.

Figure 6.5(b) still shows some peaks in $\Delta\mathcal{C}_p^{dec}$ that might seem undesirable; however, allowing fluctuations is important for the optimization as long as it does not indicate an instable behavior and the bounds of Equation 5.4 are not violated. Hence in Figure 6.6 the fluctuations in the resource buffer fullness is plotted along with the corresponding $\mathcal{Q}^c$. It can be seen in comparison with Figure 6.5(a) that around frame index 100, the controller reduces the quantizer significantly because of less resource demand, and as soon as the resource demand increases (also evident by the spike in buffer fullness of Figure 6.5(a)) the $\mathcal{Q}^c$ increases once again. Instantaneous PSNR plot is not given since in an instantaneous plot like this, it is difficult to decouple scene-content related fluctuations from the resource optimization fluctuations; these effects are quite clear from the average plots.




(a) Unconstrained resource usage by the decoder




(b) PROP used on decoder (Secondary Y-axis), unconstrained resource usage by the encoder

Figure 6.5: Instantaneous results for selected frames of “Foreman”

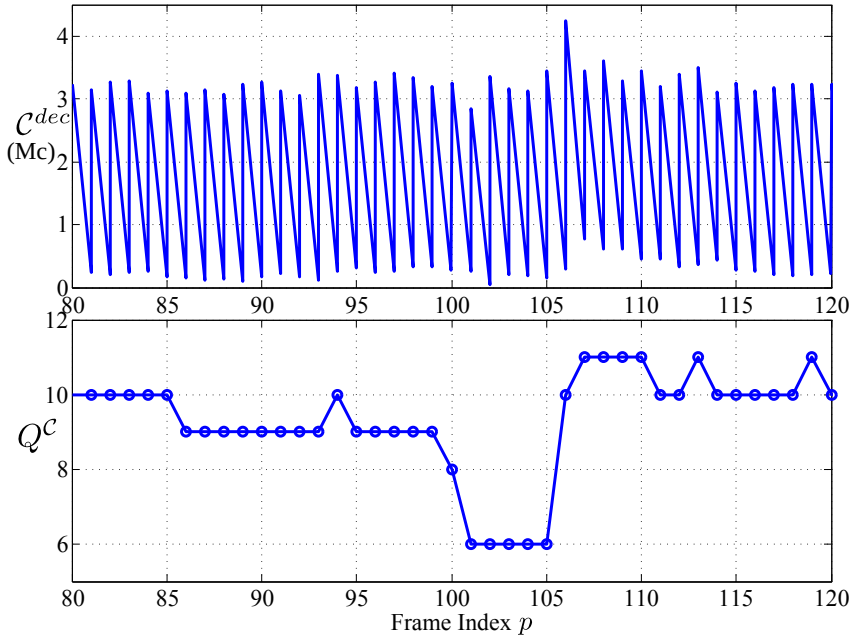



Figure 6.6: The resource buffer fullness (above) and the $\mathcal{Q}^c$ for a selected portion of the sequence

A comparison of this with the results from CCAM is worthwhile to see the reason for its slightly inferior performance. It can directly be seen from Figure 6.7 that the management technique is not quite able to converge well to the target computational resources. This can also be seen from the instantaneous buffer fullness. The reason is evident from the frame-level control parameter shown in the bottom of the figure. For this technique, the numbers 1 thorough 5 on the Y-axis correspond to the 5 different motion estimation levels as listed in Section 3.1.7 at the frame-level, and consequently the motion compensation process. However, the granularity of this control is only at the frame-level. Also these levels are quite coarse (only 5 different configurations), and the different coding options e.g. different block partitions and intra-prediction modes etc. are decoupled from the frame-level control. These effects impact adversely on the performance. However, this is understandable since the mechanism is mainly devised keeping in view encoding-end optimizations only; decoder optimizations were never a consideration for this technique.

In the second set of plots in Figure 6.8, the resources at the encoding-end are constrained on HA2, while the decoder is operated in an unconstrained fashion. As for the previous sets, results for "Stunt" and "Foreman" are shown. In comparison to the optimization at the decoding-end, the gap in the performance between CCAM and PROP is reduced, although PROP still outperforms CCAM by as much as approximately 2.5dB for the sequence "Stunt" and 2dB for "Foreman." The reason of this improvement may also be contributed by the special focus on ME optimization by CCAM. FCC performance is distinctly worst in the entire range of optimization. The higher dynamic range of PSNR for "Foreman" observed for the previous set of plots for the decoder is also observable at the encoding end as well owing to the same reasoning.

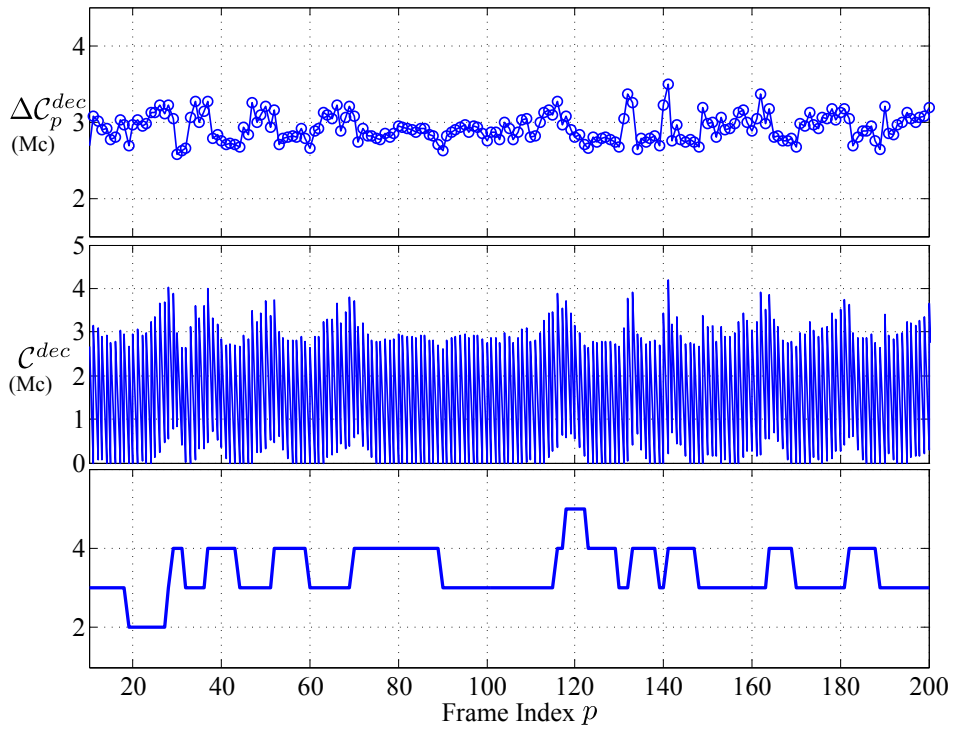
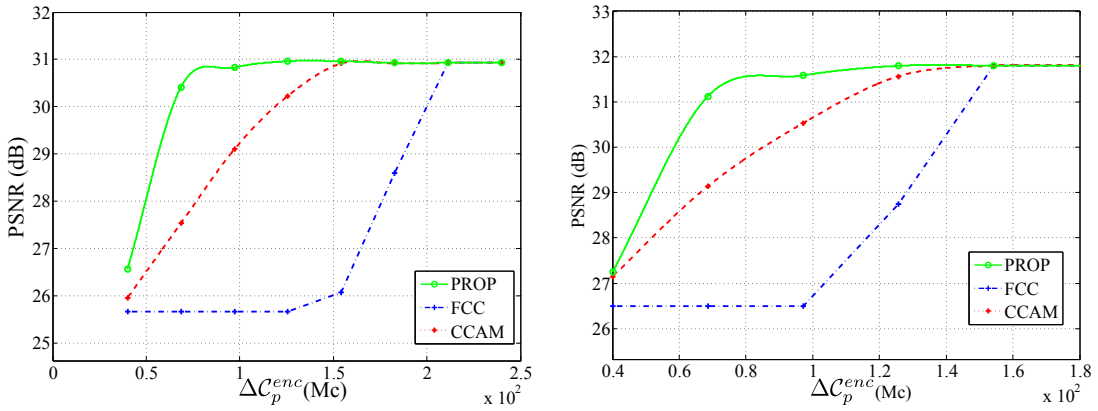



Figure 6.7: Computational resource usage (top), the resource buffer fullness (middle) and the complexity index (bottom) for a selected portion of the sequence "Foreman" using CCAM




(a) Sequence "Stunt" (b) Sequence "Foreman"

Figure 6.8: Computationally resource constrained encoder average performance

A look on the instantaneous resource usage show how PROP manages to converge to its target resource usage ($\Delta\bar{\mathcal{C}}_p^{enc} = 8$ Mc in this run) for the sequence "Foreman", as compared to the unconstrained resource usage, e.g. in Figure 6.5(b). The corresponding computational resources at the decoder end are also shown for HA1, but in this case the decoder resource usage is not constrained. It can be seen that the encoder resource usage is quite well-converged to its target setting.

In comparison however CCAM shows a significant problem. As shown in Figure 6.10, the results for a few selected frames show significant instability at certain instances. At frame indices 104 and 118, $\Delta\mathcal{C}_p^{enc}$ peaks to 125% of its target value, while for the remaining frames this is compensated by lower values. This ensures that the buffer level is maintained within the required bounds.

Upon a close look at how the frame-level motion estimation configuration is chosen, the reason of this instability becomes clear. In the frames coded immediately preceding the above mentioned frames, the conditions seemed feasible to select level-1 complexity at frame level. Full motion estimation is performed at this level. However, there is a huge increase in the resource demand at this level, and this results in the instability.

Such instability is avoided in PROP by keeping all control parameters suitably reconfigurable at sub-frame level, since frame level decisions are too coarse to achieve a stable control. The fluctuations shown in Figure 6.10 result in a slight increase in the quality of the previously mentioned frames on the cost of the adverse effects on the quality on a number of following frames.

In the typical usage scenario of PSC, the resources at both the transmitting and receiving end are constrained. In order to visualize the performance of such a system, a 3D curve is required, with two axis for the computational resource values $\Delta\mathcal{C}_p^n$ at each of the terminals and the z-axis (vertical) for the PSNR plot. As before, the encoder is operating on HA2 while the decoder is operating on HA1.

Since from the results of the previous two sets of data have shown that the most robust technique in all configuration is PROP, only the results for PROP are shown.

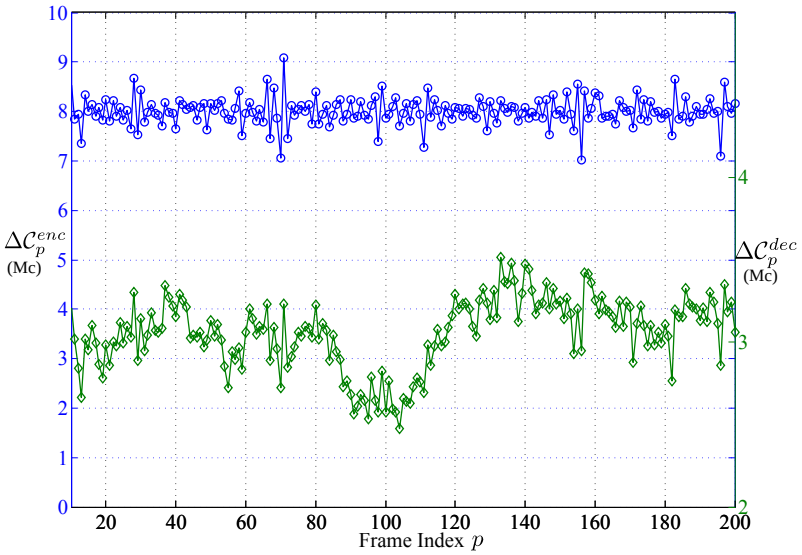



Figure 6.9: PROP used on encoder, unconstrained resource usage by the decoder (secondary Y-axis)

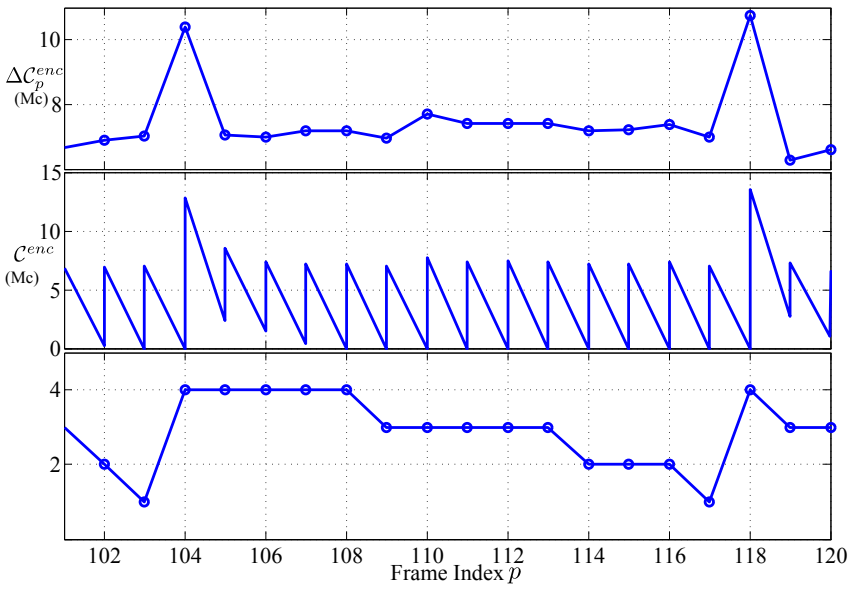



Figure 6.10: Computational resource usage (top), the resource buffer fullness (middle) and the complexity index (bottom) for a selected portion of the sequence "Foreman" using CCAM

The curves, as shown in Figure 6.11(a) and Figure 6.11(b) show that the performance converges to the plots for individual performance optimization of the encoder and the decoder, when the resources of the other terminal are unconstrained. The significantly flat "roof", the sharp facing edge of the surface, and numerical comparisons to the individual curves in the preceding sections indicate that as the resources at one of the terminals are reduced, the effect on the performance at the other terminal is a smooth transition to a reduced value; the reduced resources at one terminal do not cause any unexpected adverse effects on the performance of the other. The resulting performance can be determined at any operating point of resource configuration from these curves.

