On the Effectiveness of Video Recolouring as an Uplink-model Video Coding Technique

A thesis submitted to the
College of Graduate and Postdoctoral Studies
in partial fulfillment of the requirements
for the degree of Master of Science
in the Department of Computer Science
University of Saskatchewan
Saskatoon

By
Kai Langen

Permission to Use

In presenting this thesis in partial fulfillment of the requirements for a Postgraduate degree from the University of Saskatchewan, I agree that the Libraries of this University may make it freely available for inspection. I further agree that permission for copying of this thesis in any manner, in whole or in part, for scholarly purposes may be granted by the professor or professors who supervised my thesis work or, in their absence, by the Head of the Department or the Dean of the College in which my thesis work was done. It is understood that any copying or publication or use of this thesis or parts thereof for financial gain shall not be allowed without my written permission. It is also understood that due recognition shall be given to me and to the University of Saskatchewan in any scholarly use which may be made of any material in my thesis.

Disclaimer

Reference in this thesis to any specific commercial products, process, or service by trade name, trademark, manufacturer, or otherwise, does not constitute or imply its endorsement, recommendation, or favoring by the University of Saskatchewan. The views and opinions of the author expressed herein do not state or reflect those of the University of Saskatchewan, and shall not be used for advertising or product endorsement purposes.

Requests for permission to copy or to make other uses of materials in this thesis in whole or part should be addressed to:

Head of the Department of Computer Science
176 Thorvaldson Building, 110 Science Place
University of Saskatchewan
Saskatoon, Saskatchewan S7N 5C9 Canada

OR

Dean
College of Graduate and Postdoctoral Studies
University of Saskatchewan
116 Thorvaldson Building, 110 Science Place
Saskatoon, Saskatchewan S7N 5C9 Canada

Abstract

For decades, conventional video compression formats have advanced via incremental improvements with each subsequent standard achieving better rate-distortion (RD) efficiency at the cost of increased encoder complexity compared to its predecessors. Design efforts have been driven by common multi-media use cases such as video-on-demand, teleconferencing, and video streaming, where the most important requirements are low bandwidth and low video playback latency. Meeting these requirements involves the use of computationally expensive block-matching algorithms which produce excellent compression rates and quick decoding times.

However, emerging use cases such as Wireless Video Sensor Networks, remote surveillance, and mobile video present new technical challenges in video compression. In these scenarios, the video capture and encoding devices are often power-constrained and have limited computational resources available, while the decoder devices have abundant resources and access to a dedicated power source. To address these use cases, codecs must be power-aware and offer a reasonable trade-off between video quality, bitrate, and encoder complexity. Balancing these constraints requires a complete rethinking of video compression technology.

The uplink video-coding model represents a new paradigm to address these low-power use cases, providing the ability to redistribute computational complexity by offloading the motion estimation and compensation steps from encoder to decoder. Distributed Video Coding (DVC) follows this uplink model of video codec design, and maintains high quality video reconstruction through innovative channel coding techniques. The field of DVC is still early in its development, with many open problems waiting to be solved, and no defined video compression or distribution standards. Due to the experimental nature of the field, most DVC codec to date have focused on encoding and decoding the Luma plane only, which produce grayscale reconstructed videos.

In this thesis, a technique called "video recolouring" is examined as an alternative to DVC. Video recolouring exploits the temporal redundancies between colour planes, reducing video bitrate by removing Chroma information from specific frames and then recolouring them at the decoder.

A novel video recolouring algorithm called Motion-Compensated Recolouring (MCR) is proposed, which uses block motion estimation and bi-directional weighted motion-compensation to reconstruct Chroma planes at the decoder. MCR is used to enhance a conventional base-layer codec, and shown to reduce bitrate by up to 16% with only a slight decrease in objective quality. MCR also outperforms other video recolouring algorithms in terms of objective video quality, demonstrating up to 2 dB PSNR improvement in some cases.

Acknowledgements

Foremost, I would like to express my sincere thanks to my advisor, Dr. Dwight Makaroff for his support and encouragement.

Besides my advisor, I would like to thank the rest of my thesis committee: Dr. Derek Eager, Dr. Mark Eramian, and Dr. Khan Wahid for all of their excellent comments and questions. I would also like to thank Dr. Ketan Mayer-Patel for his assistance in refining the ideas in this thesis and related conference papers.

Finally, I would like to thank my wife, Brittany, for her love, patience, and understanding.

Contents

List of Tables

List of Figures

List of Abbreviations

BMA	Block Matching Algorithm
BR	Bit-Rate
CGOP	Chroma Group of Pictures
CIF	Common Intermediate Format
CNM	Channel Noise Model
CRC	Cyclic Redundancy Check
DCT	Discrete Cosine Transform
DVC	Distributed Video Coding
FR	Frame-Rate
GOP	Group of Pictures
LDPC	Low Density Parity Check
LDPCA	LDPC Accumulate
MAD	Mean Absolute Difference
MC	Motion Compensation
MCI	Motion-Compensated Interpolation
MCX	Motion-Compensated Extrapolation
MCR	Motion-Compensated Recolouring
ME	Motion Estimation
MSE	Mean Square Error
MV	Motion Vector
OBMC	Overlapped Block Motion Compensation
PDWZ	Pixel Domain Wyner-Ziv
PRISM	Power-efficient, Robust, hIgh compression Syndrome-based Multimedia coding
PSNR	Peak Signal-to-Noise-Ratio
QP	Quantization Parameter
RD	Rate-Distortion
SAD	Sum of Absolute Difference
SI	Spatial Information Measurement
SSIM	Structural Similarity
TDWZ	Transform Domain Wyner-Ziv
TI	Temporal Information Measurement
WZ	Wyner-Ziv

1 Introduction

1.1 Motivation

Wireless Sensor Networks (WSN) consist of a system of distributed, low-powered sensors that cooperate to deliver information [30]. WSNs are often deployed in remote locations, with limited access to power or Internet, and are generally used to collect low-rate environmental data including temperature, humidity, wind speed, and air quality [69]. In recent years, however, the collection and distribution of high-rate video data has become an increasingly important use case for WSNs. These networks, referred to as Wireless Video Sensor Networks (WVSN), consist of spatially distributed video-capture devices that collect raw video data and transmit it to an off-site location for real-time analysis [56].

Researchers at the University of Saskatchewan set up a cluster of video cameras to monitor the development of phenotypes among Canola plots in a test field. This "Camera On A STick" (COAST) system was created by the Plant Phenotyping and Imaging Research Centre (P^{2}IRC),[1] funded by the Canada First Research Excellence Fund (CREF),[2] and managed by the Global Institute for Food Security (GIFS).[3] In the COAST system, a single camera monitored each individual plot and captured a series of time-lapse images taken a minute apart. The cameras were left in the field to record with researchers returning at periodic intervals to harvest the image data. The purpose of these time-lapse images was to observe the presence of different phenotypes in various Canola test plots, including the number of flowers each plant produces over the Summer and each breed's resistance to "lodging". In an agricultural context, lodging refers to a situation in which plant stems have been broken or bent near ground-level, making the crop difficult to harvest [7]. Flowering and lodging are two physical characteristics of Canola that interest plant scientists due to the impact they have on crop yield.

Unlike WVSN systems, COAST was not designed with a networking infrastructure to transmit the video data. Instead, data collection was performed manually, requiring researchers to access and dismount each camera from its position in the field. This manual data collection increased the risk that the data-gatherers might either disturb the test-plots or alter the camera angle or focus, rendering the next batch of images unusable. Due to the long period between image collection dates, such a disruption to the camera could result in significant data loss.

[1] https://p2irc.usask.ca
[2] https://www.cfref-apogee.gc.ca
[3] https://www.gifs.ca

Plant scientists desired a more convenient method to collect and analyze the data from these cameras. One proposed technique was to transform this camera cluster into a WVSN, taking advantage of the cameras' proximity to one another. In this scenario, each node would be equipped with a low-range wireless communicator, which it could use to transmit video data to neighbouring nodes. The video packets would then be relayed to local base-station that would upload videos to the cloud. This use case is illustrated in Figure 1.1.

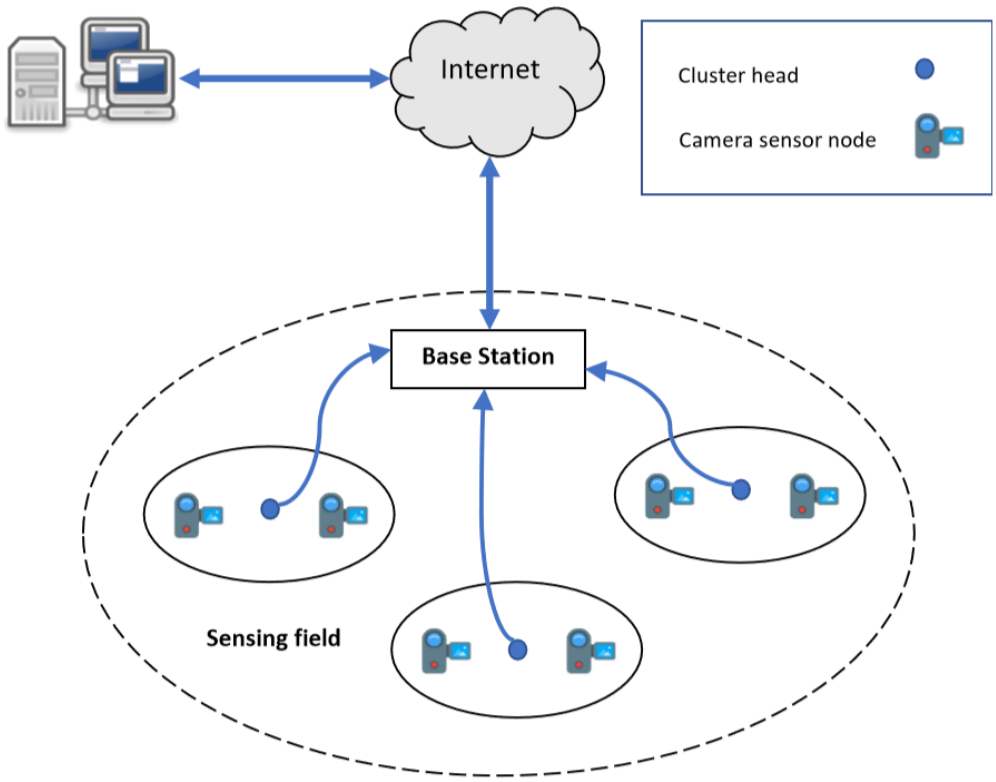


Figure 1.1: Canola video data collection

A lower-quality preview of the time-lapsed images could be created using video compression, to enable scientists to verify the continued functionality of the image capture device as well as select specific frames or windows of time for further analysis. Higher quality versions of the selected frames could then be transmitted for use by computer vision applications. Each WSN node would be battery-powered, and would consist of camera attached to a low-powered, single-board computer such as the Raspberry Pi. In this environment, it would be important to conserve power at the encoder and offset as much computational work as possible to the high-performance computers on the university grounds. Reducing power consumption would need to be balanced against the competing constraints of limited transmission rate and a desire for high image quality. At the same time, these image sequences contain a high degree of redundant temporal information, so it

makes sense to use video compression to ensure a reasonable tradeoff between bitrate and quality.

Video compression applications can be split into two models: *downlink* and *uplink*. Conventional video codecs are designed to address *downlink* or broadcast video coding scenarios where a video is transmitted from a single encoder to multiple decoder/play-back devices. In this downlink application model, fast decoding is the top priority, while encoder power-efficiency is deemed less important. The *uplink* or wireless video application model represent the inverse scenario; it aims to minimize encoding computation, offloading the work to a decoder with a dedicated power source. Alternatively, an intermediate device called a "transcoder" may be employed, which acts as a buffer between these two models. The transcoder receives a video transmitted by an uplink-style encoder, and then performs the computationally expensive task of decoding or reconstructing the original video. This reconstruction is then re-encoded using a downlink-style codec into a format that can be quickly decoded and viewed by an end-user. Some examples that represent the downlink model include the H.26x and VPx families of codecs as well as more recent innovations such as the Thor [47] or Daala [70] codecs. Distributed Video Coding (DVC) architectures such as the DISCOVER codec [4] are examples of uplink model solutions [30].

Generally speaking, DVC codecs offload computation by moving the inter-frame prediction loop from the encoder to the decoder. In a simple example, the DVC encoder might transmit only the even frames from a video, as well as "hints" to assist the decoder in correctly guessing the missing, odd frames. These hints could take the form of hash-codes or error-correcting channel codes such as Turbo [2] or Low-Density Parity Check (LDPC) codes [4]. There are two common techniques that a DVC decoder uses to predict missing frames: interpolation and extrapolation. These frame predictions are referred to as side-information that the decoder combines with the provided hints in an attempt to reconstruct the missing frames completely. If the information provided is insufficient to reconstruct a missing frame, the decoder iteratively requests more detailed hints from the encoder using a feedback loop until it has enough information or a threshold number of iterations has been reached. Frames encoded using this process are referred to as "Wyner-Ziv" or WZ-frames, named after the founding researchers of the field [1].

Based on their ability to reduce encoder computation, DVC video codecs then seem well-suited to the COAST use case. The low-power encoder devices can avoid expensive block-motion estimation, while the large-scale computers can operate as transcoders, both decoding the DVC videos received from the WVSN network and converting them into a more playable video format (e.g. H264 video). Unfortunately there are two major drawbacks of DVC codecs which make them impractical for the COAST use case described above. First, the field of distributed video coding is still in its early stages, and so support for colour video is currently lacking in most major DVC codecs. Plant scientists, however, rely on colour information in the videos to identify the number and location of Canola flowers, which is an important source of phenotype data. Any solution based on DVC would need to be modified to include this colour information. Another major drawback of the DVC video coding scheme is that it relies on a real-time feedback loop between the encoder and decoder to iteratively decode frames. In the described WVSN use case, constant two-way

communication between the video encoder and decoder is not viable due to the added decoding delays, code buffering requirements, and communication overhead.

Researchers at the University of Saskatchewan [24] have introduced a novel technique called video-recolouring that provides an alternative uplink-model coding strategy to DVC. In contrast to DVC, where the decoder estimates only the *Luma* plane, a video recolouring scheme works by removing *Chroma* plane data from the encoded bitstream and reproducing that colour information at the decoder using block-based motion prediction. This scheme limits the potential for prediction error, since Chroma information is deemed less important to the human-visual system (HVS). With decreased prediction error, both channel reconstruction and the feedback-loop can be removed from the system, along with the issues and overhead associated with them. This thesis examines how video recolouring can be combined with Intra coding and Chroma subsampling to optimize compression efficiency in an uplink model video codec.

1.2 Scope

This thesis is not a full treatise on video coding, but aims to address specific topics within that research domain. Coverage and explanation of video compression concepts is restricted to those necessary for understanding the topics of distributed video coding and video recolouring.

Due to the limitations of DVC and the constraints imposed by our WVSN use case, this thesis does not attempt to implement a DVC codec. Instead, research work focuses on the implementation and analysis of different Chroma-prediction methods, exploring how strategies from DVC can be used to improve upon the current video-recolouring research.

1.3 Thesis Statement

This thesis seek to answer the following questions:

- Is the effectiveness of video recolouring content- or context-specific? In other words, does this technique depend on characteristics of the raw video sequence and the encoding parameters?
- Is video-recolouring a useful tool for improving compression efficiency when computational resources are limited at the encoder?
- Finally, can techniques from DVC be used to improve the prediction accuracy of video recolouring?

To address these questions, a new video recolouring framework was implemented, and the following analysis was performed:

- To determine whether video-recolouring is a content-specific optimization technique, a series of benchmark videos was selected with varying spatial, temporal, and colour characteristics. A novel metric

was created to measure the "colourfulness" of a video as a function of its spatial complexity in the Luma and Chroma planes. This independent metric was compared against the actual bit-rate savings achieved by removing Chroma information from a subset of the video frames.

- Next, the rate-distortion performance of various recolouring methods was compared against Intra-only video coding, the baseline low-power video coding method, as well as Inter No Motion, a widely-used benchmark coding profile used in the DVC literature [18, 52]. Four video recolouring techniques were implemented and experiments were run to compare their rate-distortion performance against these baselines.

- Finally, to demonstrate how Chroma-estimation could be improved using innovations from the field of DVC, a new algorithm was implemented and compared against a prior video-recolouring technique. This new method, called Motion Compensated Recolouring (MCR) was also compared against the standard algorithm used in DVC, called Motion Compensated Interpolation (MCI).

1.4 Outline

The remaining chapters of this thesis are structured as follows: Chapter 2, provides a review of the relevant literature in both conventional and distributed video coding. Next, Chapter 3 describes the implementation of multiple recolouring codecs, Chapter 4 covers the experimental design methodologies, and Chapter 5 explains the analysis and results. Finally, Chapter 6 concludes the thesis, summarizing the work to date, and makes recommendations for future work.

2 Background

This chapter covers background material on conventional video codec design, Distributed Video Coding (DVC), and recent work that relates to the specific area of study for this thesis: video recolouring. To better understand DVC, it is helpful to review the technologies used in conventional video and image compression; these topics are covered in Section 2.1. Next, Section 2.2 describes both the theoretical foundations of DVC and some of the major advancements in the field of DVC codec design. Finally, Section 2.3 covers research work related to the topic of Chroma estimation, with selections from the fields of conventional and distributed video coding.

2.1 Conventional Video Coding

The following section provides an overview of video compression topics. First, the information theoretical foundations is presented. Next follows a description of the high-level design of H.264/AVC, a popular video codec. Finally, the specific blocks and algorithms used by this codec are described in greater detail.

2.1.1 Information Theory: Entropy

The theoretical foundations for data compression were introduced in Claude Shannon's seminal work, "A Mathematical Theory of Communication" [63]. In it, Shannon presents his *Source Coding Theorem*, which defines the minimum number of bits required to represent a message without loss of information. Given a finite alphabet of possible symbols $\boldsymbol{A}$, with a probability distribution P, the *information content* of a specific symbol, $a_i \in \boldsymbol{A}$, provides the smallest number of bits that can be used to represent or encode it, and is given by the following equation:

$$i(a_i) \equiv log_2 \frac{1}{p(a_i)} \,. \tag{2.1}$$

Here, $p(a_i)$ is the probability of selecting a_i from the alphabet $\boldsymbol{A}$. Another interpretation of this equation is that more probable symbols convey less information. For a source $\boldsymbol{X}$ that selects randomly from an alphabet, $\boldsymbol{A}$, Shannon defines that source's *entropy* as the probability-weighted average of $\boldsymbol{A}$'s information content per symbols. In other words, entropy describes the smallest number of bits per symbol that can be used to encode $\boldsymbol{X}$ *on average*. The entropy of $\boldsymbol{X}$ is given by the equation

$$H(\boldsymbol{X}) \equiv \sum_{a_i \in A} p(a_i) log_2 \frac{1}{p(a_i)} \,. \tag{2.2}$$

An interesting feature of entropy is that alphabets with a more skewed probability distribution will have a smaller entropy and can be encoded more efficiently. For example, two alphabets ***C1*** and ***C2*** are given in Table 2.1, each consisting of 4 symbols, but with different probability distributions. ***C1*** has a uniform distribution, that produces the maximum entropy for a four-symbol alphabet (2 bits per symbol). ***C2*** has a more skewed distribution and thus has a reduced entropy of only 1.75 bits per symbol.

Table 2.1: Entropy Comparison

Alphabet	$\boldsymbol{p(a_1)}$	$\boldsymbol{p(a_2)}$	$\boldsymbol{p(a_3)}$	$\boldsymbol{p(a_4)}$	**Entropy Equation**	**Entropy**
C1	0.25	0.25	0.25	0.25	$= 4 \times (0.25 \times 2)$	2
C2	0.5	0.25	0.125	0.125	$= (0.5 \times 1) + (0.25 \times 2) + 2 \times (0.125 \times 3)$	1.75

The terms "lossless coding" and "entropy coding" refer to techniques that encode a source by taking advantage of the statistical redundancies of its symbols; lossless coding can be used to compress a source without loss of information, and is lower-bounded by the source's entropy. The minimum bitrate R_{min} of lossless coding a source $\boldsymbol{X}$ is given by the equation

$$R_{min} = H(\boldsymbol{X}) + \varepsilon \quad bits/symbol \,, \tag{2.3}$$

where ε is a small positive value that can be made arbitrarily small [66]. A common form of lossless coding, variable length coding, is described in the proceeding section.

2.1.2 Entropy Coding

To ensure that symbols are apportioned bits according to their information content, it is necessary to encode symbols with variable length *codewords*. More probable symbols are encoded using fewer bits, which minimizes the average codeword length. The resulting codewords are referred to as *entropy codes*. Two popular entropy coding methods are Huffman and arithmetic coding [66]. The details and implementation of Huffman coding have been provided in Appendix A.

2.1.3 Lossy Compression

As previously discussed, the minimum bitrate R_{min} of lossless compression is lower-bounded by the signal's entropy; for image and video, typical lossless compression ratios are fairly small, between 1:2 and 1:3 [50]. Lossless coding is also reversible, in that the decompressed signal is identical to the original input signal. In contrast, *lossy compression*, reconstructs the original signal with some level of noise or distortion.

In the lossy case, R_{min} is no longer lower-bounded by entropy, but is instead a function of the distortion that is allowed to enter the output signal. With video and image compression, lossy codecs are able to exploit

particular aspects of the human psycho-visual system to produce outputs that appear similar to the original input. The human eye is less sensitive to specific frequencies and so the codec can distort or discard this information, reducing the bitrate with a minimal loss of subjective quality. Common elements in a lossy image compression system are the following:

1. *Transform* (T) applies a one-to-one mapping to the image, converting it into another domain or basis that is more easily compressed. The most common example is the Discrete Cosine Transform (DCT), which maps from the spatial to the frequency domain. Most of the image's energy (and important shape information) is packed into the low-frequency coefficients. In H.264, the image is partitioned into 4x4 or 8x8 *transform blocks* and a DCT is applied to each block. The DCT transform function by itself is a lossless transformation, since there exists an inverse transform function, IDCT, that can perfectly reconstruct the original signal.

2. *Quantizer* (Q) divides the transform coefficients by a scalar value or quantization matrix, the magnitude of which is decided by a *Quantization Parameter* (QP). Finally, the resulting values are truncated to integers. The inverse of quantization is de-quantization, where the truncated integers are multiplied by the same scalar or matrix values. The act of quantizing and de-quantizing a signal introduces some noise, since the quantization function is a many-to-one mapping, and there is no guarantee that the reconstructed signal will be identical to the original. Thus, the Quantizer is the major source of lossy compression in the video codec.

3. *Coder* (C) encodes the symbols that are output from the Quantizer using entropy coding and other lossless coding techniques. Due to the division present in the quantization step, small-valued symbols are more likely to occur, skewing the probability distribution and lowering entropy. Reducing entropy enables more efficient encoding of each image [66].

Within the *Coder* module, another form of lossless compression is performed prior to entropy coding. High frequency coefficients are typically small and quantization often converts these small values to zero. A technique called "Zigzag scan" is used to reorder the two-dimensional coefficient matrix into a one-dimensional array, as depicted in Figure 2.1. The reordering occurs in such a way that the zero values tend to cluster together at the end of the array.

Next, run-level coding (RLC) is applied to eliminate symbol redundancy by compressing long runs of repeating zeros into a more compact form. RLC converts each symbol element into a (run,level) pair, where the **run** indicates the number of zeros preceding each non-zero coefficient and the **level** indicates the coefficient's magnitude. A special symbol is used to indicate the final non-zero coefficient in the block [61]. These techniques form the most basic components of the H.264/AVC encoder, depicted in Figure 2.2, where the transform, quantizer, and coder blocks are shown surrounded by dashed blue lines. Other components include the intra- and inter-frame prediction modules. These modules are used to eliminate other forms of redundancy found in video data, and are described in the next section.

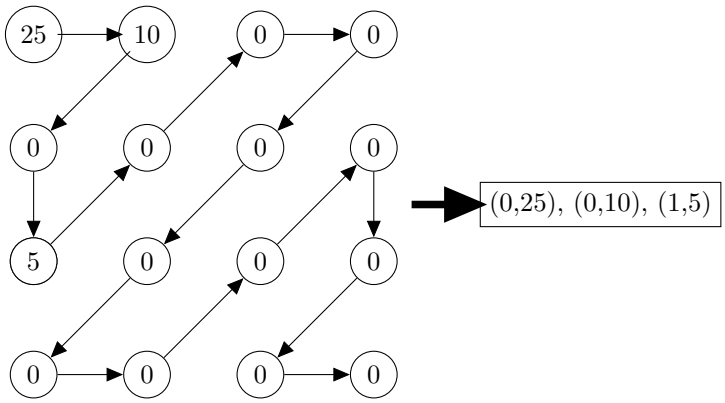


Figure 2.1: Zig-zag scan followed by run-level encoding

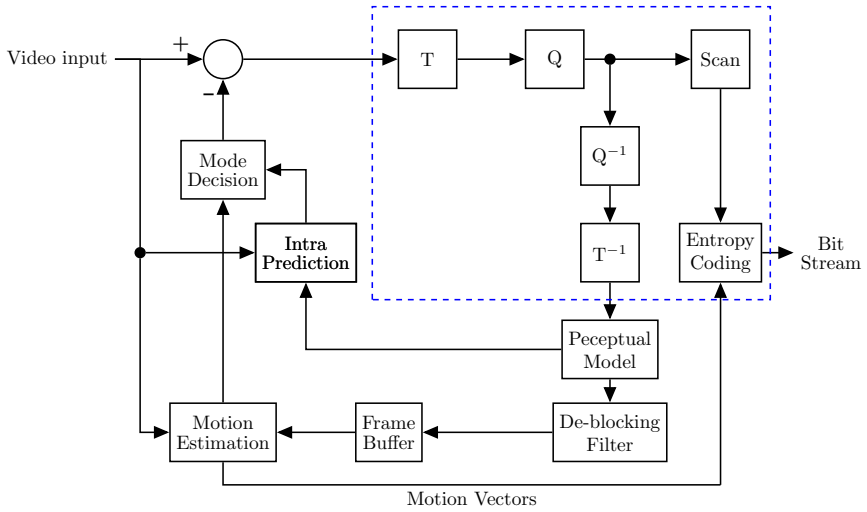


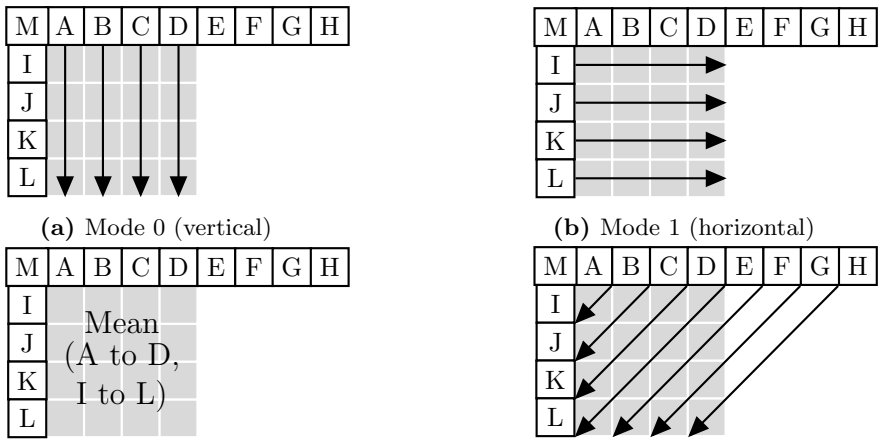
Figure 2.2: H.264 Encoder [33]

2.1.4 Prediction

Video sequences contain a high degree of redundant information. "Prediction models" are a key technology for eliminating this redundancy and efficiently encoding the video data. The goal of a prediction model is to create an estimate of the video sequence that can be represented using as few bits as possible. This prediction is then subtracted from the current frame. More accurate predictions cause the resulting difference or "residual" to have lower energy (smaller pixel values), which is more easily compressed.

A common technique used in video encoding is to split each frame into fixed-sized, non-overlapping blocks. The current block can be predicted from information in neighbouring blocks, referred to as **intra** prediction, or it can be predicted from blocks in nearby frames, referred to as **inter** or **motion compensated prediction**. The former is used to eliminate *spatial redundancy*, while the latter eliminates *temporal redundancy* [61].

- **Intra Prediction** estimates the current block from previously-coded samples in the same frame. Generally, blocks are coded in raster-scan order, from top to bottom, left to right; this makes blocks above and to the left of the current block available for use as a reference. In H.264/AVC, a technique called spatial extrapolation creates predictions using the edges of neighbouring blocks. For each predicted block, the direction of extrapolation (or prediction mode) is signalled in the encoded bitstream [61]. Figure 2.3 depicts some of the Intra prediction modes that are used in H.264. Neighbouring pixels are labelled using capital letters, while arrows indicate the direction of extrapolation. For example, in Mode 0, pixel A is used to predict the pixels directly below it.



(a) Mode 0 (vertical) **(b)** Mode 1 (horizontal)

(c) Mode 2 (DC) **(d)** Mode 3 (diagonal down-left)

Figure 2.3: H.264/AVC Spatial Extrapolation [61]

- **Inter Prediction** estimates the current block using a "reference" frame that can be either in the past or future relative to the current frame. Frame differencing is an example of a simple inter prediction technique that uses the entire previous frame as a predictor of the current frame. For videos with low motion, this technique produces a low energy residual that is easily compressed. In most videos, however, it is common for objects to move between subsequent frames or for the camera itself to move.

When faced with such inter-frame motion, frame differencing becomes less effective, and produces high energy residuals. A more effective prediction technique is to use **motion compensation**, described below.

2.1.5 Motion Estimation/Compensation

Motion estimation/compensation improves upon the predictive power of frame differencing by modeling the motion between frames. A common model of object motion is the *block translation* model developed by Jain and Jain [32]. First, each frame is split into non-overlapping *Macroblocks*. Next, for each Macroblock, the best matching subregion is selected from one or more reference frames. Finally, a prediction of the current frame is created by replacing each Macroblock with its best matching block. This model only accounts for translational movement of objects between frames and does not consider object rotation or scaling. Despite these inaccuracies, the block translation model is widely used due to the efficiency of its motion estimation calculation [27].

Many strategies have been created for finding optimal block matches, referred to as **block matching algorithms** (BMAs). A simple example of a BMA is Exhaustive Search, which iterates over every candidate in a predefined search window and compares them against the current block, as depicted in Figure 2.4. This algorithm is time consuming and computationally expensive, and so many algorithms have been proposed that reduce the number of comparisons and speed up computation. Examples include Logarithmic Search [32], Three Step Search [38], and Diamond Search [79]. For any given BMA, the accuracy or fit of each match is determined using a block matching criteria, with common examples including mean square error (MSE), mean absolute difference (MAD), Peak Signal to Noise Ratio (PSNR), and sum of absolute difference (SAD) [34].

2.1.6 Prediction Structure

Historically, methods for intra/inter-frame prediction were decided on a per-frame basis. For example, the MPEG-2 standard divided frames into a Group of Pictures (GOP) structure consisting of up to three different kinds of frames:

- **Intra-coded pictures (I-frames)** are entirely intra-coded. These typically have poorer rate-distortion performance compared to other types of frames, but are quick to encode and decode. Furthermore, inserting I-frames throughout a sequence at regular intervals allows the playback device to skip to random segments throughout the video.
- **Predicted Pictures (P-frames)** are predicted from a previously decoded frame, which can be either an I-frame or another P-frame. P-frames typically fit between I-frames and B-frames in terms of rate-distortion efficiency and computational cost.
- **Bidirectional Predicted Pictures (B-frames)** are predicted from both previous and future reference frames (which can be either I- or P-frames). For each bidirectionally predicted Macroblock, a matching

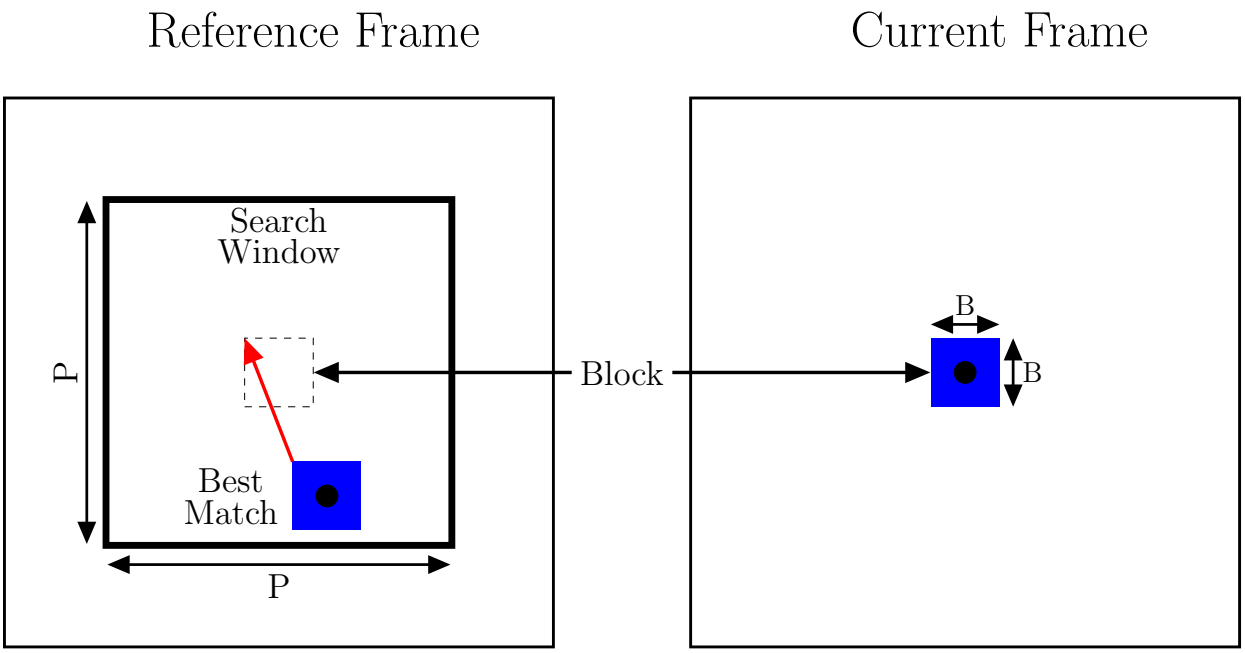


Figure 2.4: Exhaustive Search

block is selected for one past *and* one future frame. These block matches are then averaged to generate the resulting prediction. To accommodate B-frames' use of "future" frames, the transmission/decoding order is restructured such that I- and P-frames are sent prior to the B-frames that depend on them.

