ADVANCING VIDEO COMPRESSION WITH ERROR RESILIENCE AND CONTENT ANALYSIS

A Dissertation

Submitted to the Faculty

of

Purdue University

by

Di Chen

In Partial Fulfillment of the

Requirements for the Degree

of

Doctor of Philosophy

August 2020

Purdue University

West Lafayette, Indiana

THE PURDUE UNIVERSITY GRADUATE SCHOOL
STATEMENT OF DISSERTATION APPROVAL

Dr. Fengqing Zhu, Chair

School of Electrical and Computer Engineering

Dr. Edward J. Delp

School of Electrical and Computer Engineering

Dr. Amy Reibman

School of Electrical and Computer Engineering

Dr. Stanley Chan

School of Electrical and Computer Engineering

Approved by:

Dr. Dimitrios Peroulis

Head of the School of Electrical and Computer Engineering

ACKNOWLEDGMENTS

First of all, there are no words to express my gratitude towards my doctoral advisor, Prof. Fengqing Zhu. She has always encouraged me to learn new things and to try new ideas. She guided me to achieve the milestones through the graduate program. I am very grateful to her for molding me to think like a scholar and teaches me how to research, how to manage projects, and how to communicate. I feel very proud and accomplished to have worked with her. I have enjoyed our technical discussions when her insightful feedback and suggestions guided my projects onto the right track. She was also gracious to provide suggestions on how to get more involved in the lab and on my career development.

I also want to personally thank all the members of my Ph.D. advisory committee: I would like to thank Prof. Edward Delp for providing invaluable guidance since I joined the Video and Image Processing Laboratory (VIPER). He led me through my first project and my first published paper at Purdue. He participated in my weekly meetings where his valuable insights led to fruitful research ideas. I would like to thank Prof. Amy Reibman and Prof. Stanley Chan for their suggestions, discussions, and supports. I feel fortunate to have learned Digital Video Systems from Prof. Reibman, which serves as the knowledge fundamentals of my research.

VIPER is a great place to work on image and video processing problems with a group of talented engineers. I have been very fortunate to be a part of this vibrant research environment. I would like to thank Dr. Neeraj Gadgil who has been a great mentor to me and a very supportive teammate for the video compression with error resilience project. I really appreciate his guidance during the early stages of my doctoral research. I want to thank Dr. Chichen Fu for being a great teammate for the video compression with block-based texture segmentation project. I would like to

thank Dr. Qingshuang Chen for working with me on the project video compression with content analysis.

I would like to extend my gratitude to all my former and current VIPER lab members for their friendships and supports: Dr. Albert Parra Pozo, Dr. Neeraj J. Gadgil, Dr. Joonsoo Kim, Dr. Yu Wang, Blanca, Dr. Chichen Fu, Dr. Shaobo Fang, Dr. Javier Ribera Prat, Dr. David Joon Ho, Blanca Delgado, He Li, Chang Liu, Sriram Baireddy, Enyu Cai, Alain Chen, Di Chen, Qingshuang(Cici) Chen, Yuhao Chen, Jeehyun Choe, David Gera, Jiaqi Guo, Shuo Han, Hanxiang (Hans) Hao, Jiangpeng He, Jnos Horvth, Han Hu, Soonam Lee, Runyu Mao, Daniel Mas Montserrat, Ruiting Shao, Zeman Shao, Changye Yang, Sri Kalyan Yarlagadda, Yifan Zhao. In addition, I would also like to thank the students of Prof. Reibman and Prof. Chan who voluntarily participated in my subjective study in the project.

I wish to express my gratitude to Prof. John Woods who got me started my research direction on video coding.

I would like to thank my father, Zhuoning Chen, sister, Xi Chen, for their endless love and support.

TABLE OF CONTENTS

Page

Page

LIST OF TABLES

LIST OF FIGURES

Figure Page

ABBREVIATIONS

ADST	asymmetric discrete sine transform
AOM	Alliance for Open Media
ARMA	Auto-Regressive Moving Average
AV1	AOMedia Video 1
AVM	Artifact-based Video Metric
BD	Bjontegaard delta
BMA	Boundary Matching Algorithm
CABAC	Context-Adaptive Binary Arithmetic Coding
CDEF	Constrained Directional Enhancement Filter
CNNMCR	Convolutional Neural Network-Based Motion Compensation Refinement
CTU	Coding Tree Unit
CW-SSIM	Complex Wavelet Structural Similarity Index
DCT	Discrete Cosine Transform
DF	Deblocking Filter
DP	Data Partitioning
DPI	Duplicated Prediction Information
DT-CWT	Dual-Tree Complex Wavelet Transform
DVC	Deep Video Compression
FAST	Features from Accelerated Segment Test
FC	Frame Copy method
FCN	Fully Convolutional Network
FMO	Flexible Macroblock Ordering
FRUC	Frame Rate Up Conversion

FV	Fisher Vector
GAM	Generative Adversarial Nets
GF	Golden Frame
GLCM	Gray Level Co-occurrence Matrix
GOP	Group Of Pictures
GPU	Graphics Processing Unit
GVCNN	Grouped Variation Convolutional Neural Network
HEVC	High Efficiency Video Coding
HM	HEVC Test Model
HMVE	Hybrid Motion Vector Extrapolation
IDR	Instantaneous Decoding Refresh
IP	Internet Protocol
IPCED	Intra Prediction method via Convolutional Encoder-Decoder network
IPCNN	Intra Prediction Convolution Neural Network
IPFCN	Intra Prediction using Fully Connected Network
ISO	International Standards Organisation
ITU	International Telecommunications Union
LBP	Local Binary Patterns
LSTM	Long Short Term Memory
MASCNN	Multi-scale Adaptive Separable Convolutional Neural Network
MDC	Multiple Description Coding
ME	Motion Estimation
MMA	Multi-frame Motion vector Averaging method
MPEG	Moving Picture Experts Group
MRF	Markov Random Field
MSE	Mean Square Error
MV	Motion Vector
MVE	Motion Vector Extrapolation

PCM	Pulse Code Modulation
PMCNN	Pixel Motion Convolutional Neural Network
PS-RNN	Spatial Neural Recurrent Network
PSNR	Peak Signal-To-Noise Ratio
PU	Prediction Unit
QP	Quantization Parameter
RANSAC	Random Sample Consensus
RD	Rate Distortion
RMM	Recurrent Neural Network
ReLU	Rectified Linear Unit
SATD	Sum of Absolute Transformed Difference
SI	Switching I-frame
SIFT	Scale Invariant Feature Transform
SP	Switching P-frame
SSIM	Structural Similarity Index
STSIM	Structural Texture Structural Similarity Index
SURF	Speeded Up Robust Features
SVM	Support Vector Machine
UGC	User Generate Content
VAE	Variational Auto-Encoder
VMAF	Video Multimethod Assessment Fusion
VRF	Virtual Reference Frame
WARN	Wide Activation Residual Network
WCW-SSIM	Weighted Complex Wavelet Structural Similarity Index
WHT	Walsh Hadamard Transform

ABSTRACT

Chen, Di Ph.D., Purdue University, August 2020. Advancing Video Compression With Error Resilience And Content Analysis. Major Professor: Fengqing Zhu.

In this thesis, two aspects of video coding improvement are discussed, namely error resilience and coding efficiency.

With the increasing amount of videos being created and consumed, better video compression tools are needed to provide reliable and fast transmission. Many popular video coding standards such as VPx, H.26x achieve video compression by using spatial and temporal dependencies in the source video signal. This makes the encoded bitstream vulnerable to errors during transmission. In this thesis, we investigate an error resilient video coding for the VP9 bitstreams using error resilience packets. An error resilient packet consists of encoded keyframe contents and the prediction signals for each non-keyframe. Experimental results exhibit that our proposed method is effective under typical packet loss conditions.

In the second part of the thesis, we first present an automatic stillness feature detection method for group of pictures. The encoder adaptively chooses the coding structure for each group of pictures based on its stillness feature to optimize the coding efficiency.

Secondly, a content-based video coding method is proposed. Modern video codecs including the newly developed AOM/AV1 utilize hybrid coding techniques to remove spatial and temporal redundancy. However, the efficient exploitation of statistical dependencies measured by a mean squared error (MSE) does not always produce the best psychovisual result. One interesting approach is to only encode visually relevant information and use a different coding method for "perceptually insignificant" regions in the frame. In this thesis, we introduce a texture analyzer before encoding the input

sequences to identify detail irrelevant texture regions in the frame using convolutional neural networks. The texture region is then reconstructed based on one set of motion parameters. We show that for many standard test sets, the proposed method achieved significant data rate reductions.

1. INTRODUCTION

1.1 Video Basics

In this section, we provide a brief overview of basic principles and terms of video coding used throughout this thesis. We describe the timeline of the development of the video coding standards, the main methods that the video codec use to reduce the redundant information in the video and the video transmission channel.

1.1.1 Video coding standards

Over the last few decades, a series of video coding standards have been developed by the standardisation bodies, such as the international standards organisation (ISO) and the international telecommunications union (ITU). Figure 1.1 shows the timeline of the major video coding standards. The MPEG-1, MPEG-2, MPEG-4 are ISO coding standards and the H.26x line is the ITU coding standards. Other video coding standards such as Theora, Daala, VPx and AV1 have been developed in recent years.

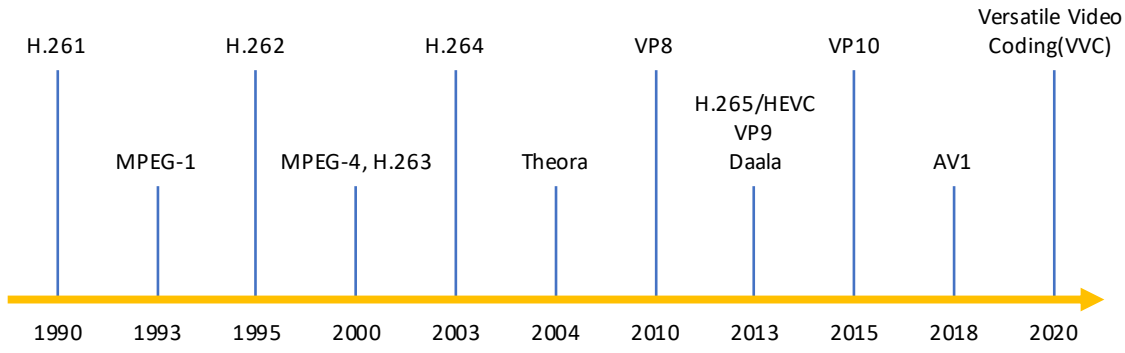


Fig. 1.1.: Majoy video coding standards timeline

In this thesis, our studies are based on VP9 [1, 2] and AV1 [3–5] codecs. VP9 is a video codec developed by Google initially released in 2013 as a compete to Moving Picture Experts Group(MPEG) [6]'s High Efficiency Video Coding (HEVC or H.265) [7, 8]. In 2015, Google and several other industrial leaders jointly founded the Alliance for Open Media (AOM) [9] to define and develop media codecs, media formats, and related technologies that are open-source and loyalty-free to meet the expanding need in web-based video consumption. VP9 is developed into the base of the first edition of the AOM video codec and released in 2018, namely AV1. Multiple new royalty-free coding tools have been contributed by members of AOM. The AV1 codec introduced several new features and coding tools such as switchable loop-restoration [10], global and locally warped motion compensation [11], and variable block-size overlapped block motion compensation [12]. AV1 has exceeded HEVC on almost all Derf's collection of test sequences [13] with respect to BD-PSNR [14]. Like most typical video coding standards, VP9 and AV1 describe the bitstream structure and syntax to standardized the decoding process that takes a sequence of compressed frames and turns it into a sequence of decompressed video frames that are ready to be displayed in correct order. The encoding process is flexible as long as the encoded bitstream can be correctly decoded by the codec. There are many ways of choosing how to encode the original video which can be better or worse depending on how many bits they end up using and how the reconstructed video looks to the human visual system.

1.1.2 Video compression by redundant information removal

The goal of video compression is to represent a video with the least amount of bits possible by reducing the redundant information in the video. There are two major kinds of redundancy: spatial redundancy and temporal redundancy. Spatial redundancy means that the nearby pixels in a frame is correlated since their pixel values usually do not change abruptly. Temporal redundancy means that the consecutive

frames in a video sequence are usually very similar within the same scene. Video codecs usually utilize the redundancy and make prediction of the pixel values in the frame and only send the prediction error which has significantly lower entropy than that of the original frame.