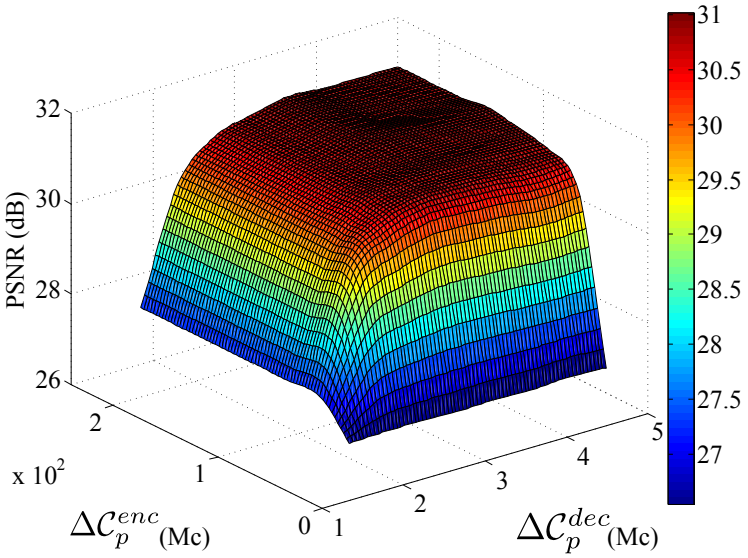
Figure 6.12 gives an inside look into how the controller achieves optimization at a specific operating point for both the terminals. In this figure, the target $\Delta\mathcal{C}_p^{enc} = 8$ Ms, while $\Delta\mathcal{C}_p^{enc} = 3$ Ms for the sequence "Foreman". As before, since the architectures are different, the numbers of the encoder and the decoder cannot be compared with each other, the relative shape of the curve is of interest though.

It can be seen that till frame index 120, the encoding-end mostly determines the optimization. This changes from this point onwards till the end of plot, and in this second section the decoder determines the optimization. Evidently, the two regions are quite well distinguished in terms of scene content as well. The first part consists of medium to high camera motion, while the second part is quasi-stationary. It can be expected that the encoding effort is significant in the high motion area because of the relatively difficult convergence in motion estimation algorithm.

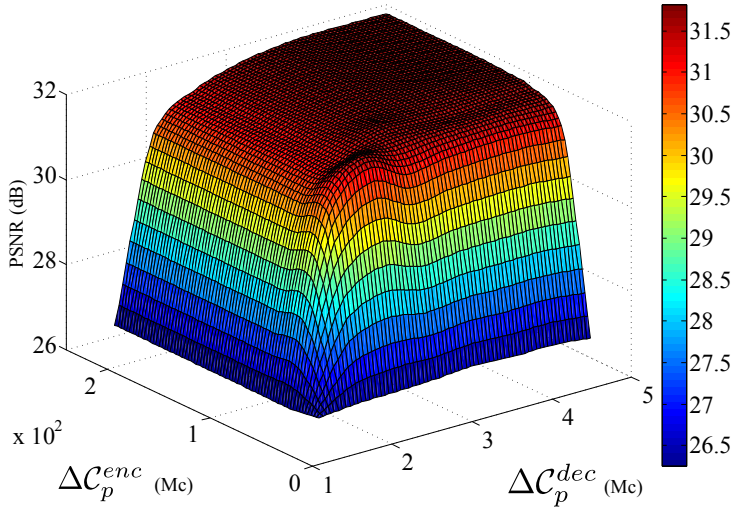
**Discussion on results:** The previous sections provided performance comparison of the various resource management techniques. The results were compared by first optimizing at a single terminal, either encoder or the decoder and letting the other terminal run unconstrained. For both encoder and decoder optimizations, PROP performed the best followed by CCAM. FCC performed the worst overall. Instantaneous resource usage was investigated in detail to reveal the reasons for this performance. Finally, results for joint encoder-decoder optimization was presented using PROP. This technique is able to converge relatively quickly to its optimal performance. For joint optimization at various terminals, different terminals may be most resource constrained at different times, and this joint optimization at the encoding-end is able to cater for this to achieve optimal results. As already seen, the decoder model used at the encoding end for optimization is quite accurate, yet it is required to update the sending end about actual resource usage to avoid any effects of drift between the modeled results and the actual results.

### Performance with an error-prone channel

As seen in the preceding section, for error-free case, PROP performs the best in the dynamic range used in these simulations in terms of the available computational resources. In this section we evaluate the performance of PROP in an error-prone channel. Section 5.4.4 commented on the impact of channel losses on the computational resources of the system, and it was established how it impacts the proposed




(a) Sequence "Stunt"




(b) Sequence "Foreman"

Figure 6.11: Both encoder and decoder computationally resource constrained

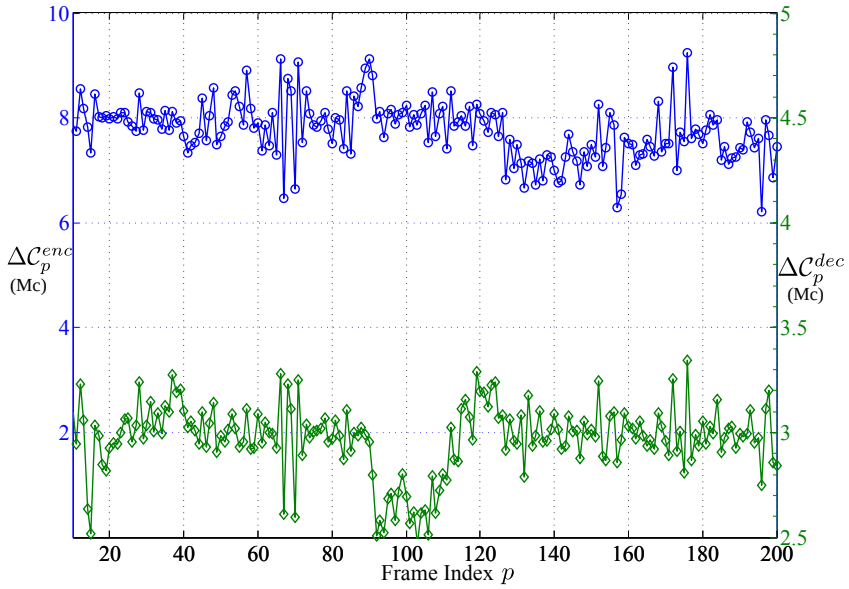



Figure 6.12: PROP used for both the encoder and the decoder (secondary Y-axis)

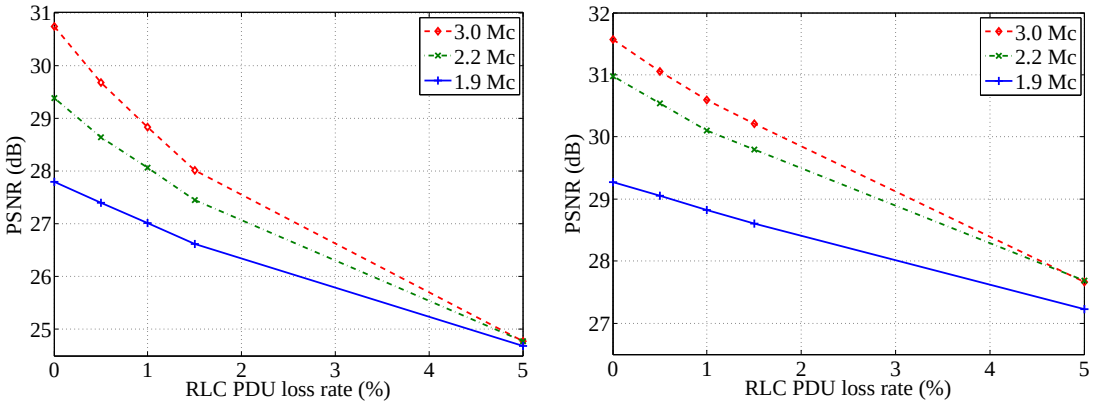
resource optimization framework.

In order to assess the impact of the channel losses on the system, we will use the framework already presented in Section 4.1.3. The most suitable robustness technique in this regard has been established to be IEC-1, which uses LTM-MCP feature of H.264/AVC to achieve robustness in conjunction with an error-tracking mechanism. Section 5.4.4 also noted that this technique is suitable for the computational resource optimization since it does not dramatically impact the computational resource performance of the system with increasing losses in the system.

Hence in Figure 6.13 the above configuration is used to generate the results for a system where the resources on the decoding end are constrained. This figure shows the PSNR performance of the techniques with varying RLC-PDU loss rates ($\nu$), with computational resource usage as the parameter, and the same two video test sequences "Stunt" and "Foreman" as in the preceding sections. The computational resource configuration selected to see the behavior of the codec is around the "knee" of the curve in Figure 6.4; the lower region sees a rapid decrease in performance while the region above this "knee" is already converging to the best performance.

It can be readily seen from the curves that because of the hectic motion in sequence "Stunt" it is strongly impacted by the channel losses. "Foreman" on the other hand does not have such rapid motion and hence it is not so adversely effected by spatio-temporal error propagation. Still compared to the results of various error-robustness techniques presented in Section 4.1.3, the drop in the performance with increasing losses is significantly minimized already. At the higher end of resource availability the PSNR achieved by "Foreman" is also higher than that of "Stunt".

The observation of interest here is that the configurations with lower resources behave in a more robust fashion; the slope of the curves with lesser resource availability is less than that of higher resources. Similar observation can be made for the case




(a) Sequence "Stunt" (b) Sequence "Foreman"

Figure 6.13: Performance of codec with resource-constrained decoder in a lossy channel scenario

where the resources on the encoding end are constrained as shown in Figure 6.14. As an example, the slope of the 3 Mc and 2.2 Mc curve at 30 dB in Figure 6.13(b) is -72.8 db/$\nu$ and -59.3 db/$\nu$ respectively, indicating a much rapid performance descend in performance for the higher resource configuration (3 Mc) at the same PSNR level.

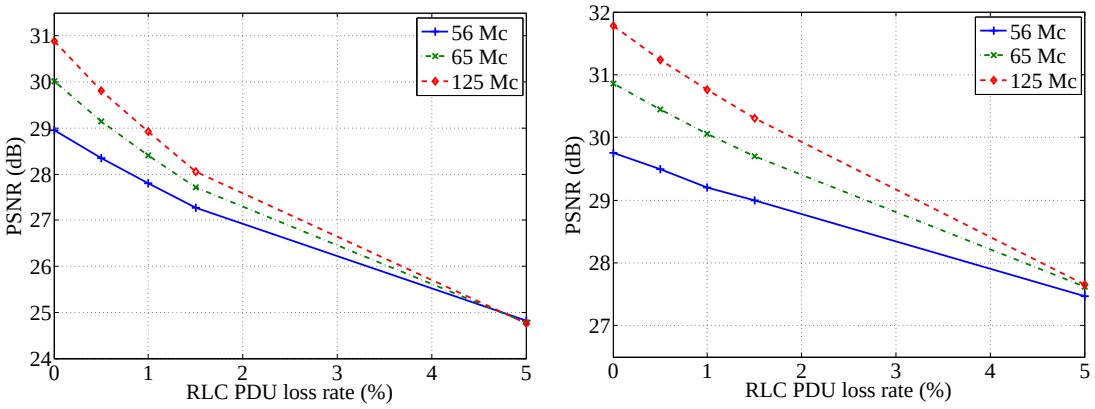
This observation is inline with the analysis in Section 5.4.2; more available resources allow for selection of modes that utilize higher resources to generate complex intra- and inter-predictors to achieve better compression. However, typically the more complex the predictors are, the more neighboring samples they utilize to generate a predictor and hence are more prone to error-propagation. It is this effect that is observable from these curves.