Figure 2.5 depicts an example of an MPEG-2 GOP prediction structure. Each GOP starts with an I-frames while P-frames are inserted at regular intervals throughout. The remaining frames are coded as B-frames, because they perform best in terms of RD efficiency.

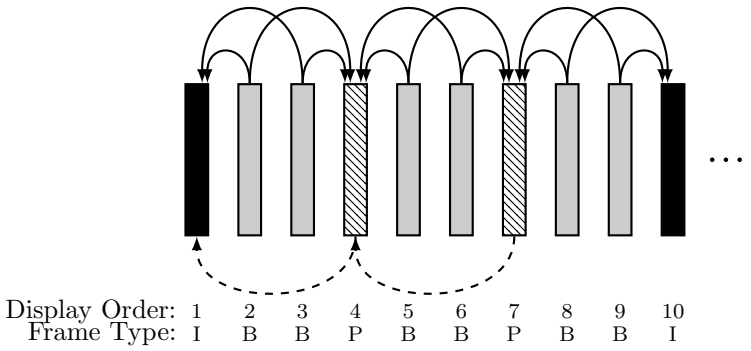


Figure 2.5: GOP Prediction Structure

With the H.264/AVC standard, a number of enhancements have been made to this earlier model; for example, H.264 separates each frame into groups of Macroblocks called "slices". Instead of selecting the prediction method *per-frame*, H.264 allows this selection *per-slice*. Furthermore, multiple different frames can be used as references by slices in the same P- or B-frame. Despite these and other enhancements specific

to B- and P-frames, the simple model depicted in Figure 2.5 provides sufficient high-level detail for the purposes of this thesis.

2.1.7 Colour-space and Chroma Subsampling

A colour space is a mathematical model which represents colour values as coordinates in a multi-dimensional space. In the RGB colour space, each coordinate is given by three numbers representing the proportion of red, green, and blue that make up the colour. The RGB colour space is based on the physiology of the human eye, which uses three kinds of photo-receptive cells to represent visual information [74], and so RGB is commonly used by displays and image sensing devices. Images are constructed from a matrix of individual picture elements or "pixels" that each consist of three 8-bit components (red, green, and blue).

However, the Human Visual System (HVS) is more sensitive to luminance (brightness) than to chrominance (colour) [61]. By using a colour space whose axes can be mapped to luminance and chrominance values, it is possible to discard more information from the planes that represent chrominance during compression. Thus, any image can be represented in a way that is visually indistinguishable from the RGB representation, but that uses fewer bits. One such colour space is called YUV. The Y plane, also called the Luma, represents pixel brightness, while the U and V planes are collectively referred to as the Chroma and represent pixel colour. The following system of linear equations can be used to convert a pixel from RGB to YUV:

$$\begin{aligned} Y &= 0.299R + 0.587G + 0.114B \\ U &= 0.564(B - Y) \\ V &= 0.713(R - Y) \end{aligned} \quad . \tag{2.4}$$

Since the HVS is more sensitive to Luma than Chroma, the resolution of Chroma planes can be reduced without a loss in perceived quality [54]; this technique is referred to as "Chroma subsampling". One common subsampling format, referred to as YUV4:2:0 is achieved by downsampling both Chroma planes so that their vertical and horizontal resolution is half that of Luma, resulting in a single U and V sample for every four Y samples. This format is depicted in Figure 2.6. After Chroma subsampling, the Luma plane retains 8 bits per pixel, while Chroma planes each have 8 bits per 4 pixels (equivalent to 2 bits per pixel). In other words, conversion to YUV combined with Chroma subsampling reduces the total bit depth from 24 bits per pixel in RGB to 12 bits per pixel in YUV, halving the raw image size.

2.2 Distributed Video Coding

Historically, in a compression application, all the information to be compressed would be available in a single location leading to the use of a *centralized encoding* scheme. A number of emerging applications focus on the concept of *distributed encoding*, where the encoder either lacks access to the entire source information

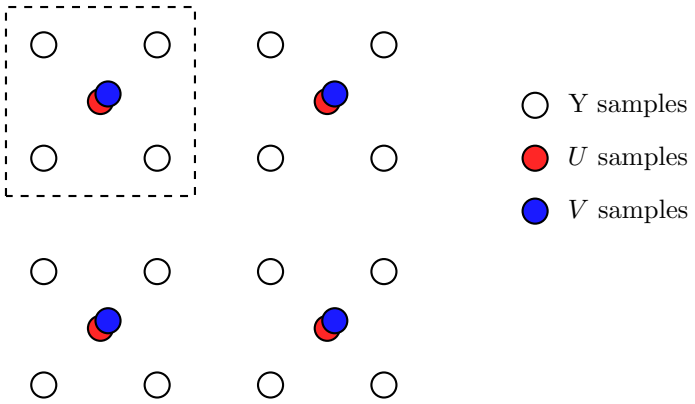


Figure 2.6: Chroma Sub-Sampling (YUV 4:2:0) [61]

to be compressed, or wishes to offload computation to the decoder [18]. Distributed Video Coding (DVC) applies these concepts to the field of video compression. Similar to Conventional Video Coding, DVC has its origins in Information Theory. This section will begin with a brief description of the underlying theorems, followed by a few examples of DVC codec designs used in the past. Finally, the key DVC technology of "side information Generation", will be explained.

2.2.1 Distributed Source Coding

The Source Coding Theorem not only describes the entropy of a single source, but also gives an equation for the combined entropy of multiple sources, called *joint entropy* [63]. The entropy equations for two correlated sources, $\mathbf{X}$ and $\mathbf{Y}$, are given by the following equation:

$$H(\mathbf{X},\mathbf{Y}) = \sum_{x,y} p(x,y) log_2 \frac{1}{p(x,y)}, \tag{2.5}$$

$$H(\mathbf{X}|\mathbf{Y}) = H(\mathbf{X},\mathbf{Y}) - H(\mathbf{Y}), \tag{2.6}$$

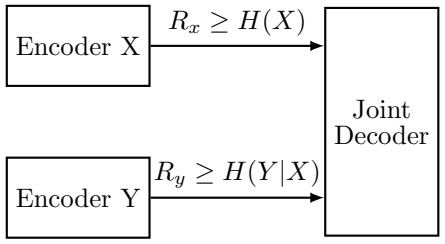
$$H(\mathbf{Y}|\mathbf{X}) = H(\mathbf{Y},\mathbf{X}) - H(\mathbf{X}), \tag{2.7}$$

$$I(\mathbf{X};\mathbf{Y}) = H(\mathbf{X}) - H(\mathbf{X}|\mathbf{Y}) = H(\mathbf{Y}) - H(\mathbf{Y}|\mathbf{X}), \tag{2.8}$$

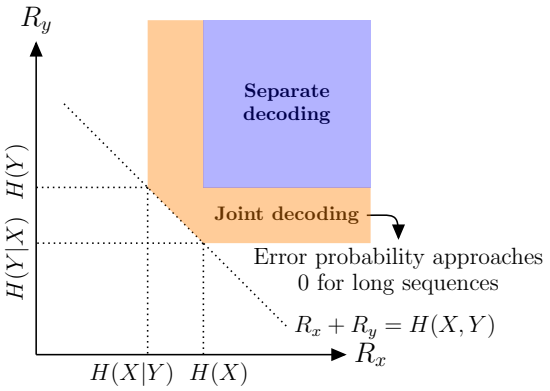
where $H(\mathbf{X},\mathbf{Y})$ is the joint entropy of $\mathbf{X}$ and $\mathbf{Y}$, $H(\mathbf{Y}|\mathbf{X})$ is the conditional entropy of $\mathbf{Y}$ *given* $\mathbf{X}$, and $I(\mathbf{X};\mathbf{Y})$ is the mutual information shared between $\mathbf{Y}$ and $\mathbf{X}$. Finally, $p(x,y)$ is the *joint probability* across all symbols belonging to the alphabets X and Y. If the two sources are independent, in other words they share no mutual information, then their joint entropy is merely the sum of their separate entropies, $H(\mathbf{X})+H(\mathbf{Y})$ [63].

The Slepian-Wolf Theorem. Slepian and Wolf extended Shannon's Source Coding Theorem by postulating that the minimum coding rate for two correlated signals should approach their joint entropy whether the correlation was known at the encoder (*joint encoding*) or the decoder (*joint decoding*) [64]. In an extreme example, consider a scenario where different parts of the source information are available at two separate encoders that cannot cooperate. So long as these encoders transmit to the same decoder, the Slepian-Wolf

theorem states that they should still be able to encode their separate signals at a combined rate approaching the signals' joint entropy with a vanishingly small probability of error [18]. Figure 2.7a gives a diagram of this application scenario, while Figure 2.7b depicts the achievable rate regions for two dependent variables.



(a) Distributed encoding, joint decoding example [18]



(b) Achievable rates for two statistically dependent random signals [19]

Figure 2.7: Slepian-Wolf Diagrams

To illustrate how the Slepian-Wolf Theorem might be applied in practice, consider the following example [57]. Imagine that you have two correlated signals, $\boldsymbol{X}$ and $\boldsymbol{Y}$, each consisting of three bits and given the values [0 1 0] and [0 1 1] such that $\boldsymbol{X}$ and $\boldsymbol{Y}$ have a Hamming distance of one. In other words, it is known that there is at most a single bit different between the two signals. In a joint encoding scenario, $\boldsymbol{Y}$ is available at both the encoder and the decoder to use for coding $\boldsymbol{X}$ and the decoder has knowledge of the Hamming distance. In this context, the signal $\boldsymbol{Y}$ is referred to as "side information" since it can be used as an additional source of information at the decoder to help decode $\boldsymbol{X}$. Given this scenario, what is the smallest number of bits that can be used to represent signal $\boldsymbol{X}$? If it is known that the Hamming distance between the two signals is at most one, then there are four possible values that $\boldsymbol{X}$ can have:

1. a value identical to $\boldsymbol{Y}$: [0 1 1],
2. same as $\boldsymbol{Y}$ but the first bit differs: [1 1 1],
3. same as $\boldsymbol{Y}$ but the second bit differs: [0 0 1],
4. same as $\boldsymbol{Y}$ but the third bit differs: [0 1 0].

Since there are only four values, it is possible to encode $\boldsymbol{X}$ using two bits. In this particular example, the difference occurs in the third bit, so the code [1 1] would be sent, the binary representation of 3.

Now consider a joint decoding scenario, where again, the two signals have a Hamming distance of one, but this time the signal $\boldsymbol{Y}$ is known only at the decoder. How many bits does it take to encode $\boldsymbol{X}$? According to the Slepian-Wolf theorem, the size of the codeword should be the same as in the first scenario. And indeed, this can be shown using the following approach. First, at the encoder generate four sets of codewords, each

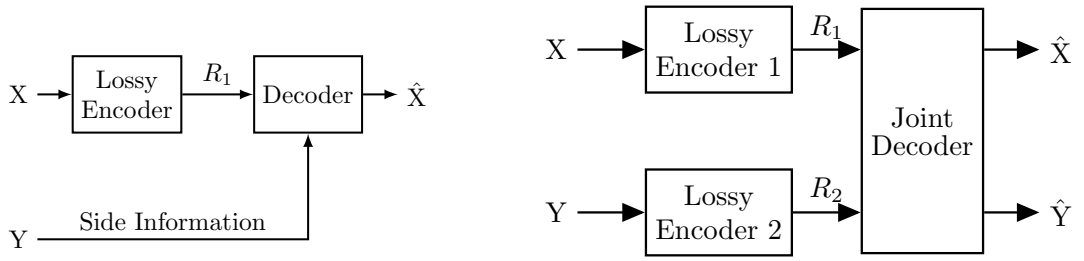
set containing members that are the maximum Hamming distance apart. These are Coset 0: ([0 0 0] and [1 1 1]), Coset 1: ([0 0 1] and [1 1 0]), Coset 2: ([0 1 0] and [1 0 1]), and Coset 3: ([1 0 0] and [0 1 1]). These sets cover all possible 3-bit values, and are placed in ascending order of the first member of each coset.

Next, find the codeword that matches the value of $\boldsymbol{X}$, and transmit the index of its coset. The decoder generates the same list of sets, and uses $\boldsymbol{Y}$ to disambiguate the correct value of $\boldsymbol{X}$ using the Coset index received from the encoder. In this example, the codeword that matches the value $\boldsymbol{X}$ is contained in Coset 2: (**[0 1 0]** and [1 0 1]), so the encoder would transmit an index of two.

The decoder receives the coset index or *syndrome*, and this gives it a choice between the two codewords in Coset 2: [0 1 0] and [1 0 1]. Finally, the decoder selects the codeword with the smallest distance from signal $\boldsymbol{Y}$; in this case [0 1 0] has a Hamming distance of one, while [1 0 1] has a Hamming distance of two, so the former is selected. In this scenario, the encoder/decoder had the choice of four possible syndromes, therefore, this message can also be encoded using only two bits.

The Wyner-Ziv Theorem. Wyner and Ziv further extended the Slepian-Wolf theorem to the lossy compression domain [77]. They considered the case where a source $\boldsymbol{X}$ is encoded and decoded using a correlated side information channel $\boldsymbol{Y}$ to assist the decoder, as seen in Figure 2.8a. Wyner and Ziv demonstrated that for any given distortion level, D, the achievable minimum rate of $\boldsymbol{X}$, $R_x(D)$, is theoretically equivalent whether $\boldsymbol{Y}$ is known at the encoder or the decoder only [15]. One simple application of this theorem is to *quantize* the source at each encoder, similar to conventional video coding, then to *bin* the quantized values into coset codes, as in the prior Slepian-Wolf example; this is named the *quantize-and-bin* strategy.

The Wyner-Ziv theorem relies on a couple of simplifying assumptions: first, the sources are jointly Gaussian and an MSE distortion measure is used [19]; second, the side information itself is not distorted. The general lossy version of the Slepian-Wolf case where both $\boldsymbol{X}$ and $\boldsymbol{Y}$ are distorted (as shown in Figure 2.8b) is an unsolved problem and the rate-distortion region is unknown for this context [15].



(a) Wyner-Ziv scenario [18] **(b)** General lossy Slepian-Wolf scenario [15]

Figure 2.8: Wyner-Zyv Diagrams

The combination of the Slepian-Wolf and Wyner-Ziv theorems is referred to as Distributed Source Coding. Although both theorems were established in the 1970's, there were no practical video codecs based of these theorems until the early 2000's with the advent of DVC. DVC was largely made possible due to recent innovations in channel coding: Turbo and LDPC codes, which are described in the next section.

2.2.2 Channel Coding Theorem

The field of Channel Coding is complementary to the study of Source Coding described above; it was established by the second part of Shannon's 1948 paper, the *Noisy-Channel Coding Theorem.*

This theorem provides a theoretical upper limit on the coding rate at which a signal can be transmitted over a noisy channel with a low probability of error. The theorem states that given a noisy physical channel with capacity C and a transmission rate over that channel R, then for rates $R < C$, it should be possible to encode the input signal such that there is an arbitrarily small probability of error, ϵ, in the decoded output. However there is no such guarantee for rates $R > C$. The paper also provides a definition for the capacity of a channel as

$$C = MAX\big[I(X;Y)\big], \tag{2.9}$$

where X and Y are the input and output signals respectively. In other words, the capacity is defined as the greatest possible mutual information between any given input and output signals that can be transmitted over the channel [63]. Based on this definition, it is possible to derive the capacity of an Additive White Gaussian Noise (AWGN) channel, a common noise model. AWGN channels have a capacity defined as a function of their channel bandwidth, W, and signal-to-noise ratio (SNR):

$$C = W * log_2(1 + SNR)\frac{bits}{second} \text{ [68].} \tag{2.10}$$

In recent decades, two channel codes have emerged, demonstrating the ability to approach Shannon's theoretical channel capacity. This breakthrough has enabled the development of practical DVC implementations that treat inter-frame correlation as a form of channel noise. At the decoder, an estimate of an unknown frame, X, can be produced from its adjacent frames. This estimate is commonly referred to as side information, and can be treated as a noisy version of X. The side information can be used at the decoder to reconstruct X with the help of error-correction codes.

The difference between the estimated side information and the original frame is referred to as correlation noise. Older DVC architectures tend to model correlation noise using a parameterized function whose arguments are calculated offline [22], whereas more recent works use an online learning model that estimates parameters based on statistics from previously-decoded frames [4, 20]. Research has moved towards using online models due to the non-stationary nature of correlation noise: sudden changes in lighting or motion between frames can affect the accuracy of predicted SI, meaning a single choice of parameters may not suit an entire video sequence [12].

The accuracy of both side information and the correlation noise model (CNM) affect the bitrate required to reconstruct a frame, X. If both the side information and CNM are reasonably accurate, a frame can be reconstructed at a much smaller bitrate than it would take to represent the frame itself; this is only feasible, however, if the coding rate is also sufficient, hence the need for high-efficiency codes like Turbo and LDPC.

2.2.3 Turbo and LDPC Codes

In the years following the publication of Shannon's paper, a number of highly structured channel codes were developed such as Reed-Muller (RM) [60], Bose-Chaudhuri-Hocquenghem (BCH) [10], and convolutional codes [73]. These codes had guaranteed minimum distance properties, offered good error-correction capabilities, and were supported by efficient decoding algorithms, but were unable to achieve the channel capacity performance stated by Shannon's theorem. However, in the early nineties, two promising capacity-approaching codes emerged: Turbo and LDPC codes. The random code construction of both Turbo and LDPC codes increased decoding time, but enabled them to exceed all prior channel codes in terms of coding rate [41].

Turbo codes [43] are constructed using a combination of convolutional coders and an equal number of interleavers. Decoding is iterative, using the Maximum a Prosteriori Proability Estimate or "MAP" algorithm. LDPC codes, on the other hand, were first studied by Gallager in 1962 [21], but were largely ignored until their rediscovery by MacKay *et al.* in 1996 [44]. The codes are defined in terms of a sparse parity-check graph, (or alternatively a sparse matrix, H), which is randomly constructed based on a set of constraints.

LDPC provides extremely good coding rates, assuming an optimal decoder. In practice, however, optimal decoding is an NP-hard problem, so the near-optimal Belief Propagation algorithm is used instead [43]. Belief Propagation is an iterative message passing algorithm commonly used in machine learning and information theory to compute the marginal probability distributions of nodes in a graph. On every iteration, nodes in the graph send a message to each of their neighbours, then update their own marginal probability based on the received messages. The Belief Propagation algorithm checks if the graph state has converged or if the number of iterations has reached some predefined threshold; otherwise, it continues iterating. The algorithm either returns its final converged state or an error value indicating that convergence was never reached [11].

The LDPC parity check graph satisfies the condition $Hx = 0$ for all valid codewords x; that is, the inner product of H with any valid codeword produces a zero syndrome, while invalid codewords produce non-zero syndromes. This property is used to check that converged Belief Propagation results are valid. The belief propagation algorithm used by LDPC is a form of "soft-decision" decoding. In contrast to "hard-decision" decoders, where nodes must take one of a fixed set of states at each iteration, soft-decision decoders may take a range of values in between; this allows the nodes to express uncertainty about their state [55]. Such soft-decision decoding algorithms are more difficult to implement, but provide improved performance over hard-decision coding [41].

Finally, although LDPC and Turbo codes have the same theoretical upper bounds, it has been shown that LDPC codes are superior to Turbo codes in the context of a practical DVC codec. The difference lies in the increased robustness of LDPC codes, which are less sensitive to inaccurate CNM model parameters [8].

2.2.4 Early DVC Codecs

Two independent DVC codecs were presented in 2002: the Berkeley Architecture, also known as PRISM (Power-efficient, Robust, hIgh compression Syndrome-based Multimedia coding) [58] and the Stanford Architecture [2]. These two codecs represent fundamentally different designs for DVC, but with the same overarching goal of moving computation from encoder to decoder.

PRISM. The PRISM Architecture depicted in Figure 2.9 took a block-based approach to DVC. Similar to conventional intra-encoding, frames were partitioned at the encoder into non-overlapping Macroblocks, DCT-transformed, and quantized. Here PRISM differed from intra-encoding; instead of source-coding the quantized codewords, PRISM "syndrome-encoded" them, generating coset indices, similar to the quantize-and-bin strategy described earlier. The decoder then performed motion-estimation to generate a number of candidate predictors that were tested one at a time in an attempt to decode the quantized codeword [30, 57].

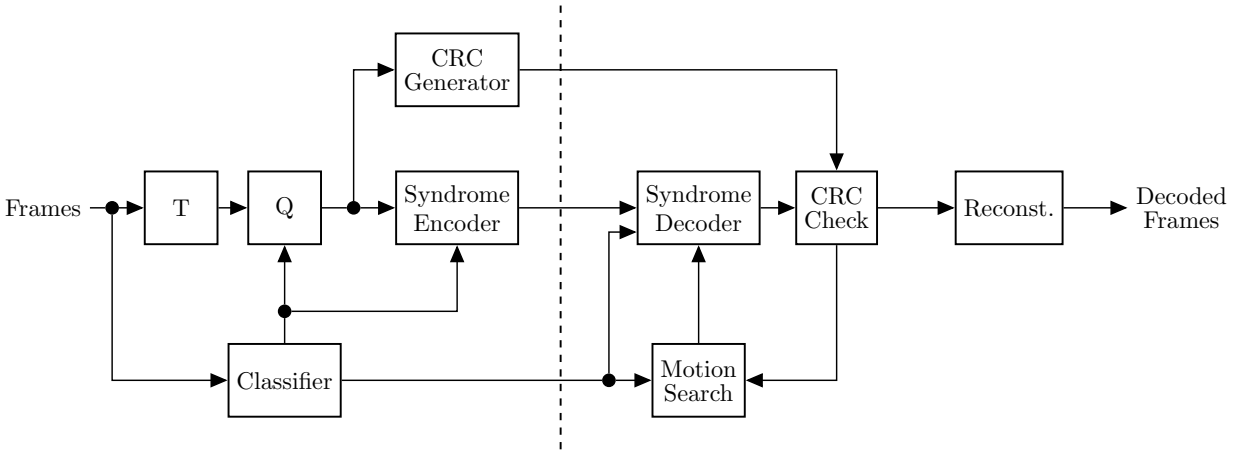


Figure 2.9: PRISM Architecture, from Dufaux *et al.*, 2009 [19]

The Stanford Architecture. While the PRISM architecture selected between channel coding or intra-coding at the blocks level, the Stanford Architecture instead encoded entire frames using a channel-coding scheme. This architecture is depicted in Figure 2.10. Solid blocks represent components that were present in the original Pixel-Domain Wyner-Ziv (PDWZ) codec, while dotted blocks were added to the subsequent Transform-Domain Wyner-Ziv (TDWZ) design to improve compression efficiency. In this scheme, even-numbered frames were encoded using conventional intra-coding techniques and treated as key-frames, while odd-numbered frames, referred to as Wyner-Ziv (WZ) frames, underwent Turbo encoding, and the resulting codes were stored in a buffer at the encoder side. A small portion of these codes were initially transmitted to the decoder, corresponding to the minimum theoretical rate required to decode the frame. Unlike PRISM, the Stanford Architecture included a feedback loop to request additional Turbo codes if the decoder was unable to reconstruct the current frame. The Stanford architecture later grew in popularity, and a number

of works were developed as enhancements of its original design [4, 8, 17, 28, 30, 37, 45, 49, 71, 78].

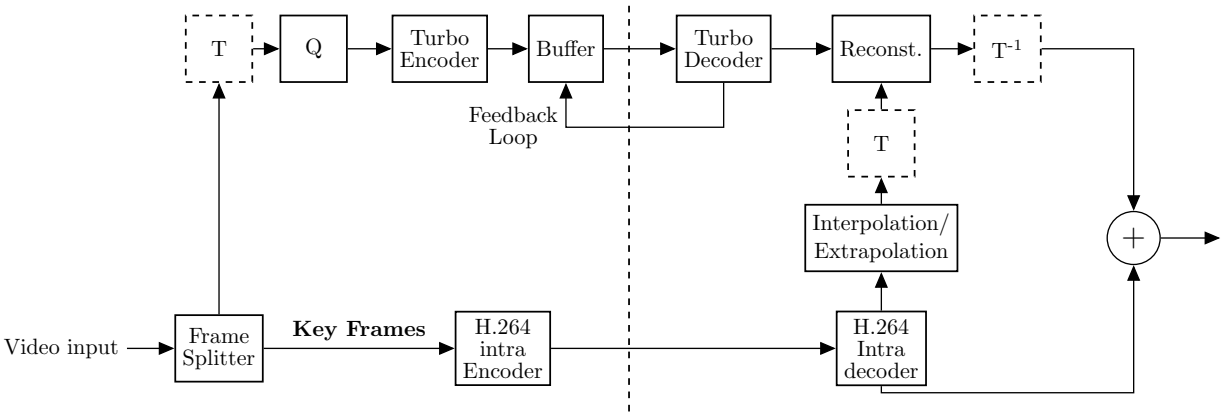


Figure 2.10: Stanford Architecture, from Dufaux *et al.*, [19]

2.2.5 DISCOVER

The DISCOVER codec (Figure 2.11) was the result of a joint European project, and was first presented in 2007 by Artigas *et al.* [4]. This codec was based on the Stanford Architecture TDWZ codec, but made significant advances in terms of rate-distortion (RD) efficiency and rate adaptability. The DISCOVER binary was also made publicly available online for download, which helped it to become the most commonly used reference codec in DVC literature.

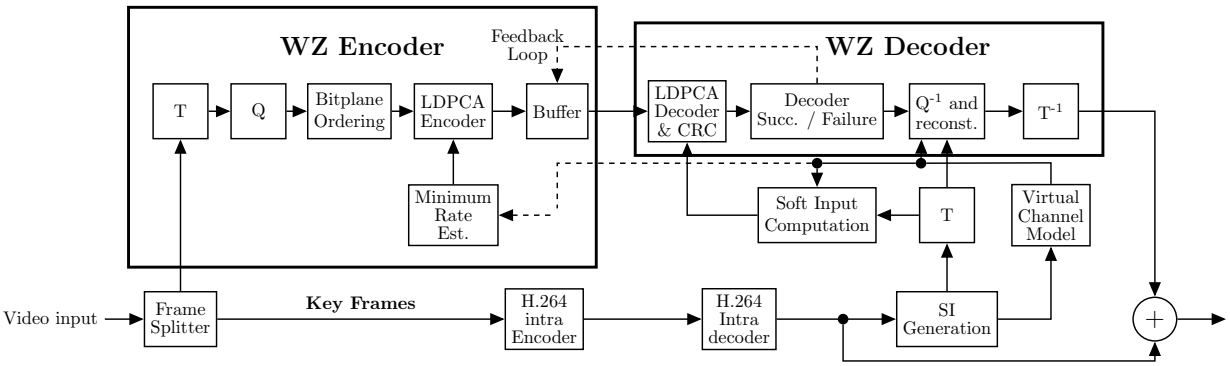


Figure 2.11: DISCOVER codec, from Artigas *et al.* [4]

Unlike the Stanford Architecture, DISCOVER divided video sequences into GOPs, similar to the strategy used by conventional video codecs. For each GOP, the first frame, called the key-frame, was coded using H.264 Intra-coding. The remaining frames in the GOP underwent WZ coding. In DISCOVER, this GOP was selected adaptively, with the GOP size selection based on activity metrics calculated at the encoder. DISCOVER's Encoder and Decoder modules are presented in detail below.

The encoder first divided frames into key and WZ-frames, based on the adaptive GOP size. Like the later TDWZ version of the Stanford codec, DISCOVER operated in the Transform Domain; the addition of

Discrete Cosine Transform (DCT) and inverse DCT blocks improved coding efficiency by taking advantage of psycho-visual redundancy. WZ-frame encoding started with a 4x4 DCT transform, and coefficients were organized into bands 1-16, where band 1 was the DC and 16 was the highest frequency AC component. Next, a uniform scalar quantizer was used for the DC band, while AC bands were quantized using a dead-zone quantizer. Frame data was reordered such that the bands from each DCT block were grouped together (e.g. all DC bands were binned within a single bit-plane, all band 2 coefficient were binned, etc.). Finally, each bit-plane was channel encoded using a Low-Density Parity Check Accumulator (LDPCA). LDPCA codes were chosen because they have been shown to approach channel capacity better than Turbo codes for the noisy virtual channel in DVC [4]. As with Stanford, the resulting channel codes were stored in a buffer, and only a small portion were transmitted initially along with CRC codes. In DISCOVER, this initial transmission rate was estimated adaptively by the "Minimum rate estimator" block.

A strategy called Motion Compensated Interpolation (MCI) was used to generate side information, which was then DCT transformed. A channel decoder was used to reconstruct the original frame using the provided side information as well as LDPCA codes received from the encoder. In the Virtual Channel Model block, a parameterized Laplacian function was used to compute the correlation noise model (CNM), with parameters estimated online at the decoder.

Decoder success/failure was determined in two steps. First, the decoder calculated the Hamming distance between the syndrome of received codes and the one generated by the LDPC decoder. Non-zero Hamming distances caused the decoder to continue iterating until some threshold number of iterations had been reached, at which point, a request for additional LDCPA codes was sent via the feedback loop. Once a Hamming distance of zero was achieved, then decoding success was further verified using the CRC codes. CRC failure also resulted in a request for additional LDPC codes [4].

Finally, after both checks had passed, the bit-plane was considered fully reconstructed by the LDPC decoder. Bit-plane information was then reordered, inverse quantized, and inverse transformed to produce the original frame.

2.2.6 Luma side information

In DVC decoders, side information generation refers to the process by which missing frames are predicted at the decoder. side information is used as input into a channel-decoder block, which reconstructs the frames by iteratively correcting the errors in this prediction [4]. Accurate side information reduces the number of errors to be corrected, lowering the number of channel code requests. Fewer requests reduces video bitrate, and also requires significantly less time to decode, so creating accurate side information is a vital component to DVC video coding [5].

Some of the simplest prediction methods estimate the current frame without motion estimation or compensation. One simple prediction method copies the entire previous frame, X_{i-1}, and uses that as the side information for the current frame, Y_i. Another simple method, linear interpolation, takes the frame average

of the previous and future reference frames as side information, $Y_i = (X_{i-1} + X_{i+1})/2$. However, the side information created through both of these simple techniques grows increasingly inaccurate for higher motion video sequences [5].

Motion Compensated Interpolation (MCI). MCI is *the* main technique used to generate side information, and is found in DVC codecs such as DISCOVER [4] and its derivatives. A block diagram of MCI is shown in Figure 2.12. MCI interpolates the current frame using two reference frames, one past and one future, which can be either key-frames or previously decoded WZ-frames. This MCI scheme works as follows:

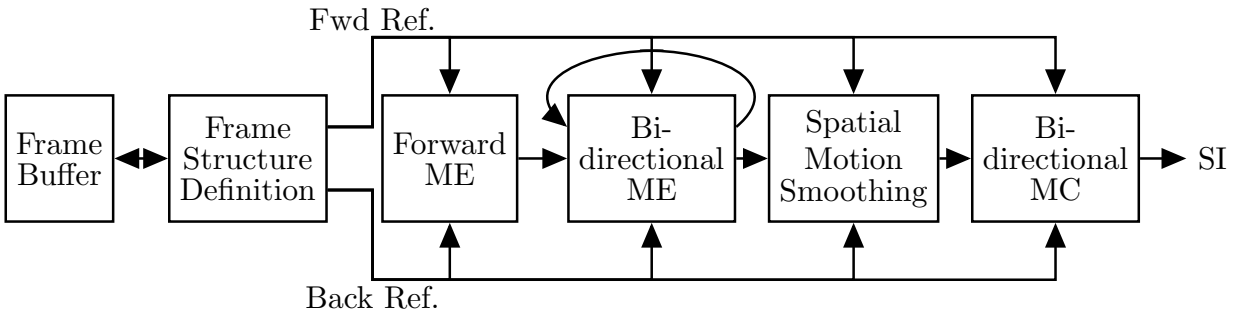


Figure 2.12: Motion Compensated Interpolation, from Dragotti & Gastpar [18]

1. First, both reference frames are sent through a low-pass filter to increase motion vector reliability.

2. Next, forward motion estimation is performed to create motion vector candidates between the past and future frames, as well as vice versa. A modified full-search ME is used that favours motion vectors that are closer to the origin [18].