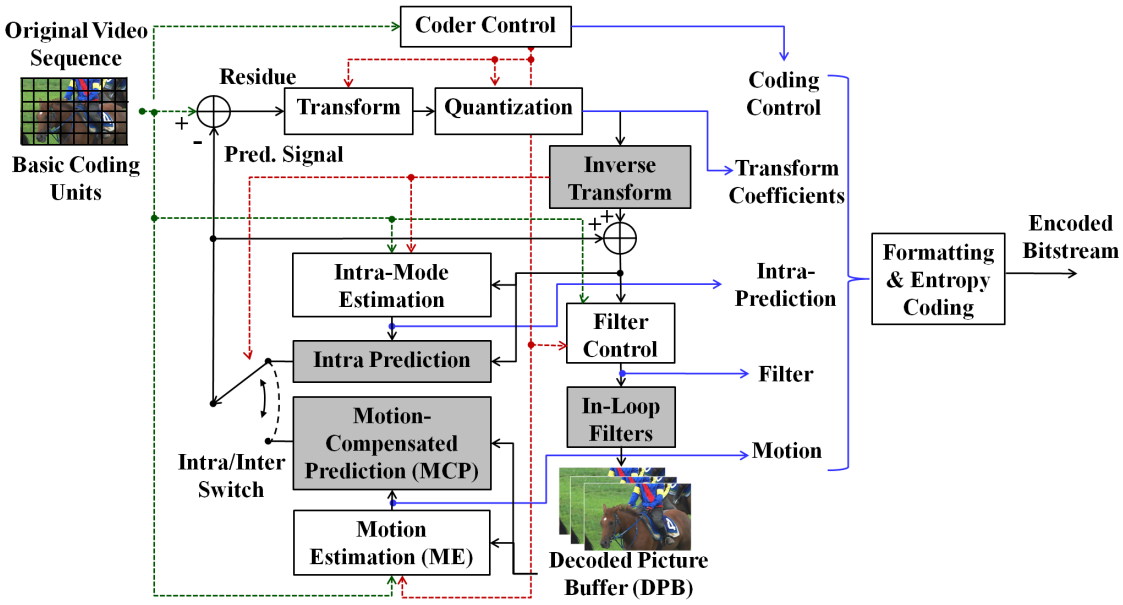


Fig. 1.2.: Block diagram of a typical video encoder. See [15]

Figure 1.2 shows the block diagram of a typical video encoder. An original video frame is first divided into non-overlapping basic prediction units or blocks. The largest block, called the superblock, is 64x64 pixels in VP9 and 128x128 in AV1. The blocks are processed in raster order. The superblocks can be subdivided into smaller blocks just like HEVC. The smallest partition is of size 4x4. The subdivision is done with a recursive quadtree. Unlike HEVC, VP9 and AV1 support horizontal or vertical subdivision. A typical partitioning of a 64x64 block is shown in Figure 1.3. Each block is predicted using a single or multiple previously encoded blocks on the difference frames or the same frame, called the inter prediction and the intra prediction respectively. The blocks using inter prediction are called inter blocks. An inter-block contains a motion vector (MV) that specifies the offset in the reference frame of the part of the image to use as a prediction for this block. The MV can specify a

fractional pixel offsets in X and Y direction. In VP9 and AV1, $\frac{1}{8}$ sub-pixel motion compensation is used. The process of searching the MV for inter blocks is called motion estimation (ME). A new global & warped motion mode is introduced in AV1. In addition, VP9 support three reference frame candidates while AV1 supports up to six reference frame candidates.

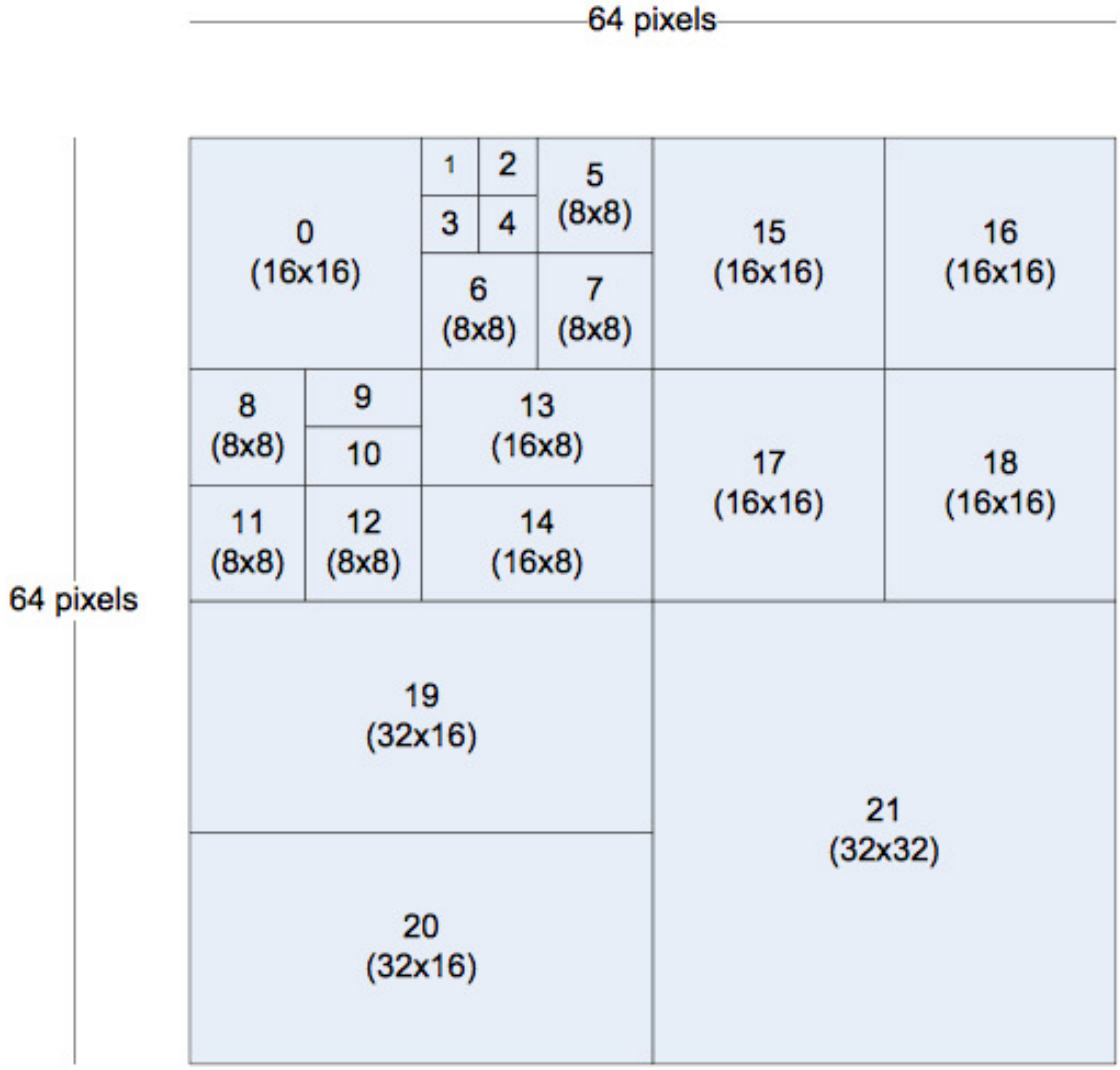


Fig. 1.3.: A typical partitioning of a 64x64 block

Intra prediction uses the pixels on the upper and/or left side of the current block. The pixel values of the current block are estimated using one of several directional

modes of intra prediction. AV1 has 56 angular intra prediction modes and several other intra prediction modes.

As mentioned above, a block can use either a single reference frame or a combination of two reference frames. The latter is called the compound prediction, where the prediction is first formed from each reference frame then the final prediction is generated as the average of these two. It is expected that in certain cases this average is a better predictor than both single predictors even with extra motion information required.

1.1.3 Global motion coding tool of AV1

Global motion coding tool is one of the new inter prediction modes merged into AV1 on the base of VP9. It has shown great coding gain for videos that contain strong non-translational motion [11]. The pure translational model used by block motion compensation has been approved to be an effective technique which is used by modern video codecs, such as VP9, H.264 and HEVC. Blocks of a suitable partitioning size in a frame is predicted using a block in a previously encoded frame where the whole block is assumed to have the same translational motion represented by a motion vector. However, the real motion in video is rarely translational only. For videos with non-translational scenes, such as camera panning and zooming, the only way to get an accurate prediction with a purely translational motion model is to use smaller block sizes which makes coding less efficient. A set of coding tools that based on affine or perspective motion model, including global motion coding tool, is introduce into AV1. They allow complex motion to be captured at larger block sizes by warping the blocks. The global motion model is described by the following:

$$\alpha \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{11} \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{1.1}$$

α represents a weight factor, (x, y) represents coordinates of the original pixel in a current frame and (x', y') represents the corresponding warpped coordinates in a reference frame. The 3x3 homography matrix H has 8 degrees of freedom which allows blocks to be transformed into arbitrary quadrilaterel. It can be decreased into lower order motion model. For example, the matrix H for affine projection with 6 degrees of freedom is restricted by:

$$h_{31} = h_{32} = 0, \quad h_{33} = 1 \tag{1.2}$$

The current implementation of AV1 supports up to 6-parameter affine model for the whole frame. The parameters in the motion model are estimated using a feature matching scheme followed by robust model fitting. First the FAST features [16] are computed in the current frame and the reference frame. After finding a set of correspondence between the feature points in the current frame and the reference frame, a desired motion model is computed using RANSAC [17]. There is a set of global motion parameters for every reference frames of the current frame. These parameters are transmitted to the decoder in the frame header.

1.1.4 Rate distortion optimization

Rate distortion optimization is used for block subdivision decision as well as the choice of prediction modes and reference frames in motion compensated prediction. The goal of an encoder is to minimize the distortion D subjective to a constraint R, where R is the number of bits used. The optimization problem can be solved using Lagrangian optimization where the distortion term is weighted against a rate term [18]. The Lagrangian optimization problem is given by

$$min\{J\}, \quad where \;\; J = D + \lambda R \tag{1.3}$$

where the Lagrangian rate-distortion function J is minimized for a particular value of Lagrange multiplier λ.

The perceived distortion is very hard to measure as the characteristics of the human visual system are complex and not well modeled. In practice, highly imperfect distortion models are used such as MSE. λ value is subject to approximations and is specified by video codec. The R contains the bits for encoding the prediction error/residual and the corresponding motion information such as the MV and the prediction mode.

The shape of the block subdivision quadtree and the reference frames and the prediction mode of each block are chosen so that the Lagrangian rate-distortion function J is minimized.

1.1.5 Transformation and quatization

The prediction error/residual then undergoes an orthogonal transformation such as discrete cosine transform (DCT), asymmetric discrete sine transform (ADST) or Walsh Hadamard Transform (WHT). The transform reduces the statistical correlation within the residual and packs most of the signal's energy into fewer coefficients in the frequency domain. The quantization of the transform coefficients allows different level of lossy compression. The encoder controls the compression ratio by adjusting the quantization parameter (QP) which specifies the width of the quantizer bin. The higher QP value is, the fewer bits are used.

Thus, each block of the input video frames is expressed in terms of prediction mode, reference frames, transform coefficient and control signals. Besides control signals for each block, there are also control signals at frame level such as the reference frame candidates for the current frame, error resilience mode, etc. Above described data is then concatenated and entropy-coded using arithmetic coding.

1.2 Video Coding And Transmission

During the last decade, there has been a significant increase in the amount of video traffic over the Internet. With the development of 3G/4G and WiFi networks, people are not only watching more online videos but also command better quality online videos which occupy more bandwidth. According to [19], traffic from wireless and mobile devices will account for more than 63 percent of total IP traffic by 2021. Globally, Internet video traffic will be 82 percent of all consumer Internet traffic by 2021, up from 73 percent in 2016. The increase in live Internet video traffic, Internet surveillance video traffic, virtual reality and augmented reality video traffic, consumer video-on-demand traffic and content delivery network traffic are accounted for this estimation. Also, mobile data traffic is expected to increase sevenfold between 2016 and 2021 which is twice as fast as fixed IP traffic. This dramatic increase in video traffic over the Internet has raised challenges for developing efficient video coding techniques.

As shown in Figure 1.4, the original video signals are encoded and transmitted over noisy channels. The decoder reconstructed the video signals based on the received encoded data. The video compression codecs act as the source encoder and source decoder.

Fig. 1.4.: A commen video transmission system. See [15]

A defining characteristic of a wireless channel is the variation of the channel strength over time and frequency [20]. This can cause packet loss during signal transmission. In real-time applications such as video chat or live streaming, retransmission of lost packets is not feasible. As a result, only a subset of transmitted packets is available at the receiver, which must reconstruct the signal from the available informa-

tion [21]. Many well-established video compression standards such as H.264 [22, 23], HEVC, VP8 [24], VP9 and AV1 are mainly aimed at achieving better compression efficiencies. Due to the use of spatial-temporal correlations for compression, such compressed bitstreams typically become vulnerable to errors. A typical video encoder uses "in-loop" decoder so that error-free frames are used as a reference at the encoder, however the reference frames at the decoder may contain errors. These errors can then propagate to the frames decoded next, until an instantaneous decoding refresh (IDR) or a key frame is successfully received by the decoder [21]. Therefore, error concealment and resilience methods are indispensable for video delivery over unreliable channels.

1.2.1 Channel model

In a typical network scenario, packet loss patterns are usually bursty. Under sufficiently high transmission load, there are spontaneous overload peaks causing packet loss. In order to study the error resilience performance of the video codec in the transfer of real-time data over the Internet, a Gilbert-Elliott model is used as a stochastic channel model to simulate the bursty packet loss pattern [25, 26].