This still however does not define any optimal operation points since higher resources still provide a higher PSNR performance. Although for a sequence like "Stunt" the curves seem to converge to a point in the vicinity of 5% RLC-PDU losses, but at this high a loss ratio, the overall performance of the system is barely satisfactory (below 25 dB) and this does not indicate a suitable operating region. Still the important observation from these results is that devices with lower processing power, although will observe a reduced video quality, but will observe a more uniform performance experience for a lossy channel.

### 6.4.3 TPTA Applications

In this section we apply the principles developed in Section 5.4 to perform the resource optimization of the TPTA system introduced in Section 4.2. The reference techniques used to benchmark the system are the same as in Section 6.4.1, however there are several important differences in terms of resource optimization considerations:

- Only the encoding-end is resource constrained, as discussed in Section 4.2.



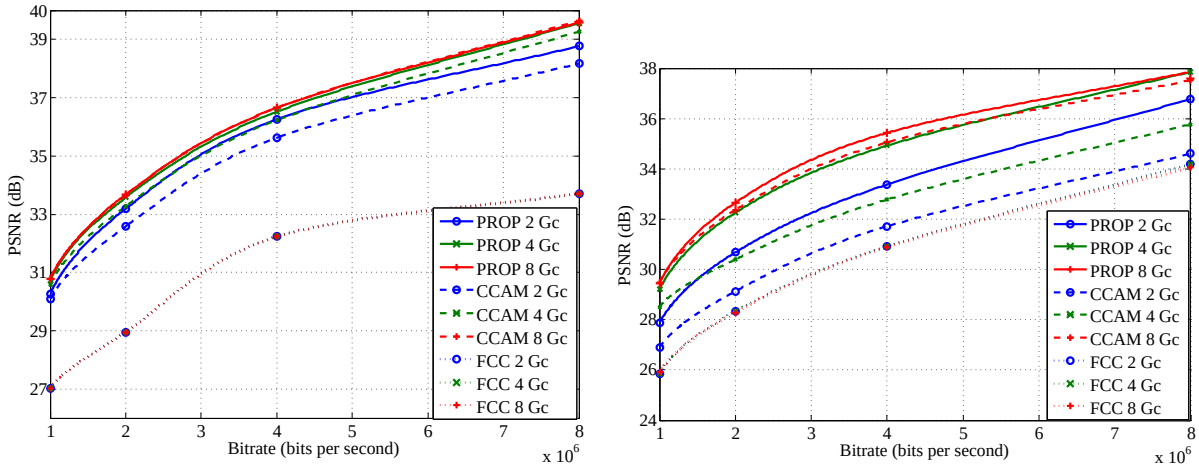

(a) Sequence "Stunt" (b) Sequence "Foreman"

Figure 6.14: Performance of codec with resource-constrained encoder in a lossy channel scenario

- A reliable communication channel is typically enabled, this is especially important for the accompanying heptic information exchange.
- A variety of communication technologies can be used on communication link, unlike some pre-defined RABs as used for the mobile communication system evaluated in Section 6.4.2.

Keeping in view the last two considerations, the RD and CD performance assessment of various techniques in Section 6.4.1 give the best insight into the system performance. For the purpose of evaluation, we have used an extensive set of video test sequences, including MVC content used in Section 4.2.5 as well as test sequences used for the evaluation of HEVC [152]. In our evaluation we have observed two disparate classes of video content, which we term as class A and class B, respectively. The reason for this classification is the dependance of the performance of the evaluated techniques on the video scene content, and will be commented upon in the following. Here as an example, we present results for two test sequences: "BasketballDrill" from class A (resolution: 832x480, 50 fps) and "RaceHorses" from class B (resolution: 832x480, 30 fps) [152]. For this set of investigations the encoding could is not performed in real time because of implementation issues. Such issues are resolved by multi-threaded implementations in practice, and hence the analysis presented here does not loose generality.

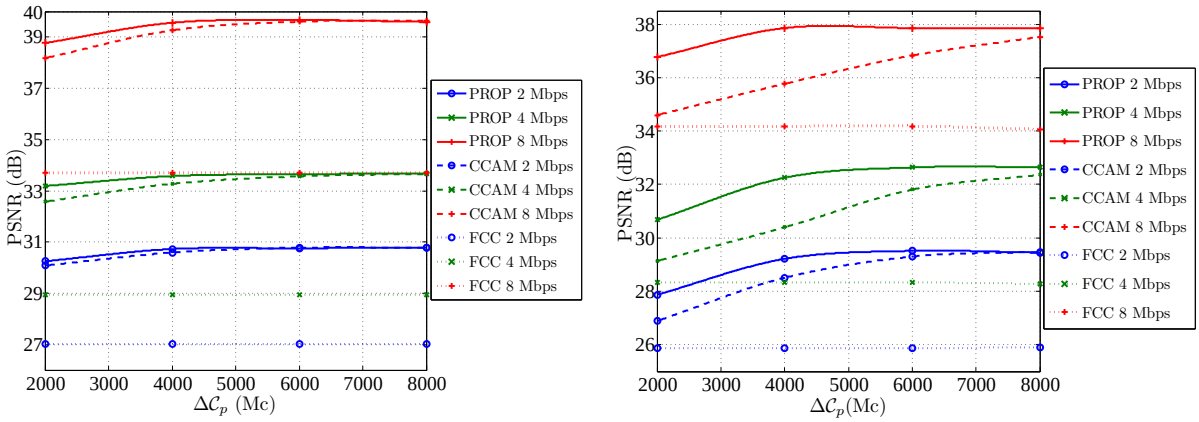
The first set of results in Figure 6.15 show the RD performance of various techniques with the encoder resource usage as a parameter. Upon an initial sight it is clear that the results for the two classes are quite different; the "BasketballDrill" performance of most configurations is already converged except for FCC. At 8 Gc the performance of PROP and CCAM is identical, indicating that at this amount of resources, for this test sequence the scenario is of virtually unconstrained resources. At 4 Gc, PROP provides 0.5 dB gain compared to CCAM. Alternately, Figure 6.16 shows the RC performance with bitrate as a parameter. This provides a more insightful view and it can be observed from the results for "BasketballDrill" that the curves for both PROP and CCAM are quite flat, with marginal improvement with the increasing




(a) Sequence "BasketballDrill" [152]

(b) Sequence "RaceHorses"

Figure 6.15: Comparative RD performance of various resource optimization techniques.




(a) Sequence "BasketballDrill"

(b) Sequence "RaceHorses"

Figure 6.16: Comparative CD performance of various resource optimization techniques.

computational resources. Both the techniques show significant improvement over FCC for most of the range of computational resources. For FCC however, all the three performance curves are overlapping indication virtually no gain with increasing resource availability. This will be described in the following.

Comparing all these results with the performance of the other sequence "RaceHorses" show an entirely different scenario. As can be seen in Figure 6.15(b), "RaceHorses" requires significantly more resources for all the techniques to converge to the optimal performance. PROP and CCAM are still performing quite close at 8 Gc, but PROP is performing significantly better for 2 and 4 Gc. In fact PROP is converging to its optimal performance at 4 Gc. In contrast FCC cannot achieve any optimization at all, this is more clear from Figure 6.16(b) which shows flat curves for FCC. Figure 6.16(b) also shows a larger performance improvement achieved by PROP for larger bitrates as compared to CCAM.

This difference in the performance of various techniques for the two selected test sequences presented here is the basis of the classification we have observed for the variety of test sequences evaluated in this study; for one set our reference techniques CCAM performs quite close to the proposed technique PROP compared to the other. The reason for this disparity is contributed by reliance of CCAM on good zero motion-vector inter-prediction matches to reduce computational resource demand, as described in Section 3.1.7 (for zero motion vector, no motion search is performed). Availability of good matches with zero motion vector is highly scene-content dependent; "RaceHorses" has significantly complex camera and object motion at a smaller frame rate as compared to "BasketballDrill", resulting in lack of good zero motion vector matches for the former. Also, as noted in Section 6.4.2, a separation between frame and macroblock-level resource optimization also costs CCAM in terms of performance.

It can also be observed from Figure 6.16(b) that there is an interdependence between bitrate and computational resource usage for the case of CCAM; it converges to its best performance more quickly for smaller bitrates. This can be expected; higher bitrate for video coding results in a larger amount of residual data and more blocks coded than being skipped. A similar, yet not so profound effect is also observed from Figure 6.16(a).

FCC shows no performance gain in any of the presented scenario. For class A, this is as expected, since the training set used to select the performance point (as discussed in Section 6.4.1) also consists of class B test sequences, which require significantly more computational resources. However, there is no performance improvement event in the case of "RaceHorses" which belongs to class B. This is so because for high quality video encoding such as the target system, the peak to average computational resource demand for a video sequence is quite high, and it is the peak computational resource demand that decides the actual operating point for this technique. The peak resource demand in this case is clearly beyond 8 Gc shown in this figures, resulting in no observed performance improvement.

## 6.5 Offline Optimizations

In this section we present the performance optimization results for the offline optimizations as discussed in Section 5.5. As already discussed in Section 5.5, receiver-end offline optimizations are the most relevant for several practical applications. Hence in this section we present a few selected results for receiver-end GOP-based adaptation for computational resource management. First, the reference algorithms employed for evaluation are described.

### 6.5.1 Reference System

We have used two types of reference systems to compare to the proposed technique in Section 5.5.1. The latter technique is abbreviated here as GA.

#### Adaptation Based on Training Set:

This represents an approach where the adaptation is performed based on a representative set of training sequence $\mathfrak{X}$, where $\mathfrak{X} \supseteq \mathcal{X}$. All the sequences have the same number of representations made available. The representations are sorted and indexed in ascending order of coding bitrate. For this reference system the selected representation is governed by:

$$\mathbf{x}^{max} = \arg\arg\min_{x \in \mathfrak{X}} \left( \sum_{\Delta t_x} \mathcal{C}_t^x - \mathcal{H} \cdot \Delta t_x - \mathcal{C}^{max} \right) \Bigg|_{\sum_{\Delta t_x} \mathcal{C}_t^x - \mathcal{H} \cdot \Delta t_x \leq \mathcal{C}^{max}}, \tag{6.4}$$

where $\Delta t_x$ is the duration of the representation $x$ with an index $\mathbf{x}$. The first arg function in Equation 6.4 returns the index $\mathbf{x}$ of the representation $x$. Hence $\mathbf{x}^{max}$ represents the *largest* index for any sequence within $\mathfrak{X}$ that can be selected by the outer optimization algorithm (e.g. the bitrate adaptation algorithm).

It is evident that such "perfect-knowledge" is not possible in general at client side for practical applications. Still this is reference is of theoretical interest as it reflects the upper performance bounds for "design-time" optimizations, i.e. optimization performed once for the lifetime of an implementation. This technique is abbreviated as TA.

#### Sequence-based Adaptation

For this technique, the representation is selected as

$$x = \arg\min_{x \in \mathcal{X}} \left( \sum_{\Delta t_x} \mathcal{C}_t^x - \mathcal{H} \cdot \Delta t_x - \mathcal{C}^{max} \right) \Bigg|_{\sum_{\Delta t_x} \mathcal{C}_t^x - \mathcal{H} \cdot \Delta t_x \leq \mathcal{C}^{max}}. \tag{6.5}$$

Hence the difference between this technique and the proposed technique GA is that instead of a finer-scale GOP-based adaptation, the adaptation is done based on the entire duration of a sequence $\Delta t_x$. The results of this technique are of interest since they will indicate the cost in terms of performance of not adapting to the varying computational resource demands within a single content. This technique will be abbreviated as SA.