3. Bidirectional motion estimation (BDME) is then used to improve the previous ME. For each block in the current frame, the motion vector that intercepts closest to the block centre is selected from the list of previous MV candidates [5]. Motion-estimation is then refined using an hierarchical block-size strategy. First, ME is performed using large block sizes (16x16) to track large object motion. Each block is then divided into smaller block sizes (8x8) and motion estimation is repeated with a smaller search window based around the candidate block match [18].

4. After the BDME step, MVs often have low spatial coherence, meaning that adjacent blocks have widely varying motion estimates. A technique called spatial motion smoothing helps to reduce the number of false MVs. This technique applies a weighted vector median filter to each MV, with weights determined by the matching success of each block [3]. The implementation of spatial motion smoothing is described in Section 3.5.3.

5. Finally, after spatial motion smoothing, the interpolated frame is created by applying bidirectional motion compensation (BDMC) [5], the same technique used to create B-frames in conventional video

codecs. BDMC uses a weighted average of two separate motion-compensated prediction blocks [76], in this case, one past and one future. Assuming that both frames are equidistant from the current frame, the same weight (1/2) is used for each reference block [5].

Many early MCI implementations used a fixed bidirectional structure (Figure 2.13a), where reference frames were required to be equidistant from the current interpolated frame. For this structure, GOP sizes were required to be a power of 2. DISCOVER researchers later implemented a flexible GOP size interpolation framework (Figure 2.13b) that allowed GOPs of any length to be used [6]. The flexible GOP size works by iterating over the following steps:

1. Select the longest frame distance between two key or previously decoded WZ frames, X_B and X_F.

2. Interpolate the frame X_i that sits between the chosen reference frames, such that

$$X_i = \lfloor (X_B + X_F)/2 \rfloor . \tag{2.11}$$

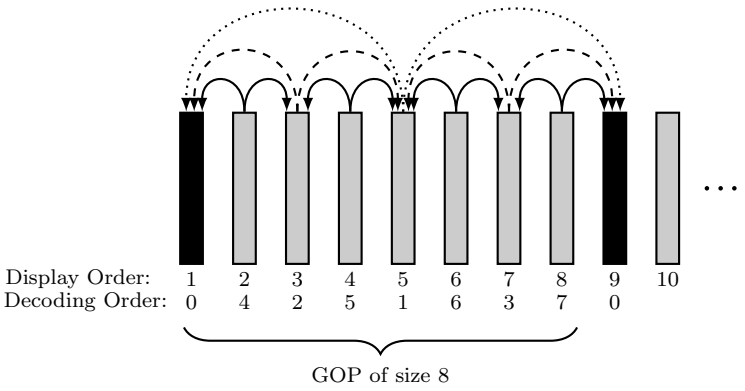
3. Repeat steps 1 and 2 until all WZ frames in the GOP are decoded.

Motion Compensated Extrapolation (MCX). MCX offers an alternative side information generation technique, using only past frames to predict motion. Motion vectors are again calculated between reference frames, but in this case, motion is extended or *extrapolated* from the past to the current WZ-frame. Extrapolation techniques have been studied extensively by Borchert, who proposed a novel three-frame MCX scheme [8, 9]. According to Borchert, extrapolation offers a tradeoff between prediction quality and key-frame cost. Compared to interpolation, extrapolation is less accurate on a per-frame basis, however, it uses a unidirectional prediction structure that offers larger and more flexible GOP sizes compared to the bidirectional structure used by MCI.

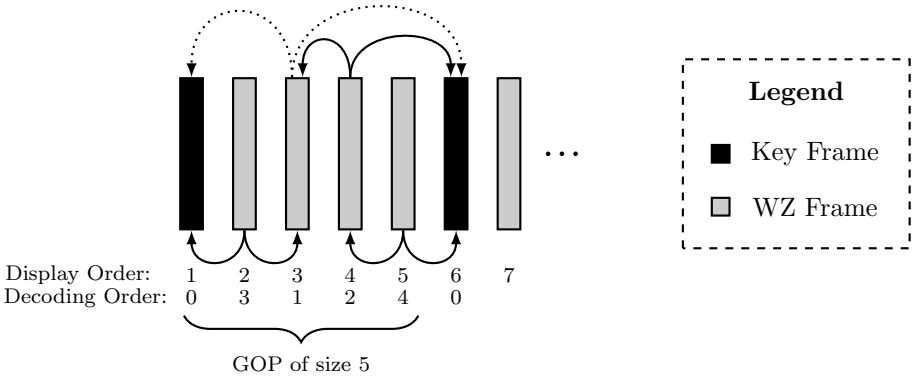
In a unidirectional prediction structure (Figure 2.13c), each step predicts only one frame into the future, meaning that frame distance is fixed regardless of GOP size. Similar to interpolation, extrapolation assumes there is no information available about the current frame, and that motion between frames is smooth and linear.

2.3 Related Work

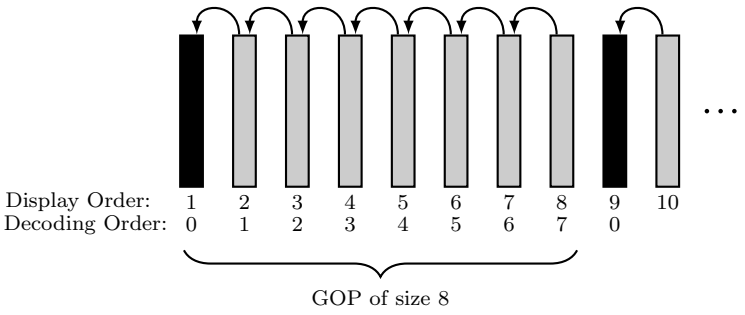
The following section describes related works that, in some form or another, exploit inter-plane correlation to improve video coding. Some works use *motion* correlation to reduce decoder computation, and enhance inter-frame prediction. Other related works take advantage of *spatial* correlation between colour planes to improve intra-frame prediction.



(a) Fixed Bidirectional



(b) Flexible Bidirectional



(c) Unidirectional

Figure 2.13: DVC Prediction Structures

2.3.1 Colour Video Support for DVC

As the field of distributed video coding is still in its early stages, support for colour video is currently lacking in most major DVC codecs [4, 18, 36]. Instead, research has focused on coding only the luma plane, while ignoring the Chroma components. To address the issue of colour-video support in DVC, Kodavalla & Mohan have designed a number of multi-plane DVC codecs based on the DISCOVER architecture [35, 36, 37]. Their first paper [36] focused specifically on feedback-free DVC coding, with Chroma prediction described as follows:

(a) The Chroma of each WZ frame was not channel-code. The PSNR of Chroma side information estimation was deemed high enough, and therefore channel codes and rate estimation were not required.

(b) No separate motion estimation was performed for the Chroma. Instead, the Luma motion vectors were used during Chroma side information generation.

(c) Chroma side information (like Luma side information) was generated at the decoder using the MCI method.

Their results demonstrated that the feedback loop could be removed from the DISCOVER architecture, while still maintaining comparable rate-distortion efficiency (especially for the Chroma planes).

Kodavalla & Mohan later expanded upon their original codec design [37], but with the re-addition of feedback channels to incrementally improve Luma side information accuracy. Following the original DISCOVER design, key-frames were coded using an H.264 Intra stream, with WZ-frames coded independently. Their main contributions were to encode each colour plane as a separate WZ bitstream, and to define quantization matrices for encoding DVC Chroma. At the decoder, half-pixel precision MVs were computed between reference frames within the Luma plane. As before, motion compensation was performed for all three colour planes using this same set of MVs This second codec design is shown in Figure 2.14.

This thesis relates to the work of Kodavalla and Mohan in a number of ways. This thesis work was similarly inspired by the lack of colour-video support in DVC, and sought to address that need. Also, similar to their work, Chroma motion-prediction was achieved by taking advantage of inter-plane motion correlation. However, in Kodavalla and Mohan's codecs, MVs were calculated solely between key-frames, and MV reuse was intended to reduce decoder computation [36]. In contrast, the Motion-Compensated Recolouring scheme (MCR) proposed in this thesis computes motion vectors between key-frames and the coincident Luma plane of the current frame.

2.3.2 Using Chroma to Enhance side information Estimation

Huang and Forchhammer proposed an enhancement to the DISCOVER architecture, which used a modified side information generation scheme (Figure 2.15) to improve upon the interpolation-based scheme used by DISCOVER [5]. As with the earlier DISCOVER design, MVs were computed between key-frames and

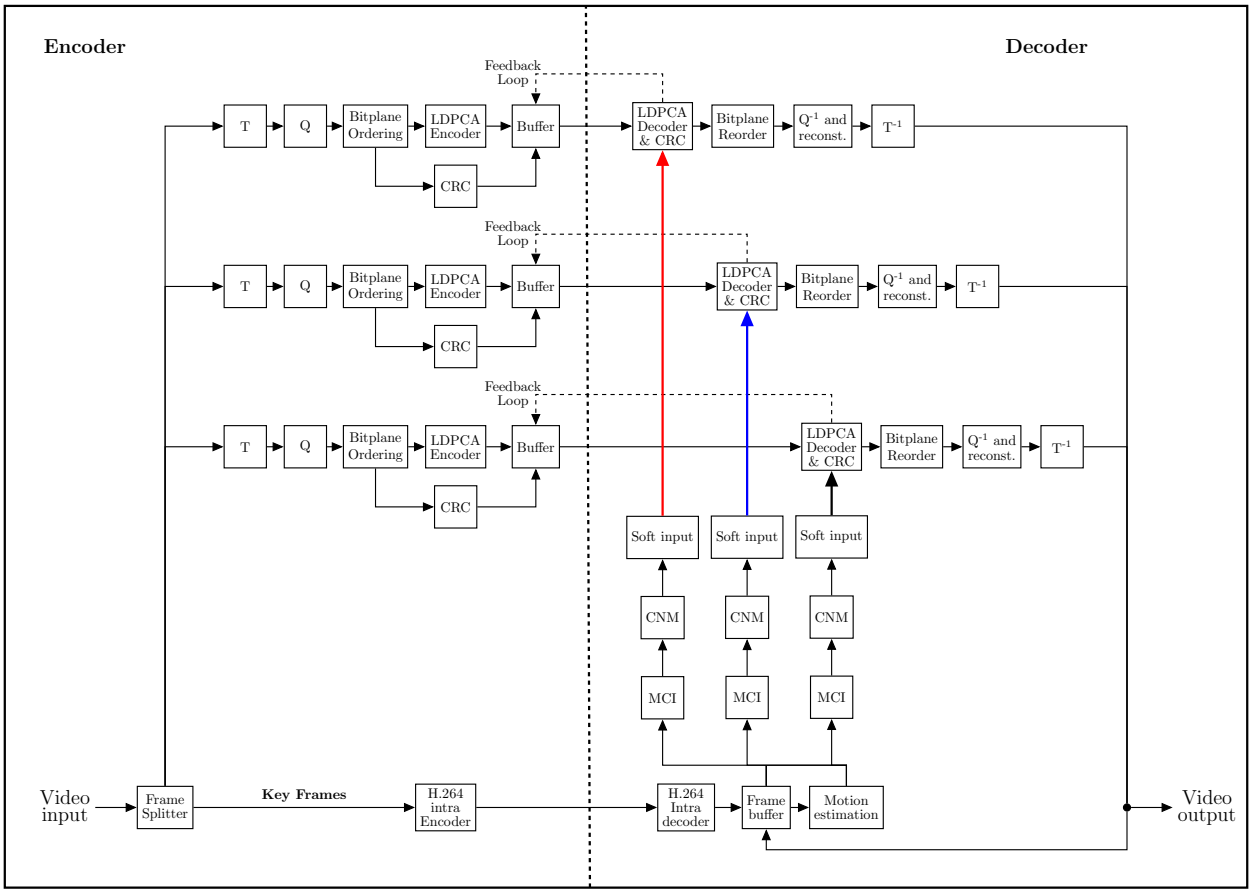


Figure 2.14: Multi-plane Codec, from Kodavalla & Mohan [37]

spatial motion smoothing was performed to increase MV coherency. Their work contributed the following improvements to DISCOVER's MCI algorithm:

- **YUV motion estimation.** According to the authors, prior MCI techniques did not use all information available at the decoder. Specifically, key-frame Chroma information was provided, but not used to assist in ME. Huang and Forchhammer developed a codec that performed block-motion search using all three key-frames colour channels. The block-matching criteria, MSE, was determined using the following equation:

$$\begin{aligned} \text{MSE} = & argmin\{\xi_{(m,n)\in block}\{(X^{Y}_{2i-1}(m,n) - X^{Y}_{2i-1}(m+\Delta m, n+\Delta n))^2\} \\ & + \lambda * \xi_{(m',n')\in block}\{(X^{UV}_{2i-1}(m',n') - X^{UV}_{2i-1}(m'+\Delta m', n'+\Delta n'))^2\}\}, \end{aligned} \tag{2.12}$$

  where $X^{Y}_{2i-1}(m,n)$ and $X^{UV}_{2i-1}(m',n')$ are the Luma and Chroma values at coordinates (m,n) and (m',n') in key frame X_{2i-1}, and λ is the weighting parameter to balance Luma and Chroma.

  $(\Delta m, \Delta n)$ and $(\Delta m', \Delta n')$ represent the corresponding Luma/Chroma motion vectors. The relationship between Luma/Chroma coordinates is given by the equations $\Delta m = 2\Delta m'$, $\Delta n = 2\Delta n'$, and $m = 2m' and n = 2n'$ for a YUV 4:2:0 format.

- **Variable Block Size ME Refinement.** After spatial motion smoothing, the codec applied an additional MV refinement step. Variable block-sizes (8x8 and 4x4) were used, because they could more accurately represent irregular motion.

- **Overlapped Block Motion Compensation.** Finally, a technique called Overlapped Block Motion Compensation (OBMC) was used to reduce blocking artifacts. More information on OBMC can be found in Appendix B.

Similar to the codec proposed in this thesis (MCR), Huang and Forchhammer's work exploited inter-plane correlation to enhance side information generation. However, there are significant differences between Huang and Forchhammer's design and the MCR codec. The former takes an interpolation-based approach, using

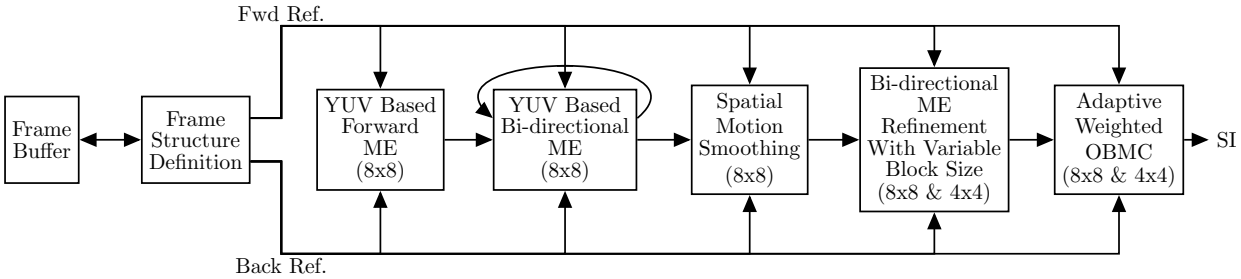


Figure 2.15: Huang and Forchhammer: Improved side-information generation scheme [28]

only key-frame information to predict MVs. The MCR codec uses information in a coincident plane of the current frame to ensure reliable MV accuracy. Furthermore, while MCR produces full-colour video, Huang and Forchhammer's codec does not, only using key-frame Chroma to enhance Luma estimation, and not to recolour WZ-frames.

2.3.3 Chroma Recolouring for Low-Power Video

The work of Hasan *et al.* [24] served as a direct inspiration for the experiments in this thesis. Like DVC, the authors designed a codec that moved the temporal prediction loop from encoder to decoder; unlike DVC however, this codec did not include channel code or the associated feedback loop to reconstruct the original video. In the Hasan *et al.* design, every N^{th} frame that was transmitted to the decoder contained both Luma and Chroma information, while the remainder were transmitted in grayscale (only the Luma planes). This effectively downsampled the Chroma data in the temporal dimension, reducing the total bitrate, but without an adverse effect on video quality.

At the decoder, the original colour video was reconstructed using the following method. Each incoming colour frame was buffered, while incoming grayscale frames were input into a "recolouring" module. Each frame was divided into non-overlapping Macroblocks (MBs), and for each MB in the grayscale frame, the absolute mean difference was calculated between the current and previous Luma. If this difference was below a predefined threshold, then the MB was classified as a "SKIPBLOCK", and the previous Chroma data was copied directly into the current frame. Otherwise, the MB was classified as a "MOTION BLOCK", and ME was used to determine the closest matching block in the previous frame. Specifically, the Three Step Search algorithm [38] was used, due to its computational efficiency. MC was applied to these blocks, and the results were combined with the SKIPBLOCKs into a motion-compensated prediction frame. Finally, the reconstructed results were written to an output video file, and buffered for use in the reconstruction of future frames. Algorithm 1 depicts the full recolouring logic.

This recolouring scheme is similar conceptually to the side information generation described in DVC literature. Specifically, the design is similar to MCX, in that it contains a uni-directional prediction structure, using previously-transmitted frames to estimate the current frame. Unlike MCX, the Hasan codec is able to exploit the information present in the current frame to assist in its motion-prediction. The proposed MCR codec aims to leverage the benefits of both strategies, including the use of coincident Luma information taken from the Hasan *et al.* codec, as well as bidirectional motion-prediction techniques taken from the DVC literature.

2.3.4 Chroma From Luma

Another research area that relates conceptually to this work is Chroma from Luma (CfL). CfL is a promising new technique explored in conventional video coding literature to predict the value of Chroma pixels from coincident Luma pixels, and use that prediction model in Intra-coding. So far, it has been adopted into the

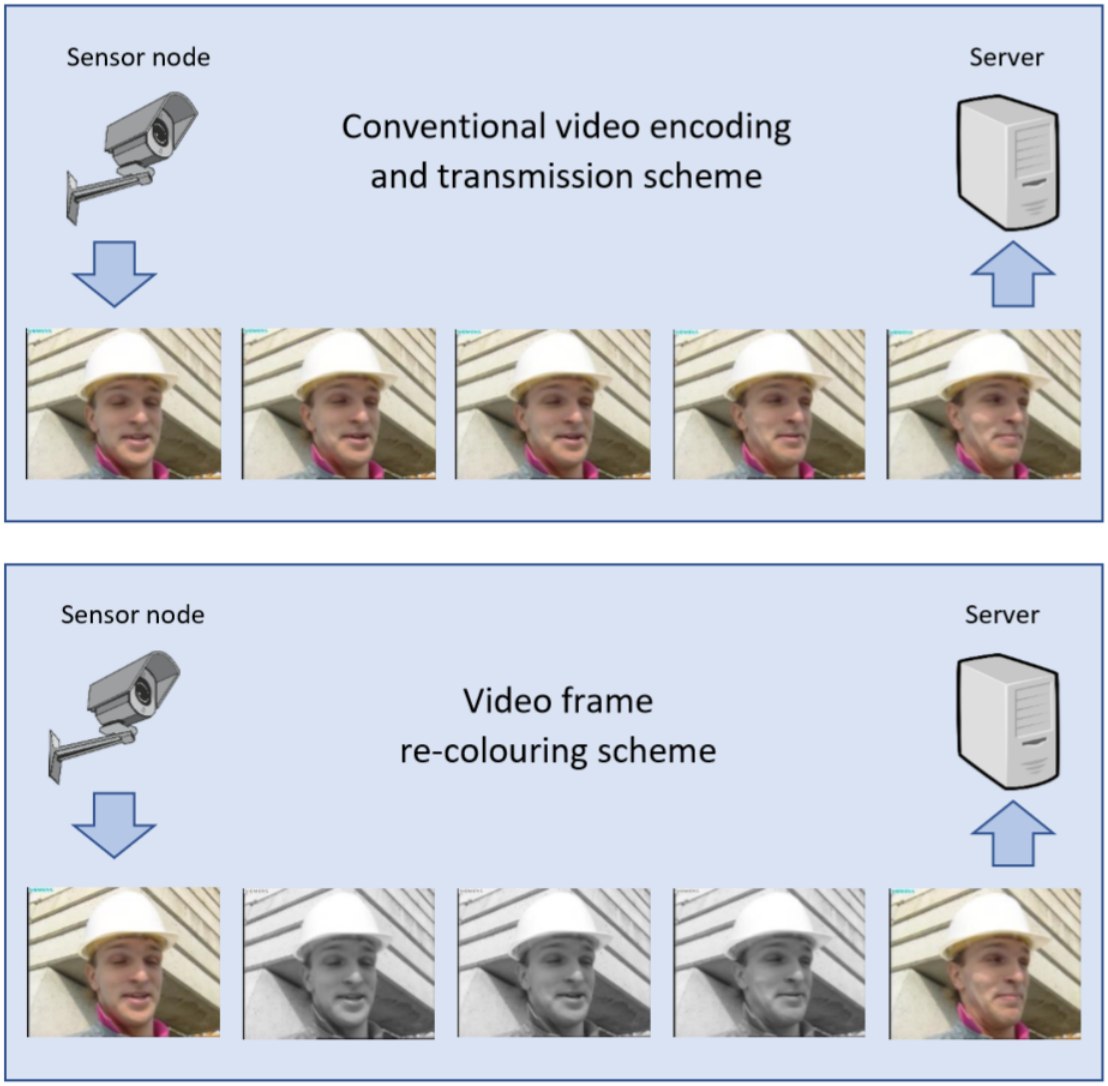


Figure 2.16: Hasan *et al.* recolouring scheme

Algorithm 1 Recolouring Algorithm

```
1: for each incoming frame do
2:     if frame is colour then
3:         buffer frame
4:         Y_p ← Y component of frame
5:     else
6:         for each Macroblock of frame do
7:             amd ← absolute mean difference w.r.t. coincident Luma (Y_p for corresponding Macroblock)
8:             if amd > threshold then
9:                 calculate Luma MVs using TSS algorithm
10:                apply motion compensation to reconstruct Chroma
11:            else
12:                copy Chroma from previous frame
13:            end if
14:            buffer reconstructed colour frame
15:            Y_p ← Y component of reconstructed frame
16:        end for
17:    end if
18: end for
```

Thor, Daala, HEVC, and AV1 codecs [67]. CfL works by modelling the linear relationship between Luma and Chroma pixels in a macroblock, and then using that model to estimate those Chroma pixel values. Lee *et al.* provided the first implementation of CfL, which works as follows: at the encoder, the Luma pixels in a macroblock are first encoded, reconstructed, and then downsampled to the same resolution as the Chroma components to provide a one-to-one pixel mapping. Next, two parameters, α and β, are calculated to model the linear relationship between this reconstructed Luma and the coincident Chroma pixels. These parameters are calculated as follows:

$$\alpha = \frac{R(L', C)}{R(L', L')}, \qquad \beta = \bar{\mathrm{C}} - \alpha \times \bar{\mathrm{L}}', \tag{2.13}$$

where $R(L', C)$ is the correlation calculated between the Chroma and Luma, and $\bar{\mathrm{C}}$ and $\bar{\mathrm{L}}'$ are the mean Chroma and Luma values in the Macroblock. Next, an estimation of each Chroma pixel value is calculated from the α and β parameters:

$$\hat{\mathrm{C}}(\mathrm{x,y}) = \alpha L'(x, y) + \beta, \tag{2.14}$$

where $\hat{\mathrm{C}}$(x,y) is the predicted Chroma pixel and $L'(x, y)$ is the downsampled and reconstructed Luma pixel at coordinates (x, y). The difference between the estimated and original Chroma pixels are then encoded and transmitted.

For CfL to work, the α and β parameters must also be present at the decoder; these can either be provided *explicitly*, as part of the video bitstream, or *implicitly*, by estimating the parameters from pixels in adjacent blocks. One example of an implicit CfL implementation comes from LG's LM mode [13]. In this mode, α and β parameters are estimated using the adjacent pixels directly above and to the left of the current macroblock (as shown in Figure 2.17). The adjacent Luma pixels must be downsampled (interpolated) so that they can be mapped to each Chroma pixel sample.

Next, the decoder calculates the linear regression model in Equation 2.14, using these estimated α and β parameters. Finally, the transmitted Chroma residuals are added to these estimates. Since the estimated parameters differed from the real parameters at the encoder, a degree of error is introduced to the reconstructed Chroma pixels. Despite this margin of error, CfL demonstrated an improvement in RD efficiency over other intra-coding techniques [13, 39].

The Daala codec took a different path with its CfL design. First, prediction was performed in the frequency domain not the spatial domain; this was the result of previous design choices in Daala that made frequency prediction more efficient. Second, Daala used a technique called Perceptual Vector Quantization (PVQ), a form of gain/shape encoding. PVQ obviated the need for α encoding while the use of frequency domain prediction removed the need for β encoding (because the mean of each AC component is always zero, there is no β offset). These design decisions allowed Daala to use CfL with very little overhead, requiring that only a sign bit be added per macroblock. Since the decoder had knowledge of all parameters, it was able to perform explicit CfL, which removed the linear fit error present in the LG LM mode and reduced

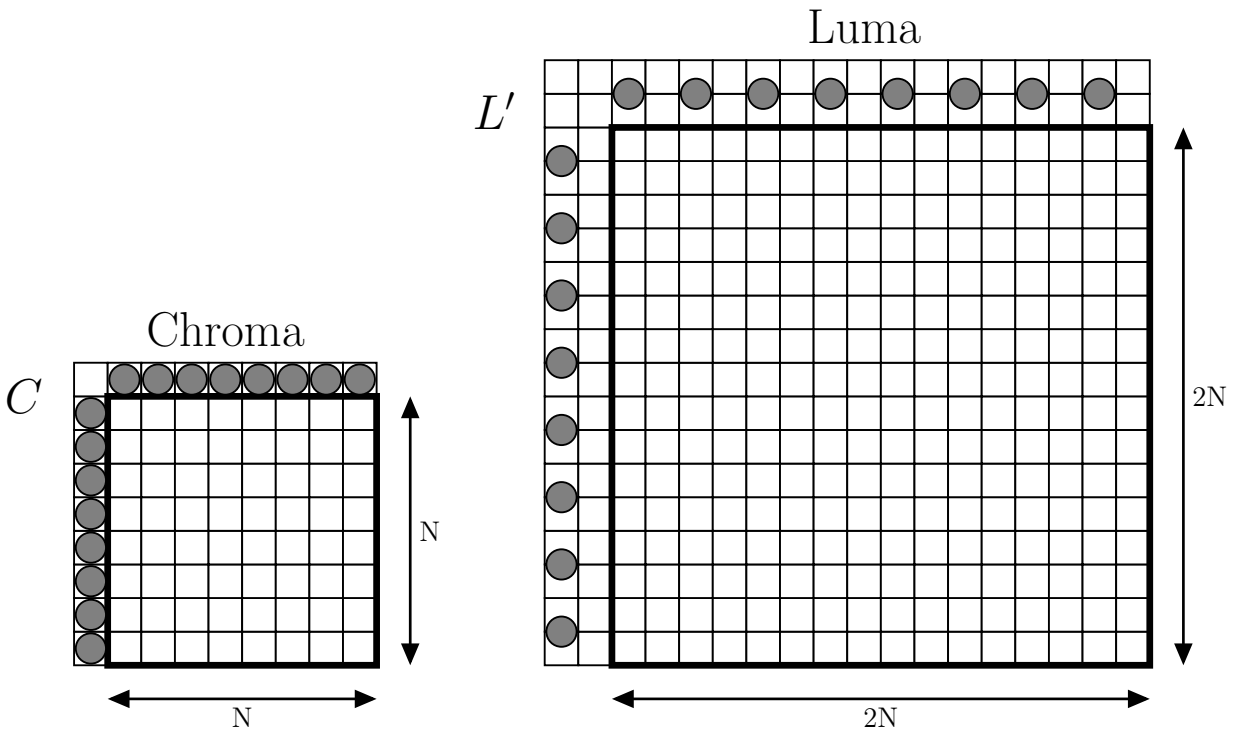


Figure 2.17: Chroma from Luma: Reference pixels [13]

decoder computation [70]. Table 2.2 was taken from a Xiph.org demo on CfL [48]. It summarizes the above descriptions, categorizing the designs according to their key traits.

Table 2.2: Chroma from Luma Research [48]

CfL Traits	**LM Mode**	**Thor CfL**	**Daala CfL**	**HEVC CCP**	**AV1 CfL**
Prediction Domain	Spatial	Spatial	Frequency	Spatial	Spatial
Bitstream Signalling	No	No	Sign bit	index + signs	joint sign + index
Activation Mechanism	LM_MODE	threshold	signal	binary flag	CfL_PRED
Requires PVQ	no	no	yes	no	no
Decoder Modelling?	yes	yes	no	no	no

CfL is conceptually related to the work in this thesis, in that it exploits inter-plane correlation, but it differs substantially in execution. The key difference is that CfL compresses *spatial* correlation between planes, while the proposed MCR technique takes advantage of *temporal* or *motion* correlation. In other words, CfL is used as a form of intra-frame prediction, while the MCR technique is inter-frame prediction, requiring motion-estimation at the decoder.

2.4 Summary

This chapter summarized topics in the fields of conventional and distributed video coding, as well as works relating to video recolouring and Chroma prediction. The chapter began by describing the theoretical foundations of lossless and lossy compression, before going on to detail techniques and design used by a practical video codec. Topics specific to *distributed* video coding were introduced including the theory of channel coding and the history and design of DVC codecs. Finally, the topic of inter-plane correlation was discussed including techniques that exploit both temporal and spatial correlation between colour planes.

3 Design & Implementation

3.1 Motivation and Comparison

The focus of this thesis is to explore the efficacy of motion-based video recolouring techniques. As mentioned in the previous chapter, the concept of using decoder-side Chroma prediction to reduce transmission rate was first proposed in the work of Hasan *et al.* [24], whose initial video-recolouring technique involved trimming Chroma information from the transmitted bitstream to save on transmission rate. The missing Chroma data was then predicted at the decoder to "recolour" the video.

The Hasan scheme was similar in motivation to distributed video coding (DVC); it shared the same goal of offloading computation to the decoder. However, unlike a DVC codec, the Hasan scheme did not use channel codes to reconstruct the original frame data to a desired quality. Instead, the Chroma that was estimated at the decoder was added without correction to the output video sequence. The increase in video distortion from Chroma prediction inaccuracies was taken as a tradeoff for the resulting savings in video rate.

This simple design offers a number of potential benefits. For example, Kodavalla and Mohan have previously demonstrated that decoder-side Chroma prediction, performed without the use of error-correction codes, improves rate-distortion efficiency over comparable DVC codecs that included Chroma channel coding [36]. However, Kodavalla and Mohan were exploring this within the context of a codec that combined Chroma estimation with DVC-style Luma prediction and that used channel encoding to correct the errors in this reconstructed Luma plane.

Another benefit of video recolouring is that it does not require channel coding, which simplifies the codec design substantially compared to a DVC codec. Both the encoder and decoder are affected by this design simplification; with WZ-encoding no longer necessary, the encoder can be implemented using a traditional H.264 encoder with added Chroma bitrate trimming. From the decoder, the complicated channel noise model and iterative channel decoder can be removed, leaving only the motion-prediction model (called side-information generation in DVC).

Finally, with the Hasan codec there is no feedback loop requirement. In DVC, the feedback loop is a point of contention in the literature, with two main schools of thought. Proponents of the feedback loop point out that it ensures decoding is successful and offers precise rate control, so that no more than the requisite number of channel codes must be transmitted to reconstruct each frame [72]. Other researchers have pointed to the disadvantages of feedback loops including prohibitive decoding delays, code buffering requirements, and high communication overhead introduced [36]. Depending on the embedded device, feedback requests

can also introduce additional power costs to keep the radio active, undoing the power-savings achieved with DVC. Therefore, a codec that can efficiently offload computation from encoder to decoder without requiring a feedback loop is desirable for many use cases. A Hasan-style recolouring codec could be especially useful in such real-time or near-real-time coding scenarios, and in other situations where hardware or timing requirements make a feedback loop unfeasible.

While the Hasan codec provided a good proof-of-concept for the use of video recolouring, there is further analysis that can be performed in this area, namely, comparison against traditional low-power video codecs such as Intra coding and advanced frame prediction techniques developed in the field of DVC. To that end, a video recolouring framework has been implemented as part of this thesis work which enables the comparison of various strategies for Chroma prediction within the decoder.

3.2 Video Recolouring Framework

A high-level block diagram of the proposed recolouring framework is shown in Figure 3.1. Unlike Hasan's original codec, the proposed recolouring framework includes intra coding using an H.264 video codec: the JM 18.5 reference codec, which is commonly used as a baseline in video coding research.[1] The use of the JM codec serves two purposes: first, it simplifies codec design, eliminating the need to re-implement intra video coding systems; second, it allows all gains in rate distortion made by the recolouring modules to be compared against a standard benchmark.