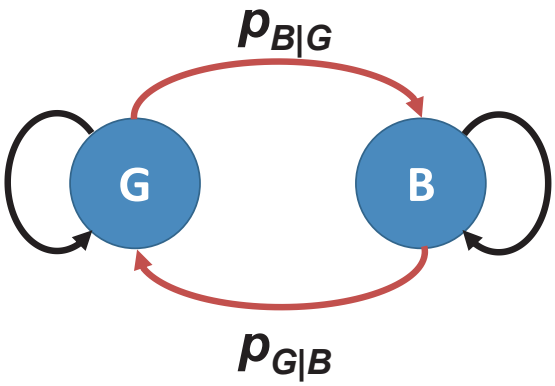


Fig. 1.5.: Gilbert-Elliott model (Two state markov model)

Figure 1.5 illustrates the architecture of the Gilbert model. In this two-state model, condition "good" denoted as "G" indicates that the packet is correctly received

and condition "bad" denoted as "B" indicates that the packet is lost. The transition probability $P_{M|N}$ is the probability of changing the state from N to M. For example, $P_{B|G}$ is the probability when the previous state is "G" and the current state is "B", and $P_{G|G}$ is the probability when the previous state is "G" and the current state is also "G". Therefore:

$$\begin{aligned} P_{G|G} &= 1 - P_{B|G} \\ P_{B|B} &= 1 - P_{G|B} \end{aligned} \tag{1.4}$$

The probability of packet loss can be expressed as:

$$\begin{aligned} P_B &= P_{B|B} \times P_B + P_{B|G} \times P_G \\ &= (1 - P_{G|B}) \times P_B + P_{B|G} \times (1 - P_B) \end{aligned} \tag{1.5}$$

After reorganization,

$$P_B = \frac{P_{B|G}}{P_{B|G} + P_{G|B}} \tag{1.6}$$

The packet loss burst length is the average length of "consecutive stays" in the state "B:"

$$\begin{aligned} L_B &= 1 \times P_{G|B} + 2 \times P_{B|B} \times P_{G|B} + 3 \times P_{B|B}^2 \times P_{G|B} + ... \\ &= P_{G|B}^{-1} \end{aligned} \tag{1.7}$$

In [27] a typical packet loss rate P_B is between 0 and 0.6, and burst length L_B is between 2 and 20. It can also be shown that when P_B is small, L_B is large; and vice versa.

1.3 Overview Of Region-based Video Coding

Region-based video coding is an alternative approach to the traditional schemes of block-based video coding, where the coding units are not square image patches but regions provided by processes such as image segmentation. Such research can be categorized into two groups: saliency-based coding and object-based coding. Saliency-based coding uses image saliency as the description of region-of-interest. For example, in [28], a video frame is first segmented into several connected regions of arbitrary

partition. Then it uses region-based motion estimation to reconstruct each segment in the frame. The method proposed in [29] decomposes a frame into frequency sub-bands and search for region-of-interest within each sub-band. The wavelet coefficients are transmitted in the order of the importance of each region. Object-based codecs automatically detect objects from the video and generate a foreground/background segmentation map. The segmentation maps can then be used to reconstruct the pixels with different models. In [30], the authors investigated object-based video compression for surveillance videos, in which several segmentation methods are considered to extract the background. Coding overhead and the corresponding coding performance are also investigated. Authors in [31] proposed an object-based coding method by pixel state analysis. The pixel state analysis is utilized to detect the foreground and background regions in the video and the pixels of the foreground are compressed using lossless compression.

Since most of the state-of-art video codecs use block-based hybrid coding techniques, many recent works of region-based video coding are integrated into the block-based video coding framework. Several region-based rate-control schemes are proposed by assigning quantization levels to blocks based on the regions they belong to [32–34]. In [35], frames are partitioned into multiple slices. Then the best macroblock mode and suitable motion compensation search pattern for a given slice is chosen. In [36], the authors propose to reduce the bit rate of fixed background by reconstructing macroblocks in the fixed background using pulse code modulation (PCM) in H.264. Other approaches involve controlling the number of non-zero DCT coefficients of the region-of-interest [37, 38].

1.3.1 Texture analysis/synthesis based video coding

A popular region-based video coding approach is the texture analysis/synthesis approach. An overview of this approach is shown in Figure 1.6. Two additional modules are typically included compared to conventional video codecs. The texture

analyzer identifies the texture regions as a pre-processing step before the encoder. Outputs from the texture analyzer such as model parameters are treated as side information, which is transmitted to the decoder for texture region reconstruction. We describe here a few representative works. A comprehensive survey of many promising texture analysis/synthesis based video coding methods is discussed in [39].

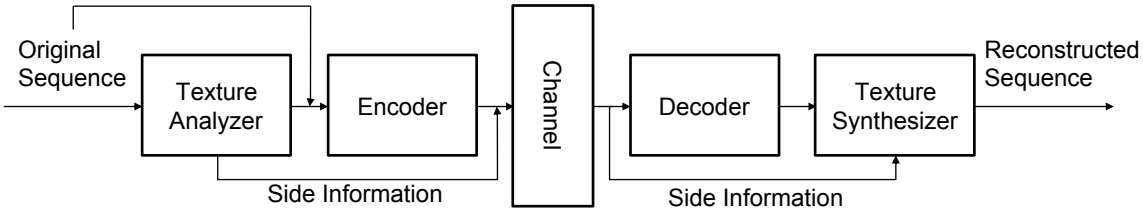


Fig. 1.6.: Overview of texture analysis/synthesis based video coding

In [40–42], Ndjiki-Nya *et al.* proposed a close-loop texture analysis/synthesis method. Texture is identified with dense motion field followed by a split-and-merge algorithm. Then the texture region is reconstructed by texture warping with an overlap region on the texture/non-texture boundary that minimizes the subjective annoyance of the blending. A video quality metric that comprises both global and local components is used for in-loop optimization and video quality assessment.

Bosch *et al.* proposed a segmentation-based video compression using texture and motion models [43] . Two texture features, gray level co-occurrence matrix (GLCM) and Gabor filters, in combination with two segmentation strategies, split-and-merge and K-means clustering, are investigated to identify texture regions. Texture region is reconstructed with texture warping using global motion models with an open-loop framework. Only subjective video quality assessment was performed for the reconstructed videos.

Zhang and Bull proposed a parametric framework for video compression based on texture warping and synthesis [44, 45]. Texture regions are segmented using features derived from the complex wavelet transform and further classified into static and dynamic texture regions according to their spatial and temporal characteristics

and reconstructed by texture warping and an auto-regressive moving average process (ARMA), respectively. An artifact-based video metric (AVM) is proposed to evaluate the quality of the reconstructed video. This method is also employed in-loop to prevent warping and synthesis artifacts.

More recently, a texture analysis/synthesis approach is integrated in HEVC video codec [46]. The proposed method identifies and processes static and dynamic textures based on 2D dual tree complex wavelet transform and steerable pyramid transform [47]. Comparing to previous work [48], different strategies for identifying static and dynamic textures are considered in [46]. In [49], correspondence analysis is explored for the analysis of motion patterns in a video on the basis of optic flow data. Optic flow residual is used as an indicator for dynamic textures. The flow lines generated from optic flow data are used in texture synthesis for creating an illusion of continuously flowing texture.

1.4 Video Coding Using Neural Network

Deep neural network has shown its superiority in many challenging tasks, especially in image processing and computer vision. Recent years, adapting deep neural networks into the field of video coding has become an active research area [50, 51]. There are two main categories of work, namely module based video coding schemes which apply deep network with traditional video coding, and end-to-end deep video coding schemes that are built primarily using deep neural networks.

1.4.1 Module based video coding

Module based method uses a hybrid video coding framework that incorporates deep neural networks with traditional video coding schemes. The use of deep neural networks to improve coding efficiency has been explored in almost all coding modules [50,51]. In this section, we review different coding modules including intra prediction, inter prediction, quantization and entropy coding, in- and out- loop filtering.

Intra Prediction

An input video sequence passes through a video compression framework is divided into frames, and frames are divided into blocks. Blocks are compressed in a predefined order and the compressed blocks can be used to predict subsequent blocks in the same frame. The process of using compressed blocks to predict subsequent ones is known as intra prediction.

In traditional video coding scheme, there are several predefined intra prediction modes and the one with the minimal rate-distortion cost is selected when comparing with other modes. Recent hybrid video coding frameworks use deep neural networks in intra prediction have shown comparative and even improved performance compared to traditional methods. Cui *et al.*proposed an intra prediction convolution network (IPCNN) that integrates CNN into the HEVC intra prediction module [52]. The IPCNN has 10 convolutional layers and batch normalization is used except for the first and the last convolutional layer. Rectified linear unit (ReLU) is selected as the nonlinear activation function. The IPCNN takes the best estimation predicted by HEVC intra prediction and three neighboring reconstructed blocks as input, and use the subtraction of the original blocks and the input as target to train a residual model. The output of IPCNN is the residual block that is used to refine the prediction obtained from HEVC. This method uses neighboring blocks as additional context and the residue learning approach achieved marginal gain.

Instead of using CNN based scheme to refine the traditional intra prediction result, Li *et al.*proposed a fully connected network called IPFCN as a new intra prediction mode in HEVC, among the other 35 modes [53]. Except for the non-linear layer PReLU [54], all layers in IPFCN are fully connected layers. Similar to IPCNN, IPFCN uses neighboring pixels as context. For the current $N \times N$ block Y, IPFCN learns a mapping from Y to its reference pixels R, where R contains L lines above and L lines to the left with $4NL + L^2$ pixels in total. They reported around 3% BD-Rate [14] saving compared to the HEVC reference software HM [55]. Pfaff *et al.*also proposed a fully connected network for intra prediction [56]. They took prediction modes into

consideration and trained multiple networks for different prediction modes. They also proposed a second neural network to predict modes from reconstructed samples. They reported around 6% BD-Rate saving compared to HM.

Hu *et al.*adopted a progressive spatial neural recurrent network (PS-RNN) for intra prediction [57]. RNNs are widely used to process time series data, usually one dimensional data. In order to process 2D image signal, the proposed PS-RNN stacks RNNs in two orthogonal directions and uses a convolutional layer to fuse the prediction. The PS-RNN consists of three spatial recurrent units, and it progressively infers information from reference to the current prediction unit (PU). They also proposed to use sum of absolute transformed difference (SATD) as the loss function to train the network since SATD is able to consider rate-distortion cost in a residue block compared to Mean Square Error.

Jin *et al.*designed an convolutional encoder and decoder network (IPCED) and a GAN [58] based training framework for intra prediction with PUs of 32×32, 16×16 and 4×4 [59]. IPCED uses a multi-scale skip architecture as encoder to combine deep global information with the shallow local information. The decoder follows mulit-level branches to generate prediction for each branch. The training framework also includes an auxiliary context discriminator network. They reported $3.41\%, 3.07\%$ and 3.44% bitrate saving for the $Y/Cb/Cr$ channel respectively compared to HEVC baseline.

Neural network based techniques can also be used for mode decision [60], down- and up-sampling [61] in intra prediction. Seki *et al.*proposed a two-stage neural network to predict mode decision instead of utilizing rate distortion optimization [60]. Authors in [61] designed a compact and efficient CNN for up-sampling instead of using hand-crafted method in conventional intra coding.

Inter Prediction

In inter prediction, previously compressed frames are used as references to predict the current frame to be encoded, therefore removing the temporal redundancy. Deep learning can also facilitate inter prediction by improving fractional-pixel motion com-

pensation, reference frame generation, and motion compensation refinement to name a few.

Yan *et al.*proposed a CNN based interpolation filter for half-pixel interpolation [62]. In [63], an image super-resolution approach is used to obtain fractional interpolation results. Later authors in [64] proposed a fractional interpolation method based on a grouped variation convolutional neural network (GVCNN), which showed round 2.2% data rate saving. The network extracts features from integer position samples and then the group variations is generated using the same feature maps. The sub-pixel result is obtained by adding the variations back to the integer sample. Instead of training a separate network for each QP and each half-pel position, a shared feature map is used to infer sub-pixel samples under different QPs at once [62, 63].

Deep neural networks not only demonstrate superiority in factional motion compensation, but also inspired work to select or generate proper references in inter prediction. In [65], a deep learning based frame rate up conversion (FRUC) algorithm in HEVC is introduced to generate virtual reference frame (VRF). The VRF is further enhanced by a CNN model to reduce the compression artifacts and improve the prediction accuracy. In addition, a CTU level coding mode is proposed to better adopt VRF in coding process as well as considering trade-off between compression performance and computational complexity.

Xia *et al.*proposed a multi-scale adaptive separable convolutional neural network (MASCNN) to generate additional reference samples which are pixel-wise closer to the to-be-coded frames [66]. With additional references sample generated from MASCNN, pixel-wise motion is better modeled which saves bits for coding. In addition, a multi-scale sum of absolute transformed difference (SATD) is introduced as the loss function for the multi-scale architecture. In [65, 66], Adaptive Separable Convolution [67] is used since 1D kernels has reduced number of parameters.

Deep neural networks have also been used in refining motion compensation [68,69] . In [68], a network is proposed that takes integer position samples as input, extract feature maps to infer the residual and then obtain the integer-position refinement.

Instead of using network to enhance the reference before motion compensation, a CNN network (CNNMCR) is proposed to refine motion compensation results in [69]. CNNMCR follows the network structure in [70], which considers both the spatial contextual information and the temporal information by taking motion compensated prediction and the neighboring reconstructed region as inputs to the network.