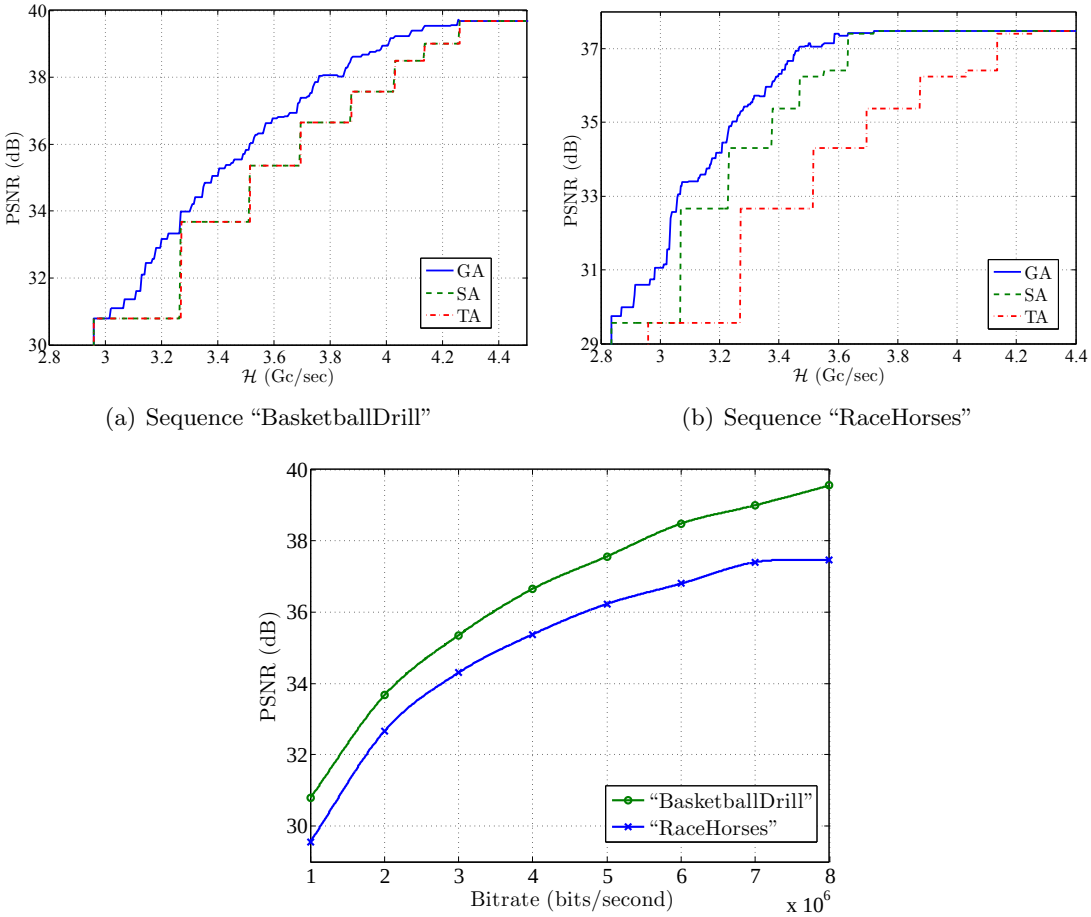
### 6.5.2 Selected Performance Results

To assess the performance of the reference technique in comparison with the proposed technique GA, we have used the same high-quality test content as in Section 6.4.3. The results for "BasketballDrill" and "RaceHorses" will be presented here as an example. Figure 6.17(c) show the RD statistics of both the sequences, with each point representing a unique representation. It can be seen that "RaceHorses" represents a much more complex content to encode than "BasketballDrill", since a lower PSNR is achieved for a similar bitrate by the former, even though it has a lower framerate.

As can be seen from Figure 6.17(a), 6.17(b), that $\mathbf{x}^{max}$ for TA is being determined by "BasketballDrill" for all the representations. This could be expected, since it has a 40% higher frame rate than that of "RaceHorses" and hence although it might have lower $\Delta\mathcal{C}_p^x$ in general, $\sum_{\Delta t_x} \mathcal{C}_t^x$ is still larger because of a larger number of frames to be encoded in a given time interval $\Delta t_x$. As a result, both SA and TA are showing the same performance results in Figure 6.17(a). However, the performance toll of TA becomes apparent in Figure 6.17(b) where it shows significantly lower performance for most computational resource configurations. This is because as expected: different video sequences have a vastly varying usage of resource, and tuning the system for the worst-case scenario (in this case for "BasketballDrill") is a pessimistic approach resulting in significant performance toll for sequences with lower resource demand (as in this case for "RaceHorses"). Figure 6.17(b) shows TA lagging at least couple of dBs behind SA.

The effect of fine-grain adaptation of GA is already evident from Figure 6.17(a) and 6.17(b); while TA and SA appear to consolidate their performance in large steps, GA seems to increase its performance much more smoothly. This effect is caused by the variation of resource usage for coding a sequence. SA has to cater for the worst-case computational resource demand within the entire sequence, resulting in suboptimal performance for rest of the sequence, while GA can switch up to a more complex representation in those latter parts. This gives a typical performance enhancement of 1.5 dB to GA for most resource configurations. Hence the overall best performance is achieved by GA for the presented results as well as the other sequences evaluated during this work.

One anomaly observable from the results of GA is a that the performance increase of this algorithm with increasing computational resource is not monotonic, and an occasional loss of performance is observable at some points of the curves. This is because the algorithm of GA (as depicted in Equation 5.17) is greedy in nature. It happens at some instances that switching up to a better representation for a given




(a) Sequence "BasketballDrill"

(b) Sequence "RaceHorses"

(c) RD characteristics

Figure 6.17: Comparative D-C performance of various resource optimization techniques for streaming along with RD characteristics of representations.

GOP might not be optimal globally for the whole duration of the sequence, as it translates into switching down in another following GOP. In some cases the increase of PSNR achieved by switching up cannot compensate for a greater PSNR reduction for the later switching down since this is scene-content dependent, and this explains a slight reduction in the average PSNR observed at some points on the curve.

# Chapter 7

# Conclusion and Outlook

This work is a step in the direction of defining a framework for computational resource management for video communication systems. This framework is essentially missing in comparison to the well-known rate-management framework (the realm of rate-distortion optimization). One aspect of this endeavour is to employ a pyramid-shaped approach for the proposals: the design principles applicable to a wide range of video communication systems make the foundation, while the performance is evaluated for a selected few systems at the apex. This ensures that the approach will find relevance for other similar systems, e.g. a newer video codec, or a newer video communication application, etc.

After the initial background literature review, a detailed insight is provided into robust system design strategies and tools as well as quantitative evaluation techniques for the target resource constrained video communication systems, namely: mobile video conversational applications and telepresence/teleaction systems. For the former system, the main challenge of robust video communication over a lossy channel in conjunction with real-time, low-delay and complexity constraints is tackled. Feedback-based error mitigation techniques have been proposed that result in an extremely robust system design without adding on complexity and delay constraints. Analysis of conventional robustness tools such as slice-based coding is also presented for this system. The proposed system exhibits almost linear PSNR loss with increasing channel loss rate, enabling to proceed further on with resource optimizations.

In relation to telepresence/teleaction systems, quantitative criteria is defined to assess the impact of video compression artefacts on the performance of the system. Specifically, the adverse effects of video compression on the performance of various stereo-matching algorithms is analyzed. This enables identification of the most robust system configuration defined by the optimal coding bitrate, video codec and stereo-matching algorithm, given a maximum allowed value of PBP.

Since video communication applications come in a wide variety of variants, a categorization is defined in this work in terms of codec topology and the type of adaptation. Hence optimization techniques are proposed based on these categories. The two aforementioned video communication systems serve as an example where the

objective performance of the proposed techniques is demonstrated.

The first step in proposing computational resource optimization algorithms is to provide detailed system-wide timing analysis, since the traditional VBV timing analysis is not valid for the target system. This timing bounds defined by this timing analysis form the basis of the optimization algorithms defined later.

A crucial cornerstone in resource optimization is the prediction of computational resource usage at a given terminal with reasonable complexity and accuracy. An accurate, compact, and low complexity resource usage model is developed for this purpose. This enables deploying easily integrable resource optimization algorithms in a wide range of applications, especially online optimized applications. The model has a compact representation and hence a low communication overhead.

The first optimization framework is targeted for online encoding systems. The optimization is performed at the encoding-end. It is established that instead of the traditional RDC approach, CD mode-ranking is more feasible because of underlying differences between RD and CD optimization. A robust CD mode-ranking is specified for optimization in conjunction with well-established RD optimization. A complexity quantization mechanism is defined that enables accurate resource usage configuration when used in conjunction with the specified quantizer control mechanism. These design principles are further specified for the selected video codec (H.264/AVC).

The second framework is targeted for offline optimizations such as video streaming applications. A GOP-based optimization strategy is defined that can either be employed at the sending-end or at the receiving-end. The pros and cons in terms of the signaling overhead and scalability are analyzed for each alternative..

Finally, quantitative effect on performance enhancement achieved by the above proposed framework is presented for 3GPP PSC applications, TPTA systems and an adaptive streaming application. Suitable reference systems are also specified for this comparison. This comparison is based on the performance metrics analyzed in the preceding sections.

As an outlook, dynamic mode-ranking can be integrated into the framework to adapt to the scene content in semi-stationary manner. The reduction of the communication overhead for offline optimizations at the receiving-end has a strong potential for further optimizations. Also the specified frameworks have to be extended to the new codecs on the horizon such as the HEVC. In addition to newer codecs, new emerging application scenarios (like the ongoing work on Modern Media Transport within MPEG) should also be considered for the applicability of resource optimization framework.

In terms of computational resource modeling, the task of defining the VCV and the implementation-independent model should be carried out at the SDO which specifies a video codec, since such a model is strictly codec dependent and demands in-depth understanding of the coding techniques employed.

# Appendix A

# Evaluation Framework

In this annex we describe the simulation environment setup used to generate the results. Mainly two types of systems were emulated: mobile video communication systems, described in Section A.1, and telepresence/teleaction (TPTA) systems described in Section A.2.

## A.1 3GPP conversational application

As specification work for video codecs is mainly carried out within ISO motion picture expert group (MPEG) and ITU-T video coding expert group (VCEG), these standardization bodies specify the operation of a video decoder for an error-free bitstream only. Since video codecs such as H.264/AVC are designed to cater for a wide range of performance and application scenarios, significant flexibility is provided so that the codec can operate effectively for a variety of scene contents, transmission environments, and service constraints.

This flexibility poses a huge challenge for mobile environments. In order to ensure proper functioning and compatibility, the video codecs have to be rigorously assessed for these systems. This involves definition of performance requirements for various services offered by the system in terms of some well defined metrics. A simulation system that can sufficiently represent a broad range of service scenarios is necessary to accomplish this task. Definition of such a system is a significant undertaking in itself because of the challenges discussed in the rest of this section. Hence the simulation environment proposed for this purpose in this work is aligned to efforts in 3GPP SA4 video adhoc group (VAG).

The use of the proposed environment is in no way limited to just giving insight into the performance of existing systems. This software is an indispensable requirement to access original ideas and techniques proposed to enhance the system performance for the future. The components and tools required for realizing the simulation framework are described in Section A.1.1. Two different approaches for a simulation framework are introduced and discussed in Section A.1.2, which are suited for different applications.

### A.1.1 Simulation Environment Components

A complete mobile video communication system consists of a large number of components. For a simulation and testing environment, different modules need to be provided to properly assess the service. Note that a real multimedia transmission system involves a significant amount of further components, but for the sake of proper assessment we abstract such services. The description of tools here is aligned to the 3GPP SA4 VAG data base [3] which collects tools necessary for video performance assessment.

#### Video Test Sequences and Formats

**Test Sequences:** In order to access the performance of video codecs, a suitable set of video test sequences has to be collected. The technical body SA-WG4 is responsible for specification of multimedia codecs within 3GPP. As a result of some activities within the group [3], a suitable set of video sequences has been selected for performance evaluation. These sequences are in YUV 4:2:0 format with either quarter-VGA or quarter common intermediate format (QCIF). The details about the test content can be found in [3].

**Video Codec:** A software implementation of video codec to be accessed is required to complete the simulation framework. H.264/AVC is the latest and most suitable choice for mobile environments because of its high compression efficiency and error resilience features. An open source implementation of H.264/AVC codec is for example available for simulation purposes [144] referred to as joint video model (JM). Before the video source is fed to the encoder, pre-processing might be appropriate in terms of temporal and/or spatial sub sampling. The encoder has to be configured according to the service requirements, and this is done via combination of a configuration file and command line arguments. The operation is file I/O based; all inputs are read from files, and outputs are written to files. For most real-time applications of interest in mobile environments, it is required that the output of source coder is encapsulated in communication protocol, most prominently in a real-time protocol (RTP) [153] before it could be transmitted. Like wise the reverse operation has to be performed before the source decoding can be done. Packetization modules based on RTP were developed to fulfill this purpose.