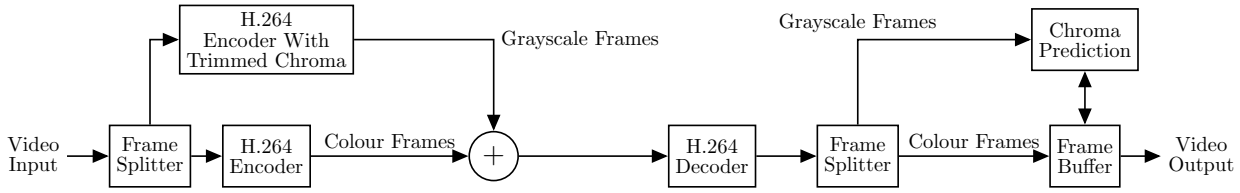


Figure 3.1: Proposed Design

At the encoder, frames are coded using H.264 Intra, and Chroma coefficients are "trimmed" from a subset of the coded frames, reducing the total bitrate. During video frame transmission, a single colour frame is followed by one or more grayscale frames, before another colour key-frame is transmitted. In this thesis, the grouping of 1 colour frame followed by N grayscale frames is referred to as a Chroma Group of Pictures (CGOP).

At the decoder, missing Chroma information was predicted using a video recolouring module, and re-added to the video. Three strategies for Chroma prediction were compared:

1. **HASAN** - the original video recolouring technique implemented by Hasan *et al.* [24]. For this thesis, a recolouring module was implemented according to the description provided in section 2.3.3.

[1] http://iphome.hhi.de/suehring/tml/

2. **MCI** - the Motion Compensated Interpolation technique used in the DISCOVER architecture [18], and described in section 2.2.5. This module was based off of an open-source implementation of the DISCOVER codec, OpenDVC [14], created by researchers at National Taiwan University. To re-use this module for video recolouring, the OpenDVC MCI algorithm was modified to interpolate frames in the Chroma planes rather than the original Luma.

3. **MCR** - A new technique called Motion Compensated Recolouring was created that combines aspects of the first two strategies. Like the Hasan codec, motion estimation is performed in the Luma plane and applied to Chroma plane, however, many of prediction strategies used within this codec were borrowed from the MCI design. As such, this MCR module was also implemented using OpenDVC as reference code.

The following section describes the implementation of each of these three strategies, including design decisions and assumptions.

3.3 Hasan Codec Implementation

In Hasan *et al.*'s original paper, there is no mention of Chroma subsampling being applied to remove redundant Chroma information. However, it does describe frame conversion from RGB to various colour spaces including YC_bC_r, YUV, YEF, YCgCo and HSV by multiplying the RGB by a constant matrix [23]. From this description, it can be inferred that the YUV colour format used was YUV4:4:4, where all three colour planes are transmitted with the same resolution (same width and height). A video recolouring scheme would be more effective on videos using a YUV4:4:4 colour format, because there is more Chroma information present in the raw video, and thus more potential bit-rate can be saved by efficiently removing it.

For the use case motivating this thesis work, raw videos are stored in the YUV4:2:0 format, where the Chroma planes both have a quarter the resolution of the Luma plane. Not only is video recolouring less effective for this colour format (due to the reduced potential bit-rate savings), the difference between Chroma and Luma resolutions also complicates the decoder design since the resolution now differs between the motion-estimation and motion-compensation steps.

To use the Hasan technique with this YUV4:2:0 use case and compare it against other proposed recolouring methods, the original design had to be modified by adding a motion-vector translation scheme. In keeping with the original codec's goal of both fast encoding and decoding, a simple scheme was chosen to minimize computation time. Motion vectors calculated from the Luma plane were halved and truncated, allowing them to be used at the lower resolution. This relationship is given by the following equation:

$$mvX_C = \lfloor \frac{mvX_L}{2} \rfloor \tag{3.1}$$

$$mvY_C = \lfloor \frac{mvY_L}{2} \rfloor, \tag{3.2}$$

where mvX_C and mvY_C are horizontal and vertical offsets of a Chroma block and mvX_L and mvY_L are the offsets of the corresponding Luma block.

As in the original design, the Three Step Search (TSS) algorithm was used to perform block-motion search [38]. The TSS algorithm pseudo-code is given in Algorithm 2. Figure 3.2 gives a depiction of the operation of TSS. The initial search grid is labelled "1", and centred around the current macroblock position. In the first iteration, the central block of the reference frame is found to be the best match, and so the search grid is reduced, while the keeping the same centre coordinates. In the second iteration (labelled "2"), the upper-right candidate is found to be the best match, and so the search grid is reduced yet again, but this time centred around the upper-right candidate block. A final iteration is performed, and the best match is returned as the search result. This TSS algorithm reduces the computational cost of motion estimation compared to Exhaustive Search (ES) by lowering the number of block comparisons required before a match is selected.

Algorithm 2 Three Step Search Algorithm

```
1: for each macroblock in the target frame do
2:     centre ← macroblock position
3:     for candidate in 3x3 grid around centre do
4:         match[i] ← SumOfAbsoluteDifference(candidate)
5:     end for
6:     centre ← Min(match)
7:     reduce grid area
8: end for
9: return centre
```

3.4 MCI Modified for Chroma Prediction

Motion Compensated Interpolation (MCI) was developed within the field of DVC, where it was used to interpolate frames missing at the decoder using macroblocks from temporally adjacent key-frames. These interpolated frame predictions were treated as a noisy representation of the original Luma frame at the encoder, and would be fed into a channel-decoding system to reconstruct the missing frame. For the purposes of this thesis, however, a modified MCI technique has been implemented to interpolate missing *Chroma* information, using the same motion prediction model in the original design.

As with the original Luma MCI, both motion estimation and compensation are performed in the same plane; for Chroma reproduction, this requires two separate pipelines to be used to compute the U and V Chroma planes independently. Finally, while the DISCOVER codec allows a flexible GOP size [18], OpenDVC was designed using a fixed bidirectional prediction structure (described in Chapter 2.2). This same fixed prediction structure was retained in the modified Chroma MCI scheme implemented for this thesis.

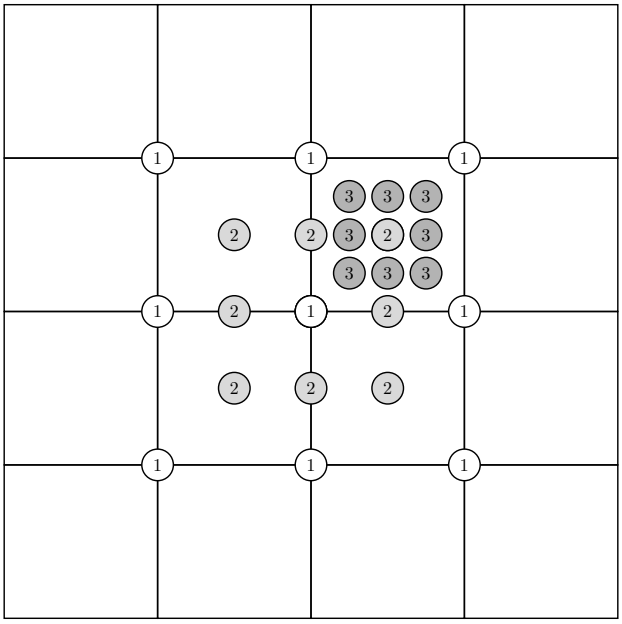


Figure 3.2: Three Step Search Algorithm

3.5 Motion Compensation Recolouring

The MCR recolouring strategy can be considered a combination of Hasan's Chroma recolouring and DISCOVER's MCI. A block diagram of MCR's design is shown in Figure 3.3, and can be broken into four main subcomponents: Chroma upsampling and decimation, Luma-ME, spatial motion smoothing, and weighted MC. Chroma upsampling and decimation is used prior to motion compensation to ensure that the motion field can be applied to the Chroma field at the appropriate scale. Luma-ME calculates the motion vectors within the Luma plane, spatial motion smoothing increases spatial coherence of the motion field, and weighted MC translates each reference frame by their respective motion field and calculates a weighted sum to create a prediction of the current frame. Each of these components is described in greater detail below.

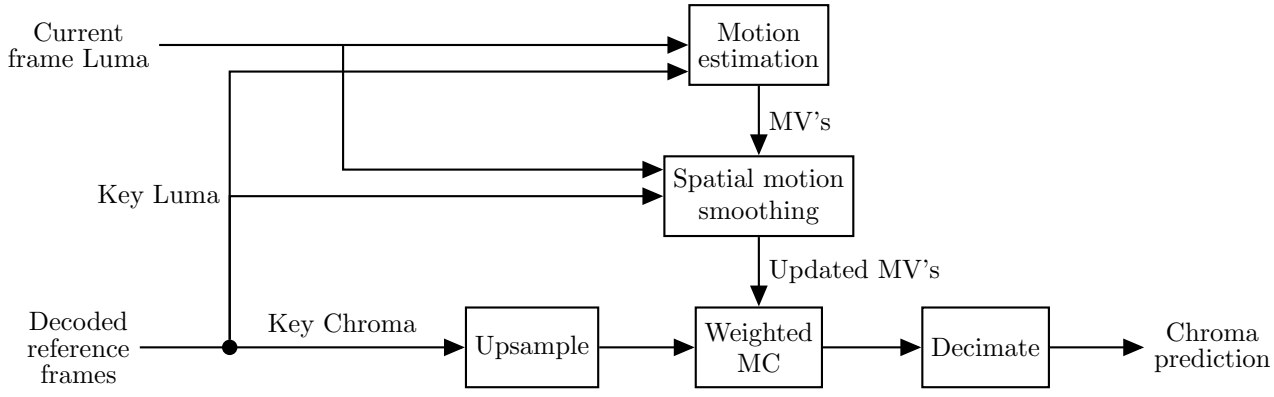


Figure 3.3: MCR Video Recolouring Strategy

3.5.1 Chroma Upsampling and Decimation

Like in the Hasan recolouring method, MCR performs motion-estimation in the Luma plane, and motion-compensation in the Chroma planes. However, the raw videos used for analysis have already been Chroma subsampled to a YUV:4:2:0 format, which causes a mismatch between the pixel resolution for these two steps. To circumvent this issue, the Hasan algorithm was modified for this thesis to divide and truncate the calculated motion vectors so they could be used at the correct scale. In the case of MCR, a more computationally expensive method is used at the decoder. First Chroma planes are upsampled to the same scale as Luma prior to motion compensation, and then downsampled or decimated after. This sequence of steps provides higher motion precision to the predicted Chroma.

The full sequence of upsampling, motion-estimation, motion-compensation, and decimation is as follows:

- First, both the U and V planes of each reference frame are upsampled to match the Luma resolution.
- Next, these upsampled planes are saved in memory (cached) to reduce computation steps for future grayscale frame recolouring.
- Each block in the reference Chroma planes is then fully motion-compensated by the pixel offsets calculated in Luma motion-estimation.
- Following motion compensation, the blocks are decimated down to the correct resolution, by sampling every second Chroma pixel value (see Figure 3.4).
- Finally, motion vector candidates from each reference frame are combined according to the weighted motion-compensation algorithm to create the Chroma frames.

This use of greater-than-integer precision in motion compensation has parallels in conventional video coding. Subpixel precision is used in codecs like H.264 and H.265 to improve prediction accuracy, though in the case of these codecs, both ME and MC are performed on the same colour plane. For these cases, integer motion estimation (full-pixel search) is calculated first, with half-pixel search calculated only in the area immediately surrounding the best integer MV. This approach may be extended using quarter-pixel search in the area around the best half-pixel match. Often, quarter-pixel search is skipped due to the massive increase in motion estimation computation, and comparatively small RD improvement [61]. In these conventional codecs, the approach of successive motion refinement is used to reduce the computational expense of subpixel search. While upsampling the entire frame and calculating MVs at higher resolution might increase accuracy, it also dramatically increases the number of motion candidates, requiring more computation. For MCR, however, it was decided that the entire frame should be upsampled, for the following reasons:

- In MCR's case, upsampling is from quarter-scale to full-scale (with regards to Luma resolution), so the number of motion candidates is no larger than usual for conventional ME.

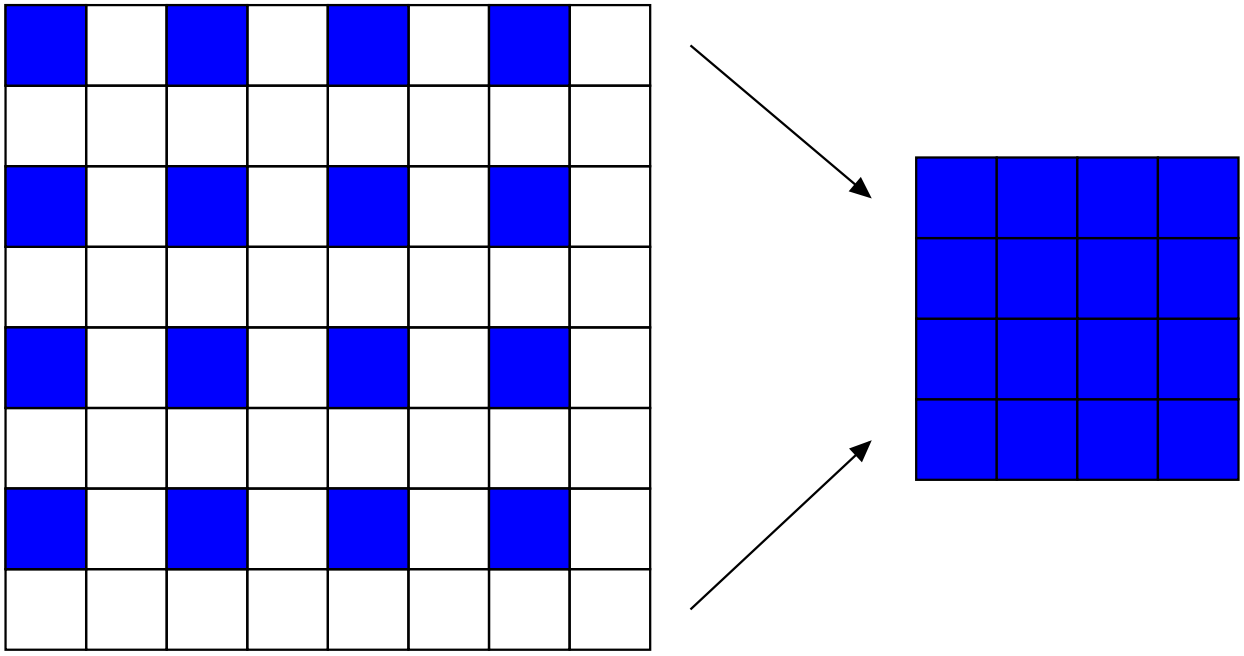


Figure 3.4: Block Decimation: down-sampling to quarter-scale

- It is easier to reason about motion vectors at the full Luma scale, rather than keeping track of two separate resolutions. Upsampling the Chroma allows Luma-scale ME, which simplifies programming complexity.
- ME is done at the decoder, where compute time is less of a concern.

3.5.2 Luma Motion-Estimation

The Hasan codec focused on reducing computation at both the encoder and decoder, using Three Step Search (TSS) and a SKIPBLOCK threshold algorithm to lower the costs associated with motion compensation. For the use case described in this thesis, decoding time was considered less of a hard constraint. Long computation times at the decoder would be acceptable if the corresponding increase in rate-distortion efficiency was sufficiently high. To that end, MCR explored both TSS and the more computationally expensive Exhaustive Search (ES) algorithm. Due to its higher prediction accuracy, ES was treated as the default algorithm used by MCR, while the use of TSS was treated as a separate decoding profile: MCR_{fast}.

One other difference between the motion estimation used by the Hasan codec and MCR is that the Hasan codec calculated only forward motion vectors from the previous key-frame, while MCR uses both the past and future reference frames. The MCR algorithm is able to reference frames set ahead of the current frame temporally by re-ordering the frames during decoding; this is possible since transmission and recolouring are performed in separate stages. The ability to use future reference frames increases the number of potential motion candidates and enables weighted motion-compensation, which improves the algorithm's predictive power. At the same time, these changes further increase MCR's decoding time compared to the Hasan codec.

3.5.3 Spatial Motion Smoothing

After motion estimation, it is common for the motion field to have low spatial coherence. ME is performed separately for each block without considering the motion of neighbouring blocks. These neighbours can often contain parts of the same objects; their motion may be correlated, which is potentially useful in determining the current block's true motion.

A technique called spatial motion smoothing can take advantage of this correlation, and help remove motion outliers [5]. In this algorithm, a weighted vector median filter is applied to each neighbourhood of block, with weights determined by each block's matching success [3]. In this case, Sum of Absolute Difference (SAD) is the measure for matching success, since that is the criteria used in the motion estimation algorithm. Weights for each block are determined from the following equation:

$$w_i = \frac{\mathrm{SAD}_B(\overrightarrow{v_c})}{\mathrm{SAD}_B(\overrightarrow{v_i})} \qquad i = 1, 2, ..., N\,, \tag{3.3}$$

where $\overrightarrow{v_c}$ is the candidate vector for the current block, and $\overrightarrow{v_i}, i = 1, 2, ..., N$ are the vectors computed for each of the 8 neighbouring blocks (as well as the current block, $N = 9$). $\mathrm{SAD}_B(\overrightarrow{v})$ is the SAD calculated between the current block and the block translated by a motion vector, $\overrightarrow{v}$. Better matches for the current block will have a smaller SAD, resulting in lower weighting.

From each of the 9 candidate motion vectors, the weighted vector median filter, selects the most "central" motion vector, $\overrightarrow{v}_{med}$, which minimizes the weighted *p-distance* from all other vectors. The weighted vector median filter equation is given below:

$$\sum_{i=1}^{N} w_i ||\overrightarrow{v}_{med} - \overrightarrow{v_i}||_p \leq \sum_{i=1}^{N} w_i ||\overrightarrow{v_j} - \overrightarrow{v_i}||_p, \qquad i, j = 1, 2, ..., N\,. \tag{3.4}$$

3.5.4 Weighted Motion-Compensation

MCR uses multiple reference frames for each prediction, and so is able to increase accuracy by employing a weighted-MC approach. According to Leontaris *et al.* [40], the use of multiple references can improve prediction accuracy over single-reference frame prediction for a variety of reasons, including the following:

- **periodic motion**: objects may disappear and reappear throughout the video sequence, so the best block match may not occur in the previous frame.
- **uncovered background regions**: objects in the previous frame may temporarily cover or occlude parts of the background, which may be visible in other coded frames.
- **lighting changes**: small, rapid changes in lighting may occur in the video. More reference frames increases the chance that shadow and lighting conditions match the current block.

- **Increased error resiliency**: noise can be introduced by the image capture device or during transmission. Multiple reference frames increases the number of possible block-matching candidates, improving error robustness.

A novel multi-hypothesis MC algorithm was implemented, and works as follows: first, a motion field is calculated using Chroma-ME between the current frame and each frame in the reference buffer. For each block motion candidate, a motion vector is stored containing the vertical and horizontal offset information, along with the SAD of the block match. Next, the algorithm iterates over each block in the current frame; the SADs of all candidates motion vectors for the current block are summed, and a weighted pixel averages are calculated from the following equation:

$$f(i,j) = \frac{\sum_{k=1}^{R} g_k(i + \hat{x_k}, j + \hat{y_k}) \cdot W_k}{\sum_{k=1}^{R} W_k}, \tag{3.5}$$

where $f(i,j)$ is a pixel at coordinates (i,j) of the current frame, R is the number of reference frames, and g_k is the k^{th} reference frame. $\hat{x_k}$ and $\hat{y_k}$ denote the horizontal and vertical offsets of the candidate motion vector from frame g_k, and W_k is the weight given to the k^{th} candidate. This W_k is calculated with the equation

$$W_k = \frac{1}{\text{SAD}_k + \epsilon}, \tag{3.6}$$

where SAD_k is the Sum of Absolute Difference calculated between the candidate and current blocks. A small value, ϵ, is added in the divisor to prevent divide-by-zero errors when a perfect match occurs (i.e. an SAD of zero).

3.6 Summary

In summary, a video recolouring framework was implemented as part of thesis work, allowing the comparison of various Chroma estimation strategy. A version of Hasan's original recolouring technique was implemented along with a modified version of MCI, the frame interpolation method taken from the DISCOVER architecture, and a new recolouring scheme, MCR, that combines aspects of the other two strategies. Finally, a second MCR profile was evaluated called MCR_{fast} that sped up decoding time by using the Three Step Search algorithm. Table 3.1 provides a high-level comparison of these algorithms in terms of prediction methods and modules used.

Table 3.1: Comparison of video recolouring techniques

	Hasan Method	**MCI**	**MCR**	**MCR Fast**
Motion Estimation Plane	Luma	Chroma	Luma	Luma
Motion Estimation Direction	Forward	Bidirectional	Bidirectional	Bidirectional
Block Search Algorithm	Three Step Search	Spiral Search	Exhaustive Search	Three Step Search
# Reference Frames	1	2	2	2
Skip Block	Yes	No	No	No
Spatial Motion Smoothing	No	Yes	Yes	Yes
Chroma Up/Down-sampling	No	No	Yes	Yes

4 Experimental Setup

This chapter details the setup of experiments used to analyze the proposed recolouring algorithms, and is broken into five sections:

- **Test Video Data** provides a description of the video collections used as data sets,
- **Comparison and Metrics** details the objective rate and quality measurements used to compare the recolouring techniques,
- **Experimental Parameters** describes the set of independent variables used throughout the experiments to tune codec performance,
- **Profile Configuration** outlines the different research codecs and profiles used as for base-layer encoding, and finally
- **Predicting Percent Chroma** describes the design of experiments used to predict the percent of an encoded video's bitrate that is required to encode Chroma information.

4.1 Test Video Data

Two sets of videos were prepared as datasets for the video recolouring experiments. The specific details of these datasets, including collection methodology and preprocessing are described below.

4.1.1 Benchmark Videos

The first dataset contained a number of benchmark videos, taken from the Xiph.org Video Test Media collection.[1] These videos were selected to represent a wide array of video content, from simple to complex scenes, and predictable to sporadic levels of motion. The videos are listed alphabetically in Table 4.1.1, along with their frame length and original source URL.

4.1.2 Canola Videos

The second set of videos was assembled as part of the P^{2}IRC COAST experiment; each video consists of a stationary camera observing a plot of Canola plants in a field. The videos were collected over the course

[1] https://media.xiph.org/video/derf/

Table 4.1: Xiph Benchmark Videos

Video Title	# Frames	Source
Akiyo	300	http://trace.eas.asu.edu/yuv/akiyo/akiyo_cif.7z
Flower	250	http://trace.eas.asu.edu/yuv/flower/flower_cif.7z
Football	260	http://www.cipr.rpi.edu/resource/sequences/sequences/sif/yuv/sif_yuv_football.tgz
Foreman	300	http://ise.stanford.edu/Video/foreman.qcif.gz
Mobile	300	http://trace.eas.asu.edu/yuv/mobile/mobile_cif.7z

of Summer 2016, with cameras programmed to capture 1280x720 resolution time-lapsed H.264 video during specific day-light hours. After collection, these H.264 videos were later converted into a series of JPEG images, one for each frame, and grouped into folders by camera and collection date.

The table below provides video metadata including the Camera ID, the series collection date, date of recording for each individual sequence, and the subfolder from which the data set was selected. Each subfolder corresponds with a single camera-day, and is numbered starting from a zero offset. In other words "Images 0" is the first date in the series, "Images 1" is the second, and so on.

Table 4.2: Custom Canola Videos

Camera ID	Date of Recording	Series Collection Date	Folder	Video Type
1109	23/06/2016	23/06/2016	Images 0	time-lapse
1108	18/07/2016	15/07/2016	Images 3	time-lapse
1109	23/07/2016	15/07/2016	Images 8	time-lapse
1108	16/08/2016	15/08/2016	Images 1	time-lapse
N/A	July/21/2018	July/21/2018	Summer2018	full-motion

Over the Summer of 2016, certain cameras had not been programmed correctly and so captured all days and nights for a given collection period into a single video sequence. The resulting images had been stored within a single folder. Scripts were developed to collate this data, partitioning it into subdirectories that corresponded with a single day (1440 images), and then remove the images captured during night hours, as these were deemed too dark to be usable.

One full-motion video was also included in the Canola video dataset. Unlike the time-lapse videos described above, this full-motion video was captured during the Summer of 2018, and at a higher resolution (1920x1080) compared to the Summer 2016 videos.

4.1.3 Data Preprocessing

The proposed codecs were designed to compress input videos in the YUV 4:2:0 planar format and at CIF resolution. The YUV 4:2:0 format (.yuv extension) contains only the uncompressed pixel values of the three

colour planes concatenated together. In contrast, many of the videos from the Xiph.org Video Test Media collection were represented in a YUV4MPEG2 format (.y4m extension); these contained uncompressed pixel data as well as a formatted header for each frame. The FFMPEG video command-line tool was used to convert the benchmark videos Y4M to YUV format (FFMPEG version 3.4.8-0ubuntu0.2). This conversion procedure is encapsulated within the script in Figure 4.1.

```
#!/bin/bash
for input in `ls /path/to/y4m/videos/*`
do
  basename=${input##*/}
  output=/path/to/yuv/"${basename%.y4m}".yuv
  ffmpeg -i $input $output
done
```

Figure 4.1: Format conversion script for Benchmark sequences

Similarly, the custom Canola images had to be converted before the videos could be encoded with the proposed codecs. 2016 Canola data was stored as a sequence of individual 1280x720 JPEG images, with a separate directory for each video. To prepare the video sequences for the recolouring codec, the JPEG images were cropped/resized to the appropriate resolution using the ImageMagick command-line tool "convert" (ImageMagick version 6.9.7-4), and FFMPEG was used once more to convert the images to the appropriate video file type. The bash script in Figure 4.2 was written to automate the entire conversion process.

```
#!/bin/bash
# crop entire sequence of N frames
for i in {1..N}
do
  # filenames are enumerated and padded with zeros
  inputName=/path/to/input/$(printf frame%06d.jpg $i)
  outputName=/path/to/output/$(printf frame%06d.jpg $i)

  convert inputName -resize 352x288! outputName
done

# Convert to .yuv format
ffmpeg -i /path/to/output/frame%06d.jpg output.yuv
```

Figure 4.2: Format conversion script for 2016 Canola sequences

This full sequence of steps: downsampling, format conversion, and video compression would represent the actual steps needed to create a low-quality preview video for a given sequence of Canola images. Unlike the 2016 data, the original 2018 Canola data was stored as a real-time H.264 encoded video, and so preprocessing for this video was reduced to a single FFMPEG command. The scale 352:288 was selected to match the expected CIF format, and 1000 frames from the original video were selected to reduce the size of the raw video generated. Finally, frames were chosen starting from the 30 second mark to avoid irregular video

artifacts and lighting that occurred at the start of the sequence. The final FFMPEG command is represented in Figure 4.3.

```
#!/bin/bash
ffmpeg -i /path/to/original/video.h264 \
    scale=352:288 \
    -ss 30 \
    -vframes 1000 \
    -c:v rawvideo \
    -pix_fmt yuv420p \
    Summer2018.yuv
```

Figure 4.3: Format conversion script for 2018 Canola sequence

4.2 Comparison and Metrics

Three types of metrics were considered as part of analysis, representing **objective video quality**, **computation time**, and **transmission bitrate**. A detailed explanation of these three categories is given below, along with the a description of the specific metrics chosen for the analysis.

4.2.1 Objective Quality Assessment

When evaluating video data compression and reconstruction, it is necessary to compare the reconstructed result against the original sequence. Similarly, there must be a means by which to compare the compression and reconstruction methods against one another. One simple technique is to visually inspect the results alongside the original and assign a *subjective* quality score, however, these subjective evaluations can be extremely time-consuming, especially when comparing long video sequences [53]. Instead, *objective* quality metrics have been created, with deterministic mathematical properties, and that have been designed to correlate well with the subjective results [66]. Furthermore, these objective evaluation methods produce simple, numerical outputs, which can be easily compared against one another.

Mean square error (MSE). MSE is an objective quality metric that is calculated using a pixel-by-pixel comparison of the reconstructed and original videos. MSE is the normalized cumulative squared error across the entire video sequence; it is given by the following equation:

$$MSE = \frac{1}{N_1 N_2 N_3} \sum_{n_1=1}^{N_1} \sum_{n_2=1}^{N_2} \sum_{k=1}^{N_3} \left(s[n_1, n_2, k] - \hat{s}[n_1, n_2, k] \right)^2 , \tag{4.1}$$

where N_1 and N_2 are the width and height, and N_3 is the number of frames in the video sequence [66]. Here, $s[n_1, n_2, k]$ represents a pixel in the k_{th} frame of the original sequence and $\hat{s}[n_1, n_2, k]$ is the corresponding pixel in the reconstructed sequence.

Peak signal-to-noise ratio (PSNR). PSNR is another common evaluation metric, calculated from the ratio of the maximum signal power, MAX, to MSE. In this instance, MAX represents the maximum possible pixel value, or 255, for an 8-bit integer pixel [66]. PSNR is calculated by the following equation:

$$PSNR = 10 \log_{10} \left(\frac{MAX^2}{MSE} \right) . \tag{4.2}$$

PSNR reaches its upper limit when the reconstructed image/video is identical to the original. In this case MSE is equal to 0, causing PSNR to approach infinity. When each pixel in the reconstructed video has the maximum possible difference from the original ($|s[n_1, n_2, k] - \hat{s}[n_1, n_2, k]| = 255$), then MSE is equal to MAX^2. This causes PSNR to drop to zero, its lower limit. PSNR values are expressed on a logarithmic decibel (dB) scale. MSE and PSNR are the most widely used quality metrics in engineering and video processing, because the values are easy to calculate and compare.

In video processing, three PSNR values are reported, because each colour plane (Y, U, and V) represents a separate signal. In the case of YUV 4:2:0 colour format, the common convention in video coding is to represent PSNR results as a 6:1:1 weighted average of Luma and Chroma PSNRs, as follows [26, 29]:

$$PSNR_{Avg} = \frac{6 \times PSNR_Y + PSNR_U + PSNR_V}{8} , \tag{4.3}$$

where $PSNR_Y$ is the average Luma PSNR, and $PSNR_U$ and $PSNR_V$ are average Chroma PSNRs computed over all frames in the video.

The structural similarity (SSIM). The SSIM index was designed to more closely approximate the human visual system. Given two $N \times N$ luminance blocks, $\mathbf{x}$ and $\mathbf{y}$, SSIM performs three comparisons: luminance, contrast, and structure. The luminance comparison is estimated as the mean intensities of the two signals, μ_x and μ_y. Signal contrast is estimated using standard deviation, σ_x and σ_y. Finally, the structure comparison uses the correlation between the signals, σ_{xy} [75]. The resulting SSIM equation is given below:

$$SSIM(\mathbf{x}, \mathbf{y}) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)} , \tag{4.4}$$

where c_1, c_2, and c_3 are small constants to avoid divide-by-zero errors.

For this analysis, PSNR was used to compare the recoloured videos. A major reason for selecting PSNR over SSIM was that there was an industry standard weighting equation for PSNR, where Luma and Chroma PSNR are combined to create a weighted average (as described above). This weighting captures the loss of video quality in the Chroma planes due to Chroma trimming and reconstruction, while accounting for the lower value that the Human Value System assigns to colour information. In the case of SSIM, there were examples of weighting equations that included the Chroma plane data, including two implementations called "CSSIM" [31, 25], but an industry-standard weighting such as exists for PSNR could not be identified by a search of the literature. In the end, the use of an industry-standard measurement ensures that the results in this thesis can be easily compared against other implementations and video coding strategies.

4.2.2 Bit Rate

Another important factor in evaluating a video compression technique is determining the bitrate of the compressed video, measured as the number of bits per second transmitted from encoder to decoder. For this thesis, implementation work focused on the decoder side. Modifying the existing JM encoder to trim Chroma coefficients would require a substantial software engineering effort, and this would only be beneficial if it could be proven that video recolouring offers significant bitrate and quality benefits. Thus, it is the goal of this thesis to demonstrate the potential benefits of video recolouring, while leaving a full encoder implementation to future work.

Instead of trimming Chroma at the encoder, the effective bitrate savings of this technique were modelled using data from the output statistics file. These statistics files were generated by the JM codec, and contained summary bitrate information broken into categories. Average bitrate per second was determined by taking the reported bits per frame, subtracting the fraction of Chroma coefficients that would be dropped, and multiplying the result by the frame-rate. This calculation is given by the equation below:

$$BR_{Avg} = (BR_{Tot} - BR_C * \frac{CGOP - 1}{CGOP}) * FR\,, \tag{4.5}$$

where BR_{Tot} is the average bits-per-frame for all bitrate categories and BR_C is the average bits-per-frame for only the Chroma coefficients. Finally, the entire equation was multiplied by FR, the video's frame rate, to derive the average bitrate in seconds.