Quantization and Entropy Coding

Quantization and entropy coding are two key steps to binarize compressed representations. Work on leveraging quantization strategies using deep neural networks is still quite limited. In [71], a three layer neural network with 894 trained parameters is used to predict the local visibility threshold C_T for each 64×64 CTU that indicates HEVC distortions. Then the quantization step is derived by regression in

$$log(Q_{step}) = \alpha C_T^2 + \beta C_T + \gamma \tag{1.8}$$

where α, β and γ are three coefficients that are predicted from three two layer networks respectively. A 11% improvement is reported for HEVC when using coefficients predicted from CNN models.

Recent advances in entropy coding mostly focus on achieving accurate probability estimation of coding modes and transform coefficients. In [56] and [72], a fully connected network and a convolutional neural network are designed to predict probability distribution of intra prediction modes respectively and further improve the entropy coding efficiency. Puri *et al.*proposed a 12 layered CNN to predict transform index in multiple transforms framework from the quantized coefficient blocks [73] . Then one is able to use the predicted probability to code the transform mode in variable length. In [74], a network-based arithmetic coding strategy called CNNAC is proposed to directly estimate the DC coefficients probability for HEVC intra coding instead of using hand crafted context-adaptive binary arithmetic coding (CABAC).

In-Loop and Out-Loop Filtering

Lossy video compression usually causes visual artifacts, such as blocking, flickering, ringing and blurring. Typically, filtering is performed to reduce the reconstruction artifacts. Depending on whether a filter is involved in the encoding processing, it is categorized as either in-loop filter or out-loop filter. Recent research shows that deep neural network has a strong ability to learn how to restore degradation generated from compression.

AV1 codec contains three in-loop filters, Deblocking Filter (DF), Constrained Directional Enhancement Filter (CDEF) [75] and Loop-Restoration [76]. Different from these in-loop filters, a CNN called SimNet is proposed to build the relationship between the reconstructed frame and the original frame in [77]. SimNet is based on residual framework, and the depth of the network is varied with the distortion level of the reconstructed frame. SimNet can be applied to both intra- and inter-prediction, with a skip enhancing strategy for inter coding to improve the coding efficiency by avoiding the effect of double enhancement. This approach achieves 7.27% and 5.57% BD-Rate saving for intra- and inter- respectively. With similar skipping enhancement strategy, [78] designed a Wide Activation Residual Network (WARN) that allocates network depth and parameters with wide activation [79] to achieve better performance. This approach achieves 14.42% and 9.64% BD-Rate saving for intra- and inter- coding.

Post processing using neural networks to reduce compression artifacts are first applied to image compression [80–85]. Later on, out-loop filters are investigated using deep neural networks. Wang *et al.*proposed a 10 layer CNN model to enhance each reconstructed frame for HEVC [86]. Instead of using one frame as input, Yang *et al.*proposed a method to use neighboring frames to enhance the current frame [87] . An auto-encoder model is designed to learn the residual on encoder side, and then the residual is transmitted to reconstructed the frame at the decoder [88]. In [89], a learning based multi-frame video quality enhancement is introduced which enlarges the spatial-temporal correlation among frames.

Although module based video coding with deep neural works is able to achieve comparable or even better compression quality compare to the traditional video coding frameworks, the trade-off between computational complexity and compression efficiency is under explored. In addition, the computational cost is often the obstacle for deploying deep module based video coding in practical systems.

1.4.2 End-to-end deep video coding

Another popular direction for video compression is extended from end-to-end deep network image compression schemes [90–92]. Deepcoder [93] is one of the earliest end-to-end deep video scheme that combines several auto-encoders. For intra-prediction, the Deepcoder builds an auto-encoder to compress the block. For inter-prediction, motion estimation and compensation are performed and then compressed by the auto-encoder. Both intra- and inter-prediction residuals are further encoded using a another network. All representations from auto-encoders are quantized and encoded using Huffman entropy coding. This method shows the possibility of building end-to-end deep neural network in the video coding framework, however the performance is not comparable to H.264.

The Pixel Motion CNN (PMCNN) [94] introduces a spatio-temporal modeling and proposes an iterative analysis/synthesis learning approach. The spatio-temporal modeling is extended from the PxielCNN [95] to sequentially predict frames in chronological order. The residuals are then analyzed and synthesized using a LSTM-based auto-encoder with connection between adjacent stages which was introduced in [90]. This method achieves comparable results to H.264.

DVC introduced in [96] takes advantage of traditional video compression architecture and nonlinear ability of neural networks. A learning based optical flow estimation was proposed, and the method uses two auto-encoders to compress predicted motion and residues. Instead of using two auto-encoders to compress predicted motion and residues, Rippel *et al.*only uses one to achieve the same goal, along with multi-flow

representation and a more sophisticated spatial rate control algorithm [97]. This method is reported to outperform commercial codecs in the low-latency mode.

Other deep video coding schemes includes [98], [99] and [100]. Wu *et al.*views video compression as repeated image interpolation and builds an end-to-end trained network for video compression [98]. Their method uses RNN based image compression [90] to compress the key frames, and builds a conditional interpolation model to interpolate B frames hierarchically. A variational inference approach based on variational autoencoder (VAE) is proposed in [99] for video compression. Their results show a better visual quality on specialized content videos, and achieved comparable quality on generic videos compared to VP9. In [100], a deep generative modeling is proposed which aims to learn a continuous latent representation that can be discretized with minimal information loss for further binarization, instead of generating new videos in [101].

The end-to-end deep video coding frameworks have not outperformed HEVC in terms of PSNR from our observation. Replacing traditional video compression with neural networks is a challenging task, and research is still at its infancy. In addition to coding performance improvement, other issues such as computational complexity, hardware adoption, and power efficiency.

1.5 Texture Analysis And Synthesis

1.5.1 General scene understanding

Region-based video coding relies on having a good understanding of the objects in the scene, so bit allocation for the different regions can be optimized. Scene understanding involves multiple tasks, for example maintaining the coherency of objects in the scene and recognizing the event in a scene. Scene understanding is a challenging computer vision task, because the perception of a scene could be varied from different human point of views. In general, there are two main approaches for scene understanding, context based and semantic based. Context based approaches use

contextual information gathered from nearby objects to analyze the dependencies of different objects in presence, while semantic based approaches rely on the semantic meaning of the global scene or explicit objects to help understand the image. The goal of semantic scene segmentation is to partition a scene based on its semantic meaning. Recent advances in deep neural networks and the availability of large-scale datasets like ImageNet [102], COCO [103] and ADE20K [104] have enabled improved performance in semantic scene segmentation [104–106]. For example, the Fully Convolutional Network (FCN) [105] is one of the most commonly used network architectures for semantic scene segmentation. The major issue with FCN [105] is the lack of global contextual information to categorize global scene, which could lead to segmentation error. The pyramid scene parsing network (PSPNet) [106] addresses this issue by adding a global pyramid pooling module to extract global information from the image.

1.5.2 Texture based approaches

Texture analysis

Many region-based video coding methods use different coding modes for texture in a frame, since regions containing texture are "perceptual insignificant", yet costly to encode. Texture is an essential visual cue in many types of images and videos. In general, texture represents region with no explicit objects, and texture region usually follows a structural pattern or can be approximated by probabilistic models. Analyzing texture regions in an image or video such as detection and synthesis can help understand the scene for subsequent processes, which has drawn a lot of research interest in computer vision and video compression. The key process of analyzing texture is to formulate the texture representation, which is a vector transformed from the input image. Early development of texture representations can be categorized into two areas, filtering based approaches and statistical modelings. Gabor filters is one of the common filter based approaches which convolves the image, and is fol-

lowed by nonlinearity to obtain texture representation [107,108]. Texture can also be represented by a statistical model, e.g., the Markov Random Field (MRF) [109,110]. Later on, local hand crafted features, such as the Scale Invariant Feature Transform (SIFT) [111], Speeded Up Robust Features (SURF) [112] and Local Binary Patterns (LBP) [113], are widely used as texture descriptors. In recent years, neural network has shows success in learning and extracting features from large labeled datasets. CNN based texture representations are very popular, and the most straightforward approach is to directly extract from the fully connected layers, e.g., from the $FC6$ or $FC7$ layer of AlexNet [114]. In [115], the output of the convolutional layer is further encoded with a traditional encoder, namely the Fisher Vector (FV) [116], which shows the orderless pooling of CNN features is a good texture descriptor.

Texture synthesis

Depending on how texture is represented and encoded, texture synthesis may be required to reconstruct pixels corresponding to the texture regions in a frame. There are two main approaches of texture synthesis in general. Non-parametric based approaches resample texture patches from the reference image [117–119]. Parametric statistical models reconstruct the texture regions by optimizing statistic models and sampling from the models [47, 120, 121]. Markov random field (MRF) is one of the most frequently used models. Recently, a pre-trained CNN model on image classification task is used to generate texture patches [122] which showed a new direction for texture synthesis. In addition, generative models have also shown promising results in image synthesis [58, 123]. Li *et al.*proposed a Markovian generative adversarial network for texture synthesis which showed remarkable results in terms of synthesis quality [124]. However, consider the computational cost and implementation challenges, utilizing generative models as a hybrid module at the decoder side of video codec is still not well exploited.

1.6 Contribution Of This Thesis

The main contributions of this thesis are:

- VP9 Video Coding For Lossy Transmission Channels Using Error Resilience Packets

  - We presented a VP9-based error resilient video coding method that uses error resilience packets that consist of the frame-level macroblock prediction information and encoded keyframe information.
  - We compared our proposed method with several previously developed error resilience methods for video compression. Experimental results exhibit that our method performs well in terms of both PSNR and image quality under typical lossy network conditions.

- Multi-Reference Video Coding Using Stillness Detection

  - We proposed an automatic Golden Frame (GF) group stillness feature detection method. Each GF groups is classified into still GF group and non-still GF group based on three metrics.
  - We utilized the GF group stillness feature to adaptively choose the coding structure for each GF group based on its stillness feature to optimize the coding efficiency.

- AV1 Video Coding Using Texture Analysis With Convolutional Neural Networks

  - We propose a Convolutional Neural Network architecture which identifies the "perceptually insignificant" region before encoding the video sequences using conventional video codec.

- We developed a new CNN based texture analysis/synthesis coding tool for AV1 codec that reconstructs the texture region differently for the B-frames and the P-frames which largely reduces the temporal visual artifacts.
- We proposed a visual quality assessment metric for evaluating visual quality of videos with texture synthesized region.
- We compared our proposed method with the original AV1 codec. Experimental results show that our proposed method can achieve significant data rate reduction with satisfying visual quality for both standard test sets and user generated content, which are verified by a subjective study and objective quality assessments.
- Our proposed method is a pioneering work that integrates learning-based texture analysis and reconstruction approach with modern video codec for enhancing video compression performance.

1.7 Publications Resulting From This Work

1. **Di Chen**, Neeraj Gadgil and Edward J. Delp, "VPx video coding for lossy transmission channels using error resilience packets", *Picture Coding Symposium (PCS)*, December, 2016, Nurembreg, Germany.

2. Chichen Fu, **Di Chen**, Edward J. Delp, Zoe Liu and Fengqing Zhu, "Texture segmentation based video compression using convolutional neural networks", *Electronic Imaging (EI)*, January, 2018, Burlingame, CA.

3. **Di Chen**, Zoe Liu, Yaowu Xu, Fengqing Zhu, Edward Delp, "Multi-Reference Video Coding Using Stillness Detection", *Electronic Imaging (EI)*, January, 2018, Burlingame, CA.

4. **Di Chen**, Chichen Fu, Zoe Liu and Fengqing Zhu, "AV1 Video Coding Using Texture Analysis With Convolutional Neural Networks", *arXiv preprint, p. arXiv:1804.09291*, 2018.

5. **Di Chen**, Qingshuang Chen and Fengqing Zhu, "Pixel-level Texture Segmentation Based AV1 Video Compression", *International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, May, 2019, Brighton, UK.

6. **Di Chen**, Qingshuang Chen, Zoe Liu and Fengqing Zhu, "Advances In Region-Based Video Coding Using Deep Neural Network", *Proceedings of the IEEE*, 2020, *under revision.*

2. VP9 VIDEO CODING FOR LOSSY TRANSMISSION CHANNELS USING ERROR RESILIENCE PACKETS

2.1 Video Error Concealment And Resilience

As mentioned in chapter 1, the variation of the channel strength and overloaded transmission may cause bursty packet loss during video transmission. Therefore, the increasing video traffic over error-prone wireless channels has raised some significant challenges for developing efficient coding techniques for this purpose. In real-time applications such as video chat or live streaming, retransmission of lost packets is not feasible. As a result, only a subset of transmitted packets is available at the receiver, which must reconstruct the signal from the available information [21]. This may be achieved by error concealment and error resilience techniques for video compression and transmission.

Various error concealment approaches such as spatial pixel interpolation, frequency domain reconstruction have been proposed [21]. Temporal error concealment is another popular error concealment technique. It uses the correlation between a lost frame and its temporal neighbors. In [125], a method that estimates the motion vector by doing a full motion search in the neighborhood of a lost pixel-block is proposed. Another temporal approach called boundary matching algorithm (BMA) [126] selects the lost motion based on a boundary difference. Multi-frame motion vector averaging method (MMA) [127], motion vector extrapolation (MVE) [128] and hybrid motion vector extrapolation (HMVE) [129] methods estimate the entire missing frame using the received frames. A redundant motion vector method for H.264/AVC is proposed in [130], in which motion vectors are sent as additional information. Many error resilience methods have been proposed to make compressed bitstream robust to losses [21]. One such scheme proposes the use of redundant pictures that are trans-

mitted using a reliable channel [131]. Scalable video coding (SVC) [132] and multiple description coding (MDC) [133] are two popular error resilient coding techniques.