**3GPP Transport Module:** The transport of real-time services over mobile networks results in different kind of distortions and effects. For testing such services, a channel transport software implementation is available from 3GPP [154]. This software models the entire communication stack to the extent being necessary for the testing. In addition, it supports most 3GPP multimedia service bearers. A service bearer describes the channel parameters e.g. bitrate, delay, loss pattern etc. for a given service. Like the case of the video codec, it is configurable via configuration files and is file I/O based. The random seed initialization for a channel realization is also configurable.

### A.1.2 Simulation And Testing Environment

Based on the components described in previous section, two versions of the required simulation environment are proposed here. These are described in detail in this section along with their pros and cons.

#### File-I/O based Environment

Commonly, tools as for example discussed in Section A.1.1, have file I/O interfaces. Based on these tools a simulation environment employing simple file interfaces is introduced in the following. This simulation and testing environment is depicted in Figure A.1.

**Design Considerations:** Several video applications are entirely unidirectional (e.g. data flowing from the transmitter to the receiver). In this case the modules can be connected with minimum changes to their existing interfaces and internal architecture. The connected configuration of modules in this system is shown in Figure A.1. As an example, in order to connect, run and control all the modules, UNIX shell scripts are useful, which sequentially invoke the different modules. In this system, a file is generated by each processing unit, which is numbered in Figure A.1. The timing information is also embedded in this file.

**Operation for Single Test Run:** For a single run of the software, following steps are performed:

- For an individual test file numbered "1", determined by the source parameter, a pre-processed file "2" is fed to the encoder.

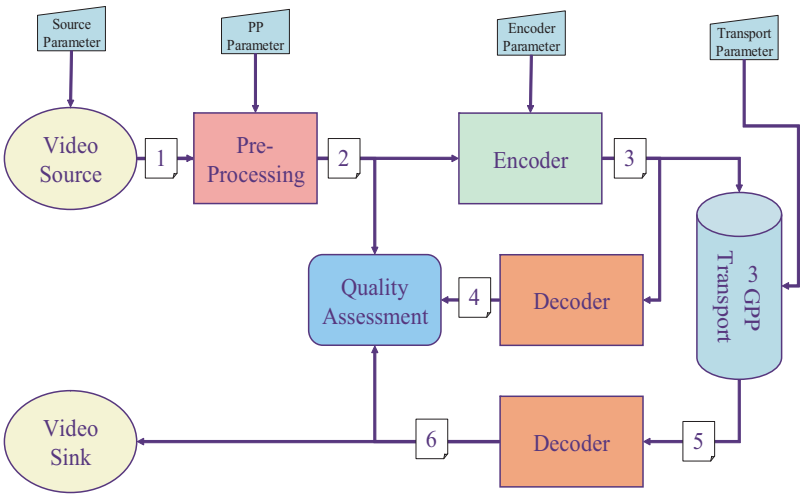



Figure A.1: File I/O Simulation Environment

- The encoder generates the compressed RTP according to the encoding parameters. Simulation time information is also embedded in the file by the encoder. The resultant output file is marked as "3".
- The RTP file with embedded time information "3" is then processed by the transport software according to the transport parameters. A "simulated time" is maintained by the module during this processing. The embedded timing information of the input file is used to determine if a packet should be processed at a given simulated time. Delays and losses are introduced to the data during the processing. The delay information is communicated to the decoder by updating the embedded timing information of the file "5".
- Following this, the decoder takes up the received file "5", and decodes it after de-packetization. The decoded video is given by file "6".
- After a complete file has been decoded, the quality assessment module takes up the original video file "2", the reconstructed video "4" and the decoded video "6" to generate the resulting quality metrics for this run.

For statistical significance, in general multiple runs need to be carried out. This is accomplished by applying different random seeds in the channel simulator software from 1,..., *N*, each seed results in an individual run.

**Principles and Problems:** This simulation approach is sufficient in case of unidirectional transmission, e.g. in case of services like MBMS. However, this approach does not provide any room for investigations where a backward information flow is proposed, e.g. in case of conversational applications. The reason is that all the software modules discussed here have internal states. Consider the principle in Figure A.2. For example, if module 2 changes the state, then module 1 might want to or has to change its state accordingly to optimize the system performance. This is not possible for this simulation system since the modules do not interact with each other in the backward direction. This prevents proper simulation and assessment of techniques that incorporate feedback messages. Therefore, in the following section, we propose a new simulation environment, which allows addressing these issues to obtain meaningful results and to draw practical and relevant conclusions for media handling and interaction in multimedia telephony services.

### Time-sliced Multi-process Simulation and Testing Approach

To address the problems identified in the previous section, we propose a time-sliced simulation environment. In what follows we discuss our consideration for developing this environment:

**Design Considerations:** I/O interfaces of all modules that are expected to change their state based on reverse information flow are modified. These modules are the encoder, the decoder, and the channel module, and will be referred to as interactive

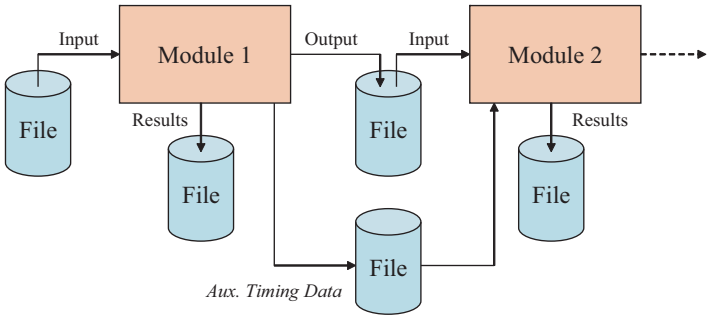



Figure A.2: File-based Simulation Approach

modules. Since we expect a backward information flow, an additional channel module in the backward direction is incorporated. Instead of the simple file I/O, the modules should communicate with each other by message queues, so that information generated by one module can be made available to another module immediately, where the latter can act on this. Most operating systems have support for message queues, and all have their own considerations. All interactive modules have to be trigger-able and interruptible, so that their execution could be halted to wait for the processing of other modules. Most operating systems offer either processes and/or threads to accomplish this. The performance overheads of processes are much more than that of threads in terms of switching. However using processes results in lesser changes to individual existing modules, hence UNIX processes were selected for implementation along with inter-process communication queues for data and message transfer.

**Operation of Testing and Simulation System:** The operation and data flow for the simulation system is shown in Figure A.3. Compared to the file-based approach as shown in Figure A.1, media encoder, forward channel, media decoder, and backward channel are controlled by a central controller or *executive* with the main task of properly scheduling the operations in the simulation environment using the time-sliced simulation architecture. Pre-processing and quality evaluation is still done offline, as for example user interaction with the played-out content is not assumed. Note also that simulation is not necessarily real-time, which is of significant advantage as it might allow to speed up simulations or to simulate complex operations which do not run in real-time.

In addition to file interfaces, the simulation system also includes data queues and control queues which can be implemented by UNIX IPC queues. The following basic operations are carried out

1. The video or media source is available in a file (indicated by interface “1”) which might be pre-processed, e.g. sub-sampled, to obtain a pre-processed file “2”.

2. The controller advances the time in step size of $\Delta t_s$ milliseconds. Typically

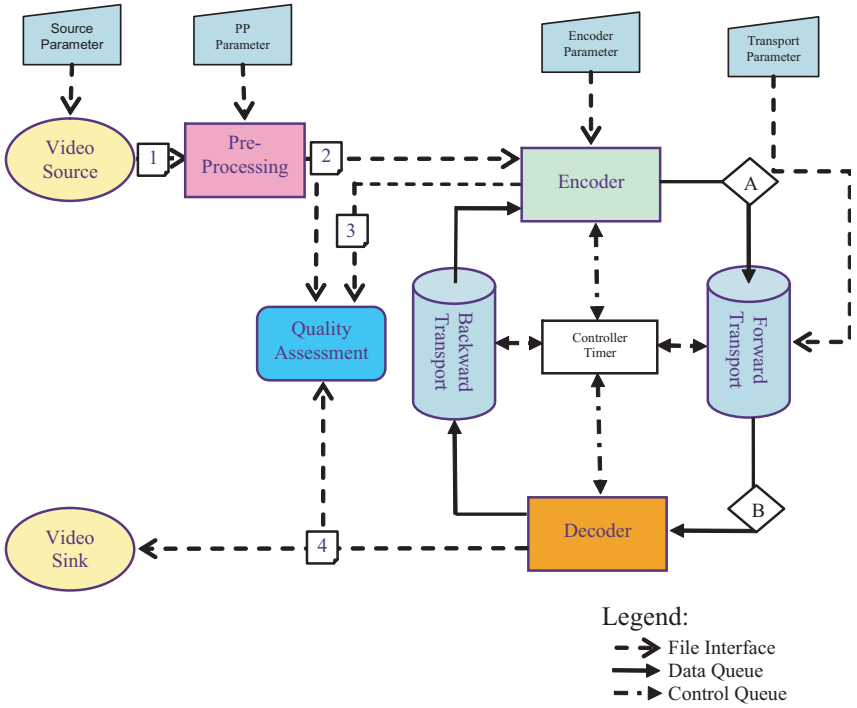



Figure A.3: Proposed multi-process simulation environment

the simulation starting time is zero millisecond. At a given discrete time step

$$t_s = K \cdot \Delta t_s, K = 0, 1, 2, ... \tag{A.1}$$

the controller communicates the time $t_s$ to the encoder. The processing of data by all the interactive modules is simulated to be executed instantaneously. The encoder executes all necessary processing which is due until the time $t_s$, e.g. it takes video frame up to time $t_s$ from file "2", compresses it and outputs all necessary information in data queue "A". It also writes the encoded and reconstructed media samples, i.e. the video frame, with appropriate timing information to file "3". Finally, it sends an acknowledgement (ACK) to the controller for completion of its tasks and becomes inactive. In the case when there is no input data at a given time $t_s$ or no output needs to be generated, the module skips the processing and sends the ACK immediately. This ACK is necessary for proper processing of the next module, as in the simulation system, the encoder might take a variable amount of time for its processing.

3. Then, the controller triggers the forward channel with the same time stamp $t_s$. The channel checks the input data queue and processes all necessary information and outputs data, if any data is available for the output data queue "B". Finally, it sends an ACK to the controller for completion of its tasks and becomes inactive.

4. Likewise the controller sequentially (after receiving an ACK for completion from the forward channel) triggers media decoder and backward channel with the same time $t_s$. The media decoder writes all decoded media samples, e.g.

video frames, with appropriate timing information to file “4”.

5. After receiving the ACK of the backward channel, all processing until time $t_s$ for the entire simulation chain has been done. Hence the controller increments the simulated time by the selected time increment $\Delta t_s$.

6. As soon as one of the modules decides it has finished its processing, it sends a completion message to the controller. The controller takes care not to trigger that module again. Also, this module sends a completion message to all the modules receiving data from it.

7. The round robin scheduling of encoder, forward channel, decoder and backward channel continues until the controller has received a completion signal from all modules. When this has happened, it terminates the current simulation run.

8. The quality assessment is then done offline based on files “2”, “3”, and ”4”.

## A.2 TPTA Evaluation Methodology and Framework

To evaluate the performance of stereo matching schemes in TPTA, we propose the evaluation framework shown in Figure A.4. The proposed framework simulates the integration of video coding and stereo matching in a TPTA scenario. The individual components of this framework along with their interactions are explained in the sequel. We will conduct the study on stereo videos since the results can easily be extended for more than two views.

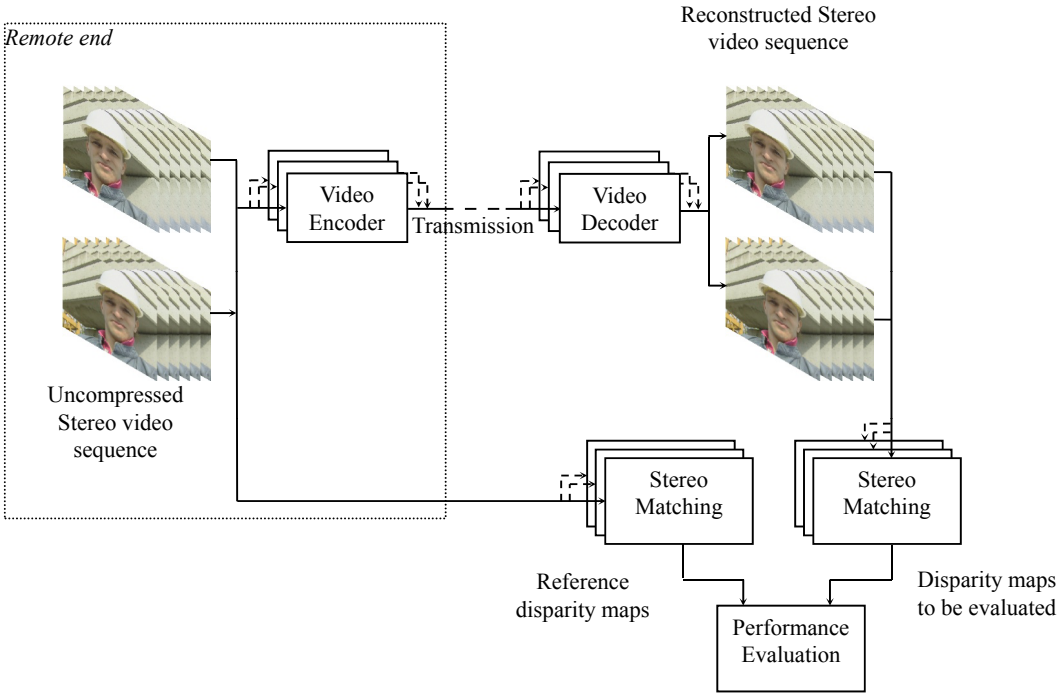



Figure A.4: Block diagram of the evaluation framework.