4.2.3 Computation Time

The driving motivation behind the uplink model of video coding is to reduce the **computation** required at the encoder. Therefore, in this thesis, it was necessary to define a metric that could be used to quantify and compare this computation. One considered measurement was the **power consumption** of the encoding device. A prior work focused exclusively on power consumption analysis of DVC encoding [14]. The authors modelled power consumption as a simple sum of four values: transmission power P_t, coding power P_c, sensor power P_s, and the power consumption of a specialized video analysis engine P_a. These measurements can be easily acquired on an embedded device, but are less practical on commodity computers. Instead, for this thesis, **elapsed time** was used as a stand-in for computation. Elapsed time is easier to acquire than power statistics: the Unix "time" function can be called via a command-line terminal to measure the real, user, and system time of a running process. In this case, real or "wall-clock" time provides a reliable comparison of two encoding methods.

Two separate run-time environments were used for experimentation and analysis respectively. A laptop was used for preliminary codec development and regression testing. A second environment, consisting of a series of networked computers, was later used to perform the analysis. The hardware specifications for the two environments are shown in Table 4.3. All encoding and decoding times were recorded within the

"Analysis Environment" and the results are represented as the average time measured on the seven worker nodes.

Table 4.3: Experimental Environment Hardware

	Development Environment	**Analysis Environment**
# of Computers	1	8
ARCH	x86_64	x86_64
CPU	Intel Core i5-6200U @ 2.30GHz x 4	Intel Core i7-2600 @3.40GHz x 8
GPU	Intel HD Graphics 520 (Skylake GT2)	GeForce GT 1030
RAM	8.3 GB	16 GB

4.2.4 Video Characteristics

Prior to the recolouring analysis, measurements were taken of the raw video data set to determine if any content-specific characteristics impact the effectiveness of video recolouring. Two standard metrics were used: Spatial Perceptual Information (SI) and Temporal Perceptual Information (TI). These metrics are detailed in the International Telecommunication Union (ITU) Recommendation P.910 [59]. Higher SI values mean that a video is more spatially complex, while higher TI indicates that there is a higher degree of motion in the video. SI is calculated by applying a Sobel filter to the Luma plane of each frame (F_n). Next, the standard deviation across all filtered Luma pixels is computed, and the max result across the entire time series is returned as the value for SI:

$$SI = max_{time}\{std_{space}\left[Sobel(F_n)\right]\}. \tag{4.6}$$

TI is based on the frame delta, acquired by subtracting pixels in the current frame, F_n, from those in the previous frame, F_{n-1}. The value for TI are then computed by taking the standard deviation across all pixels in the frame delta, and returning the maximum result across the video sequence:

$$TI = max_{time}\{std_{space}[F_n(i,j) - F_{n-1}(i,j)]\}. \tag{4.7}$$

One down-side of these measurements are that they only capture global maximums, meaning that two video sequences with completely different spatial and temporal characteristics can end up with the same or similar SI / TI values. A future work may be to explore how SI / TI measurements evolve over the course of a video, and how those changing values impact the effectiveness of video recolouring.

Another important characteristic to analyze is the amount of Chroma information in the video. Video-recolouring codecs work by removing temporal redundancy from the Chroma planes, leaving the Luma plane

unaffected. In the coding scheme examined in this thesis, video recolouring is also combined with Intra-coding, which adds entropy coding overhead and mode signifier bits to the encoded bitstream that would likewise be unaffected by the video-recolouring. Since the recolouring strategy only compresses a fraction of the total video, its effectiveness will be highly dependent on the proportion of video bitrate taken up by Chroma in the encoded bitstream. In this thesis, the ratio between Chroma bitrate and total video bitrate will be referred to using the symbol $BR_{C/T}$.

To calculate this $BR_{C/T}$ ratio, bitrate measurements were extracted from the output statistics files generated by the JM codec. These coding statistics include a table of the average bitrate consumption per frame, broken into categories containing the Chroma and Luma coefficients per frame, as well as the bitrate incurred by frame overhead and intra-coding modes. Thus, $BR_{C/T}$ can be calculated using a simple division of two values:

$$BR_{C/T} = \frac{\text{Average Chroma Coeff. / frame}}{\text{Average Total Bits / frame}}. \tag{4.8}$$

4.3 Experimental Parameters

Originally, the experimental design of the MCR codec considered a large number of parameters. To reduce the scope of parameters considered during analysis, a set of initial experiments was run. During this experimentation phase, each parameter was selected in turn and varied over a suitable range, while all other parameters were kept the same. From the results of this initial testing, it was determined that two parameters: key-quantization parameter (Key-QP) and Chroma Group of Picture size (CGOP), had the greatest impact on the output metrics (PSNR and bitrate). For many other parameters, the differences in performance were miniscule or, beyond a certain point, the increase in computation was not worth the negligible increase in video quality.

For each of these other parameters, a single value was selected that offered a favourable tradeoff in terms of video quality and decoding time. These constants were combined into a video compression profile, shown in Table 4.4.

Table 4.4: Experimental parameters

Parameter	**Final Selection**
Key-frame QP	independent variable
CGOP	independent variable
Number of reference frames for motion estimation	2
Type of reference frames used (key / WZ-frames)	Key-frames only
Number of spatial-motion smoothing iterations	5
motion-estimation search range (Width x Height)	11x11
Macro-block size (Width x Height)	8x8

4.4 Profile Configuration

4.4.1 Intra Codec

For the video recolouring framework, two scenarios were considered. In the first scenario, both all Chroma recolouring schemes were compared against **H.264 Intra coding**, which was configured by varying the *QPISlice* parameter. For each QP, the entire video was encoded and reconstructed using intra-coded frames. Next, a video containing only the designated colour frames was encoded, by setting the *FrameSkip* parameter equal to $CGOP-1$. From these reconstructed colour-frames, Chroma was estimated for each of the remaining grayscale frames, and the resulting Chroma planes were multiplexed with the original, reconstructed Luma.

For this intra-coded video analysis, a "Theoretical Maximum" value was calculated, representing the optimal rate-distortion efficiency that could be achieved using video recolouring. This theoretical max curve assumed that all of the bitrate improvements from trimming Chroma, but maintained the same PSNR as the original Intra-coded video. In other words, the theoretical max PSNR was the same as the original intra-frame video PSNR for each GOP and QP parameter:

$$PSNR_{MAX} = PSNR_{Orig}\,. \tag{4.9}$$

The Theoretical Max bitrate was calculated using the same equation (4.2.2) used to calculate bitrate for the recolouring codecs:

$$BR_{MAX} = (BR_{Tot} - BR_C * \frac{CGOP-1}{CGOP}) * FR\,. \tag{4.10}$$

The combination of these two equations provided a useful upper bound on the potential for video recolouring as a video compression technique.

4.4.2 Inter No Motion

The second evaluation scenario combined video recolouring with a common DVC benchmark called **Inter No Motion** (Inter-NM). Inter-NM is a form of Inter-frame video compression that does not use motion estimation, instead using only the codec's residual coding capabilities. Inter-NM is often provided as a low-power alternative to DVC [18], and is generally implemented as a configuration of conventional video codecs (for example JM H.264). However, Inter-NM does not represent an official H.264 profile, and there is no clear, consistent definition of how it is implemented or configured across the DVC literature. The DISCOVER researchers described No Motion as H.264/AVC in the Main profile, configured to an IB...IB... frame structure, but without any motion estimation [18]. In this context, "IB...IB..." refers to the encoding strategy where all even frames are coded as Intra-only I-frames, and odd frames are coded as B-frames that use a combination of Intra-coding and Inter-coding based on data from previous and future frames. Ouaret defined Inter No Motion as having an IP...IP... structure, where the motion search range is configured to zero, and P-frame blocks "predicted from the co-located block in the previous I frame" [52].

In this thesis, Inter-NM was implemented as described by Ouaret [52]. This configuration was chosen because the description was clearer, and it mapped easily to JM codec parameters. Each GOP was encoded using of a single I-frame and multiple P-frames. To achieve this, JM was configured with parameters *"DisableSubpelME=1"*, *"SearchRange=0"*, *"NumberReferenceFrames=1"*, and *"IntraPeriod"* assigned equal to the CGOP size. Both I-frame and P-frame QP were assigned the same value (represented by the JM parameters "QPISlice" and "QPPSlice" respectively). As before, QP values were selected over the range 22-34 (even).

4.4.3 Inter Basic Profile

Finally, a configuration of the JM codec called "Inter Basic Profile" was used to compare its encoding times against those of the Intra an Inter-NM encoding profiles. Inter-BP used a similar configuration to the Inter-NM profile, with P-frame encoding, *"DisableSubpelME=1"*, and "QPISlice" and "QPPSlice" set to the same values. However, Inter-BP included motion-estimation and compensation steps, using the Fast Full Search (FFS) algorithm, the default block-matching algorithm used by the JM codec [65]. For this, the "SearchRange"was set to 16, and "NumberReferenceFrames" was set to 5.

4.5 Predicting Percent Chroma

While $BR_{C/T}$ might be a good predictor for the efficacy of video recolouring, there are some problems with using this measurement to make decisions about the video coding strategy. First, these coding statistics are highly dependent on the underlying Intra codec that is used. Not all video codecs produce such statistics files, or generate statistics that break down the average bitrate by category. Second, these statistics are available

only after the encoder has finished encoding the entire video, so would not be useful in selecting between a video recolouring strategy or a conventional Intra- or Inter-coding strategy at the encoder. Instead it would be helpful if there was a method for predicting $BR_{C/T}$ using measurements that could be calculated using the information available prior to encoding.

As a final analysis, this thesis explores how $BR_{C/T}$ can be predicted using spatial information measurements calculated in the Luma (Y) and Chroma (U and V) planes; these metrics will be referred to as SI_Y, SI_U, and SI_V, respectively. The ratio of Chroma to Luma SI, $\frac{SI_U+SI_V}{SI_Y}$, was selected as an explanatory variable, and a linear regression model was used to predict the dependent variable, $BR_{C/T}$. This model was implemented in Python using the Scikit-Learn Machine Learning library, and is represented using the following equation:

$$Y_{SIOnly} = \theta_0 + \theta_1 * \frac{SI_U + SI_V}{SI_Y}\,, \tag{4.11}$$

where Y_{SIOnly} is the predicted value of $BR_{C/T}$ and θ is the parameter vector used to weigh the effect the independent variable on the prediction. The reason for selecting $\frac{SI_U+SI_V}{SI_Y}$ as the explanatory variable is as follows: more spatially complex information has more entropy and so is more difficult to compress; if the combined Spatial Information of the Chroma planes is greater in relation to the Luma plane SI, then it is likely that the Chroma planes will take up a larger share of the bitrate.

At the same time, a model that uses SI alone as a predictor of $BR_{C/T}$ is quite simplistic; the equation assumes that all encoded videos contain the same ratio of Chroma-to-total bitrate, regardless of the encoding parameters used. In reality, video codecs tend to discard more Chroma information than Luma information for low-rate videos (when the QP value is high). It was theorized that the linear regression model could be enhanced with the addition of a second term representing the quantization parameter used to encode the video. Incorporating both the SI ratio and the desired QP value into the regression model results in the following equation:

$$Y_{SI+QP} = \theta_0 + \theta_1 * \frac{SI_U + SI_V}{SI_Y} + \theta_2 * \frac{QP}{QP_{Max}}\,, \tag{4.12}$$

where QP represents the target Quantization Parameter (QP) that will be used to encode the video and QP_{Max} represents the highest possible quantization parameter that can be used. One benefit of using these two fractional terms is that both are already normalized within the range 0 and 1. This helps to prevent a large term in one variable from being over-valued in the weighting calculation.

The values for θ were determined by fitting both models onto the dataset of benchmark and Canola videos described in Section 4.1. As in previous analyses, videos were intra-encoded using QP values from the range 22-34 even. The bitrate ratio for each encoded video was determined from the corresponding statistics file generated the JM codec. The parameters used in Y_{SIOnly} and Y_{SI+QP} were selected based on experiments run with this initial video dataset. A form of cross-validation called Leave-One-Out Cross-Validation (LOOCV) was used to estimate the predictive power of different measurements and arrive at the selected models.

Finally, the scores for both models were measured using 6 videos taken from outside this dataset, and encoded with a QP value randomly selected from the same range as before. These 6 external videos are de-

scribed in greater detail in Appendix C. The score itself was calculated using the coefficient of determination, R^2, a commonly-used metric in statistical data analysis.

5 Analysis

5.1 Why Use a Uplink Coding Strategy for Low-Power Video?

In conventional video codecs, both Intra- and Inter-frame coding are used to remove redundant information from a raw video and convert it into a compressed form that can be more efficiency transmitted. Inter-frame coding is used to remove temporal redundancy, significantly improving the Rate-Distortion (RD) efficiency of the compressed bitstream compared to an Intra-only approach. However, inter-frame coding also constitutes the majority of computation time at the encoder. A key motivator for this thesis was to investigate efficient strategies for moving computation down-stream to the decoder, in what is referred to as the uplink coding model. In such a model, fast coding methods such as intra-coding are still used at the encoder, while computationally expensive block-motion estimation is performed at the decoder.

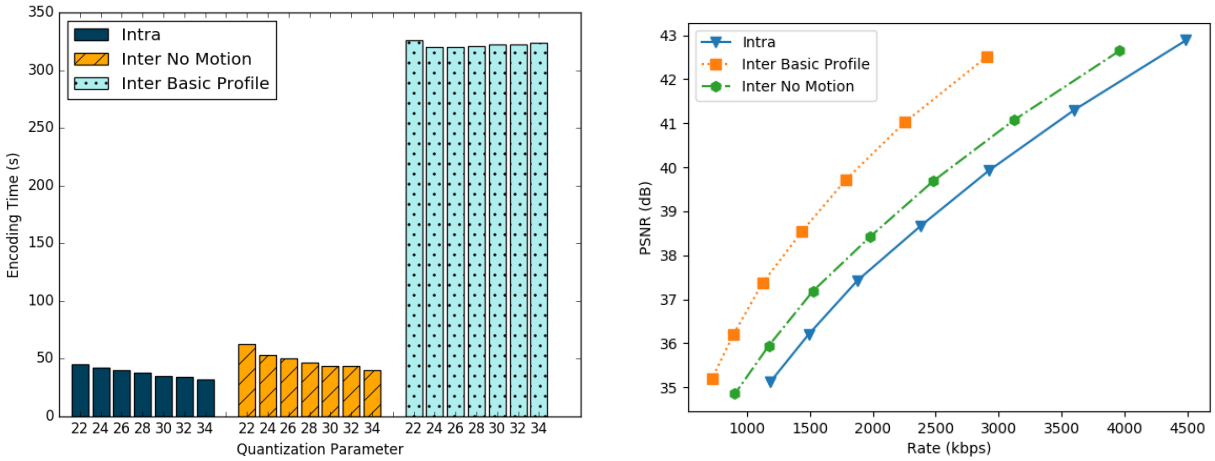
This section demonstrates the difference in encoding time between *Intra* and *Inter* coding profiles. As a sample test to compare Intra vs. Inter computation times, the JM codec was used to encode the entire Foreman sequence of 300 frames. Three encoding profiles were selected: "Intra", "Inter No Motion" (NM), and "Inter Basic Profile" (BP). Both Inter-NM and Inter-BP were configured using a GOP of 2, and an IP...IP... structure. The key distinction between these two Inter profiles is that Inter-NM does not actually perform any motion-estimation / block-motion search. Instead, the P-frame residuals are calculated by subtracting the previous frame as-is, without any motion-compensation.

Figure 5.1a displays the encoder computation time for both profiles using QP values over the range 22-34 even. With Intra coding, the entire sequence was encoded in under 45 seconds. Inter-NM required slightly more computation, requiring 63 seconds in the worst case. In contrast, Inter-BP encoding took substantially longer, requiring up to 326 seconds, roughly 7x that of Intra. It is clear from this that the inclusion of encoder-side motion-estimation significantly increases computational overhead compared to the fast Intra and Inter-NM profiles. In a resource-constrained environment, such an increase to encoding time could rapidly consume an embedded encoder's energy budget.

It should be noted that JM is primarily used for research purposes, and that faster, optimized H.264 alternatives do exist. For example, the x264 codec has been shown to encode between 20-50 times faster than JM for the same PSNR [46]. However, these faster codecs take advantage of parallel computation available on certain hardware architectures; such parallelization may not be possible on a resource-constrained systems, or may itself increase the total energy consumption.

Figure 5.1b shows the respective RD curves of the Intra, Inter-NM, and Inter-BP profiles. Due to the lack

of motion-estimation in Inter-NM, the P-frame residuals contain higher energy, and so Inter-NM offers less bitrate savings compared to Inter-BP. For example, at the highest QP value, Average Intra bitrate is 4385 kbps, whereas Inter-NM is 3870 kbps, reducing bitrate by around 11%, with only a small drop in perceived quality (Intra has a PSNR of 42.88 dB versus Inter-NM's PSNR of 42.64 dB). Compared to Inter-BP's bitrate of 2842 kbps (35% bitrate reduction) and a comparable PSNR of 42.52 dB, it is clear that motion estimation offers a significant boost to RD efficiency.



(a) Encoding Time **(b)** RD Curves

Figure 5.1: Foreman: Intra vs. Inter Encoding Configurations

This analysis showed both the computational costs of motion-estimation as well as the benefits in terms of RD efficiency. The uplink model codec aims to effectively leverage the RD benefits of motion estimation while reducing power consumption by limiting the computation at the encoder.

5.2 Video Characteristics

The hypothesis in this thesis is that these recolouring algorithms are content-specific, working more effectively for some videos than others. Therefore, it is important to have a vocabulary for describing videos characteristics in objective terms, to allow later analysis sections to interpret video coding results. This section presents measurements taken of the benchmark and Canola videos that represent the videos' spatial and temporal information, as well as a breakdown of video bitrates by category.

As expressed in Chapter 4, Spatial Information (SI) and Temporal Information (TI) are two measurements that are often used to describe a video's contents. Figure 5.2 provides a graphical representation of these values, while Table 5.1 contains all SI and TI measurements calculated for the Benchmark and Canola videos.

In Figure 5.2, it can be seen that the Benchmark videos cover a wide range of SI and TI measurements; Akiyo has both low spatial and temporal complexity, while Foreman and Football have mid-range SI measurements but high temporal complexity, and the Flower and Mobile videos have both high SI and TI. By

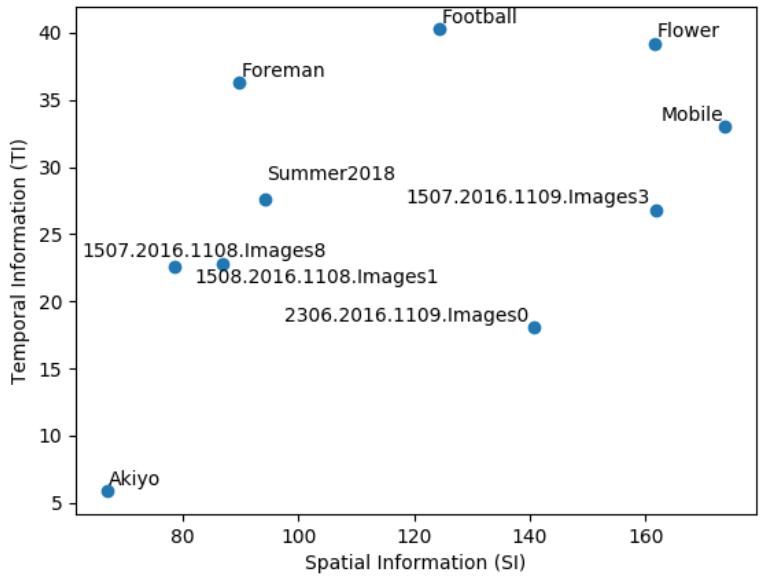


Figure 5.2: SI / TI Graph

comparison, the four Canola videos vary less in terms of TI, but cover a wide range of SI values.

Figures 5.3a and 5.3b break down the video bitrates by category for the benchmark video and Canola video datasets. These bitrates were averaged across QP values, with standard deviation represented using vertical error lines for each category. There are a number of conclusions that can be drawn from these graphs about the amount of Chroma in the encoded bitstream ($BR_{C/T}$). First, one can see that Chroma coefficients do not constitute the majority of an encoded video's bitrate, (between 4-19% of the encoded video when averaged across the full range of QP values). Second, most Canola videos have a smaller $BR_{C/T}$ ratio compared to the Benchmark dataset. Third, it is apparent that there is a high degree of variability in how bitrate is apportioned between the "Luma", "Chroma", and "Other" categories.

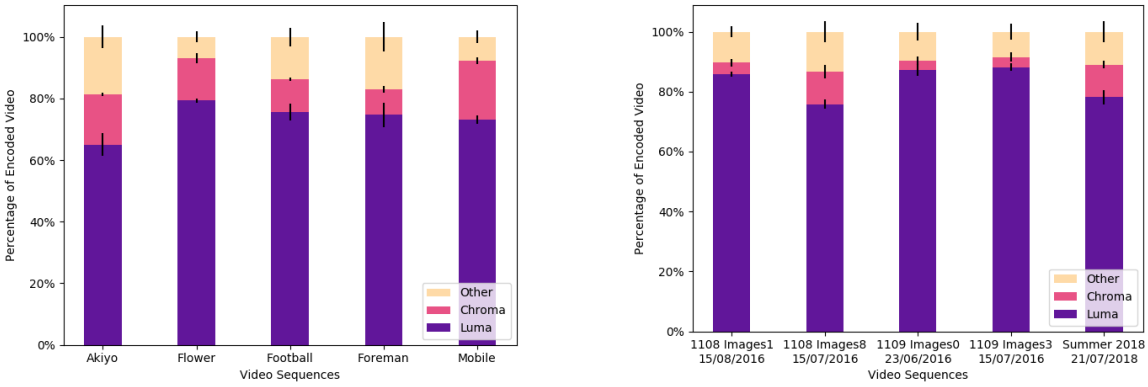
This third observation indicates that the Intra-frame codec might be more effective at compressing Chroma for some video content than for others. This variability also means that video-recolouring, when combined with Intra-coding, may be used as a kind of content-specific video compression optimization technique, and points to the need for an objective metric to help identify when the technique is useful. In the next section, the performance of different Chroma prediction strategies are evaluated, and the effect of SI, TI, and $BR_{C/T}$ on video recolouring are observed.

5.3 Video Recolouring On Intra-coded Sequences

The following section contains the Rate-Distortion analysis for video recolouring. The key video characteristics (Spatial Information, Temporal Information, and Average $BR_{C/T}$) are summarized in Table 5.2 below.

Table 5.1: Spatial and Temporal Information Measurements

Video	SI	TI
Akiyo	66.9	5.9
Foreman	89.7	36.3
Football	124.3	40.2
Flower	161.6	39.2
Mobile	173.7	33.0
1507.2016.1108 images8	86.9	22.8
1508.2016.1108 images1	78.6	22.6
1507.2016.1109 images3	161.8	26.8
2306.2016.1109 images0	140.8	18.1
Summer2018 (full-motion)	94.3	27.6



(a) Benchmark Videos **(b)** Canola Videos

Figure 5.3: Proportion of video bitrate by category

Here, average $BR_{C/T}$ values are depicted, measuring the mean across all encodings and QP values. The following subsections have been grouped according to video characteristics to help emphasize the effects they have on the proposed recolouring strategy.

Table 5.2: Benchmark Video Characteristics

Video	SI	TI	$BR_{C/T}$ (Average)
Akiyo	66.9	5.9	16.30%
Foreman	89.7	36.3	8.32%
Football	124.3	40.2	10.65%
Flower	161.6	39.2	13.79%
Mobile	173.7	33.0	19.14%

5.3.1 Low Temporal, Low Spatial Information

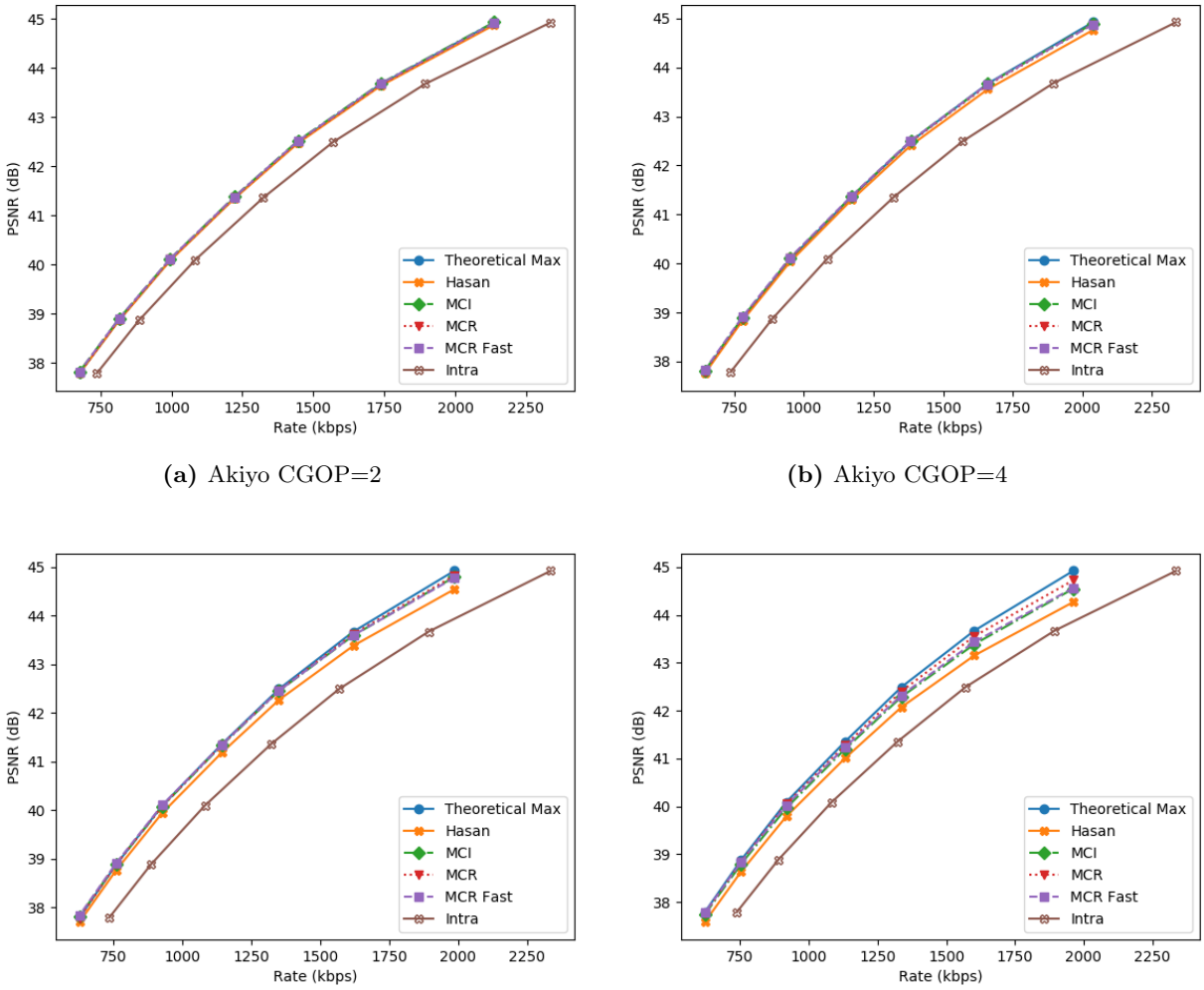
First, the four recolouring methods were compared for the "Akiyo" benchmark video (Figure 5.4), which depicted a newscaster talking in front of a static background, with no global camera motion. This video is characterized by both low SI and low TI values.

Figure 5.4: Akiyo Video

Figure 5.5 shows the results of applying video recolouring to this Akiyo sequence, with a separate RD graph presented for each of the CGOP values selected (2, 4, 8, 16). For this low-motion sequence, all recolouring strategies perform better than H.264 Intra alone, and all achieve very close to the Theoretical Maximum RD-curve; prediction accuracy is especially good given smaller CGOP values (Figures 5.5a and 5.5b), where the prediction curves are hardly distinguishable from the Theoretical Maximum curve. Applying larger CGOP values of 8 and 16 (shown in Figures 5.5c and 5.5d respectively) improves the bitrate savings afforded by video recolouring, but has a deleterious effect on objective video quality for the various video recolouring techniques. Overall, the MCR recolouring scheme performs the best in terms of RD efficiency,

though this difference is marginal unless both video quality and CGOP size is high.

The high accuracy of Chroma prediction can be explained in part by the extremely low video motion; the action in the video sequence consists of small, predictable movements, and so information in the neighbouring frames is a good predictor of the state of the current frame. Another important factor is that Chroma makes up a large percentage of total video bitrate for Akiyo (16.3%), meaning that there is a greater potential for bitrate savings by removing Chroma from the encoded bitstream.



(a) Akiyo CGOP=2

(b) Akiyo CGOP=4

(c) Akiyo CGOP=8

(d) Akiyo CGOP=16

Figure 5.5: Chroma Recolouring: Low Motion

5.3.2 High Temporal Information, Mid-Range Spatial Information

Next, the efficacy of video recolouring was observed for videos with High Temporal Information and Medium Spatial Information. Two sequences were used: "Foreman" and "Football" (Figures 5.6a and 5.6b). In the Foreman video, this motion is the result of a sudden change in camera perspective, whereas motion in the

Football sequence comes from both global camera motion, and the sudden, sporadic movements of players on a field. Both videos contained a mix of spatially complex foreground information and a spatially simple background, resulting in mid-range SI scores.

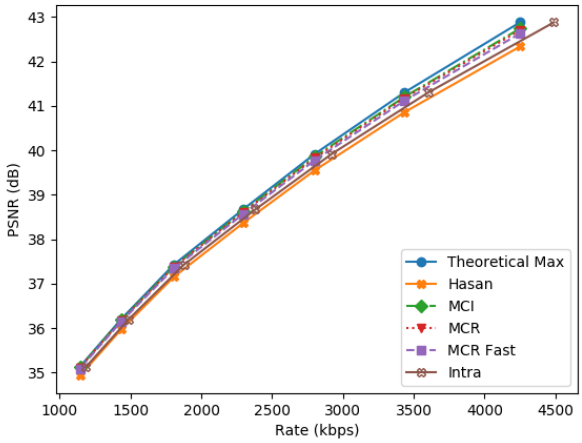
(a) Foreman **(b)** Football

Figure 5.6: Foreman and Football Videos

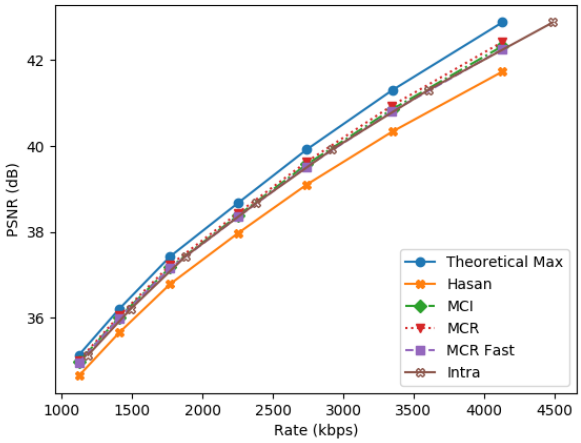
As seen in Figures 5.7 and 5.8, none of the three recolouring strategies are particularly effective for these two sequences, and few recoloured videos outperforms the baseline Intra coding method. The best rate-distortion performance in this category comes from recolouring the Foreman sequence with MCR and a CGOP of two (Figure 5.7a). For both videos, however, video recolouring techniques show a particularly rapid decline in their prediction quality as CGOP increases. For example, when the Foreman video is recoloured using the MCR technique with a CGOP of 16, the resulting video has a 1.3 dB decrease in PSNR compared to the Intra-coded baseline in the worst case. The Football video suffers even worse quality degradation; when encoded using the MCR recolouring scheme and a CGOP of 16, the worst-case result is a 3.2 dB decrease in PSNR.

One possible reason for the poor performance of the video recolouring techniques is the complex motion observed in these videos, which leads to poor motion-prediction performance. It is also important to note that both videos contained a relatively small proportion of Chroma information compared to the other benchmark videos. After Intra-coding, Football had an average C/T ratio of 10.6%, while Foreman had a C/T ratio of only 8.3%. In the case of these two videos, the loss in video quality was not worth the bitrate savings when such a small percentage of the video is affected by the bitrate optimization technique.