Several error resilience tools are proposed for the VPx and H.26x bitstreams. Specifically, H.264 offers data partitioning (DP), flexible macroblock ordering (FMO) and switching P (SP) and switching I (SI) slices [134]. In the current VP9 [1] implementation, a flag "error_resilient_mode" is used to turn on the error resilience mode at encoder. This mode restricts entropy decoding dependency across frames. The entropy coding context probabilities are reset at the beginning of each frame. Also, the colocated MV from previously encoded reference frame cannot be included in the list of MV reference candidates. However, without other scheme to prevent drift between the encoder and decoder, there can be a significant drop in image quality of the decoded video. A simple method that uses duplicated prediction information can be effective to provide a graceful degradation of performance with packet loss [135]. However, this method is less effective in the case of keyframe loss as can be seen in the failure cases. In this thesis, we use a redundant bitstream that is composed of only keyframes and duplicated prediction information for non-keyframes. This is sent to the receiver in the form of an "error resilience packet" at each time interval.

2.2 Proposed System Architecture

Encoder: As shown in Figure 2.1, the original video sequence shown in white is encoded using a standard VP9 encoder with encoded keyframes shown in green and interframes shown in blue. Our proposed system is designed to form an "error resilience packet" shown in yellow for a given interval of time, i.e. every N frames.

The error resilience packet consists of two parts. The first part is the duplicated prediction (mode/motion) information of all interframes. It includes the frame headers and per-macroblock prediction information for each interframe in this interval as used in [135].

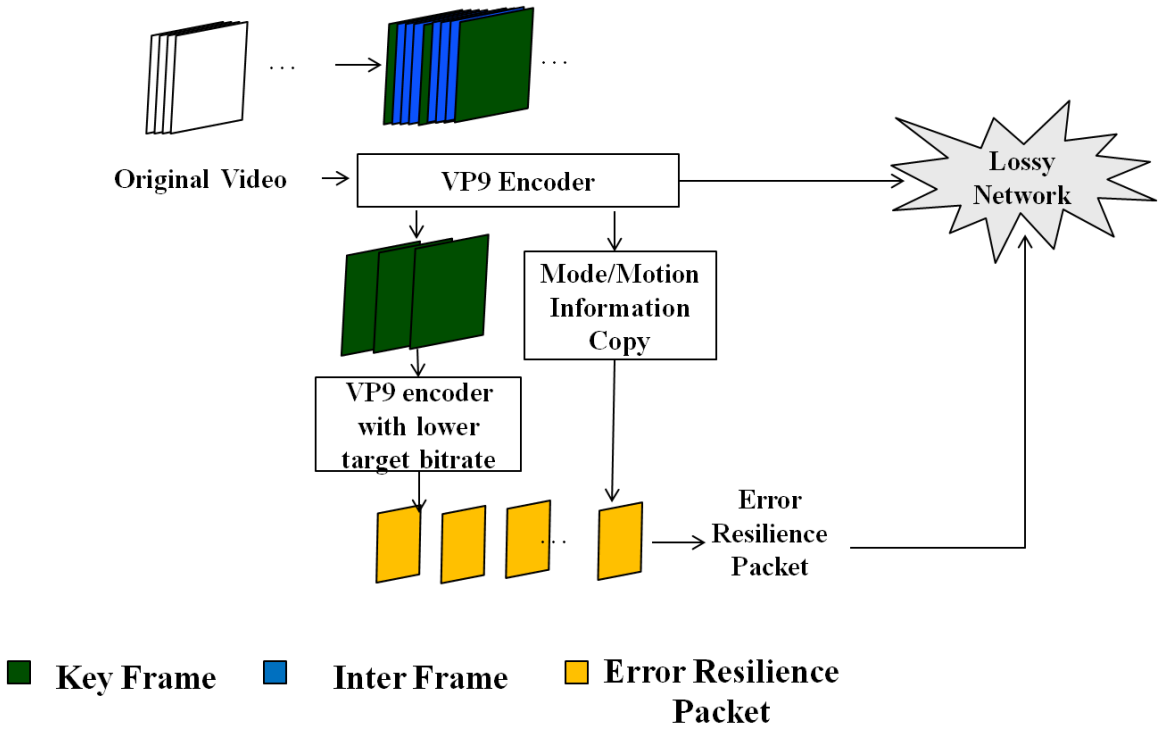


Fig. 2.1.: Proposed encoder architecture

Secondly, we extract all the keyframes from the original video sequence and concatenate them to make what we call *keyframe-sequence*. Then we encode this *keyframe-sequence* with VP9 at a lower bitrate than that of the original encoder. A suitable value of "target-bitrate" is used while encoding to adjust the amount of overhead due to redundant keyframe information. Consider N frames of the original video sequence encoded at f frames per second using $(\mathrm{BR})_{\mathrm{Seq}}$ target-bitrate. Let K be the number of frames that are encoded as keyframes out of N frames. Our *keyframe-sequence* consists of the above K original frames. Let k be the average frame rate of the *keyframe-sequence*: $k = K/(\frac{N}{f})$ frames per second. Now, we encode the *keyframe-sequence* at k frames per second using $(\mathrm{BR})_{\mathrm{kf}}$, the target-bitrate of the *keyframe-sequence*:

$$(\mathrm{BR})_{\mathrm{kf}} = \frac{1}{d}\left(k \cdot \frac{(\mathrm{BR})_{\mathrm{Seq}}}{f}\right)$$

where d (> 1) is a positive constant.

We concatenate the two above described data and a list of keyframe indexes to form an error resilience packet for every N frames of the original sequence. The error resilience packets are sent over the network and assumed to be protected from packet losses. They are sent either embedded in the bitstream or over a separate channel as side information. In this chapter, we do not discuss any details of the syntax for the error resilient bitstream generated using our proposed method. But we allocate a certain number of bytes to accommodate any additional syntax elements needed to specify the presence of the error resilience packets.

Decoder: Our error resilient decoder is shown in Figure 2.2. The error resilience packet corresponding to each N frames is first decoded to obtain redundant keyframes and per-frame prediction information. The main part of the bitstream is then decoded such that each frame is reconstructed using the received packet data. When packet loss occurs, the decoder examines the frame headers and the list of keyframe indexes obtained from the error resilience packet to get the frame type. If the lost frame is a keyframe, shown in red, the decoder uses the lower-bitrate keyframe as the concealment signal. If it is an interframe, shown in pink, the decoder uses the prediction information from the error resilience packet to reconstruct the frame. The residue signal corresponding to a lost interframe cannot be recovered since it is not sent as a part of the error resilience packet. The concealed frames are also used to update the VPx reference frame pool in the same way as it is done for the received frames.

2.3 Experimental Results And Analysis

Our proposed method is implemented by modifying the *libvpx* software available on the WebM website [136]. We used the VP9 encoding options, mainly "codec", "good", "error-resilient", "cpu-used", "target-bitrate", "kf-max-dist" to obtain different encoded bitstreams [137]. The following are our encoding configuration settings:

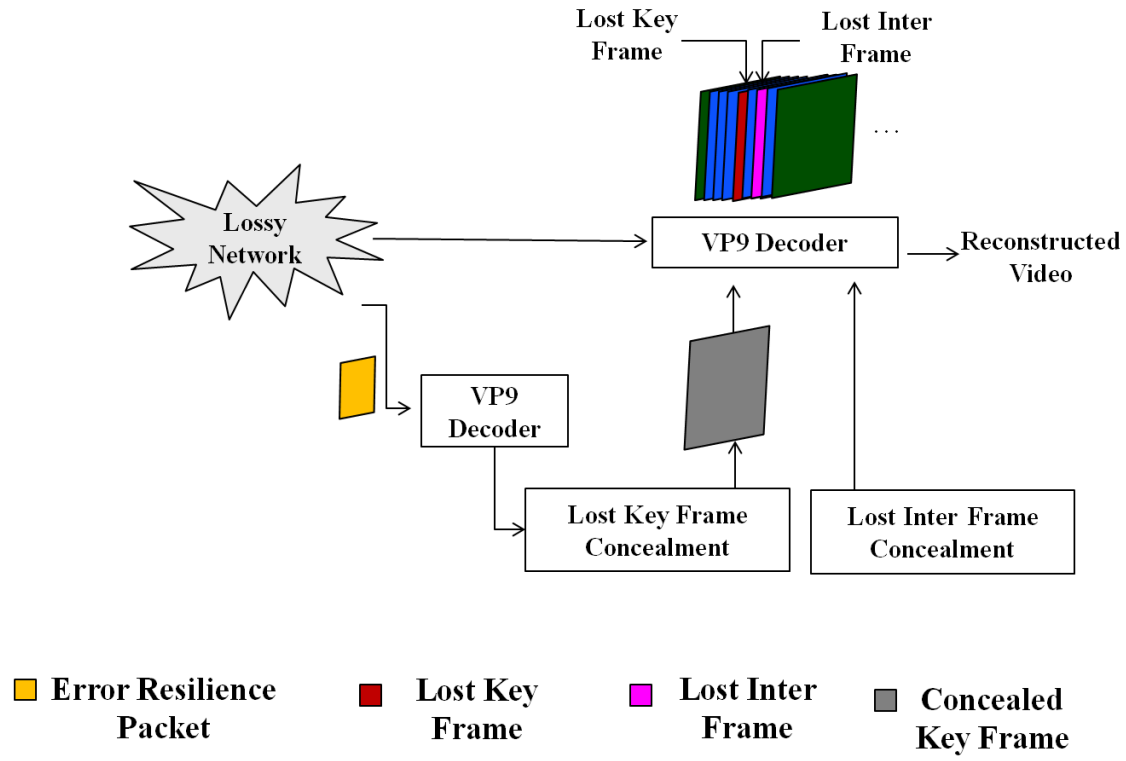


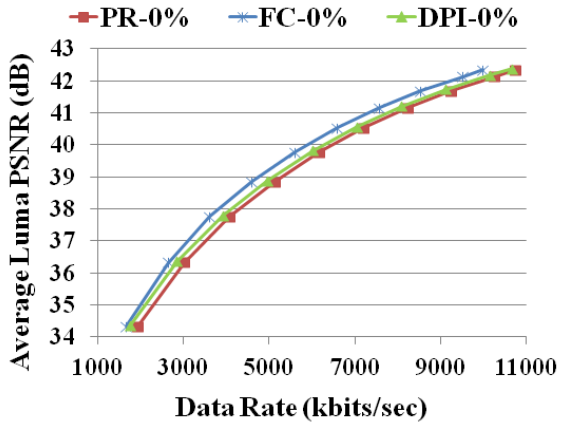
Fig. 2.2.: Proposed decoder architecture

```
./vpxenc -w <Width > -h <Height > --i420  --verbose --psnr -o <out.webm>
 --codec=vp9 --good --cpu-used=0 --end-usage=cbr --fps= < f >/1 --passes=1
--target-bitrate=<1500-15000> --kf-min-dist=0 --kf-max-dist=<kDist >
--error-resilient=1 <in.yuv>
```

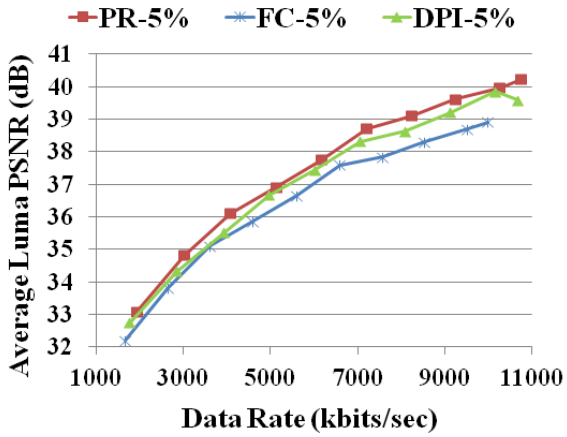
As shown in Table 2.1, three video sequences *BasketballDrill*, *PartyScene* and *KristenAndSara* are used for our experiments.

Table 2.1.: Test sequences used for our experiments

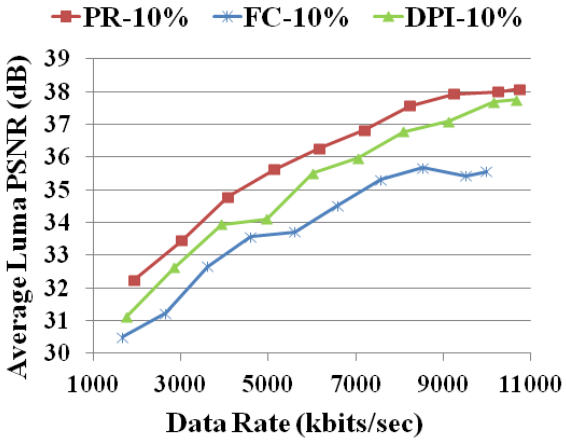
Sequence	Spatial Resolution ($Width \times Height$)	Frame Rate (f)	No. of frames	Maximum key-frame distance ($kDist$)
BasketBallDrill	832×480	50	500	25
PartyScene	832×480	50	500	25
KristenAndSara	1280×720	60	600	30



(a) Lossless



(b) Packet Loss=5%



(c) Packet Loss=10%

Fig. 2.3.: Performance comparison for *BasketballDrill*

We assume one packet contains one encoded frame. We use a Gilbert-Elliott model that is used in [138], to simulate packet loss pattern for the lossy network. When the packet loss rate is small, burst length is large, and vice versa [27]. Table 2.2 shows a list of packet loss rates and their respective burst lengths used in our experiments.