### A.2.1 Performance Evaluation

At the encoder (shown in Figure A.4) the stereo sequences are rectified using the camera parameters. The sequences are then converted to YUV 4:2:0 while the image size must be a multiple of 16 pixels both in height and width. Therefore, the color conversions are done before coding and if required, the dimensions are matched by padding the image with grey values to avoid generating redundant data. The resulting sequences are then fed to the selected video codec. The encoding quality is determined by the allowed video bitrate on the channel; therefore, each of the left and right video sequence is allowed half of the video bitrate to enable homogenous video quality.

At the receiving end, the compressed video streams are decoded to get the reconstructed video that will be used for stereo matching. Following this, the performance evaluation is done using the metrics discussed above. Since the target of this study is to asses exclusively the impact of video compression on the performance of the stereo matching, the reference disparity map for performance evaluation is the disparity map generated with no video compression applied, using the same stereo matching technique.

# Bibliography


[1] M. Meeker *et al.*, "The mobile internet report," Morgan Stanley Research, Morgan Stanley & Co. Incorporated, Tech. Rep., 2009. [Online]. Available: http://www.morganstanley.com/institutional/techresearch/

[2] "Cisco visual networking index: Global mobile data traffic forecast update, 2010-2015," Cisco Systems, Inc, Tech. Rep., Feb. 2011. [Online]. Available: http://www.cisco.com/en/US/solutions/collateral/ns341/ns525/ns537/ns705/ns827/white_paper_c11-520862.html

[3] "Technical specification group services and system aspects; video codec performance," 3GPP, 3GPP Technical Specification TS 26.902, Jun. 2007.

[4] "Advanced video coding for generic audiovisual services," ITU–T Recommendation H.264 – ISO/IEC 14496-10 (AVC), 2010.

[5] *Cortex$^{TM}$-A9 Processor*, ARM®, 2011. [Online]. Available: http://www.arm.com/products/processors/cortex-a/cortex-a9.php

[6] G. Sullivan and J. Ohm, "Recent developments in standardization of high efficiency video coding (HEVC)," *SPIE Applications of Digital Image Processing XXXIII, Proc. SPIE*, vol. 7798, 2010.

[7] J. Hicks. (2011, May) Samsung's foldable AMOLED display: no creases, even after 100,000 tries. [Online]. Available: http://www.engadget.com/2011/05/15/samsungs-foldable-amoled-display-no-creases-even-after-100-00/

[8] *Information technology - Dynamic adaptive streaming over HTTP (DASH) - Part 1: Media presentation description and segment formats*, ISO/IEC JTC 1/SC 29, 2011.

[9] R. Pantos, "HTTP live streaming," Internet Engineering Task Force (IETF), Internet draft, 2011. [Online]. Available: http://tools.ietf.org/html/draft-pantos-http-live-streaming-07

[10] J. Bankoski, "Intro to WebM," in *Proceedings of the 21st international workshop on Network and operating systems support for digital audio and video*. ACM, 2011, pp. 1–2.

[11] J. Bankoski, P. Wilkins, and Y. Xu, "Technical overview of VP8, an open source video codec for the web." [Online]. Available: http://research.google.com/pubs/archive/37073.pdf

[12] T. Wiegand, H. Schwarz, A. Joch, F. Kossentini, and G. Sullivan, "Rate-constrained coder control and comparison of video coding standards," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 13, no. 7, pp. 688–703, 2003.

[13] "Information technology – coding of audio-visual objects – part 2: Visual," ITU–T Recommendation ISO/IEC 14496-2:2004, 2004.

[14] B. Girod, A. Aaron, S. Rane, and D. Rebollo-Monedero, "Distributed video coding," *Proceedings of the IEEE*, vol. 93, no. 1, pp. 71–83, 2005.

[15] C. Brites, "Advances on distributed video coding," Ph.D. dissertation, Universidade Técnica De Lisboa, 2005.

[16] X. Artigas, J. Ascenso, M. Dalai, S. Klomp, D. Kubasov, and M. Ouaret, "The DISCOVER codec: architecture, techniques and evaluation," in *Picture Coding Symposium*, 2007.

[17] ——. DISCOVER distributed video codec. [Online]. Available: http://www.discoverdvc.org/

[18] J. Pedro, C. Brites, J. Ascenso, and F. Pereira, "Studying the feedback channel in transform domain wyner-ziv video coding," in *Sixth Conference on Telecommunications, Peniche, Portugal*, 2007.

[19] G. Sullivan and J.-R. Ohm, "Meeting report of the sixth meeting of the joint collaborative team on video coding (JCT-VC)," Tech. Rep. JCTVC-F_Notes_dB, 2011. [Online]. Available: http://wftp3.itu.int/av-arch/jctvc-site/2011_07_F_Torino/

[20] P. Chou and M. van der Schaar, *Multimedia over IP and wireless networks: compression, networking, and systems*. Academic Press, 2007.

[21] T. Wiegand, G. Sullivan, G. Bjontegaard, and A. Luthra, "Overview of the H.264/AVC video coding standard," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 13, no. 7, pp. 560–576, July 2003.

[22] G. Sullivan and T. Wiegand, "Video compression-from concepts to the H. 264/AVC standard," *Proceedings of the IEEE*, vol. 93, no. 1, pp. 18–31, 2005.

[23] T. Wiegand, N. Färber, K. Stuhlmüller, and B. Girod, "Error–resilient video transmission using long-term memory motion–compensated prediction," *IEEE Journal on Selected Areas in Communications*, vol. 18, no. 6, pp. 1050–1062, Jun. 2000.

[24] T. Wiegand and B. Girod, *Multi-Frame Motion-Compensated Prediction for Video Transmission*. KLADR: KL, 2001.

[25] T. Wedi and H. Musmann, "Motion-and aliasing-compensated prediction for hybrid video coding," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 13, no. 7, pp. 577–586, 2003.

[26] T. Wedi, "Hybrid video coding based on high-resolution displacement vectors," in *Proceedings of SPIE*, vol. 4310, 2000, p. 186.

[27] W. Zia and F. Shafait, "Reduced complexity techniques for long-term memory motion compensated prediction in hybrid video coding," in *Proceedings Picture Coding Symposium*, Beijing, China, Apr. 2006.

[28] *SAA7118*, Trident Microsystems, 2010. [Online]. Available: http://www.tridentmicro.com/producttree/tv/pc-tv/saa/saa7118/

[29] *ADV7314: Multiformat 216 MHz Video Encoder with Six NSV$^{TM}$ 14-Bit DACs*, Analog Devices, 2011. [Online]. Available: http://www.analog.com/en/digital-to-analog-converters/video-encoders/adv7314/products/product.html

[30] T. Wiegand, W.-J. Han, B. Bross, J.-R. Ohm, and G. J. Sullivan, "WD3: Working draft 3 of high-efficiency video coding," Tech. Rep. JCTVC-E603, 2011. [Online]. Available: http://phenix.it-sudparis.eu/jct/doc_end_user/current_document.php?id=2471

[31] S.-i. S. W.-J. H. Ken McCann, Benjamin Bross, "HM3: High efficiency video coding (HEVC) test model 3 encoder description," Tech. Rep. JCTVC-E602, 2011. [Online]. Available: http://phenix.it-sudparis.eu/jct/doc_end_user/current_document.php?id=2470

[32] F. De Simone, L. Goldmann, J. Lee, and T. Ebrahimi, "Performance analysis of VP8 image and video compression based on subjective evaluations," in *Proceedings of SPIE*, vol. 8135, 2011, p. 81350M.

[33] T. Stockhammer and W. Zia, *Multimedia over IP and wireless networks: compression, networking, and systems*. Academic Press, 2007, ch. Error-Resilient Coding And Decoding Strategies, pp. 13–58.

[34] P. Salama, N. Shroff, E. Coyle, and E. Delp, "Error concealment techniques for encoded video streams," in *Proceedings IEEE International Conference on Image Processing*, vol. 1, Washington DC, USA, Oct. 1995, pp. 9–12.

[35] W. Zeng and B. Liu, "Geometric-structure-based error concealment with novel applications in block-based low-bit-rate coding," *IEEE Trans. on Circuits Syst. Video Technol.*, vol. 9, no. 4, pp. 648–665, Jun. 1999.

[36] G. Bjøntegaard, "Definition of an error concealment model TCON," Boston, USA, Doc. ITU-T/SG15/LBC-95-186, Jun. 1995.

[37] Y.-K. Wang, M. Hannuksela, V. Varsa, A. Hourunranta, and M. Gabbouj, "The error concealment feature in the H.26L test model," in *Proceedings IEEE International Conference on Image Processing*, vol. 2, Rochester(NY), USA, Sep. 2002, pp. 729–732.

[38] L. Qian, D. Jones, K. Ramchandran, and S. Appadwedula, "A general joint source-channel matching method for wireless video transmission," in *Data Compression Conference, 1999. Proceedings. DCC '99*, Mar. 1999, pp. 414 –423.

[39] Y. Eisenberg, C. Luna, T. Pappas, R. Berry, and A. Katsaggelos, "Joint source coding and transmission power management for energy efficient wireless video

communications," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 12, no. 6, pp. 411 –424, Jun. 2002.

[40] C. Luna, Y. Eisenberg, R. Berry, T. Pappas, and A. Katsaggelos, "Joint source coding and data rate adaptation for energy efficient wireless video streaming," *Selected Areas in Communications, IEEE Journal on*, vol. 21, no. 10, pp. 1710 – 1720, 2003.

[41] P. Agrawal, J.-C. Chen, S. Kishore, P. Ramanathan, and K. Sivalingam, "Battery power sensitive video processing in wireless networks," in *Personal, Indoor and Mobile Radio Communications, 1998. The Ninth IEEE International Symposium on*, vol. 1, Sep. 1998, pp. 116 –120 vol.1.

[42] D. Li, Y. Sun, and Z. Feng, "Joint power allocation and rate control for real-time video transmission over wireless systems," in *Global Telecommunications Conference, 2005. GLOBECOM '05. IEEE*, vol. 4, 2005, pp. 5 pp. –2168.

[43] Q. Zhang, Z. Ji, W. Zhu, and Y.-Q. Zhang, "Power-minimized bit allocation for video communication over wireless channels," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 12, no. 6, pp. 398 –410, Jun. 2002.

[44] Z. Ji, Q. Zhang, W. Zhu, J. Lu, and Y.-Q. Zhang, "Joint power control and source-channel coding for video communication over wireless networks," in *Vehicular Technology Conference, 2001. VTC 2001 Fall. IEEE VTS 54th*, 2001.

[45] Z. Ji, Q. Zhang, W. Zhu, and Y.-Q. Zhang, "End-to-end power-optimized video communication over wireless channels," in *Multimedia Signal Processing, 2001 IEEE Fourth Workshop on*, 2001.

[46] T.-H. Lan and A. Tewfik, "Power optimized mode selection for h.263 video coding and wireless communications," in *Image Processing, 1998. ICIP 98. Proceedings. 1998 International Conference on*, vol. 2, Oct. 1998, pp. 113 –117 vol.2.

[47] X. Lu, Y. Wang, and E. Erkip, "Power efficient h.263 video transmission over wireless channels," in *Image Processing. 2002. Proceedings. 2002 International Conference on*, 2002.

[48] X. Lu, E. Erkip, Y. Wang, and D. Goodman, "Power efficient multimedia communication over wireless channels," *Selected Areas in Communications, IEEE Journal on*, vol. 21, no. 10, pp. 1738 – 1751, 2003.

[49] X. Lu, T. Fernaine, and Y. Wang, "Modelling power consumption of a h.263 video encoder," in *Circuits and Systems, 2004. ISCAS '04. Proceedings of the 2004 International Symposium on*, vol. 2, May 2004, pp. II – 77–80 Vol.2.