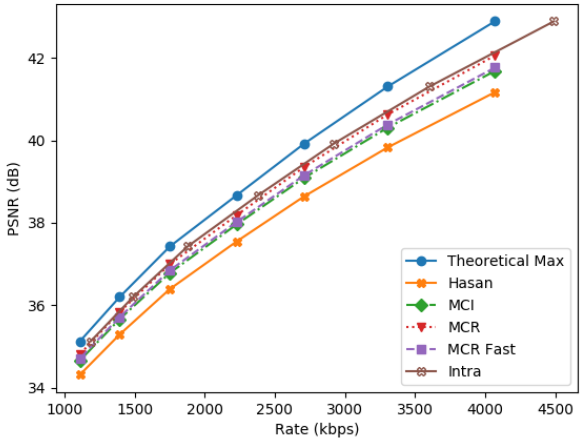
Another interesting observation is that MCR performs better than MCI for these two videos when CGOP is high. For the Foreman sequence, MCR achieves 0.4dB higher PSNR for Foreman at a CGOP of 8 and 0.6dB higher PSNR at a CGOP of 16. For the Football sequence, MCR achieves roughly 0.3dB higher than MCI for CGOPs 4, 8, and 16. For all other video sequences and CGOP values observed, there is virtually no difference between the prediction accuracy of MCR and MCI, and so these examples stand out as an exception. It is hypothesized that MCI's worse performance is due to the more sporadic motion of these sequences, which is



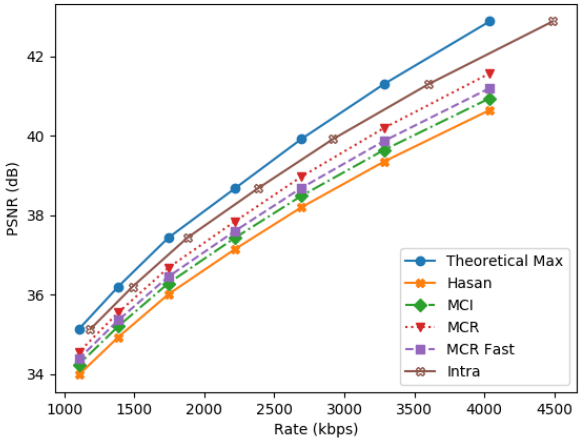
(a) CGOP=2



(b) CGOP=4



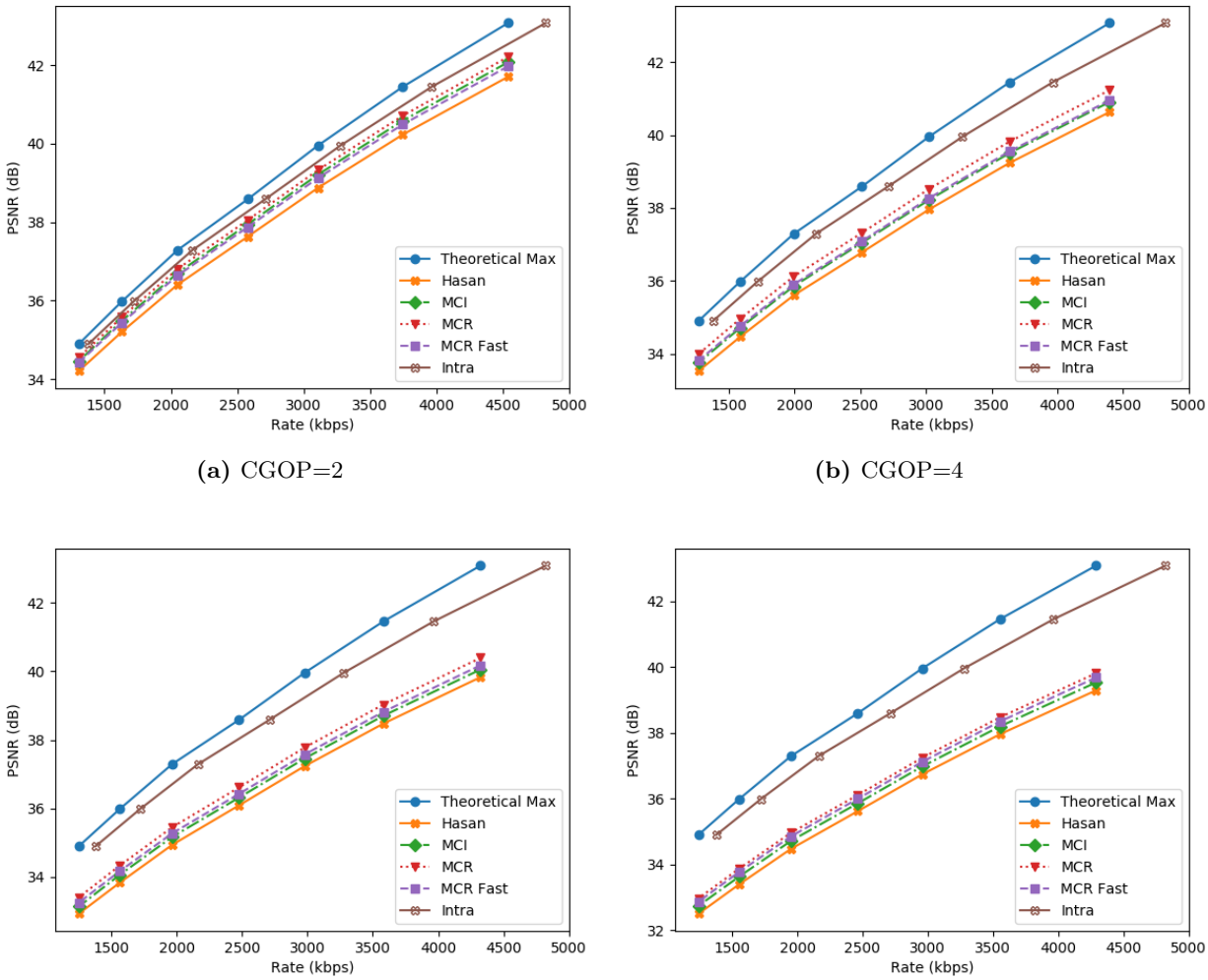
(c) CGOP=8



(d) CGOP=16

Figure 5.7: Recolouring Analysis: Foreman

not as easily predicted using an interpolation-based technique. This also points to a potential failure of the TI measurement to accurately represent video motion, since Foreman, Football, Flower, and Mobile all have similar TI values, despite significant differences in motion characteristics. Therefore, a future work might be to create a better measurement for video motion which can better-capture certain motion-characteristics: does a video contains smooth or sporadic motion and does that motion profile change throughout the course of the video?



(a) CGOP=2 **(b)** CGOP=4

(c) CGOP=8 **(d)** CGOP=16

Figure 5.8: Recolouring Analysis: Football

5.3.3 High Temporal Information, High Spatial Information

Next, the video recolouring strategies were compared for two high-motion, high-spatial-complexity benchmark videos: "Flower" and "Mobile" (Figures 5.9a and 5.9b). Both scenes are characterized by detailed, colourful scenes and smooth, global camera motion.

(a) Flower **(b)** Mobile

Figure 5.9: Flower and Mobile Videos

For these sequences, recolouring techniques are able to create very high quality predictions while offering significant bitrate savings. In particular, both the MCI and MCR techniques are quite accurate up to a CGOP of 8 (Figures 5.10c and 5.11c).

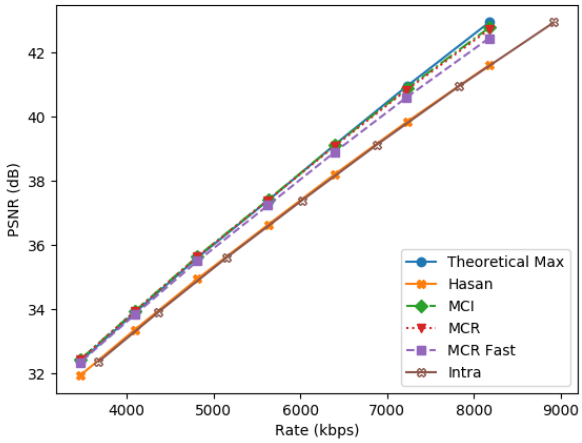
MCR_{fast} has slightly lower prediction accuracy compared these other two techniques, trading prediction accuracy for faster decoding time, though still better than the baseline Intra coding. Finally, the Hasan recolouring codec performs worse or simply as-good-as the baseline method; its motion model is less effective for these higher-motion videos than on a low-motion video like Akiyo.

At a CGOP of 16, the Flower sequence begins to see some loss of quality, with both MCR and MCI nearing the Intra curve, and MCR_{fast} dipping below it (Figure 5.10d). On the other hand, all three of these recolouring techniques are able to maintain high quality predictions for the Mobile sequence even at this high CGOP value (Figure 5.11d).

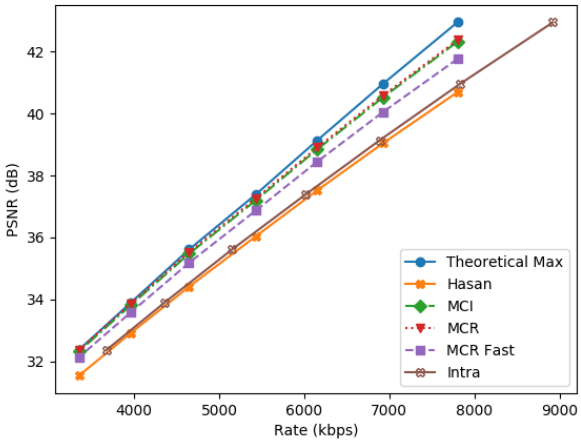
The recolouring results for the Flower and Mobile sequences are quite different from those of the Foreman and Football sequences, despite all four videos having similar temporal information measurements. One explanation for this difference is that Flower and Mobile both contain a large proportion of Chroma information with C/T ratios of 13.8% and 19.1% respectively, and this results in larger potential bitrate savings (as shown by difference between the Theoretical Maximum and Intra RD-curves). Another factor might be the type of motion contained in the videos, as both Flower and Mobile have smooth, global camera motion, while the motion in Football and Foreman appears more sporadic. A future work might be to develop a better measurement than TI for comparing and categorizing video motion.

5.3.4 Canola Time-lapse Videos

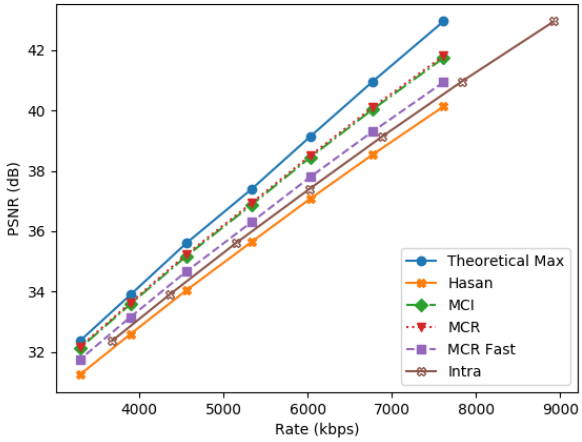
The Canola video sequences were recorded on stationary cameras taking time-lapse images at a rate of one image per minute. Each camera captured a single Canola test plot in a field. Thumbnails from these videos can be seen in Figures 5.12 and 5.13.



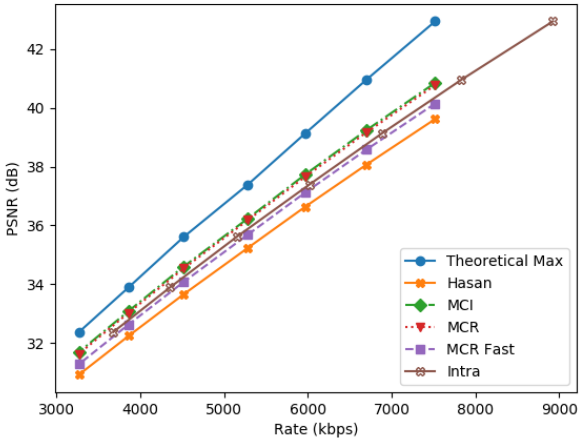
(a) CGOP=2



(b) CGOP=4

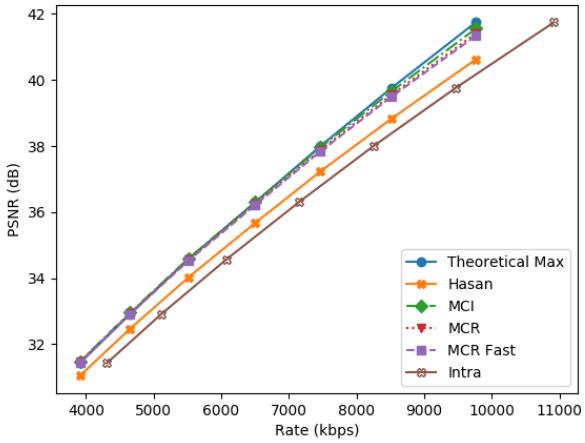


(c) CGOP=8

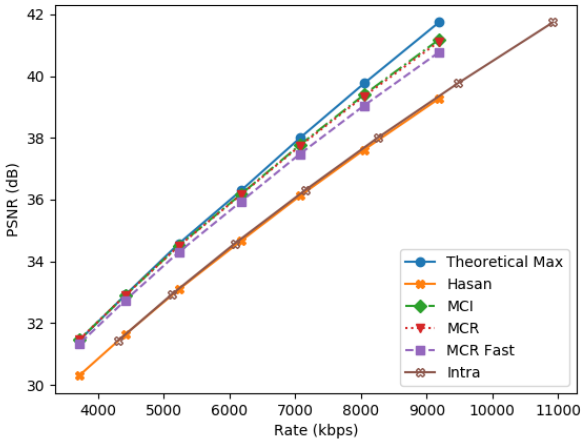


(d) CGOP=16

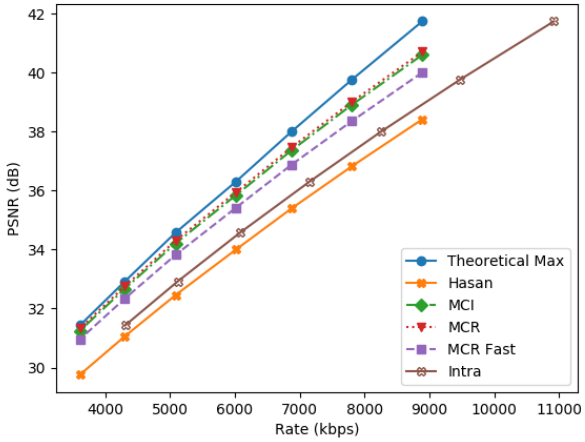
Figure 5.10: Recolouring Analysis - Flower



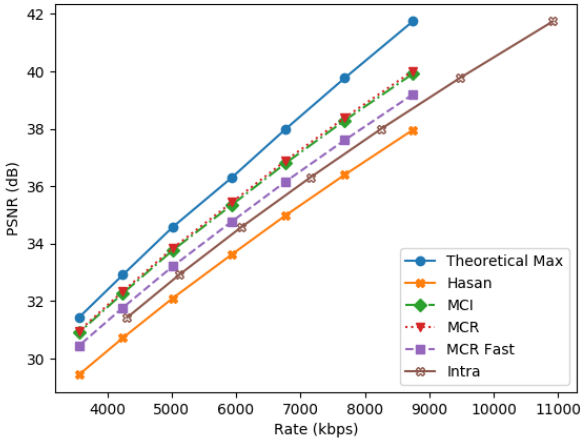
(a) Mobile CGOP=2



(b) Mobile CGOP=4

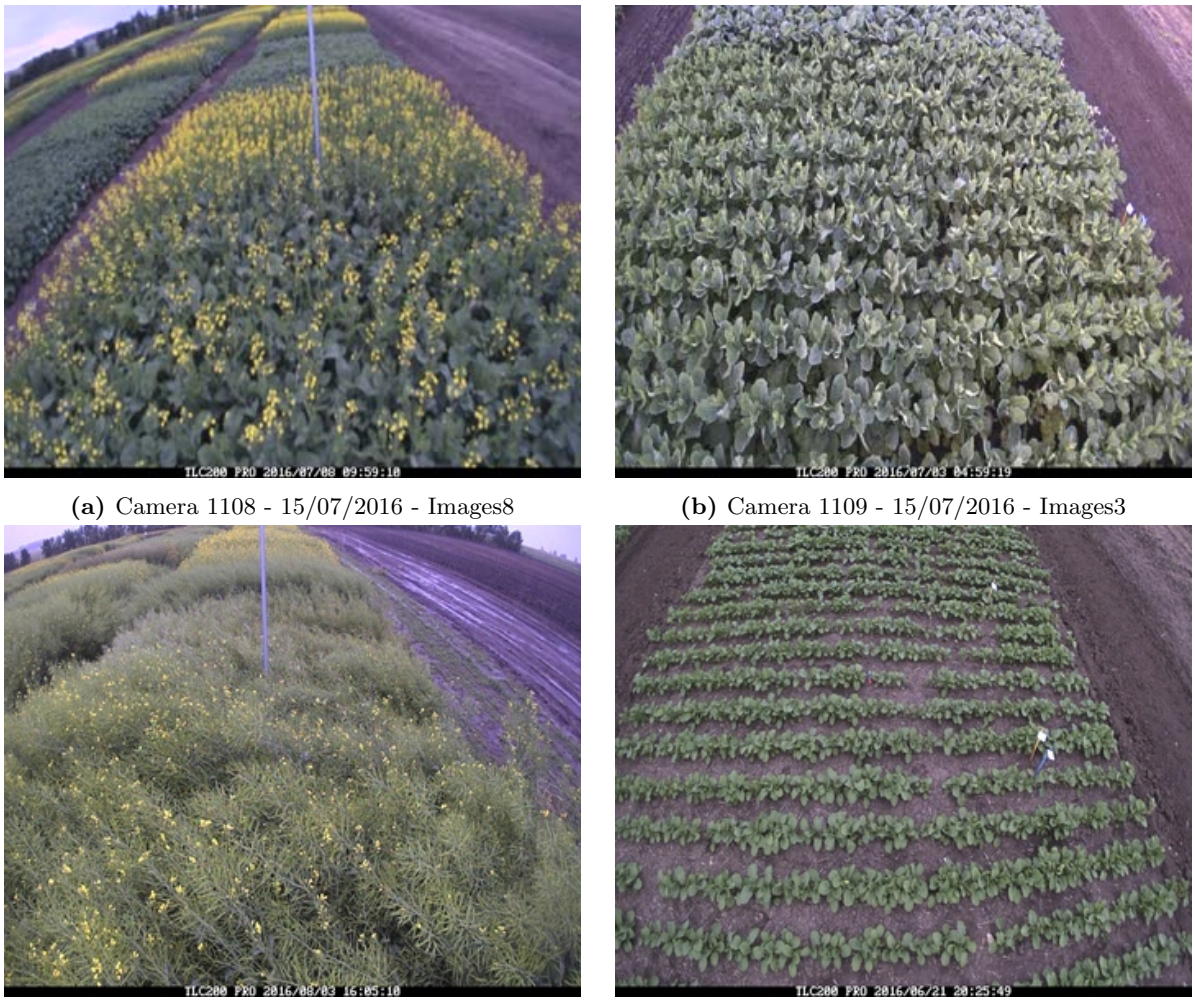


(c) Mobile CGOP=8



(d) Mobile CGOP=16

Figure 5.11: Recolouring Analysis - Mobile



(a) Camera 1108 - 15/07/2016 - Images8

(b) Camera 1109 - 15/07/2016 - Images3

(c) Camera 1108 - 15/08/2016 - Images1

(d) Camera 1109 - 23/06/2016 - Images0

Figure 5.12: 2016 Canola Videos

In terms of video characteristics, the five videos cover a range of spatial and temporal measurements, as well as bitrate compositions. Table 5.3 depicts these values for each video. From this table, it is clear that three of the five Canola videos (camera-days 1508.2016.1108.images1, 1507.2016.1109.images3, and 2306.2016.1109.images0) have very low $BR_{C/T}$, averaging between 3-4% of the total video bitrate. From their thumbnails, it can be seen that these three low $BR_{C/T}$ videos contain Canola crops that are either very early or very late in the season, and show relatively few flowers. The $BR_{C/T}$ for these videos are far below the values found in the benchmark videos, which averaged between 10-20% of the total video bitrate.

These three low-$BR_{C/T}$ videos have Theoretical Maximum RD curves that sit quite close to the baseline Intra curves, and in some cases the two lines are nearly indistinguishable. The small horizontal distance between these two curves indicates that there is almost no potential to save bitrate using a video recolouring method, even when CGOP is high (when most of the video's Chroma information is discarded). For example, Figure 5.14 show the RD graphs created by applying video recolouring to the 2306.2016.1109.images0

Figure 5.13: 2018 Canola Video - Full-motion 30 FPS Video

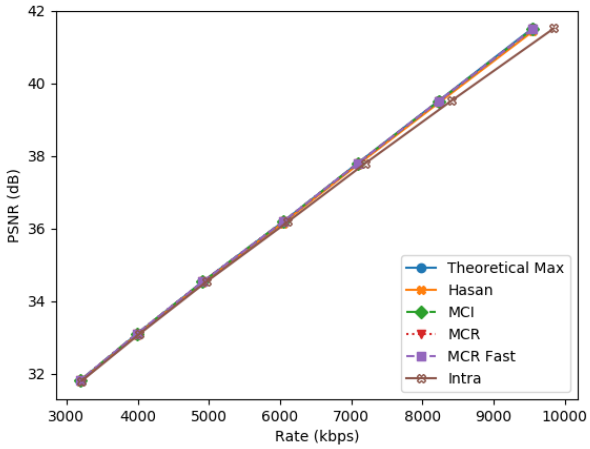
Table 5.3: Canola Video Characteristics

Video	SI	TI	$BR_{C/T}$ (Average)
1507.2016.1108.images8	86.9	22.8	10.89%
1508.2016.1108.images1	78.6	22.6	3.85%
1507.2016.1109.images3	161.8	26.8	3.39%
2306.2016.1109.images0	140.8	18.1	3.10%
Summer2018 (full-motion)	94.3	27.6	10.88%

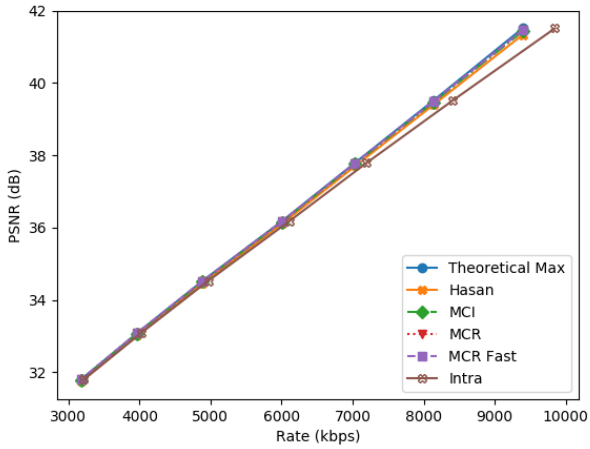
sequence. Despite the fact that all recolouring algorithms exhibit high prediction accuracy, with RD-curves close to that of the Theoretical Max line, the resulting bitrate savings are almost negligible. The only case where video recolouring offers some improvement is when the QP is the lowest and CGOP is high (e.g. QP=22 and CGOP=16); in this case, trimming Chroma bits reduces the bitrate by roughly 600 kpbs, or 6% of total.

On the other hand, the videos from camera-day 1507.2016.1108 and Summer2018 were taken during a time when most Canola flowers were in bloom; this correlated with much higher $BR_{C/T}$ values. The increased $BR_{C/T}$ of these two videos caused the Theoretical Max curves to shift to the left of the Intra baseline curves. In the best case, with highest target video quality and largest CGOP size (QP=22, CGOP=16), this enables a potential bitrate saving of up to 1 Mbps or 14% of total (seen in Figure 5.16d).

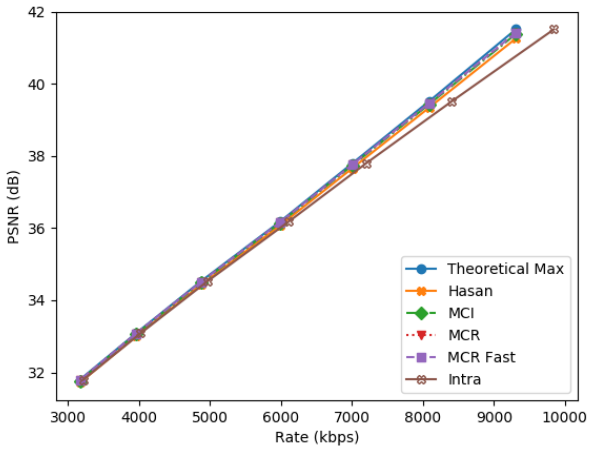
The four implemented recolouring codecs also performed well for camera-day 1507.2016.1108.images8. All four methods achieved a prediction accuracy that approached the objective quality of the Theoretical Maximum, and remained high as CGOP increases from 2 to 16. For example, given a QP of 22, the PSNR difference between MCR and the Theoretical Max was only 0.2 dB for a CGOP of 2 and 0.4 dB for a CGOP of 16. In contrast, the video recolouring algorithms did not perform as well for the Summer2018 sequence when CGOP was high. For example, when the Summer2018 sequence was encoded with a QP of 22 and a



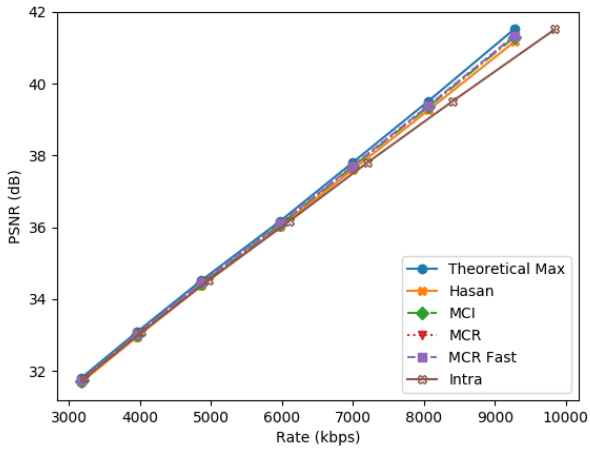
(a) 23/06/2016 - Day 0, CGOP=2



(b) 23/06/2016 - Day 0, CGOP=4



(c) 23/06/2016 - Day 0, CGOP=8

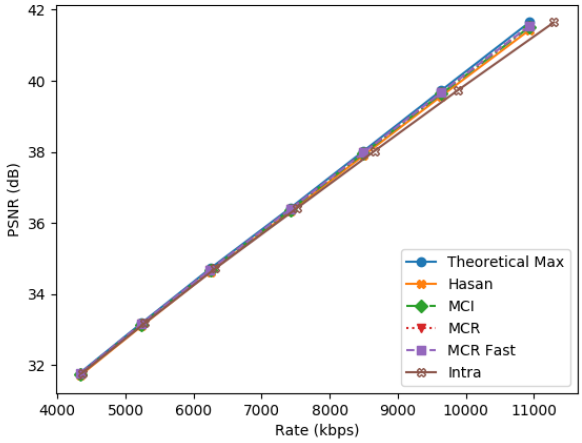


(d) CGOP=16

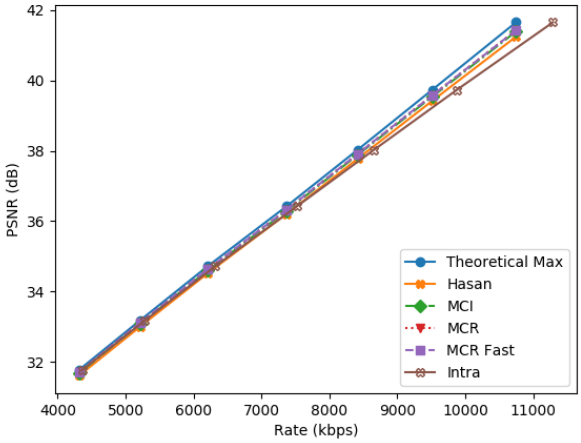
Figure 5.14: Recolouring Analysis: 23/06/2016, Camera 1109, Images0

CGOP of 2, the difference between MCR and the Theoretical Max was only 0.1 dB. However, increasing the CGOP to 16 led to a 1 dB PSNR decrease for MCR.

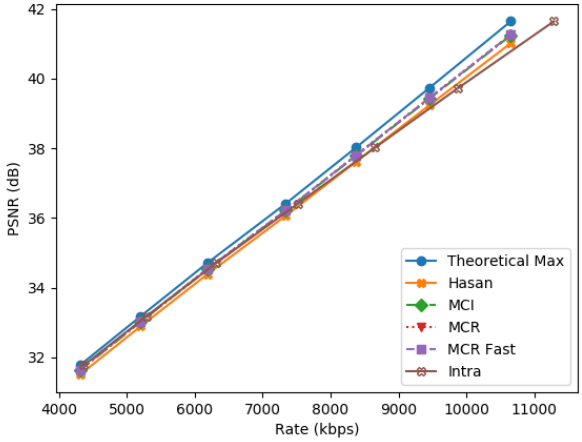
The best explanation for the difference in motion-prediction accuracy between these two sequences is their differing temporal information measurements. Summer2018 had more complex motion compared to the 1507.2016.1108.images8 video, with a TI score of 27.6 compared to the latter's score of 22.8. The impact of TI on Chroma prediction accuracy can be further demonstrated by comparing the RD graph from recolouring the Canola 1507.2016.1108.images8 video with the graphs produced for the Football benchmark sequence (Section 5.3.2). Both of these videos have roughly the same $BR_{C/T}$ (around 10%), but Football has a TI measurement of 40.2 which is almost double the Canola video's measurement of 22.8. This difference in TI corresponds to a dramatic change in motion-prediction accuracy for the two videos.