Table 2.2.: Lost packet rate and burst length

Packet Loss Rate	2%	4%	6%	8%	10%	12%
Burst Length	6	6	5	5	4	4

For all sequences, we set $d = 5$ to control the target bitrate of the *keyframe-sequence*. For our experiments, we send an error resilience packet for the entire length of the short sequences we used for testing. However, N can be adjusted according to the requirement of an application, since it controls the latency of the communication. The error resilience packet and the first frame of each sequence are assumed to be error-free. In addition to the data in the error resilience packet, we reserve 100 bytes to accommodate the syntax elements. Then the total bitrate is the addition of the original bitrate, the extra bitrate from the error resilience packet and the 100 bytes reserved for syntax elements.

We compare the results of our proposed method (PR) with the frame copy (FC) method from [139] in which the lost frame is concealed using the previously received frame. For each packet loss rate, the experiment is repeated 50 times with different packet loss patterns for our proposed method and the FC method. Each data point reported in the figures is the average PSNR of these 50 packet loss experiments.

Figure 4.14 presents the comparison of PR and FC methods for *BasketballDrill*, *PartyScene* and *KristenAndSara* sequences for the lossless (0%), 2%, 4%, 6%, 8%, 10% and 12% packet loss cases. For all sequences, FC method outperforms PR when there is no packet loss. This is because we send additional data using the error resilience packet that is redundant when there is no packet loss. For *BasketballDrill* and *PartyScene* sequences, the PSNR performances of PR and FC for the 2% loss case are close to each other. When the packet loss increases to 4%, our proposed method

begins to outperform FC. For *KristenAndSara* sequence, our proposed method has better PSNR performance from 2% packet loss rate for all our tested data rates. When packet loss rate is above 4%, PR clearly outperforms FC for all test sequences in terms of average luma PSNR. In addition, as packet loss increases, our proposed method shows a graceful degradation in performance, while the performance of the FC method decreases more rapidly, especially for higher data rates. This is because the burst packet loss (simulated using the Gilbert-Elliott model) often causes multiple successive frames to be lost at the receiver. In that case all the lost frames are replaced by the one most recently received frame. This causes a significant drift at the decoder and a visual appearance of "frame-freeze." The effects of the drift continue until a keyframe is received. Whereas, our proposed method conceals a lost interframe using the protected prediction information to reduce the drift. If a keyframe is lost, our method recovers a low-quality version of the keyframe using the error resilience packet to prevent the drift propagation.

We also compare the results of our proposed method and the FC method with that of the duplicated prediction information (DPI) method used in [135]. Figure 2.3, Figure 2.4 and Figure 2.5 present the performance of our proposed method (PR) as compared with the frame copy method (FC) and the duplication of prediction information (DPI) method used in [135] for "Lossless" (0%), 5% and 10% packet loss cases.

For all test sequences, FC performs better than PR and DPI in the lossless case as expected. This is because the other two methods contain various levels of redundant information in their bitstreams, in the form of error resilience packets. In lossless case, DPI performs better than PR because the size of error resilience packet in our proposed method is larger than that used in DPI, as a result of redundant keyframe information. For packet loss at "5%", our proposed method outperforms the other two (DPI and FC) for almost all data rates. For 10% packet loss, our proposed method shows a greater advantage over DPI and FC in terms of PSNR. This shows

that the error concealment performance of our proposed method is improved under higher packet loss conditions.

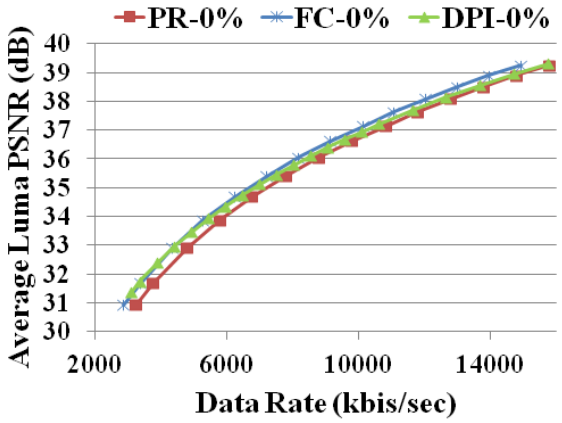
Figure 2.7 and Figure 2.8 show examples of visual comparison of the error concealment done using our proposed method and that done with FC. Each Figure contains (a) the original frame, (b) decoded frame without any packet loss, (c) frame concealed using our proposed method and (d) frame concealed using the FC method.

As seen in the *PartyScene* example shown in Figure 2.7, our proposed method has produced a better concealment than the FC method. Some parts of the frame concealed using our proposed method are blurred (e.g. the letter "P"). This is because the lost keyframe that is used to form the prediction signal for this frame is concealed using the *keyframe-sequence* encoded at a lower bitrate. There are also blocky artifacts in the moving area (e.g. the girl's hair). This is because our error resilience packet contains a copy of only the prediction information of interframes and it does not contain any information about the pixel-wise residues. The frame concealed using the FC method has relatively larger distortions in the moving areas (e.g. the girl's face). This is because the lost keyframe was concealed using a previously received frame which was inaccurately used as prediction signal for the current frame.

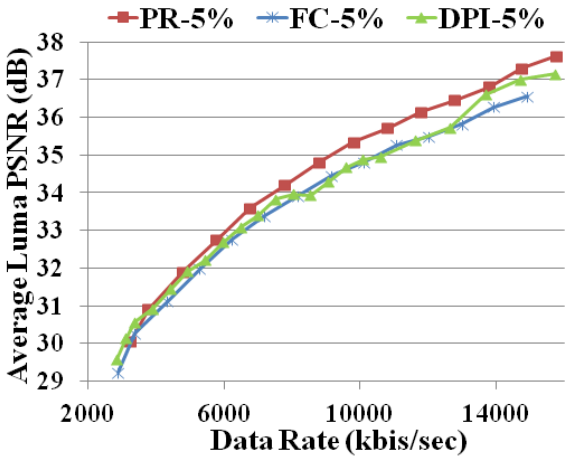
For *BasketballDrill* example shown in Figure 2.8, the frame concealed using our method contains blurriness and blockiness in the moving area (e.g. the basketball and the player's hands). Whereas, the frame concealed using the FC method contains seemingly fewer of these artifacts. However, this frame is from a considerably earlier time and the video reconstructed using this frame has a noticeable "frame-freeze" artifact, which seems visually worse than small blocking artifacts localized in a frame.

In another example (not shown in the chapter), if the lost keyframe is also a frame with the scene-change, the image quality produced by the FC method is unacceptably low for the concealed keyframe and many subsequent frames until another keyframe is received. The DPI method conceals lost frames that are subsequent to a lost keyframe with the scene-change using the concealed keyframe. This leads to the failure cases reported in [135]. This observation is also supported in terms of poor

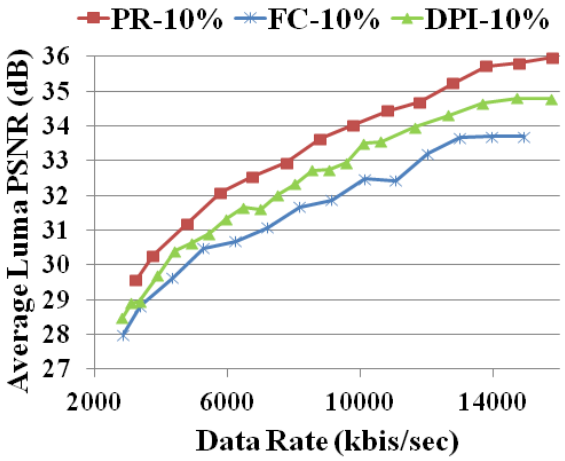
PSNR performance of individual frames concealed after the scene-change. Therefore, our proposed method performs better than other two methods in terms of both the PSNR and the visual video quality.



(a) Lossless

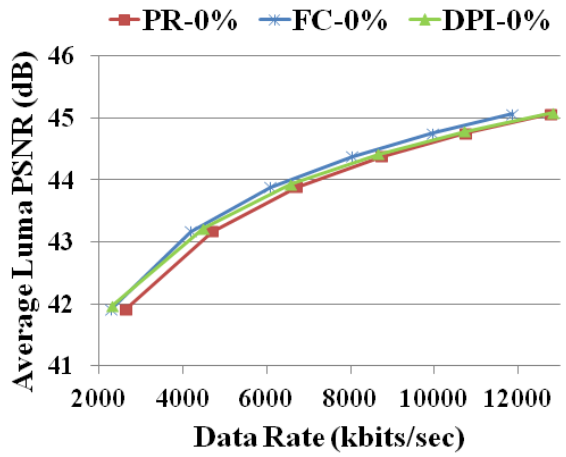


(b) Packet Loss=5%

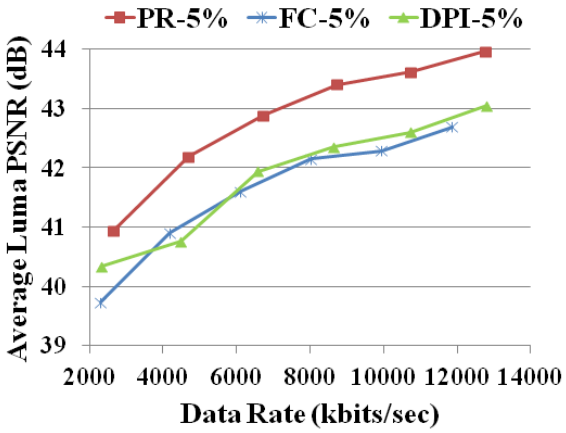


(c) Packet Loss=10%

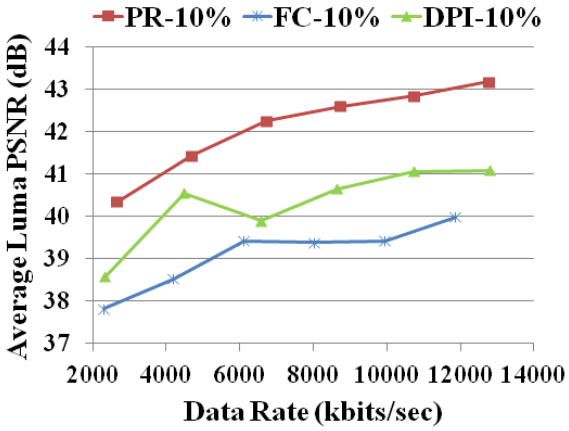
Fig. 2.4.: Performance comparison for *PartyScene*



(a) Lossless

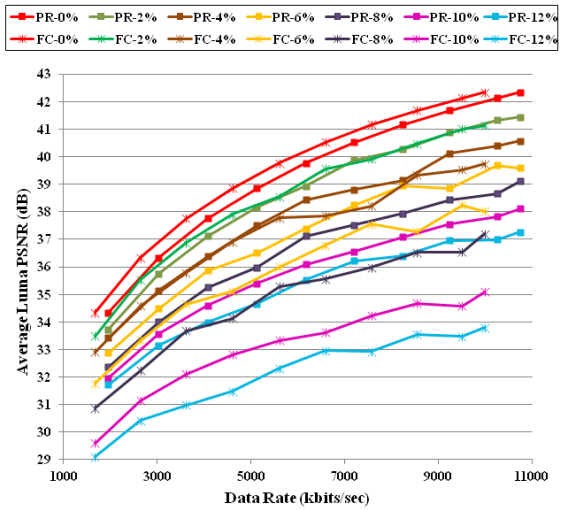


(b) Packet Loss=5%

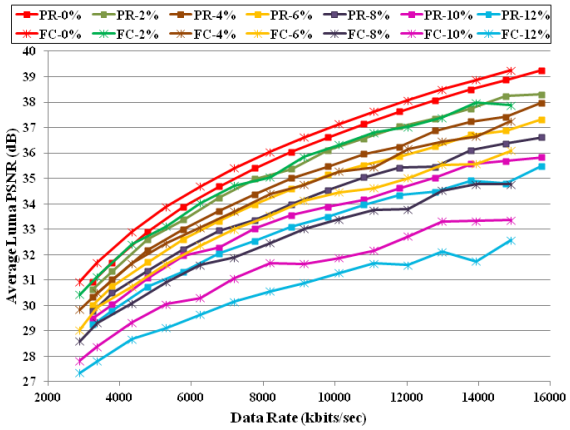


(c) Packet Loss=10%

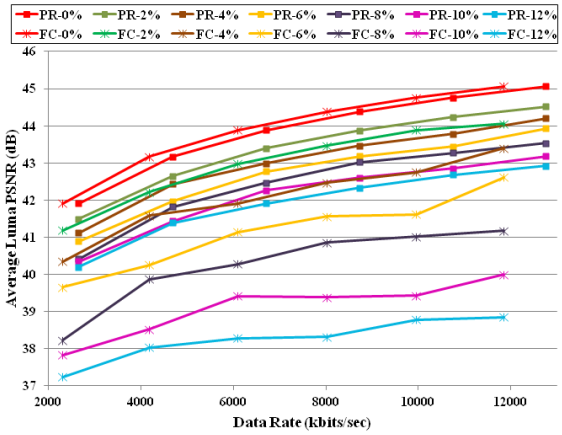
Fig. 2.5.: Performance comparison for *KristenAndSara*