[50] X. Lu, Y. Wang, and E. Erkip, "Power efficient h.263 video transmission over wireless channels," in *Image Processing. 2002. Proceedings. 2002 International Conference on*, 2002.

[51] E. Erkip, Y. Wang, D. Goodman, Y. Wu, and X. Lu, "Energy efficient coding and transmission," in *Vehicular Technology Conference, 2001. VTC 2001 Spring. IEEE VTS 53rd*, 2001.

[52] W. Pu, Y. Lu, and F. Wu, "Joint power-distortion optimization on devices with MPEG-4 AVC/H.264 codec," in *Communications, 2006. ICC '06. IEEE International Conference on*, vol. 1, 2006, pp. 441 –446.

[53] Z. He, Y. Liang, L. Chen, I. Ahmad, and D. Wu, "Power-rate-distortion analysis for wireless video communication under energy constraints," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 15, no. 5, pp. 645 – 658, May 2005.

[54] W. Cheng, X. Chen, and Z. He, "Doubling of the operational lifetime of portable video communication devices using power-rate-distortion analysis and control," in *Image Processing, 2006 IEEE International Conference on*, 2006, pp. 2473 –2476.

[55] Z. He and D. Wu, "Resource allocation and performance analysis of wireless video sensors," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 16, no. 5, pp. 590 – 599, May 2006.

[56] Y. Liang, I. Ahmad, and J. Luo, "Joint power and distortion control in video coding," A. Said and J. G. Apostolopoulos, Eds., vol. 5685, no. 1. SPIE, 2005, pp. 885–895. [Online]. Available: http://link.aip.org/link/?PSI/5685/885/1

[57] Z. He, W. Cheng, and X. Chen, "Energy minimization of portable video communication devices based on power-rate-distortion optimization," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 18, no. 5, pp. 596 –608, May 2008.

[58] C. Shannon, "A mathematical theory of communication," *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 5, no. 1, pp. 3–55, 2001.

[59] R. Vanam, E. Riskin, S. Hemami, and R. Ladner, "Distortion-complexity optimization of the H.264/MPEG-4 AVC encoder using the GBFOS algorithm," in *Data Compression Conference, 2007. DCC '07*, 2007, pp. 303 –312.

[60] E. Kaminsky, D. Grois, and O. Hadar, "Dynamic computational complexity and bit allocation for optimizing H.264/AVC video compression," in *Information Technology: Research and Education, 2006. ITRE '06. International Conference on*, 2006, pp. 167 –171.

[61] D. Grois, E. Kaminsky, and O. Hadar, "Buffer control in H.264/AVC applications by implementing dynamic complexity-rate-distortion analysis," in *Broadband Multimedia Systems and Broadcasting, 2009. BMSB '09. IEEE International Symposium on*, May 2009, pp. 1 –7.

[62] T. da Fonseca and R. de Queiroz, "Complexity-constrained h.264 HD video coding through mode ranking," in *Picture Coding Symposium, 2009. PCS 2009*, May 2009, pp. 1 –4.

[63] ——, "Complexity-constrained rate-distortion optimization for h.264/AVC video coding," in *Circuits and Systems (ISCAS), 2011 IEEE International Symposium on*. IEEE, pp. 2909–2912.

[64] C. E. Rhee, J.-S. Jung, and H.-J. Lee, "A real-time h.264/AVC encoder with complexity-aware time allocation," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 20, no. 12, pp. 1848 –1862, 2010.

[65] Y. Hu, Q. Li, S. Ma, and C.-C. Jay Kuo, "Joint rate-distortion-complexity optimization for h.264 motion search," in *Multimedia and Expo, 2006 IEEE International Conference on*, 2006, pp. 1949 –1952.

[66] J. Stottrup-Andersen, S. Forchhammer, and S. Aghito, "Rate-distortion-complexity optimization of fast motion estimation in H.264/MPEG-4 AVC," in *Image Processing, 2004. ICIP '04. 2004 International Conference on*, vol. 1, 2004, pp. 111 – 114 Vol. 1.

[67] J. Zhang, Y. He, S. Yang, and Y. Zhong, "Performance and complexity joint optimization for h.264 video coding," in *Circuits and Systems, 2003. ISCAS '03. Proceedings of the 2003 International Symposium on*, vol. 2, May 2003, pp. II–888 – II–891 vol.2.

[68] D. Kwon, P. Agathoklis, and P. Driessen, "Performance and computational complexity optimization in a configurable video coding system," in *Wireless Communications and Networking, 2003. WCNC 2003. 2003 IEEE*, vol. 3, 2003, pp. 2086 –2089 vol.3.

[69] H. Ates, B. Kanberoglu, and Y. Altunbasak, "Rate-distortion and complexity joint optimization for fast motion estimation in h.264 video coding," in *Image Processing, 2006 IEEE International Conference on*, 2006, pp. 37 –40.

[70] G. De Haan and P. Biezen, "An efficient true-motion estimator using candidate vectors from a parametric motion model," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 8, no. 1, pp. 85 –91, Feb. 1998.

[71] I. Richardson and Y. Zhao, "Adaptive algorithms for variable-complexity video coding," in *Image Processing, 2001. Proceedings. 2001 International Conference on*, 2001.

[72] T. da Fonseca, R. de Queiroz, and D. Mukherjee, "Complexity-scalable H.264/AVC in an ipp-based video encoder," in *Image Processing (ICIP), 2010 17th IEEE International Conference on*, 2010, pp. 2885 –2888.

[73] T. da Fonseca and R. de Queiroz, "Macroblock sampling and mode ranking for complexity scalability in mobile h.264 video coding," in *Image Processing (ICIP), 2009 16th IEEE International Conference on*, 2009, pp. 3753 –3756.

[74] M.-C. Chien, Z.-Y. Chen, and P.-C. Chang, "Coding-gain-based complexity control for h.264 video encoder," in *Image Processing, 2008. ICIP 2008. 15th IEEE International Conference on*, Oct. 2008, pp. 2136–2139.

[75] Y. V. Ivanov and C. J. Bleakley, "Dynamic complexity scaling for real-time H.264/AVC video encoding," in *Proceedings of the 15th international conference on Multimedia*, ser. MULTIMEDIA '07. New York, NY, USA: ACM, 2007, pp. 962–970. [Online]. Available: http://doi.acm.org/10.1145/1291233.1291444

[76] ——, "Real-time h.264 video encoding in software with fast mode decision and dynamic complexity control," *ACM Trans. Multimedia Comput. Commun. Appl.*, vol. 6, pp. 5:1–5:21, February 2010. [Online]. Available: http://doi.acm.org/10.1145/1671954.1671959

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

[78] C. Kannangara, I. Richardson, M. Bystrom, and Y. Zhao, "Complexity control of H.264/AVC based on mode-conditional cost probability distributions," *Multimedia, IEEE Transactions on*, vol. 11, no. 3, pp. 433 –442, 2009.

[79] C. Kim, "Complexity adaptation in video encoders for power limited platforms," 2010.

[80] W. Kim, J. You, and J. Jeong, "Complexity control strategy for real-time H.264/AVC encoder," *Consumer Electronics, IEEE Transactions on*, vol. 56, no. 2, pp. 1137 –1143, May 2010.

[81] L. Su, Y. Lu, F. Wu, S. Li, and W. Gao, "Real-time video coding under power constraint based on H. 264 codec," in *SPIE Visual Communications and Image Processing*, vol. 6508, 2007.

[82] ——, "Complexity-constrained h.264 video encoding," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 19, no. 4, pp. 477 –490, 2009.

[83] Z. Zhong and Y. Chen, "Complexity regulation for real-time video encoding," in *Image Processing. 2002. Proceedings. 2002 International Conference on*, 2002.

[84] J. Ribas-Corbera, P. Chou, and S. Regunathan, "A generalized hypothetical reference decoder for H.264/AVC," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 13, no. 7, pp. 674 – 687, 2003.

[85] J. Valentim, P. Nunes, and F. Pereira, "An alternative complexity model for the MPEG-4 video verifier mechanism," in *Image Processing, 2001. Proceedings. 2001 International Conference on*, 2001.

[86] ——, "Evaluating MPEG-4 video decoding complexity for an alternative video complexity verifier model," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 12, no. 11, pp. 1034 – 1044, Nov. 2002.

[87] S. Regunathan, P. Chou, and J. Ribas-Corbera, "A generalized video complexity verifier for flexible decoding," in *Image Processing, 2003. ICIP 2003. Proceedings. 2003 International Conference on*, vol. 3, 2003, pp. III – 289–92 vol.2.

[88] M. Horowitz, A. Joch, F. Kossentini, and A. Hallapuro, "H.264/AVC baseline profile decoder complexity analysis," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 13, no. 7, pp. 704 – 716, 2003.

[89] M. Holliman and Y. Chen, "MPEG decoding workload characterization," in *Proc. of Workshop on Computer Architecture Evaluation using Commercial Workloads*, 2003, pp. 23–34.

[90] Z. Ma and Y. Wang, "Complexity modeling of scalable video decoding," in *Acoustics, Speech and Signal Processing, 2008. ICASSP 2008. IEEE International Conference on*, 31 2008.

[91] S.-W. Lee and C.-C. Kuo, "Complexity modeling for motion compensation in H.264/AVC decoder," in *Image Processing, 2007. ICIP 2007. IEEE International Conference on*, 16 2007.

[92] ——, "Complexity modeling of H.264/AVC CAVLC/UVLC entropy decoders," in *Circuits and Systems, 2008. ISCAS 2008. IEEE International Symposium on*, May 2008, pp. 1616 –1619.

[93] ——, "Complexity modeling of spatial and temporal compensations in H.264/AVC decoding," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 20, no. 5, pp. 706 –720, May 2010.

[94] S.-W. Lee and C.-C. J. Kuo, "H.264/AVC entropy decoder complexity analysis and its applications," *Journal of Visual Communication and Image Representation*, vol. In Press, Corrected Proof, 2010.

[95] T. Lan, Y. Chen, and Z. Zhong, "MPEG2 decoding complexity regulation for a media processor," in *Multimedia Signal Processing, 2001 IEEE Fourth Workshop on*, 2001.

[96] M. Mattavelli, S. Brunetton, and D. Mlynek, "Implementing real-time video decoding on multimedia processors by complexity prediction techniques," in *Consumer Electronics, 1998. ICCE. 1998 Digest of Technical Papers. International Conference on*, 1998, pp. 264 – 265.

[97] C. Hentschel, M. Gabrani, K. Van Zon, R. Bril, and L. Steffens, "Scalable video algorithms and quality-of-service resource management for consumer terminals," in *Consumer Electronics, 2001. ICCE. International Conference on*, 2001.

[98] M. van der Schaar and Y. Andreopoulos, "Rate-distortion-complexity modeling for network and receiver aware adaptation," *Multimedia, IEEE Transactions on*, vol. 7, no. 3, pp. 471 – 479, 2005.

[99] N. Kontorinis, Y. Andreopoulos, and M. van der Schaar, "Statistical framework for video decoding complexity modeling and prediction," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 19, no. 7, pp. 1000 –1013, 2009.

[100] "Delay budget within the access stratum," 3GPP, 3GPP Technical Specification TS 25.853, Mar. 2001.

[101] "Packet switched conversational multimedia applications; default codecs," 3GPP, 3GPP Technical Specification TS 26.235, Mar. 2008.

[102] "Video adhoc group database for video codec evaluation," 3GPP SA4 Video Adhoc Group, Tech. Rep. S4-050789, 2005.

[103] G. Cote and F. Kossentini, "Optimal intra coding of blocks for robust video communication over the internet," *Signal Processing: Image Commun., Special Issue on Real-time Video over Internet*, vol. 15, pp. 25–34, Sep. 1999.

[104] J. Liao and J. Villasenor, "Adaptive intra update for video coding over noisy channels," in *Image Processing, 1996. Proceedings., International Conference on*, vol. 3. IEEE, 2002, pp. 763–766.

[105] Q. Zhu and L. Kerofsky, "Joint source coding, transport processing, and error concealment for H.323-based packet video," in *Proceedings of SPIE*, vol. 3653, 1998, p. 52.

[106] S. Winkler, *Digital Video Quality Vision Models and Metrics*. John Wiley & Sons, 2005.

[107] *Question ITU-R 211/11 ITU-R BT.500 Methodology for the Subjective Assessment of the Quality for Television Pictures*, ITU-R Std., Rev. 11, 2002.

[108] *Question ITU-T 21/9 ITU-T P.910 Subjective video quality assessment methods for multimedia applications*, ITU-T Std., Rev. 1, 9 1999.

[109] "Technical specification group services and system aspects; video codec performance," 3GPP, 3GPP Technical Specification TS 26.902, 2007.