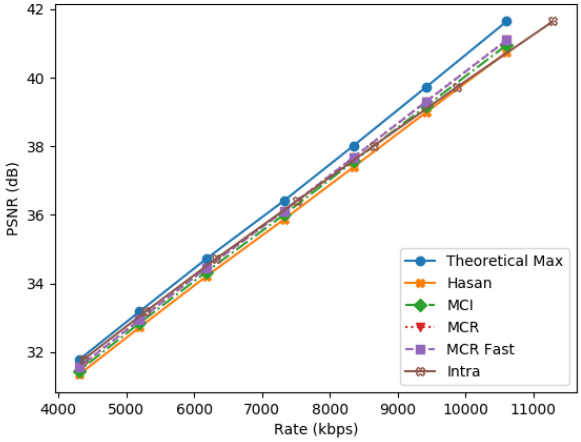
(a) CGOP=2



(b) CGOP=4

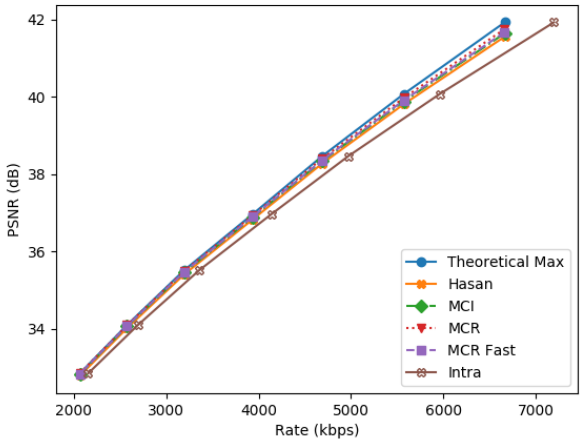


(c) CGOP=8

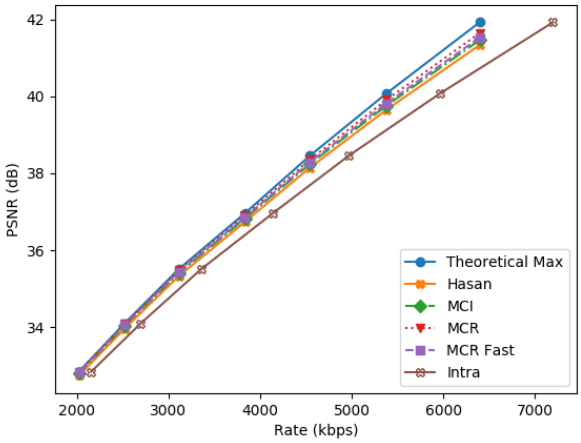


(d) CGOP=16

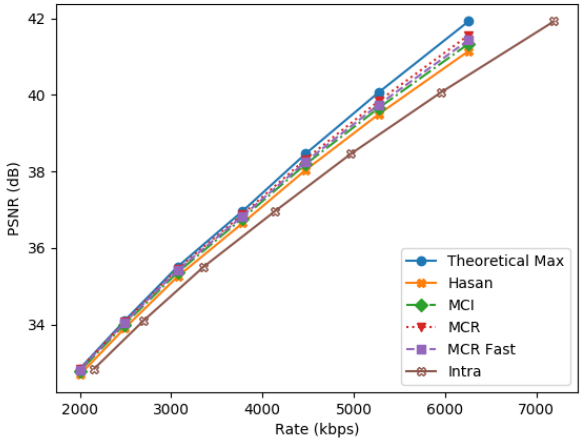
Figure 5.15: Chroma analysis: 15/07/2016, Camera 1109, Images3



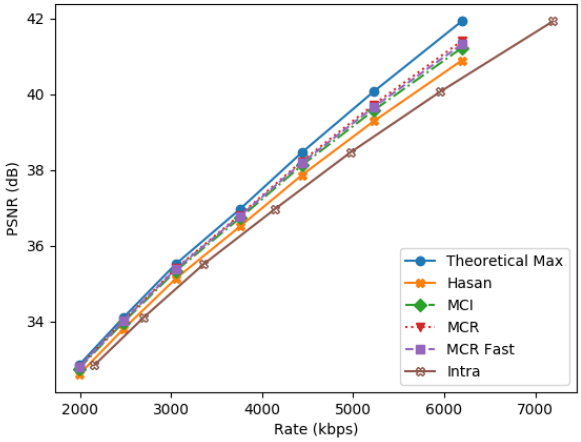
(a) CGOP=2



(b) CGOP=4

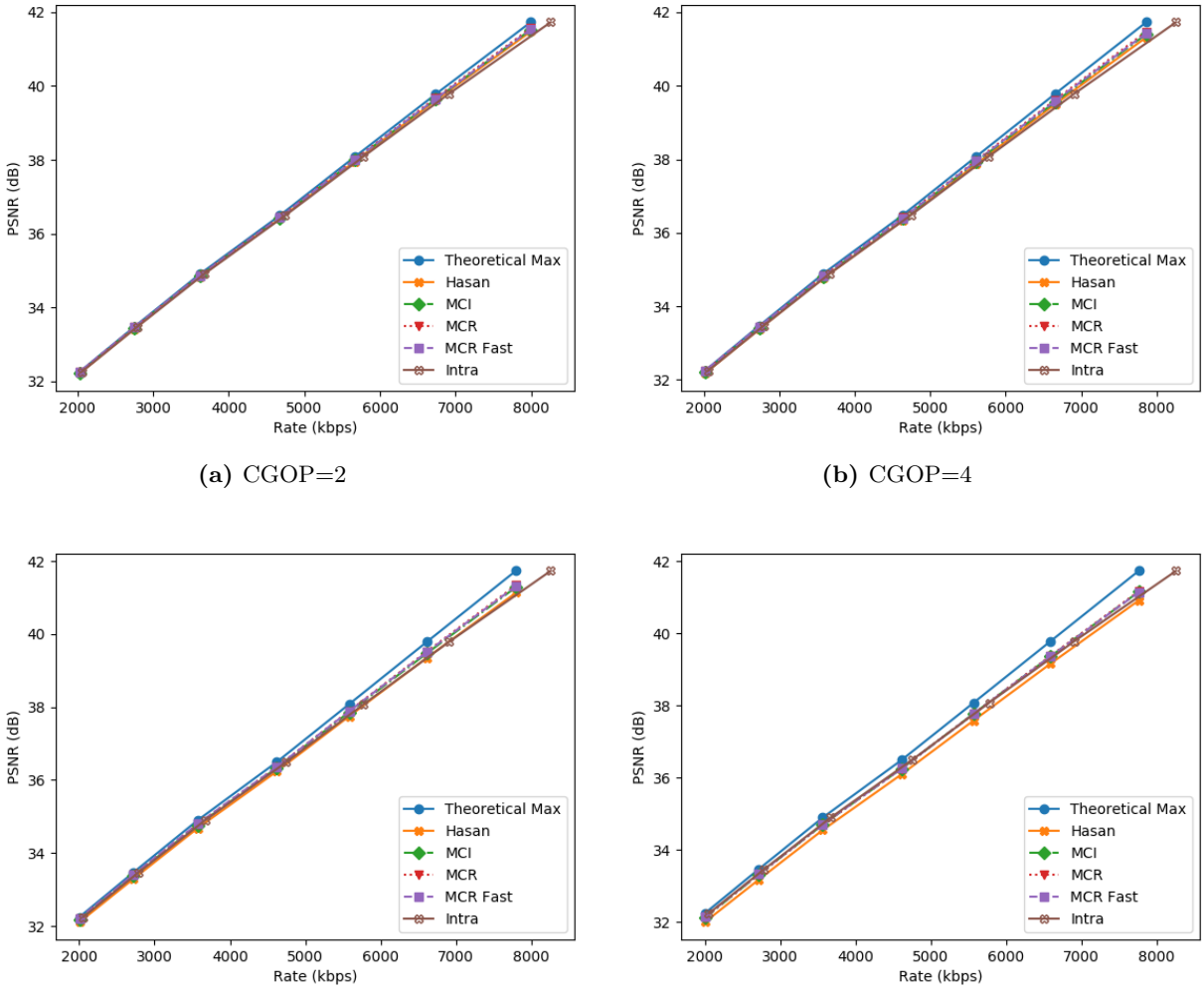


(c) CGOP=8



(d) CGOP=16

Figure 5.16: Chroma analysis: 15/07/2016, Camera 1108, Images8



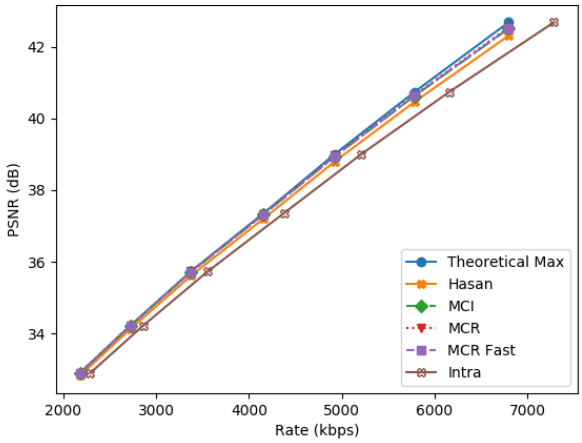
(a) CGOP=2

(b) CGOP=4

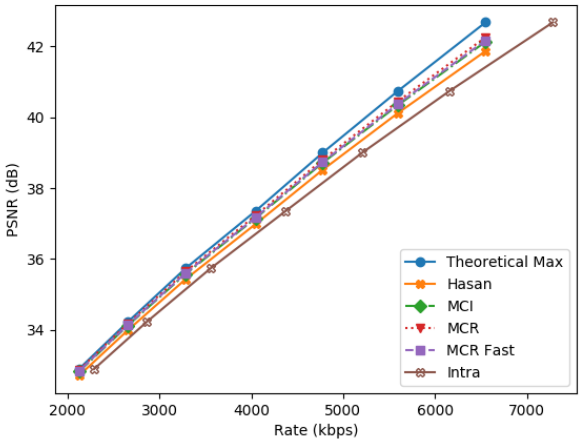
(c) CGOP=8

(d) CGOP=16

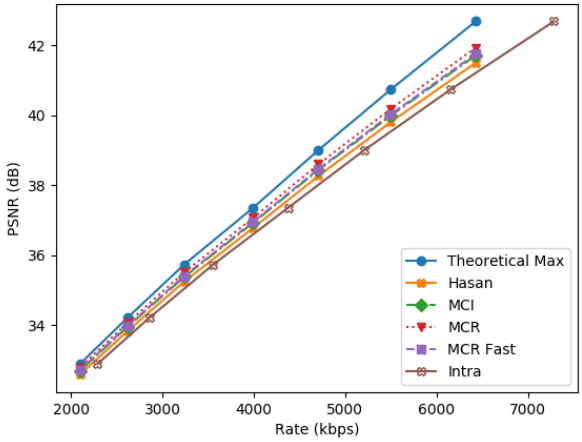
Figure 5.17: Recolouring analysis: 15/08/2016, Camera 1108, Images1



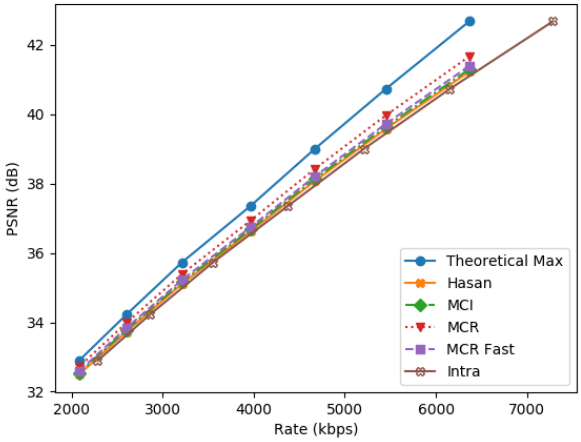
(a) CGOP=2



(b) CGOP=4



(c) CGOP=8



(d) CGOP=16

Figure 5.18: Recolouring Analysis: Summer 2018 (full-motion)

5.3.5 Comparing Recolouring Algorithms in Terms of Decoding Time

During analysis, the average decoding times per frame were measured across all videos and CGOP sizes. These results are represented as bar graphs in Figure 5.19, with standard deviation shown as vertical error lines. From these error bars, it is clear that each of the recolouring schemes have fairly consistent decoding times that do not vary by video. The Hasan codec is orders of magnitude faster than the other recolouring schemes, averaging at around 14 ms per frame. The speed of the Hasan codec is due to its efficient block matching algorithm, motion compensation scheme, and "Skip block" technique.

Next is MCR_{fast}, with an average of 278 ms per frame; MCR_{fast} uses the same Three-Step-Search algorithm as Hasan to speed up block-matching, but computation time is increased because it uses multiple reference-frames and performs Chroma up-sampling in order to apply motion vectors calculated in the Luma plane.

Finally, the slowest recolouring algorithms are Chroma MCI (with an average of 686 ms per frame) and full MCR (1028 ms per frame). Both Chroma MCI and MCR use a form of exhaustive search to find block-matches, though MCI ends up taking less time than MCR because its motion estimation is done at a smaller resolution (MCI motion vectors are calculated in the Chroma planes rather than Luma plane).

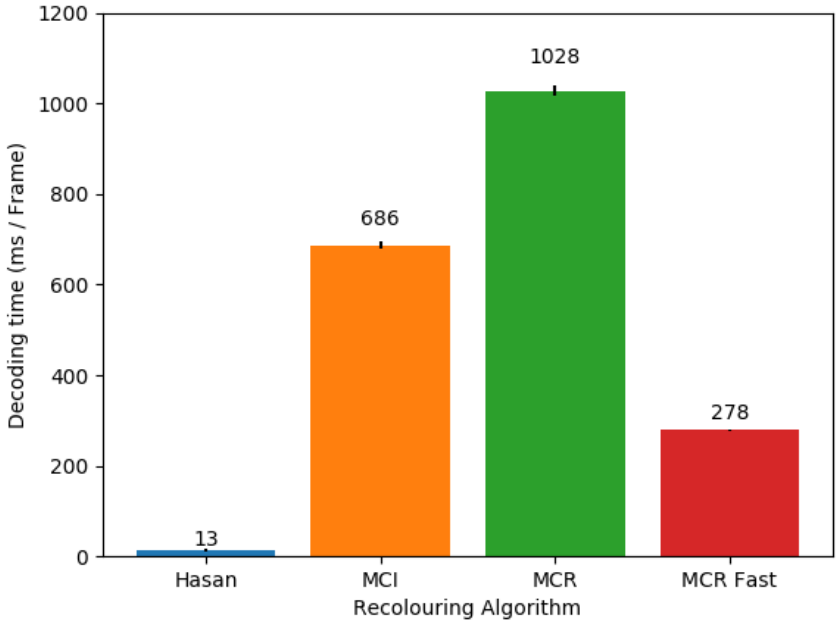


Figure 5.19: Decoding time across all videos

5.4 Video Recolouring on Inter-coded (No Motion) Sequences

In a second analysis, video recolouring was evaluated in combination with H.264 Inter No Motion (Inter-NM), and these results are presented in Figure 5.20. Only MCR was used for this analysis, since it was

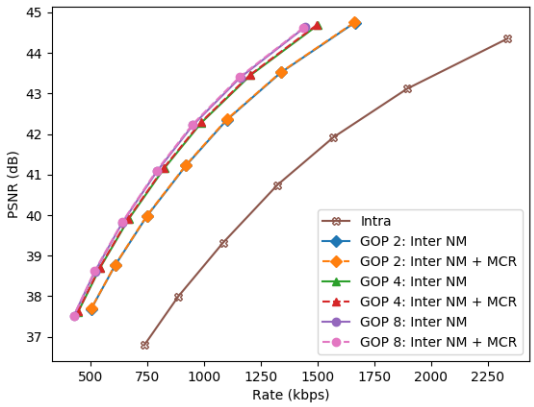
demonstrated to have the best objective quality performance out of all recolouring algorithms during the previous analysis in Section 5.3. Each video sequence was encoded with Inter-NM using GOPs of 2, 4, and 8. The MCR recolouring algorithm was parameterized such that CGOP was set to equal the GOP parameter used by Inter-NM. For a given GOP value, the pair of RD curves for Inter-NM and Inter-NM + MCR were represented graphically using the same line markers, but the former was depicted with solid lines, and the latter with dashed lines. Finally, in each of the sub-figures in Figure 5.20, an Intra-only RD curve was also included for comparison, represented as a brown line with x-shaped markers. This Intra-only curve was parameterized using the same QP values as the Inter-NM encodings, but consisting of only key frames.

In general, Inter-NM had better RD efficiency than its Intra-only counterpart, with the best performance occurring for the low-motion videos Akiyo and Canola (Figures 5.20a and 5.20b). For example, Akiyo encoded using Inter-NM with a GOP of 8 and a QP of 22 saved 892 kbps compared to the Intra-only encoding for the same QP, saving 38% of its total bitrate. At the same time, PSNR increased slightly, from 44.36 dB using Intra-only to 44.60 with Inter-NM.

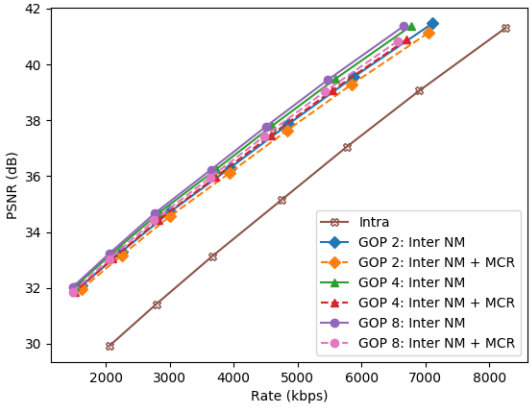
The high-motion videos Flower and Mobile (Figures 5.20e and 5.20f) saw the smallest RD increase from Inter-NM. In particular, Flower barely saw any bitrate savings from the added residual encoding; given the same set of parameters described above (Flower encoded using Inter-NM with a GOP of 8 and a QP of 22), bitrate decreased by 260 kbps compared to Intra-only (3%) and PSNR decreased from 42.91 dB to 42.76 dB. The fact that Inter-NM works better for low- than high-motion videos should be no surprise, however, since this profile is using residual encoding without any form of motion compensation. Thus, the frame residuals for higher-motion videos will have much higher energy and will be compressed less efficiently.

Another observation about Inter-NM that can be derived from this analysis is the fact that GOP size has very little effect on the compression results. For most videos, there is almost no observable difference between Inter-NM encoded with a GOP of 2 or a GOP of 8. The major outlier in this case is the Akiyo video, where GOPs of 4 and 8 offer some bitrate savings compared to a GOP of 2.

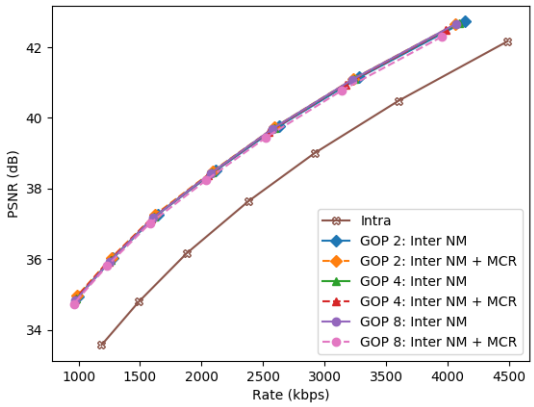
Finally, the addition of video recolouring to Inter-NM had mixed results. Video recolouring did not provide the same bitrate savings that it demonstrated during the analysis in Section 5.3. For the Akiyo and Foreman sequences, there was no discernible difference between Inter-NM and Inter-NM + MCR, and for Canola and Football the addition of MCR decreased the PSNR of the decoded results, with very little bitrate savings. Only the colourful, high-motion sequences Flower and Mobile saw a major improvement using video recolouring. In the best case, (Flower encoded with a GOP of 2 and a QP of 22), MCR was able to increase bitrate savings from 3% using Inter-NM to 9% compared to the Intra-only coding. For both Flower and Mobile, the bitrate savings from video recolouring arrived without any significant loss in video quality.



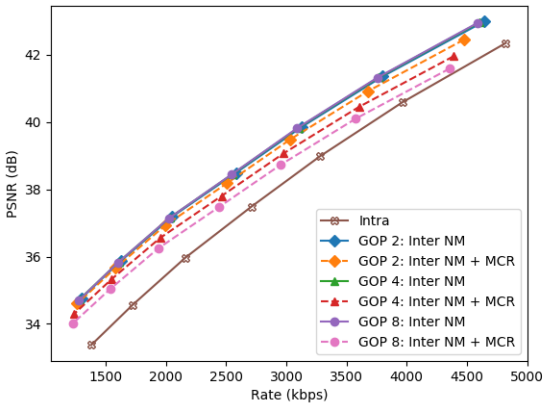
(a) Akiyo



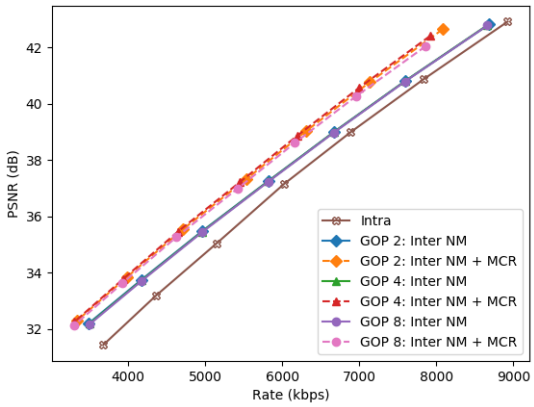
(b) Canola 15/08/2016 - Day 1



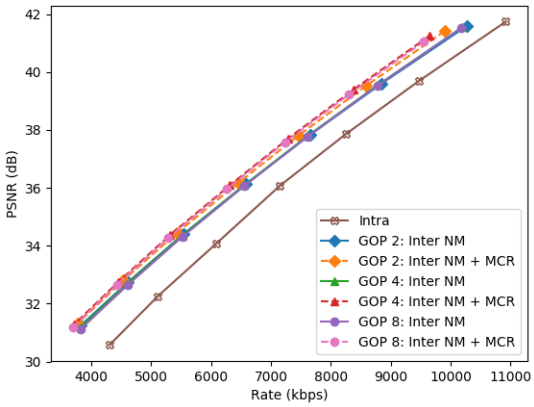
(c) Foreman



(d) Football



(e) Flower



(f) Mobile

Figure 5.20: Inter No Motion with Video Recolouring

5.5 Predicting the Percent Chroma Bitrate

As discussed in Section 5.2, the percentage of a video's bitrate taken up by Chroma coefficients, $BR_{C/T}$, can vary depending on the video content; some videos may contain more complex colour information which is harder for Intra-coding to successfully compress. In Section 5.3, it was observed that $BR_{C/T}$ had a significant impact on the success of video-recolouring as an optimization technique. Video recolouring offered greater bitrate savings to encoded videos with high $BR_{C/T}$ compared to those with low $BR_{C/T}$ videos. At the same time, it was apparent during analysis that the QP value used to encode the video also had an effect on the bitrate ratio. Higher QP values used during Intra encoding disproportionately affected Chroma bitrate, resulting in a smaller $BR_{C/T}$ value.

This next analysis attempts to predict the value of $BR_{C/T}$ using information available prior to encoding. If $BR_{C/T}$ can be estimated accurately and efficiently at the video capture device, this prediction can be used to make decisions about video coding strategy, and in particular can help to determine if video-recolouring is useful for a given encoding application. Two linear regression models were compared as predictors for $BR_{C/T}$. As described in Chapter 4, the first model, looked only at the spatial information of the input video planes to form a prediction, while the second model included the encoding parameter QP. In other words, the first measurement looks only at the video content, while the second measurement considers both video content and encoding context. The benchmark and Canola video sets were used as training data for the two regression models, while a third test dataset of five encoded videos was used to measure the accuracy of the models. Table 5.4 presents these test scores, along with the cross-validation scores, measured from the training data:

Table 5.4: Prediction Results

Strategy	**Cross-Validation Score**	**Test Score**
With SI only	0.7178	0.7202
With SI and QP	0.7745	0.8399

A few tentative conclusions can be drawn, based on the results in this table:

1. Both models provided a good prediction for $BR_{C/T}$ (the percentage of video bitrate used by the Chroma planes), and

2. By factoring in the encoding parameter, QP, it was possible to increase the coefficient of determination score, R^2, by roughly 10% compared to a model that used raw video data alone.

At the same time, there is a benefit of using the "SI only" model that is not captured by looking at prediction accuracy alone. The "SI Only" model is codec agnostic, in that it cares only about characteristics the video and does not care about the encoder used. By ignoring the underlying Intra-codec, it is possible to

reuse this model even if the JM codec is swapped for another, or if other encoding parameters are used (as was the case with the Intra No Motion profile). The second model was able to increase the prediction score by including a QP predictor, however, this came at the cost of tying the model to the JM codec. If in the future, a recolouring scheme uses another codec to perform intra-coding (such as H.265 HEVC, AV1, etc.), the "SI and QP" model will need to be retrained using this new codec.

5.6 Summary

This Chapter presented the main analysis for this thesis. Section 5.1 demonstrated the need for a DVC-style video codec, positing that motion estimation represents the majority of a video encoder's computational costs; it demonstrated that using a video profiles such as Intra only, or Inter-NM rather than conventional video profiles like Inter-BP can reduce encoding time at the expense of rate-distortion efficiency.

Next, Section 5.2 presented key video characteristics of the Benchmark and Canola video sets. SI and TI values were calculated from the raw, uncompressed video data, representing the video's spatial and temporal information. Along with these measurements, a breakdown was given of the bitrate averages for Intra-encoded videos. Encoded bitrate was broken into three categories, Luma, Chroma, and Other, and the total video bitrate was visualized in terms of these three components. Specifically, the percentage of Chroma bitrate was highlighted, and this value was represented using the symbol $BR_{C/T}$. It was shown that the chosen video dataset contained a wide range of SI, TI, and $BR_{C/T}$ values, indicating significant variance in spatial, temporal, and colour complexity.

In Section 5.3, the video recolouring codec described in Chapter 3 was used to enhance Intra-only coding of the selected videos. Four different video recolouring methods were compared against the baseline Intra RD-curves. A Theoretical Max RD curve was also depicted, representing the best-case video recolouring scenario: all the bitrate savings from the recolouring codec were retained, but with no loss to objective video quality compared to the Intra-only version. For the given test sequences, the proposed MCR recolouring technique was able to achieve the highest prediction accuracy, but also required the longest decoding times.

Video recolouring was less effective when combined with Inter-NM. The motion-compensation-free residual encoding used by this profile was able to efficiently compress temporal redundancy in many cases, especially for low-motion sequences or those without global camera motion. The resulting residual frames offered little remaining temporal information for video-recolouring to compress. Only the Flower and Mobile sequences showed an improvement from the combination due to the high degree of colour and motion information in those sequences.

Finally, Section 5.5 detailed the analysis of a predictor for $BR_{C/T}$, that was trained using information available prior to encoding the video. It was found that the ratio between Chroma and Luma SI could be used as an accurate predictor of $BR_{C/T}$, and that the accuracy could be further enhanced by factoring in the desired encoding QP.

6 Conclusions

6.1 Thesis Summary

As part of thesis work, I implemented a framework for combining conventional low-powered video coding techniques with video recolouring in order to offload computation to the decoder. Three video recolouring designs were implemented. The first method was based on an older video recolouring work by Hasan *et al.* [24], but modified to fit a YUV4:2:0 video format. This Hasan recolouring algorithm predicted the forward motion of the Chroma planes based on motion estimation done in the coincident Luma planes. The second algorithm used Motion-Compensated Interpolation (MCI) [4], a technique from the field of Distributed Coding for predicting motion at the decoder, where the target frame is not present. This MCI method was modified to perform motion estimation and compensation in the Chroma planes, using two independent prediction/compensation pipelines. Finally, a new algorithm, Motion-Compensated Recolouring (MCR) was proposed that combined aspects of the previous two. Like the Hasan algorithm, estimation was done in the Luma plane and compensation in the Chroma plane. At the same time, the motion estimation and prediction pipeline closely resembled that of MCI, using the more computationally expensive bi-directional motion-estimation, weighted motion-compensation, and spatial motion-smoothing techniques. The problem of differing resolution between Luma and Chroma was resolved using a combination of upsampling prior to motion-compensation and decimation after to achieve the target Chroma frame size.

A series of videos was collected to test the efficacy of this recolouring framework, including general-purpose benchmark videos and custom sequences used for agricultural research. For each video, three key characteristics were measured prior to recolouring analysis; these included the spatial information measurement (SI), temporal information measurement (TI), and the percentage of encoded bitrate used to represent Chroma information ($BR_{C/T}$). During analysis, the effect of these video characteristics on the recolouring algorithms was observed, and a new measurement was proposed to predict $BR_{C/T}$ prior to encoding, in order to help inform video coding decisions.

Two profiles were used for base layer video coding: Intra-only, and Inter No Motion (Inter-NM). In the video recolouring framework, the effects of Chroma trimming and data transmission were simulated to determine the rate-distortion improvement of video recolouring over the base layer coding alone. the Inter-NM profile worked much the same as the Intra-only profile, but with an added frame subtraction step. I-frames were intra-coded as before, while P-frames underwent a motion-free residual coding. Video recolouring was added on top of this Inter-NM base layer, simulating the effect of trimming Chroma from each P-frame at

the encoder. For the Intra-only profile, all video recolouring algorithms were evaluated, while only MCR was used for the Inter-NM analysis.

Finally, the difference between video recolouring algorithms was observed. It was found that MCR produced the most accurate motion-estimation results out of all recolouring methods, but at the cost of increased computation at the decoder. The potential bitrate savings of video recolouring was found to correlate with the $BR_{C/T}$ of a specific video, while recolouring accuracy correlated with the video's TI measurement, as well as the CGOP parameter used to encode the video.

6.2 Contributions

At the start of this thesis, the following questions were asked:

- What are the benefits of an uplink coding strategy for low-power video?
- When might video recolouring be a useful strategy to improve compression performance?
- Does video recolouring offer an effective improvement over Intra-only and Inter No Motion compression methods?
- Can techniques from Distributed Video Coding be used to increase the effectiveness of video recolouring?

Chapter 5 contained various analyses that attempted to address each of these questions. The findings from this thesis are summarized below.

What are the benefits of an uplink coding strategy for low-power video? In Section 5.1, it was demonstrated that a significant portion of the video encoding process occurs during the motion-estimation and motion-compensation steps. Two low-power video compression methods, Intra-only and Inter No Motion (NM) were compared against the inter-frame Basic Profile (BP) which was found to be 7 times slower. At the same time, the addition of motion-estimation at the encoder significantly improved compression efficiency, reducing the encoded video bitrate. For the example video sequence, Inter-BP required 35% less bitrate than Intra-only (for a Quantization Parameter of 22), compared to an 11% bitrate reduction achieved using Inter-NM. The use of an uplink video compression model allows the codec to optimize the tradeoffs between these three constraints; motion estimation is removed from the encoder, reducing computation time and saving on energy, while motion estimation at the decoder is used to improve bitrate savings for a desired video quality.

When might video recolouring be a useful strategy to improve compression performance? This thesis examined video recolouring as a tool to enhance conventional low-power video coding techniques (Intra-only and Inter-NM). It was hypothesized that recolouring might be more effective in certain contexts, and so in Section 5.2, three video characteristics were examined to determine their affect on video recolouring.

As seen in the Intra and Inter-NM analysis (Sections 5.3 and 5.4), two factors had the greatest impact on the effectiveness of video recolouring as a RD-optimization technique:

1. $BR_{C/T}$: the percentage of bitrate required to represent Chroma in the encoded bitstream,
2. TI: the temporal information measurement representing the amount of motion that occurs between frames in a video.

The effect of $BR_{C/T}$ could be observed by comparing the Theoretical Max rate-distortion curve on each graph against the baseline coding. Videos with a higher percentage of encoded Chroma bitrate had greater potential to save bitrate by discarding information from these planes. The effect of TI was seen in the predictive power of the four implemented algorithms. Videos with less temporal information could be more accurately predicted using motion-estimation, and so the PSNR of the reconstructed videos was closer to that of the Theoretical Max. The effects of both $BR_{C/T}$ and TI were exaggerated as CGOP size was increased. Videos with higher $BR_{C/T}$ saw greater bitrate savings at high CGOP, due to the greater percentage of recoloured frames. At the same time, the increased distance between reference frames for large CGOP sizes meant that high-TI videos suffered worse prediction accuracy compared to low-TI videos.

Does video recolouring offer an effective improvement over Intra-only and Inter No Motion compression methods? Video recolouring was demonstrated to improve over a baseline Intra-coding compression technique under certain conditions. For low-motion video, it reduced the bitrate, while incurring only a slight decrease in video quality, resulting in a better RD curve. For high-motion video, the effectiveness of video recolouring depended significantly on the $BR_{C/T}$. Videos like Mobile and Flower, which had $BR_{C/T}$ values between 15-20% around saw an improvement to RD efficiency. On the other hand, videos like the Football and Foreman sequences, with high motion and low $BR_{C/T}$ (between 5-10%), saw either no increase or actually decreased in RD efficiency.

Combining video recolouring with the Inter No Motion (Inter-NM) profile greatly reduced the benefits of video recolouring on low-motion sequences; for these videos, compression efficiency was dominated by the bitrate savings due to Inter-NM, so that there was no discernible difference to bitrate from adding video recolouring. Only for high-motion sequences with high $BR_{C/T}$ values was there a significant improvement compared to Inter-NM encoding.

Can techniques from Distributed Video Coding be used to increase the effectiveness of video recolouring? This thesis presented a video codec, MCR, that borrowed components from the MCI design used in Distributed Video Coding, but that uses information in the coincident Luma plane to inform its predictions. MCR was compared against two similar recolouring codecs: MCI and Hasan, as well as the baseline coding technique (Intra-only or Inter No Motion).

MCR outperformed the Intra-only baseline for videos with both low and high SI/TI values, though its performance proved worse for videos where Chroma made up only a small portion of the encoded video.

MCR was shown to achieve better prediction and therefore higher objective quality than the prior Hasan recolouring method for all videos, QP values, and CGOP sizes. MCR also proved as-good-as or better than Chroma MCI, seeing the greatest gains over MCI for higher video quality and larger CGOP.

In terms of decoding time, the Hasan codec was the clear winner, requiring only 14 ms to decode each frame. The MCR technique took an average of 1028 ms per frame, while MCI took 686 ms per frame. Finally, a fast configuration of MCR was implemented that used the Three Step Search (TSS) block matching algorithm in place of Exhaustive Search. MCR_{fast} was able to speed up decoding execution from 1028 ms to only 278 ms per frame, trading off speed for a slight decrease in objective video quality.

Is it possible to predict $BR_{C/T}$ prior to intra-encoding? During analysis, it was shown that video recolouring works poorly for videos where $BR_{C/T}$ is small. It would be beneficial if this information could be used to make encoding decisions, however, this ratio can only be calculated after a video had already been encoded. The ability to predict this $BR_{C/T}$ ratio from raw video data would be useful to help the encoder make decision about when to use video recolouring in place of inter-frame coding or other low-power options.

A simple linear regression model was created to estimate $BR_{C/T}$, and two sets of input parameters were used. In the first set, a parameter was derived by dividing SI measurements taken from the video's Chroma plane by similar SI measurements taken of the Luma plane. This model was trained on the encoded Benchmark and Canola videos and tested on a set of unrelated videos, resulting in a coefficient of determination of 72%.

The second set of parameters included the above SI ratio as well as the quantization parameter used to encode the video, normalized by the maximum possible QP. Inclusion of the QP encoding parameters increased the coefficient of determination to 83%, however, this increased predictive accuracy came at a cost. Factoring in QP worked for the specific codec and coding profile used, but it is likely that a change to the underlying codec or encoding profile would require the model to be retrained to stay accurate.

6.3 Future Work

Future work will be to implement a standalone video recolouring codec that trims Chroma information from the encoded bitstream and uses the MCR at the decoder, rather than just simulating the effect of Chroma trimming. Extra analysis can be performed using additional video quality metrics such as the Structural Similarity Index Measure [75] (SSIM) and a larger video dataset. For example, this larger data-set could include videos at higher resolutions than CIF, videos with different colour formats (YUV 4:4:4 instead of 4:2:0), and time-lapsed videos with differing frame-rates (with frames captured at intervals greater than or less than 1 minute apart).

Another future work will be to implement an adaptive CGOP size within this framework. A high degree of variability was observed in rate-distortion performance across different video types and CGOP sizes. A

codec that can adapt CGOP size depending on the degree of motion contained in the video would be able to take full advantage of this technique while decreasing the risk of degrading RD.

A further enhancement to this stand-alone codec would be an adaptive algorithm that can select whether or not to perform recolouring. In this thesis, the relationship between video recolouring and the characteristics TI and $BR_{C/T}$ were observed, but no explicit thresholds were determined. Future work might be to discover for which combination of these values it is feasible to use recolouring, and for which it is better to use another low-power coding method.

Section 5.3 also demonstrated that there was a significant difference in prediction accuracy for high motion videos. Specifically, video recolouring did well for both Flower and Mobile but poorly for Football, even though all three had very high TI measurements. Both Flower and Mobile depicted smooth, global camera motion, while Football was more sporadic. This indicates that the *type* of motion may play an important role in the predictive power of block-based motion-estimation, and that TI might be insufficient as a representation of video motion. An additional measurement that could differentiate between smooth and sporadic motion might be more useful. Alternatively, an adaptive learning algorithm might be employed at the encoder to make coding decisions, selecting between conventional video coding techniques or a combination of conventional and video-recolouring algorithms.

This work suggests that content-specific video coding techniques may be useful in optimizing or improving existing codecs. Future work will be to explore other video measurements and content-specific video coding strategies to improve both conventional and distributed video codecs.