(a) *BasketballDrill*

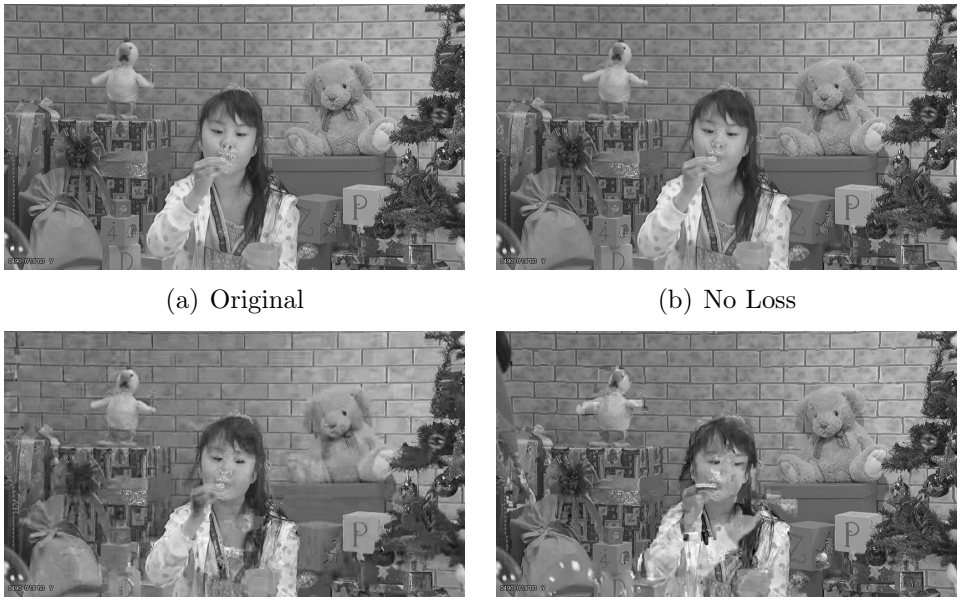


(b) *PartyScene*



(c) *KristenAndSara*

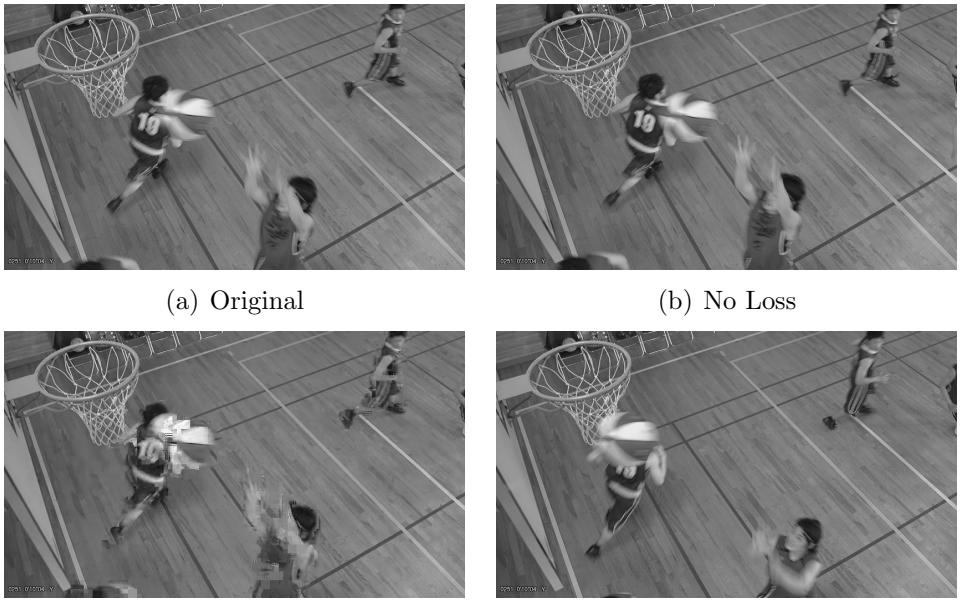
Fig. 2.6.: Packet loss performance



(a) Original (b) No Loss

(c) Loss Concealed using PR (d) Loss Concealed using FC

Fig. 2.7.: Visual comparison for *PartyScene* (Luma)



(a) Original (b) No Loss

(c) Loss Concealed using PR (d) Loss Concealed using FC

Fig. 2.8.: Visual comparison for *BasketballDrill* (Luma)

3. MULTI-REFERENCE VIDEO CODING USING STILLNESS DETECTION

3.1 Introduction

The AV1 codec is an open source, royalty-free video codec developed by a consortium of major technology companies called Alliance for Open Media which is jointly founded by Google. It followed the VP9 codec, a video codec designed specifically for media on the web by Google WebM Project. It is expected to achieve generational improvement in coding efficiency over VP9. The AV1 codec introduced several new features and coding tools, one of which being the multilayer coding structure [140].

The current AV1 codec divides the source video frames into Golden-Frame (GF) groups. The length of each GF group, i.e. the GF group interval, may vary according to the video's spatial or temporal characteristics and other encoder configurations, such as the key frame interval at request for the sake of random access or error resilience. The coding structure of each GF group is based on their interval length and the selection of reference frames buffered for the coding of other frames. The coding structure determines the encoding order of each individual frame within one GF group.

In the current implementation of the AV1 encoder, a GF group may have a length between 4 to 16 frames. Various GF coding structures may be designed depending on the encoder's decision on the construction of the reference frame buffer, as shown in Figure 3.1(a) and Figure 3.1(b). The `extra-ALTREF_FRAME`s and the `BWDREF_FRAME`s introduce hierarchical coding structure to the GF groups [140]. The VP9 codec uses three references for motion compensation, namely `LAST_FRAME`, `GOLDEN_FRAME` and `ALTREF_FRAME`. `GOLDEN_FRAME` is the intra prediction frame. `LAST_FRAME` is the forward reference frame. `ALTREF_FRAME` is the backward reference frame selected

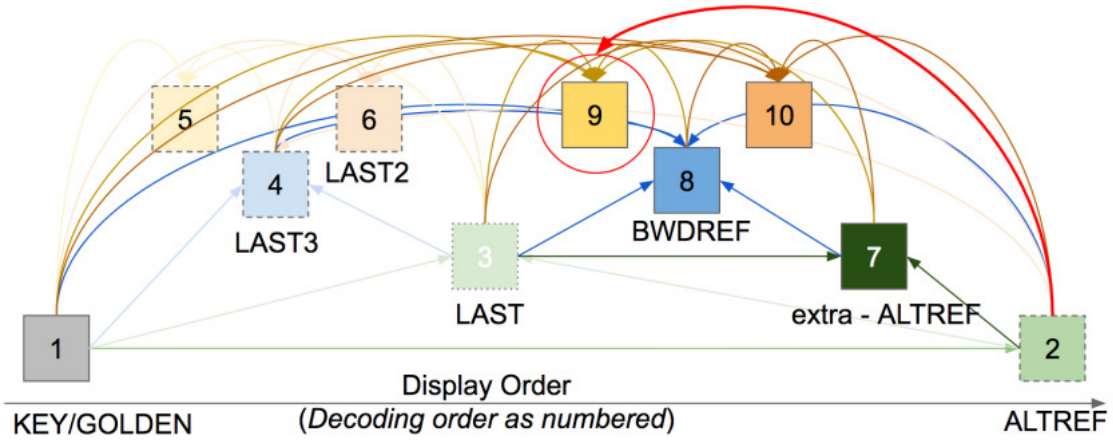
from a distant future frame. It is the last frame of each GF group. A new coding tool is adopted by AV1 that extends the number of reference frames by adding LAST2_FRAME, LAST3_FRAME, extra-ALTREF_FRAME and BWDREF_FRAME. LAST2_FRAME and LAST3_FRAME are similar to LAST_FRAME. extra-ALTREF_FRAME and BWDREF_FRAME are backward reference frames in a relatively shorter distance. The main difference is that BWDREF_FRAME does not apply temporal filtering. The hierarchical coding structure in Figure 3.1(a) may greatly improve the coding efficiency due to its multi-layer, multi-backward reference design.

The current AV1 encoder uses the coding structure shown in Figure 3.1(a) for all the GF groups. However, a comparison of the compression performance with extra-ALTREF_FRAME and BWDREF_FRAME enabled and disabled showed that the coding efficiency for some test videos was actually worse when these two reference frames were enabled. This means that the multilayer coding structure does not always have better coding efficiency for all the GF groups. One such example is the GF groups with stillness feature. In this section, we propose a new approach that adaptively designs the Golden-Frame group coding structure through the use of stillness detection. A set of metrics are designed to determine whether the frames in a GF group is of little motion. Little work has been done that investigates the use of different coding structures depending on video content. In [141], an adaptive video coding control scheme is proposed that suggests using more P- and B-frame while the temporal correlation among the frames in a group of pictures (GOP) is high. A method for using different GOP sizes based on video content is presented in [142].

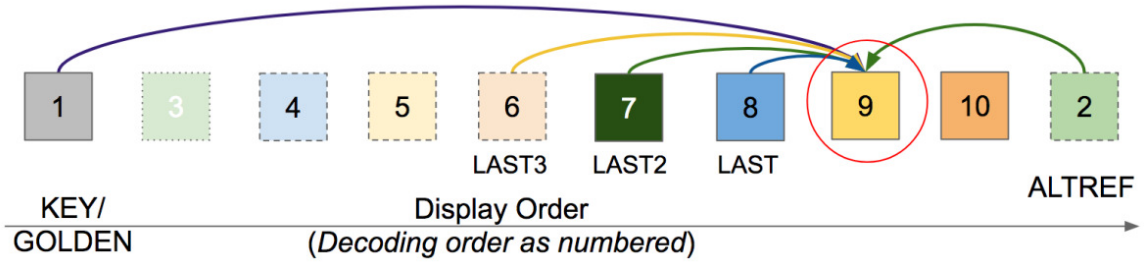
3.2 Method

3.2.1 GF group stillness

A GF group may be constructed to contain consistent characteristics to differentiate itself from other GF groups. For instance, some GF group may present stillness across its successive frames, and others may present a zoom-in / zoom-out motion



(a) GF Group Coding Structure Using Multilayer



(b) GF Group Coding Structure Using One-Layer

Fig. 3.1.: GF group coding structures

across the entire GF group. We examined the coding efficiency and the stillness feature of each GF group and found that when stillness is present in one GF group, the use of multilayer coding structure as shown in Figure 3.1(a) may produce worse coding performance, as opposed to that generated by the one-layer structure in Figure 3.1(b).

3.2.2 Automatic GF group stillness detection

An automatic stillness detection of the GF groups is proposed in this section which allows the GF groups to choose adaptively between two coding structures as shown in Figure 3.1(a) and Figure 3.1(b). Three metrics are extracted from the GF group

during the first coding pass of AV1 to determine the GF group stillness. The first coding pass of AV1 conducts a fast block matching with integer-pixel accuracy and uses only one reference frame, the previous frame. Some motion vector and motion compensation information are collected during the first coding pass. Our proposed stillness detection method uses this information to extract three metrics as described below which requires small amount of computation. It then identifies the thresholds and derives the criteria to classify GF groups into two categories: GF groups of stillness and GF groups of non-stillness. The thresholds are obtained by collecting statistics of the three metrics from GF groups of eight low resolution (*cif*) test videos. We manually labeled the stillness or non-stillness of the GF groups. Figure 3.3 shows the histograms and the thresholds of the three metrics. We intentionally included some test videos that contain GF groups of "stillness-like" characteristics in the non-stillness class because they are more likely to be misclassified as GF group of stillness. The GF group with "stillness-like" characteristics shows either very slow motion or static background with small moving objects. We obtained three criteria which are jointly applied to automatically detect stillness. Finally, the GF group is coded using the workflow given in Figure 3.2.