[110] T. Wiegand, G. Sullivan, G. Bjøntegaard, and A. Luthra, "Overview of the H.264/AVC video coding standard," *IEEE Trans. on Circuits Syst. Video Technol.*, vol. 13, no. 7, pp. 560–576, Jul. 2003.

[111] R. Zhang, S. Regunthan, and K. Rose, "Video coding with optimal inter/intra-mode switching for packet loss resilience," *IEEE Journal on Selected Areas in Communications*, vol. 18, no. 6, pp. 966–976, Jun. 2000.

[112] T. Stockhammer and S. Wenger, "Standard-compliant enhancement of JVT coded video for transmission over fixed and wireless IP," in *Fourth International Workshop on Distributed Computing*, Capri, Italy, Sep. 2002.

[113] B. Girod and N. Färber, "Feedback-based error control for mobile video transmission," *Proceeding of the IEEE*, vol. 97, pp. 1707–1723, Oct. 1999.

[114] I. Rhee and S. Joshi, "Error recovery for interactive video transmission over the internet," *IEEE Journal on Selected Areas in Communications*, vol. 18, no. 6, pp. 1033–1049, Jun. 2000.

[115] M. Kalman, P. Ramanathan, and B. Girod, "Rate–distortion optimized streaming with multiple deadlines," in *Proceedings IEEE International Conference on Image Processing*, Barcelona, Spain, Sep. 2003.

[116] W. Tu and E. Steinbach, "Proxy-based reference picture selection for real-time video transmission over mobile networks," in *Proceedings IEEE ICME*, Amsterdam, Netherlands, Jul. 2005, pp. 309–312.

[117] Z. He, Y. Liang, L. Chen, I. Ahmad, and D. Wu, "Power-rate-distortion analysis for wireless video communication under energy constraints," *IEEE Trans. on Circuits Syst. Video Technol.*, vol. 15, no. 5, pp. 645–658, May 2005.

[118] "Permanent document on test components," 3GPP, 3GPP Permanent Document S4-060515, Aug. 2006.

[119] W. Zia, T. Oelbaum, and K. Diepold, "Subjective evaluation of error control strategies for mobile video communication," in *Proc. Picture Coding Symposium*, Nov. 2007.

[120] G. M. Mair, "Towards transparent telepresence," in *Virtual Reality: Second International Conference*, July 2007, pp. 300–309.

[121] D. Scharstein and R. Szeliski, "A taxonomy and evaluation of dense two-frame stereo correspondence algorithms," *Int. J. Computer Vision*, vol. 47, no. 1, pp. 7–42, Apr. 2002.

[122] "Middlebury stereo evaluation," http://vision.middlebury.edu/stereo/data.

[123] S. Birchfield and C. Tomasi, "Depth discontinuities by pixel-to-pixel stereo," *Int. J. Computer Vision*, vol. 35, no. 3, pp. 269–293, Dec. 1999.

[124] L. Wang, M. Liao, M. Gong, R. Yang, and D. Nistér, "High quality real-time stereo using adaptive cost aggregation and dynamic programming," in *Int. Symp. 3D Data Processing, Visualization and Transmission*, Jun. 2006, pp. 798–805.

[125] P. F. Felzenszwalb and D. P. Huttenlocher, "Efficient belief propagation for early vision," *Int. J. Computer Vision*, vol. 70, no. 1, pp. 41–54, Oct. 2006.

[126] Q. Yang, L. Wang, R. Yang, S. Wang, M. Liao, and D. Nistér, "Real-time global stereo matching using hierarchical belief propagation," in *Br. Machine Vision Conf.*, Sep. 2006, pp. 989–998.

[127] A. Klaus, M. Sormann, and K. Karner, "Segment-based stereo matching using belief propagation and a self-adapting dissimilarity measure," in *Int. Conf. Pattern Recognition*, Aug. 2006, pp. 15–18.

[128] M. Morbee, L. Tessens, J. Prades-Nebot, A. Pizurica, and W. Philips, "A distributed coding-based extension of a mono-view to a multi-view video system," in *3DTV-Conf.*, Kos Island, Greece, May 2007.

[129] E. Ekmekcioglu, S. T. Worrall, and A. M. Kondoz, "Bit-rate adaptive downsampling for the coding of multi-view video with depth information," in *3DTV-Conf.*, May 2008.

[130] L. Karlsson and M. Sjostrom, "Region-of-interest 3D video coding based on depth images," in *3DTV-Conf.*, May 2008.

[131] S. Ince, E. Martinian, S. Yea, and A. Vetro, "Depth estimation for view synthesis in multiview video coding," in *3DTV-Conf.*, May 2007.

[132] K.-J. Yoon and I.-S.Kweon, "Adaptive support-weight approach for correspondence search," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 28, no. 4, pp. 650–656, Apr. 2006.

[133] W. Zia, K. Diepold, and M. Sarkis, "Optimization of video coding for telepresence applications," in *Applications of Computer Vision (WACV), 2009 Workshop on*. IEEE, pp. 1–8.

[134] "Multimedia broadcast/multicast service; stage 1 (release 9)," 3GPP, 3GPP Technical Specification TS 22.146, Jun. 2008.

[135] "Multimedia broadcast/multicast service (MBMS); protocols and codecs (release 9)," 3GPP, 3GPP Technical Specification TS 26.346, Sep. 2010.

[136] T. R. Gardos, "Video codec test model, near term, version 10 (TMN-10), draft 1," Tech. Rep. Q15-D-65d1, 1998.

[137] T. Wiegand and B. Girod, "Lagrange multiplier selection in hybrid video coder control," in *Image Processing, 2001. Proceedings. 2001 International Conference on*, vol. 3. IEEE, 2001, pp. 542–545.

[138] J. Slotine, W. Li *et al.*, *Applied nonlinear control*. Prentice hall Englewood Cliffs, New Jersey, 1991, vol. 66.

[139] M. Fliess and C. Join, "Intelligent PID controllers," in *Control and Automation, 2008 16th Mediterranean Conference on*. IEEE, 2008, pp. 326–331.

[140] "Generic coding of moving pictures and associated audio information: Systems," ITU–T Recommendation ISO/IEC 13818-1:2007(E), 2007.

[141] K. Willner, K. Ugur, M. Salmimaa, A. Hallapuro, and J. Lainema, "Mobile 3D video using MVC and N800 internet tablet," in *3DTV Conference: The True Vision-Capture, Transmission and Display of 3D Video, 2008*. IEEE, 2008, pp. 69–72.

[142] L. Aimar, L. Merritt, E. Petit, M. Chen, J. Clay, M. Rullgrd, C. Heine, and A. Izvorski, "x264-a free H264/AVC encoder." [Online]. Available: http://www.videolan.org/developers/x264.html

[143] F. Bellard and M. Niedermayer, "FFmpeg." [Online]. Available: http://ffmpeg.org/

[144] K. S. et al., "H.264/AVC software coordination." [Online]. Available: http://iphome.hhi.de/suehring/tml/

[145] *MSM7200A™ Chipset Solution*, QUALCOMM, 2007. [Online]. Available: http://www.datasheetpro.com/268119_download_MSM7200A_datasheet.html

[146] *HTC Touch HD phone specifications*, HTC Corporation, 2007. [Online]. Available: http://www.gsmarena.com/htc_touch_hd-2525.php

[147] *Intel® Atom™ Processor N270*, Intel Corporation, 2008. [Online]. Available: http://ark.intel.com/products/36331

[148] *AMD Inc. Opteron™ Processor Solutions*.

[149] T. Stockhammer, "System and cross-layer design for mobile video transmission," Ph.D. dissertation, Technischen Universität München, 2008.

[150] A. Papoulis, S. Pillai, and S. Unnikrishna, *Probability, random variables, and stochastic processes*. McGraw-hill New York, 1965, vol. 196.

[151] S. Riedl, "Iterative decodierung parallel verketteter binärer faltungscodes," Ph.D. dissertation, Technischen Universität München, Jan 1997.

[152] "Joint call for proposals on video compression technology," ISO/IEC JTC1/SC29/WG11, Tech. Rep. N11113, Jan. 2010.

[153] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, "RTP: A transport protocol for real–time applications," Internet Engineering Task Force (IETF), Request for Comments (standard) 3550, Jul. 2003.

[154] "SA4 simulator for packet-switched services," 3GPP SA4 Video Adhoc Group, Tech. Rep. S4-050685, 2005.

All internet links were last checked in 6.1.2012.

# List of Figures


2.1 An abstraction for a video communication system . . . . . 5
2.2 Performance evaluation [17] of DISCOVER DVSC for sequence "Foreman" . . . . . 9
2.3 A sketch of slice structure frame coding for H.264/AVC . . . . . 13

3.1 Energy tradeoff between video encoding and wireless data transmission [57]. . . . . 20
3.2 Decoder resource usage model being used for bitstream adaptation [98] 27

4.1 Protocol stack of 3GPP Conversational Packet Switched Video Services 30
4.2 Instantaneous impact of an error on video quality . . . . . 31
4.3 Performance comparison with varying slice sizes, with RLC-PDU loss as a parameter. . . . . 34
4.4 IEC with error tracking. The video frame rate is $1/T$ . . . . . 36
4.5 Proposed IEC configurations. . . . . 37
4.6 Average results based on PSNR vs. RLC-PDU loss rate performance 38
4.7 Average results based on PDVD vs. RLC-PDU loss rate performance 38
4.8 PSNR variation within sequence "Stunt" . . . . . 39
4.9 Subjective comparison for the investigated techniques. . . . . 40
4.10 Subjective test results for sequence "Party" . . . . . 41
4.11 A depiction of a Telepresence scenario. . . . . 43
4.12 Performance results for test sequence "Vassar" . . . . . 46
4.13 Rate-distortion performance comparison of MPEG-4 ASP and MVC for "Vassar" . . . . . 46
4.14 Comparisons for H.264/AVC and MVC. . . . . 48

5.1 An abstraction of a generic video coding application . . . . . 52
5.2 Hypothetical reference decoder (HRD) data flow and the buffering . 55
5.3 The timelines and intervals of different processes in a traditional hypothetical reference decoder (HRD) model . . . . . 55
5.4 An abstraction of generic video complexity verifier (VCV) . . . . . 56
5.5 Timelines of various processes for VCV . . . . . 58
5.6 Status of the frame buffers used for VCV . . . . . 60
5.7 A high-level abstraction of the video decoder . . . . . 62
5.8 Flow of the data in the buffered model . . . . . 63
5.9 An architectural abstraction of the decoder . . . . . 64
5.10 An abstraction of session setup for optimized delivery . . . . . 74
5.11 A computational resource demand snapshot, showing a chaotic system 76

5.12 The block diagram of the control system . . . . . 78

6.1 Performance of the simplest decoder model ($L = 1$) for sequence "Foreman" . . . . . 87
6.2 Decoder model with $L = 2$, its instantaneous, average, peak and cumulative error performance for "Foreman" . . . . . 88
6.3 Verification of the model on HA2 for sequence "Party" . . . . . 89
6.4 Computationally resource constrained decoder average performance . 92
6.5 Instantaneous results for selected frames of "Foreman" . . . . . 94
6.6 The resource buffer fullness (above) and the $\mathcal{Q}^c$ for a selected portion of the sequence . . . . . 95
6.7 Computational resource usage (top), the resource buffer fullness (middle) and the complexity index (bottom) for a selected portion of the sequence "Foreman" using CCAM . . . . . 96
6.8 Computationally resource constrained encoder average performance . 97
6.9 PROP used on encoder, unconstrained resource usage by the decoder (secondary Y-axis) . . . . . 98
6.10 Computational resource usage (top), the resource buffer fullness (middle) and the complexity index (bottom) for a selected portion of the sequence "Foreman" using CCAM . . . . . 98
6.11 Both encoder and decoder computationally resource constrained . . 100
6.12 PROP used for both the encoder and the decoder (secondary Y-axis) 101
6.13 Performance of codec with resource-constrained decoder in a lossy channel scenario . . . . . 102
6.14 Performance of codec with resource-constrained encoder in a lossy channel scenario . . . . . 103
6.15 Comparative RD performance of various resource optimization techniques. . . . . 104
6.16 Comparative CD performance of various resource optimization techniques. . . . . 104
6.17 Comparative D-C performance of various resource optimization techniques for streaming along with RD characteristics of representations. 108

A.1 File I/O Simulation Environment . . . . . 113
A.2 File-based Simulation Approach . . . . . 115
A.3 Proposed multi-process simulation environment . . . . . 116
A.4 Block diagram of the evaluation framework. . . . . 117

# List of Tables

5.1 Signaling overhead for receiver-end optimization . . . . . 82