Finally, it was observed that each of the Canola videos displayed a limited range of colour information. As well, it was known that specific colours were deemed more important to the computer-vision application used by COAST. A potential future work for the COAST project might be to explore how knowledge of this colour information could be exploited to help improve compression efficiency. For example, an application-specific colour-space could be explored to represent the raw video data, with pre- and post-processing steps added to convert between YUV and this new colour-space. During encoding, more bitrate could be invested into the colour components that are important to the application (e.g. yellow) while other colour components could be more aggressively compressed; in this way, bitrate savings could be better optimized to the application and video content.

References

[1] A. Aaron, E. Setton, and B. Girod. Towards practical Wyner-Ziv coding of video. In *IEEE International Conference on Image Processing*, pages 869–872, Barcelona, Spain, September 2003.

[2] A. Aaron, R. Zhang, and B. Girod. Wyner-Ziv coding of motion video. In *Asilomar Conference on Signals, Systems and Computers*, pages 240–244, Pacific Grove, CA, November 2002.

[3] L. Alparone, M. Barni, F. Bartolini, and V. Cappellini. Adaptively weighted vector-median filters for motion-fields smoothing. In *IEEE International Conference on Acoustics, Speech, and Signal Processing*, pages 2267–2270, Atlanta, GA, May 1996.

[4] X. Artigas, J. Ascenso, M. Dalai, S. Klomp, D. Kubasov, and M. Ouaret. The DISCOVER codec: Architecture, techniques and evaluation. In *Picture Coding Symposium*, pages 1–4, Lisboa, Portugal, November 2007.

[5] J. Ascenso, C. Brites, and F. Pereira. Improving frame interpolation with spatial motion smoothing for pixel domain distributed video coding. In *EURASIP Conference on Speech and Image Processing, Multimedia Communications and Services*, pages 1–6, Smolenice, Slovak Republic, January 2005.

[6] J. Ascenso, C. Brites, and F. Pereira. Content adaptive Wyner-Ziv video coding driven by motion activity. In *International Conference on Image Processing*, pages 605–608, Atlanta, GA, October 2006.

[7] P. M. Berry. Lodging resistance in cereals. In R. A. Meyers, editor, *Encyclopedia of Sustainability Science and Technology*, pages 6201–6216. Springer, New York, NY, 2012.

[8] S. Borchert. *Distributed Video Coding (DVC): Motion estimation and DCT quantization in low complexity video compression.* PhD thesis, Technische Universiteit Delft, Delft, The Netherlands, 2010.

[9] S. Borchert, R. P. Westerlaken, and I. L. Lagendijk. On extrapolating side information in distributed video coding. In *Picture Coding Symposium*, pages 1–4, Lisboa, Portugal, November 2007.

[10] R. C. Bose and D. K. Ray-Chaudhuri. On a class of error correcting binary group codes. *Information and Control*, 3(1):68–79, 1960.

[11] A. Braunstein, M. Mézard, and R. Zecchina. Survey propagation: an algorithm for satisfiability. *Random Struct. Algorithms*, 27:201–226, 2005.

[12] C. Brites and F. Pereira. Correlation noise modeling for efficient pixel and transform domain Wyner-Ziv video coding. *IEEE Transactions on Circuits and Systems for Video Technology*, 18:1177–1190, September 2008.

[13] J. Chen. CE6.a.4: Chroma intra prediction by reconstructed luma samples. Technical report, Meeting report of the fifth meeting of the Joint Collaborative Team on Video Coding, Geneva, Switzerland, March 2011. Sources: Samsung Electronics Co., Ltd; LG Electronics.

[14] S. Chien, T. Cheng, S. Ou, C. Chiu, C. Lee, V. S. Somayazulu, and Y. Chen. Power consumption analysis for distributed video sensors in machine-to-machine networks. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 3(1):55–64, March 2013.

[15] T. M. Cover and J. A. Thomas. *Elements of Information Theory (Wiley Series in Telecommunications and Signal Processing).* Wiley-Interscience, New York, NY, 2006.

[16] N. Deligiannis, A. Munteanu, T. Clerckx, J. Cornelis, and P. Schelkens. Overlapped block motion estimation and probabilistic compensation with application in distributed video coding. *IEEE Signal Processing Letters*, 16(9):743–746, September 2009.

[17] N. Deligiannis, F. Verbist, J. Barbarien, J. Slowack, R. Van de Walle, P. Schelkens, and A. Munteanu. Distributed coding of endoscopic video. In *IEEE International Conference on Image Processing*, pages 1813–1816, Brussels, Belgium, September 2011.

[18] P. L. Dragotti and M. Gastpar, editors. *Distributed Source Coding: Theory, Algorithms and Applications*. Academic Press, Boston, MA, 2009.

[19] F. Dufaux, W. Gao, S. Tubaro, and A. Vetro. Distributed video coding: Trends and Perspectives. *EURASIP Journal on Image and Video Processing*, 2009:1–13, February 2009.

[20] S. Fei and Y. Juanjuan. Review of distributed video coding. In *IEEE International Conference on Electronic Measurement Instruments*, pages 315–320, Yangzhou, China, October 2017.

[21] R. G. Gallager. Low-density parity-check codes. *IRE Transactions on Information Theory*, 8:21–28, 1962.

[22] B. Girod, A. M. Aaron, S. Rane, and D. Rebollo-Monedero. Distributed video coding. *Proceedings of the IEEE*, 93(1):71–83, January 2005.

[23] R. Hasan. Design framework for internet of things based next generation video surveillance. Master's thesis, University of Saskatchewan, Saskatoon, Canada, 2017.

[24] R. Hasan, S. K. Mohammed, A. H. Khan, and K. A. Wahid. A color frame reproduction technique for IoT-based video surveillance application. In *IEEE International Symposium on Circuits and Systems*, pages 1–4, Baltimore, MD, May 2017.

[25] M. A. Hassan and M. S. Bashraheel. Color-based structural similarity image quality assessment. In *IEEE International Conference on Information Technology*, pages 691–696, Amman, Jordan, May 2017.

[26] D. T. Hoang. PSNR computation on R'G'B' color system. Technical report, Meeting report of the fourth meeting of the Joint Collaborative Team on Video Coding, Daegu, South Korea, January 2011.

[27] D. T. Hoang and J. S. Vitter. *Efficient Algorithms for MPEG Video Compression*. John Wiley & Sons, Inc., New York, NY, 2002.

[28] X. Huang and S. Forchhammer. Improved side information generation for Distributed Video Coding. In *IEEE Workshop on Multimedia Signal Processing*, pages 223–228, Cairns, Australia, October 2008.

[29] Y. Huang, H. Qi, B. Li, and J. Xu. Adaptive weighted distortion optimization for video coding in RGB color space. In *IEEE International Conference on Image Processing*, pages 3141–3145, Paris, France, October 2014.

[30] N. Imran, B.-C. Seet, and A. C. M. Fong. Distributed video coding for wireless video sensor networks: a review of the state-of-the-art architectures. *SpringerPlus*, 4(1):513–543, September 2015.

[31] M. Jadhav, Y. Dandawate, N. Pisharoty, and M. M. Mandal. Performance evaluation of structural similarity index metric in different colorspaces for hvs based assessment of quality of colour images. *International Journal of Engineering and Technology*, 5:1555–1562, 04 2013.

[32] J. Jain and A. Jain. Displacement measurement and its application in interframe image coding. *IEEE Transactions on Communications*, 29(12):1799–1808, December 1981.

[33] R. Karthikeyan, G. Sainarayanan, and S. N. Deepa. Perceptual video quality assessment in H.264 video coding standard using objective modeling. *SpringerPlus*, 3:1–6, April 2014.

[34] S. T. Khawase, S. Kamble, N. Thakur, and A. S. Patharkar. An overview of block matching algorithms for motion vector estimation. In *Research in Intelligent and Computing in Engineering*, pages 217–222, Gopeshwar, India, June 2017.

[35] V. K. Kodavalla. Challenges in practical deployment of distributed video coding. In *IEEE International Conference on Microelectronics, Computing and Communications*, pages 1–6, Durgapur, India, January 2016.

[36] V. K. Kodavalla and P. G. K. Mohan. Chroma components coding in feedback-free distributed video coding. In *IEEE International Conference on Internet Multimedia Systems Architecture and Application*, pages 1–6, Bangalore, India, December 2011.

[37] V. K. Kodavalla and P. G. K. Mohan. Chroma components coding method in Distributed Video Coding. In *IEEE International Conference on Devices, Circuits and Systems*, pages 413–417, Coimbatore, India, March 2012.

[38] T. Koga, K. Iinuma, A. Hirano, Y. Iijima, and T. Ishiguro. Motion compensated interframe coding for video conferencing. In *IEEE National Telecommunications Conference*, pages G5.3.1–G5.3.5, New Orleans, LA, November 1981.

[39] S. H. Lee and N. I. Cho. Intra prediction method based on the linear relationship between the channels for YUV 4:2:0 intra coding. In *IEEE International Conference on Image Processing*, pages 1037–1040, Cairo, Egypt, November 2009.

[40] A. Leontaris, P. C. Cosman, and A. M. Tourapis. Multiple reference motion compensation: A tutorial introduction and survey. *Foundations and Trends in Signal Processing*, 2(4):247–364, April 2009.

[41] S. Lin and D. J. Costello. *Error Control Coding, Second Edition.* Prentice-Hall, Inc., Upper Saddle River, NJ, 2004.

[42] P. List, A. Joch, J. Lainema, G. Bjontegaard, and M. Karczewicz. Adaptive deblocking filter. *IEEE Transactions on Circuits and Systems for Video Technology*, 13(7):614–619, 2003.

[43] D. J. C. MacKay. *Information Theory, Inference & Learning Algorithms.* Cambridge University Press, New York, NY, 2003.

[44] D. J. C. MacKay and R. M. Neal. Near Shannon limit performance of low density parity check codes. *Electronics Letters*, 32:457–458, August 1996.

[45] A. Mahmood, L. S. Dooley, and P. Wong. Transform domain distributed video coding using larger transform blocks. In *Global Conference on Signal and Information Processing*, pages 1–5, Montreal, Canada, November 2017.

[46] L. Merritt and R. Vanam. Improved rate control and motion estimation for H.264 encoder. In *IEEE International Conference on Image Processing*, pages 309–312, San Antonio, TX, September 2007.

[47] S. Midtskogen. Improved chroma prediction. RFC draft, Cisco, August 2016.

[48] C. Montgomery. Next Generation Video: AV1, Part 1: Chroma from Luma (20180429). `https://people.xiph.org/~xiphmont/demo/av1/demo1.shtml`. Accessed: 2020-04-05.

[49] L. Natário, C. Brites, J. Ascenso, and F. Pereira. Extrapolating side information for low-delay pixel-domain distributed video coding. In *International Conference on Visual Content Processing and Representation*, VLBV'05, pages 16–21, Berlin, Heidelberg, 2006. Springer-Verlag.

[50] European Society of Radiology. Usability of irreversible image compression in radiological imaging. A position paper by the European Society of Radiology. In *Insights into Imaging*, volume 2, pages 103–115, Vienna, Austria, April 2011. Springer Berlin Heidelberg.

[51] M. T. Orchard and G. J. Sullivan. Overlapped block motion compensation: an estimation-theoretic approach. *IEEE Transactions on Image Processing*, 3(5):693–699, September 1994.

[52] M. Ouaret. *Selected topics on distributed video coding*. PhD thesis, École Polytechnique Fédérale de Lausanne, Lausanne, Switzerland, 2008.

[53] T. N. Pappas and R. J. Safranek. Perceptual criteria for image quality evaluation. In A. C. Bovik, editor, *Handbook of Image and Video Processing*, pages 669–684. Academic Press, 2000.

[54] C. Poynton. *A technical introduction to digital video*. J. Wiley, New York, NY, 1996.

[55] J. G. Proakis. *Digital Communications*. McGraw-Hill series in electrical and computer engineering communications and signal processing. McGraw-Hill, New York, NY, 2001.

[56] R. Puri, A. Majumdar, P. Ishwar, and K. Ramchandran. Distributed video coding in wireless sensor networks. *IEEE Signal Processing Magazine*, 23(4):94–106, July 2006.

[57] R. Puri, A. Majumdar, and K. Ramchandran. PRISM: A video coding paradigm with motion estimation at the decoder. *IEEE Transactions on Image Processing*, 16(10):2436–2448, October 2007.

[58] R. Puri and K. Ramchandran. PRISM: A new robust video coding architecture based on distributed compression principles. In *Allerton Conference on Communication, Control and Computing*, pages 1–10, Allerton, IL, November 2002.

[59] Rec. ITU-T P.910. Subjective video quality assessment methods for multimedia applications. Standard, International Telecommunication Union, 2008.

[60] I. Reed. A class of multiple-error-correcting codes and the decoding scheme. *Transactions of the IRE Professional Group on Information Theory*, 4(4):38–49, 1954.

[61] I. E. Richardson. *The H.264 Advanced Video Compression Standard*. Wiley Publishing, Chichester, U.K., 2nd edition, 2010.

[62] International Telecommunication Union: Telecommunication Standardization Sector. Video coding for low bit rate communication. Technical report, Recommendations: H-Series, Geneva, Switzerland, January 2005.

[63] C. E. Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27(3):379–423, July 1948.

[64] D. Slepian and J. Wolf. Noiseless coding of correlated information sources. *IEEE Transactions on Information Theory*, 19(4):471–480, July 1973.

[65] K. Sühring, A. M. Tourapis, A. Leontaris, and G. Sullivan. JM Reference Software Manual. Technical report, JVT of ISO/IEC MPEG, London, UK, July 2009. Sources: Dolby Laboratories Inc., Fraunhofer-Institute HHI, Microsoft Corporation.

[66] A. M. Tekalp. *Digital Video Processing*. Prentice Hall Press, Upper Saddle River, NJ, 2nd edition, 2015.

[67] L. Trudeau, N. Egge, and D. Barr. Predicting Chroma from Luma in AV1. In *Data Compression Conference*, pages 374–382, Snowbird, UT, March 2018.

[68] D. Tse and P. Viswanath. *Capacity of wireless channels*, pages 166—227. Cambridge University Press, 2005.

[69] S. Ullo and G. R. Sinha. Advances in Smart Environment Monitoring Systems Using IoT and Sensors. *Sensors (Basel, Switzerland)*, 20(11):1–18, 2020.

[70] J. Valin, T. B. Terriberry, N. E. Egge, T. Daede, Y. Cho, C. Montgomery, and M. Bebenita. Daala: Building a next-generation video codec from unconventional technology. In *IEEE International Workshop on Multimedia Signal Processing*, pages 1–6, Montreal, Canada, September 2016.

[71] D. Varodayan, A. Aaron, and B. Girod. Rate-adaptive codes for distributed source coding. *Signal Processing*, 86(11):3123–3130, November 2006.

[72] F. Verbist, N. Deligiannis, S. Satti, P. Schelkens, and A. Munteanu. Encoder-driven rate control and mode decision for distributed video coding. *EURASIP Journal on Advances in Signal Processing*, 2013:156, 10 2013.

[73] A. Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory*, 13(2):260–269, 1967.

[74] H. von Helmholtz. *Helmholtz's treatise on Physiological Optics*. Dover Publications, New York, NY, 1962.

[75] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, April 2004.

[76] T. Wiegand, G. J. Sullivan, G. Bjontegaard, and A. Luthra. Overview of the H.264/AVC video coding standard. *IEEE Transactions on Circuits and Systems for Video Technology*, 13(7):560–576, July 2003.

[77] A. Wyner and J. Ziv. The rate-distortion function for source coding with side information at the decoder. *IEEE Transactions on Information Theory*, 22(1):1–10, January 1976.

[78] Q. Xu. *Layered Wyner-Ziv video coding: a new approach to video compression and delivery*. PhD thesis, Texas A&M University, College Station, TX, 2007.

[79] S. Zhu and K.-K. Ma. A new diamond search algorithm for fast block-matching motion estimation. *IEEE Transactions on Image Processing*, 9(2):287–290, February 2000.

Appendix A

Huffman Coding

Huffman coding produces a form of VLC referred to as a "prefix code", which signifies that no codeword forms a prefix of another. This is important to ensure that any encoded bit-string can be uniquely decoded. As a simple counter-example, consider an alphabet consisting of three symbols: A, B, and C. If A is encoded as "110", B as "1", and C as "0", then the entire codeword for B can be found at the start of the codeword for A, forming a prefix. If this is the case, then it is impossible to uniquely decode the bit-string "110", because this could either decode to "a" or to "bbc".

Given a finite alphabet of symbols and their probability distribution, Huffman coding produces an *optimal* integer prefix code; this means that it approaches as close to entropy as possible, given the condition that the length of each codeword is an integer. Huffman coding can even achieve entropy if all symbol probabilities are powers of 2. In more complex cases, entropy can only be *approached* using Huffman coding, but never reached exactly. Arithmetic coding, on the other hand, is capable of asymptotically achieving entropy regardless of symbol probabilities [66].

The Huffman encoding process works by first generating a binary tree for the provided alphabet and then using this tree to produce the codewords. The tree is constructed according to the procedure in algorithm 3:

Algorithm 3 Huffman Tree Construction

```
1:  procedure MakeTree(symbols, probabilities)        ▷ Node = {symbol, prob, left, right}
2:      N ← length(symbols)
3:      for i ← 1 to N do                             ▷ Initialize Heap with leaf nodes
4:          heap.push(Node(symbols[i], probability[i], NIL, NIL))
5:      end for
6:      while heap.size() ≥ 2 do                      ▷ Merge min nodes to make tree
7:          min1 ← heap.pop()
8:          min2 ← heap.pop()
9:          newProb ← min2.prob + min1.prob
10:         newNode ← Node(NIL, newProb, min2, min1)
11:         heap.push(newNode)
12:     end while
13:     tree ← heap.pop()
14: end procedure
```

After the Huffman tree has been constructed, the algorithm iterates over the leaves of the tree and uses the "path" to each leaf as the codeword for that symbol. Left branches are represented by zeros and right branches are represented by ones.

For example, consider the alphabet ***C2*** from Table 2.1. Huffman tree construction would first pop a_3 and a_4 from the heap, because these have the smallest probabilities. The two smallest nodes are then merged into a tree and reinserted into the heap. This tree would then be merged with a_2, and the result would be merged with a_1 to create the final Huffman tree as depicted in Figure A.1. The resulting codewords mapping is shown in Table A.1. In this case, the average codeword length (weighted by symbol probability) is 1.75, as given by the following equation:

$$\bar{R} = 0.5 \times 1 + 0.25 \times 2 + 0.125 \times 3 + 0.125 \times 3 = 1.75, \tag{A.1}$$

This average codeword length is equivalent to the entropy for the alphabet ***C2***, as shown in Table 2.1. This example demonstrates that Huffman coding is capable of encoding a source at a bitrate equal to its entropy, in the special case where all probabilities are powers of 2. A technique known as arithmetic coding can also produce optimal codewords that approach Shannon's theoretical limit, even for more general probability distributions. However, the details of arithmetic coding are beyond the scope of this thesis.

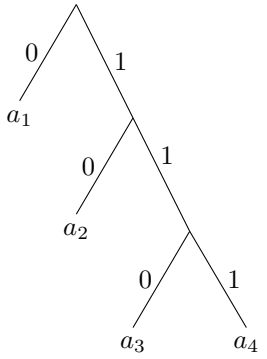


Figure A.1: Huffman Tree

Table A.1: Huffman Coding Example

Symbol	**Probability**	**Codeword**
a_1	0.5	0
a_2	0.25	10
a_3	0.125	110
a_4	0.125	111

Appendix B

Blocking Artifact Removal

One consequence of using the block translation model for inter-frame coding is the occurrence of *blocking artifacts* in the predicted frames. This term describes distortion effects that occur at Macroblock boundaries, as well as similar artifacts created as a result of the transform and quantization process, that occur at transform block boundaries. In H.264, Macroblocks are 16x16 pixel blocks composed of multiple 4x4 (or sometimes 8x8) transform blocks, so it is typical to see more pronounced blocking effects at the Macroblock edges, with smaller, less pronounced artifacts at internal transform block edges. These blocking artifacts have a negative impact on both the subject quality of the video and on objective quality metrics such as MSE or PSNR. Methods to reduce blocking distortion include *in-loop filtering* and *overlapped block motion compensation*, detailed below.

B.1 In-loop filtering

works by applying a 1-D smoothing filter to vertical and horizontal edges of 4x4 blocks in the predicted image. Up to three pixels on either side of the block boundary may be affected, as shown in Figure B.1 (p0 and q0 represent pixels at the boundaries of adjacent blocks p and q). At the encoder, such filters are applied to reconstructed reference images to improve prediction, while at the decoder, they are applied as part of image reconstruction to improve subjective quality. For H.264, filter strength is selected according to the quantization parameter used, the pixel gradient near block edges (Figure B.1), and a *boundary strength* value, which is determined by the rules in Table B.1 [61].

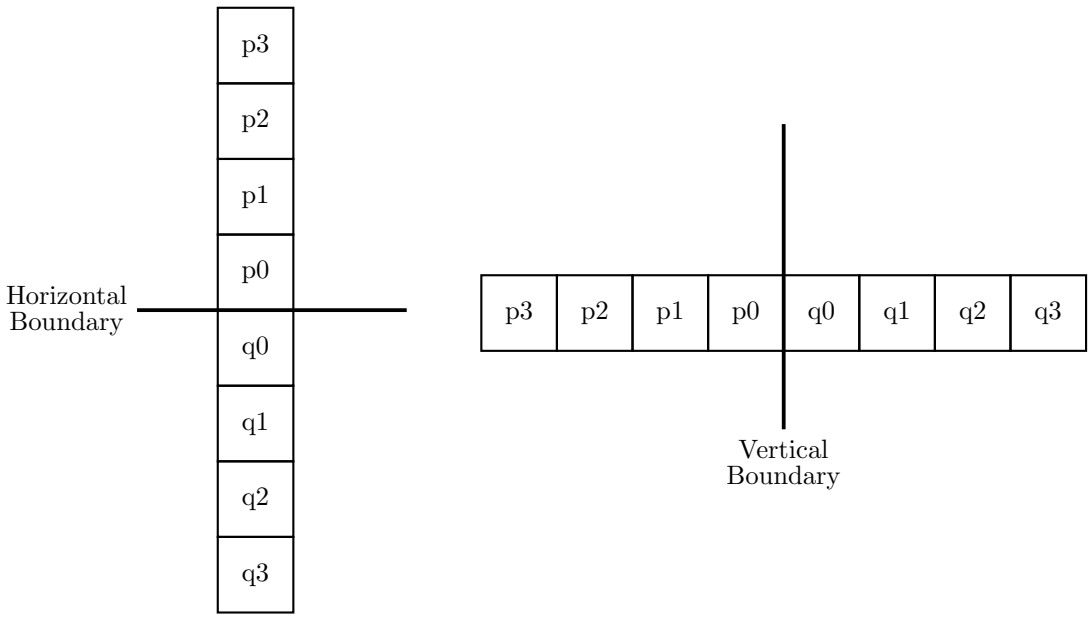


Figure B.1: Horizontal and vertical boundary regions

Finally, the filter is only applied if the following rules are satisfied:

1. Boundary strength > 0 and
2. pixel gradients are above a predefined threshold,

where the threshold increases with QP in order to accommodate the greater coding error (size of blocking artifacts) at higher QP values [42].

Table B.1: Boundary Strength Selection

Case	Boundary Strength
If either block is intra-coded and boundary is a Macroblock edge	Bs = 4 (strongest filtering)
Else either block is intra-coded and boundary is not a Macroblock edge	Bs = 3
Else either block is intra-coded and either contain non-zero coefficients	Bs = 2
Else both blocks have different reference frames or different motion vector values	Bs = 1
Else	Bs = 0 (no filtering)

B.2 Overlapped Block Motion Compensation

Overlapped Block Motion Compensation (OBMC) is another technique used to suppress blocking artifacts and smooth block boundaries. For each Macroblock, the motion vectors of neighbouring blocks are used to create additional motion hypotheses. A weighted average of the different hypotheses forms the final prediction block [40]. One example of OBMC is the advanced prediction mode (Annex F - APM) of the H.263 codec [62]. APM uses four motion vectors per Macroblock (one for each 8x8 subregion), and each block is calculated as a weighted sum of three prediction values. First, the motion vector from the current block is used, followed by two out of four MV's from adjacent blocks. For a specific pixel, $\hat{p}(i,j)$, its weighted prediction value is given by the equation below:

$$\hat{p}(i,j) = \frac{\left[\sum_{k=0}^{2} p(i+\hat{u}_k, j+\hat{v}_k) \cdot H_k(i,j) + 4\right]}{8}, \tag{B.1}$$

where $(\hat{u}_k, \hat{v}_k)$ is the motion vector for block k.For $k = 0$, the current block is used, for $k = 1$, the block above or below is used, and finally, for $k = 2$ the block to the left or right. $p(i+\hat{u}_k, j+\hat{v}_k)$ is then the pixel value from the reference block and $\{H_k(i,j); k = 0, 1\}$ are weights given by the following matrices:

$$H_0 = \begin{bmatrix} 4 & 5 & 5 & 5 & 5 & 5 & 5 & 4 \\ 5 & 5 & 5 & 5 & 5 & 5 & 5 & 5 \\ 5 & 5 & 6 & 6 & 6 & 6 & 5 & 5 \\ 5 & 5 & 6 & 6 & 6 & 6 & 5 & 5 \\ 5 & 5 & 6 & 6 & 6 & 6 & 5 & 5 \\ 5 & 5 & 6 & 6 & 6 & 6 & 5 & 5 \\ 5 & 5 & 5 & 5 & 5 & 5 & 5 & 5 \\ 4 & 5 & 5 & 5 & 5 & 5 & 5 & 4 \end{bmatrix}, H_1 = \begin{bmatrix} 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 \\ 1 & 1 & 2 & 2 & 2 & 2 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 2 & 2 & 2 & 2 & 1 & 1 \\ 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 \end{bmatrix}, H_1 = \begin{bmatrix} 2 & 1 & 1 & 1 & 1 & 1 & 1 & 2 \\ 2 & 2 & 1 & 1 & 1 & 1 & 2 & 2 \\ 2 & 2 & 1 & 1 & 1 & 1 & 2 & 2 \\ 2 & 2 & 1 & 1 & 1 & 1 & 2 & 2 \\ 2 & 2 & 1 & 1 & 1 & 1 & 2 & 2 \\ 2 & 2 & 1 & 1 & 1 & 1 & 2 & 2 \\ 2 & 2 & 1 & 1 & 1 & 1 & 2 & 2 \\ 2 & 1 & 1 & 1 & 1 & 1 & 1 & 2 \end{bmatrix},$$

APM, and OBMC in general, were excluded from all H.264 profiles, likely due to the higher computation cost that they require at the decoder [40]. Another, more computationally expensive approach to OBMC is to use a statistical model, weighting each motion hypothesis based on the probability that its match is accurate. An early example of statistical OBMC is the work of Orchard & Sullivan [51], where the researchers experimented with different window shapes or "supports". The authors also considered adaptive weightings, chosen as a function of the video data being coded, and used physically overlapped window supports.

Other researchers have used statistical OBMC models in a DVC codec to enhance bidirectional motion compensation. Deligiannis *et al.* implemented OBMC following the work of Orchard & Sullivan, using overlapping spatial blocks and adaptive weights. A special metric, Pixel Error Ratio (PER), was used to calculate block weightings [16].

Statistical OBMC may have interesting applications within the proposed codec, as motion hypotheses can be derived from both Chroma planes. In this case, since motion is estimated in a separate plane from which the motion is applied, the codec may produce imperfect or inaccurate motion vectors. As such, previous metrics such as PER may not be available to us; instead, probability could be calculated as some function of the block difference, using MSE, PSNR, or SAD.

Appendix C

Additional Benchmark Videos

In Section 5.5, Spatial Information measurements (SI) were used to predict the percent of an encoded video's bitrate required to represent Chroma information (referred to as $BR_{C/T}$). The existing data set of Benchmark and Canola videos were used to train a linear regression model, and six additional videos were selected as an independent "test set", to compliment this "training set".

Similar to the Benchmark sequences, these test set videos were chosen from the Xiph.org video collection,[1] where they were originally represented using the YUV4MPEG2 video format (.y4m). The script described in Figure 4.1 was used to convert them from .y4m to raw .yuv format, which was a prerequisite for both the SI calculation script and JM encoder. Below, Table C provides the names of the six videos, listed alphabetically, along with their frame length and original source.

Table C.1: Additional Xiph Benchmark Videos

Video Title	**# Frames**	**Source**
Bowing	300	ftp://ftp3.itu.int/video-site/sequences/Bowing/
Bus	150	http://trace.eas.asu.edu/yuv/bus/bus_cif.7z
Coastguard	300	http://trace.eas.asu.edu/yuv/coastguard/coastguard_cif.7z
Ice	240	ftp://ftp.tnt.uni-hannover.de/pub/svc/testsequences/Ice/
Tempête	260	ftp://ftp3.itu.int/video-site/sequences/tempete.zip
Waterfall	260	http://trace.eas.asu.edu/yuv/waterfall/waterfall_cif.7z

Measurements taken for each video are given by Table C.2 including their SI, TI, and bitrate characteristics. These test videos were selected to cover a wide range of spatial and temporal and colour information, similar to the training set. Figure C.1 provides a further visualization of the SI / TI distribution for the test set, while Figure C.2 depicts the breakdown of video bitrate by category.

Table C.2: Additional Videos - Measurements

Video	$\boldsymbol{SI_Y}$	$\boldsymbol{SI_U}$	$\boldsymbol{SI_V}$	**TI**	$\boldsymbol{BR_{C/T}}$ **(Average)**
Bowing	69.93	10.36	9.11	24.93	6.69
Bus	150.15	20.49	22.36	38.33	4.07
Coastguard	121.11	13.64	9.73	34.91	2.26
Ice	90.40	20.01	32.32	29.5	11.38
Tempête	110.30	53.52	32.89	21.40	10.67
Waterfall	69.26	37.92	22.25	8.25	22.17

[1] https://media.xiph.org/video/derf/

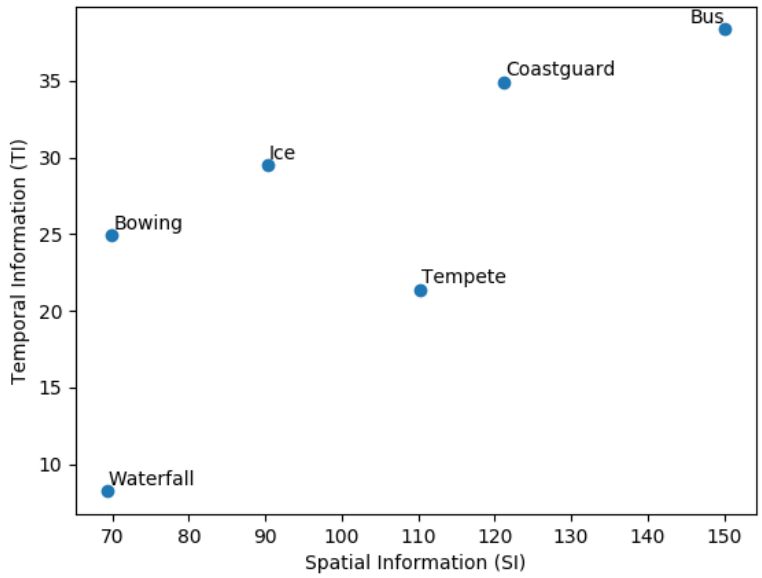


Figure C.1: SI / TI Graph

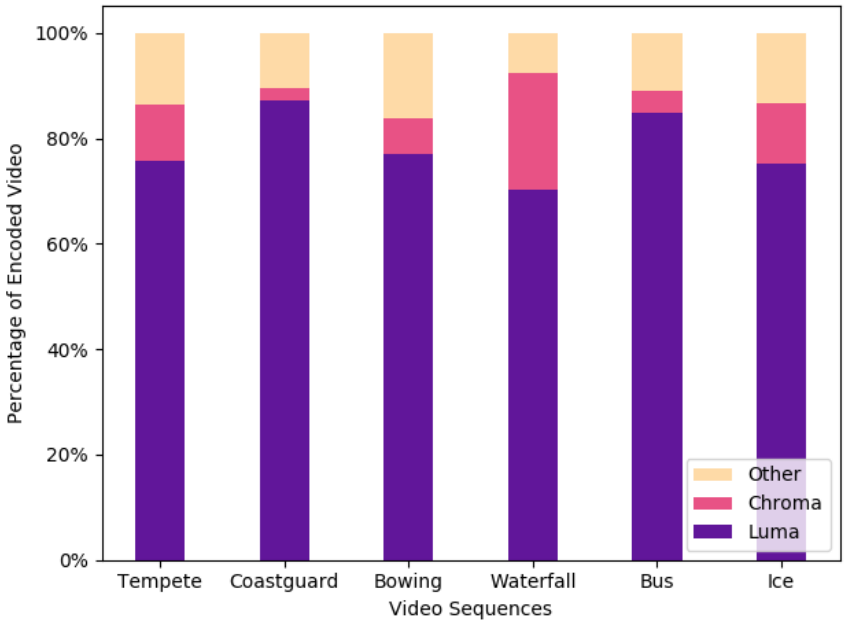


Figure C.2: Proportion of Video Bitrate by Category