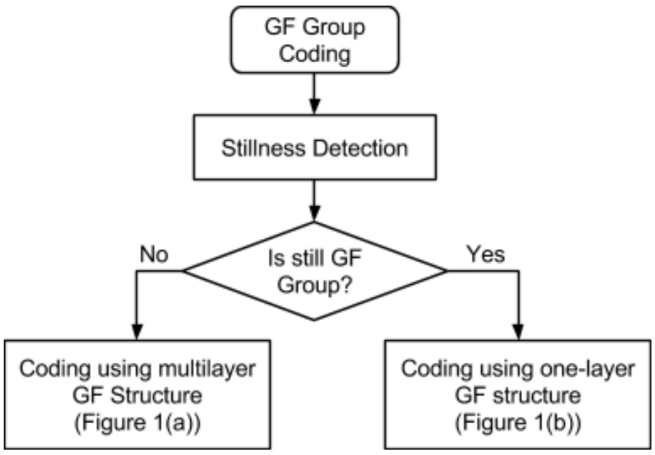
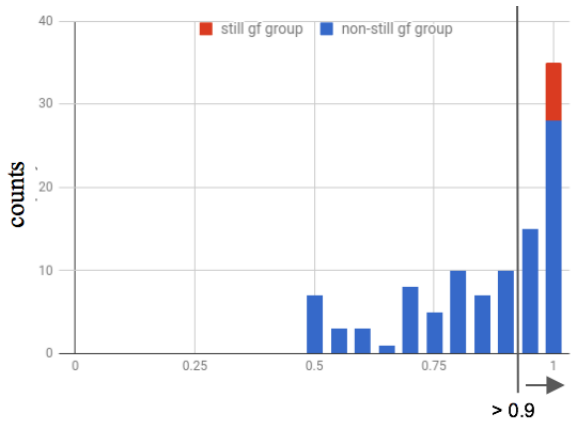
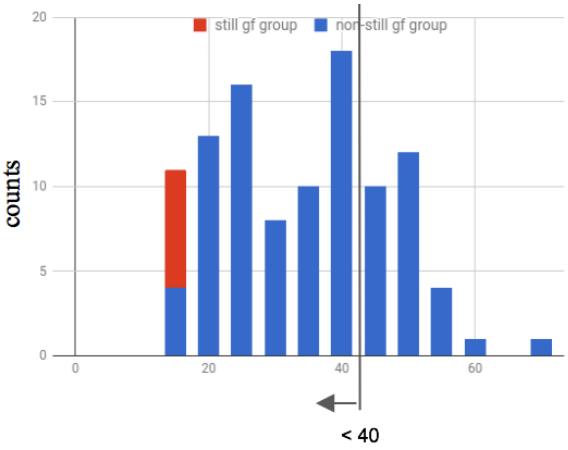


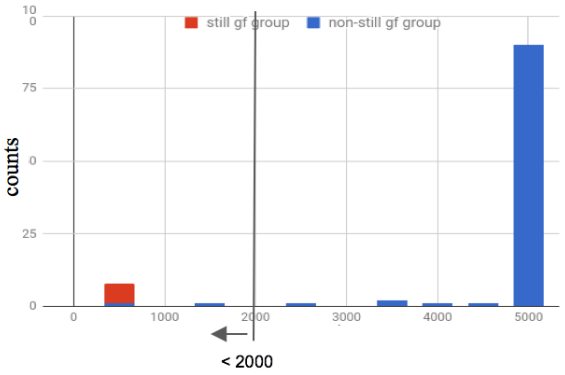
Fig. 3.2.: GF group coding with stillness detection



(a) *zero_motion_accumulator*



(b) *avg_pixel_error*



(c) *avg_error_stdev*

Fig. 3.3.: Thresholds for metrics

Stillness Detection Metrics:

1. **zero_motion_acumulator**: Minimum of the per-frame percentage of zero-motion inter blocks within one GF group:

$$zero\_motion\_accumulator = MIN(pcnt\_zero\_motion_{Fi} \mid Fi \in S) \tag{3.1}$$

where
$S = \{Fi | i = 1, 2, ..., gf\_group\_interval\}$, the set of frames in the GF group
$gf\_group\_interval$: number of frames in the GF group
$pcnt\_zero\_motion$: percentage of the zero-motion inter blocks out of all the inter blocks

2. **avg_pixel_error**: Average of per-pixel sum of squared errors (SSE) within one GF group:

$$\begin{aligned} &avg\_pixel\_error = \\ &MEAN(frame\_sse_{Fi}/number\ of\ pixels\ per\ frame \mid Fi \in S) \end{aligned} \tag{3.2}$$

where
$frame\_sse_{Fi}$ is the SSE of frame Fi

3. **avg_error_stdev**: First calculate the standard deviation of the block-wise SSEs for each frame, where block SSEs are obtained from zero-motion prediction; then obtain the mean value of the standard deviations of all the frames in one GF group:

$$avg\_error\_stdev = MEAN(STDEV_{Fi}(block\_sse_{(0,0)}) \mid Fi \in S) \tag{3.3}$$

where
$block\_sse_{(0,0)}$ is the block-wise SSEs obtained from zero-motion prediction

$STDEV_{Fi}$ is the standard deviation of the block-wise SSEs of frame Fi

We use the above three metrics to differentiate those GF groups of stillness features from other GF groups, subject to the criteria in Table 3.1.

Table 3.1.: Criteria for GF group stillness detection

Stillness Detection Metrics	Stillness Detection Criteria (Identified as GF group of stillness)
$zero\_motion\_accumulator$	>0.9
avg_pixel_error	<40
avg_error_stdev	<2000

3.2.3 Adaptive GF group structure design

Once a GF group is categorized as a GF group of stillness, no `extra-ALTREF_FRAME` or `BWDREF_FRAME` is used in the single layer coding structure as shown in Figure 3.1(b). The single layer coding structure still has multiple reference frames employed for the coding of one video frame. `LAST_FRAME`, `LAST2_FRAME`, `LAST3_FRAME` and `GOLDEN_FRAME` are used as forward prediction reference and `ALTREF_FRAME` is used as backward prediction reference. If a GF group is categorized as non-still GF group, we will further leverage the use of `BWDREF_FRAME` and `extra-ALTREF_FRAME` to help improve the coding performance.

3.3 Experimental Results

We tested the proposed method using two standard video test sets with various resolutions and spatial/temporal characteristics, as shown in Table 3.2. More specifically, the set of *lowres* includes 40 videos of *cif* resolution, and the set of *midres* includes 30 videos of 480p and 360p resolution. Each video is coded with a single

GOLDEN_FRAME and a set of target bitrates. For quality metrics, we use the arithmetic average of the frame PSNR and Structural Similarity Index (SSIM) [143]. To compare RD curves obtained by the base AV1 codec and our proposed method, we use the BD-RATE metric [14]. Experimental results demonstrated the advantage of the proposed approach. The Google test set of *lowres* has two video clips that contain detected still GF groups (*pamplet_cif* and *bowing_cif*) and test set *midres* has one (*snow_mnt*). As shown in Table 3.3, by applying the proposed approach, the BD-RATE of video clips that contain GF groups of stillness has decreased by approximately 1%. The classification results of the proposed automatic stillness detector contain no misclassification case in the videos from these two test video sets. There are mainly two reasons that the single layer coding structure has better coding efficiency on the GF groups with stillness feature. One is that the multilayer coding structure in Figure 3.1(a) involves more candidate reference frames thus requires more motion information to be transmitted to the decoder. The other reason is that the multilayer coding structure uses an unbalanced bit allocation scheme which is not preferable for GF group of stillness in which the frames are very similar.

Table 3.2.: BD-RATE reduction using proposed method on Google test set

test set	BD-RATE(PSNR)	BD-RATE(SSIM)
test set of *lowres*	-0.063	-0.045
test set of *midres*	-0.026	-0.041

Table 3.3.: BD-RATE reduction using proposed method on video clips contain GF group of stillness

video clip	BD-RATE(PSNR)	BD-RATE(SSIM)
pamplet_cif	-1.395	-1.076
bowing_cif	-1.118	-0.735
snow_mnt	-0.767	-1.235

4. ADVANCES IN REGION-BASED VIDEO CODING USING DEEP NEURAL NETWORK

4.1 Introduction

The growing demand for high quality online video content has resulted in novel video coding schemes developed to improve the coding efficiency of video compression methods. One category of these techniques is region-based video coding approaches. As opposed to conventional block-based video coding approaches, where each frame is partitioned into block structures and separately encoded, transmitted and decoded, region-based video coding approaches segment a frame into coherent regions based on features such as motion and texture, then each region is coded using different approaches or encoding parameters. Since most modern video codecs, such as H.264, HEVC and AV1, use block-based hybrid coding approach, region-based approaches need to be further integrated into the block-based coding structure [144]. Block-based hybrid coding techniques consist of 2D transforms and motion compensation techniques to remove spatial and temporal redundancy by efficient exploitation of statistical dependencies measured by MSE. However, it does not always produce the best psychovisual result. Region-based methods aim to remove the psychovisual irrelevant information and enhance the visual quality by improving the fidelity of human interested regions which leads to data rate reduction without noticeable degradation of visual quality.

A popular region-based video coding approach is the analysis/synthesis approach. It is assumed that many video scenes can be classified into textures that are either "perceptually significant" (in an MSE sense) or "perceptually insignificant". By "perceptually insignificant" pixels, we mean regions in a frame that an observer will not notice any difference without observing the original video sequence. The viewer per-

ceives the semantic meaning of the displayed texture rather than the specific details therein [145]. The "perceptually insignificant" pixel blocks usually are "noise-like" textures and are generally costly to encode using hybrid coding methods. In this paper, we define texture as "perceptually insignificant" pixels in a frame with respect to the Human Visual System, while non-texture area refers to all other pixels in a frame. The encoder only encodes areas of a video frame that are "perceptually significant" using hybrid coding techniques. The "perceptually insignificant" pixels are treated differently by fitting a statistical model. The model parameters are then transmitted to the decoder as side information. The decoder uses the model parameters to reconstruct the pixels in the frame as an approximation. Since the residual signal is not transmitted and the parameters of the statistical model generally can be represented by far fewer bits, data rate can be significantly reduced. Over the past two decades, many promising region-based video coding methods have been developed which are discussed in [39].

In recent years, adapting deep neural networks into the field of video coding has become an active research area [50, 51]. Module based methods use a hybrid video coding framework that incorporates deep neural networks with traditional video coding schemes. The use of deep neural networks to improve coding efficiency has been explored in almost all coding modules, including intra prediction, inter prediction, quantization and entropy coding, loop filtering and post-processing. In addition, attempts have been made to build end-to-end video coding schemes that primarily use deep neural networks which show a new direction of video compression.

Previous work on "analysis/synthesis" coding approaches use hand-crafted feature based texture analyzers which require a set of manually chosen parameters to achieve accurate texture segmentation for different videos. In [146], both static and dynamic texture models are investigated for image and video coding, respectively. A texture analysis-synthesis loop is developed to identify "rigid" or "non-rigid" texture regions which are then synthesized at the decoder using corresponding side information. This approach only applies to Gaussian texture. In [45], a perspective motion

model is employed to warp static textures. Texture regions are segmented using features derived from wavelet transform and further classified based on their spatial and temporal features. It relies on the proper choices of a set of parameters to achieve accurate texture segmentation. The texture synthesizer in [147] does not use motion model but instead uses pixels from the above and left macroblocks to synthesize static texture. In practice, this requires the current block to find similar contexts from the above and left area which may not be true for all encoding blocks. While they have shown good performance on a selection of test videos, it is difficult to scale up to a large variety of videos with different texture-liked areas without manual tuning of the parameters. To overcome this challenge, deep learning based methods can learn these parameters during training, therefore do not require such parameter tuning for inference.

4.1.1 Perceptual quality metrics

Research has shown that MSE based criterion such as PSNR and SSIM typically used in hybrid coding system is not an adequate quality measure for videos reconstructed with texture synthesis approaches since they are not optimized for pixel-wise similarity measure with the original video frames [148]. Therefore, the incorporation of perceptual-based quality metric instead of MSE based metric is important and necessary to assess the coding performance of a texture analysis/synthesis based video coding system. We will review existing perceptual-based quality metrics and introduce a new metric in our own work.

To assess the performance of a video coding system, video quality metrics are computed to describe the visual quality of video sequences. The most direct way to measure video quality is through a subjective test. Subjective tests capture the viewer's perception of the visual quality of the video sequences under carefully designed testing process and a controlled environment. However, the process is very resource intensive and cannot be automated. Objective video quality metrics are

mathematical models to estimate a viewer's perception. They are efficient and convenient for in-loop optimization of video codecs. The result of subjective tests can be used as the ground-truth to evaluate the performance of objective quality metrics. The most commonly used metric, PSNR compares the pixel-wise difference between the decoded frame and the reference frame, thus may not correspond well with the human subjective evaluation. In recent years, a number of visual quality metrics have been developed to have a better correlation with human perception.

Inherent structural information is an important factor for human visual perception. The SSIM [149] and the multi-scale SSIM [150] estimate the structural similarity by computing the mean, variance and co-variance of the pixels. The complex wavelet transform variant of the SSIM, CW-SSIM [151] and WCW-SSIM [152], are invariant to image scaling, translation and rotation.

A perceptually motivated reduced-reference (RR) quality metric for synthesized textures is proposed in [153]. The perceptual regularity of the original and synthesized textures is quantified through the visual saliency of the textures. But it is not sensitive to artifacts from texture warping. In [148], a compatible artifact-based video metric (AVM) is proposed to capture the artifact caused by texture synthesis and was also employed in in-loop optimization. AVM consists of blurring estimation, similarity estimation and edge artifact detection. However, the similarity estimation component uses a simple model for human visual system and relies on the accurate choice of several parameters. The STSIM [154] and STSIM2 [155] remove the pixel-wise comparison component in SSIM to compare the texture similarity. However, they do not capture temporal artifacts in the reconstructed videos.

The Video Multimethod Assessment Fusion (VMAF) [156] fuses several image quality metrics using an SVM-based regression. However, these metrics are still largely based on pixel-by-pixel comparison to the original video, which are not suitable for video reconstructed using analysis/synthesis approaches. The experiments in [157] showed that VMAF is not suitable for assessing the quality of synthesized content,

indicating that significant research is still needed to develop perceptually accurate quality metrics that can assess synthesis artifacts.

In this chapter, we propose a block-level and a pixel-level texture segmentation methods to extract texture regions in a video frame using convolutional neural networks and only encode areas of a video frame that are "perceptually significant." To model the textures both spatially and temporally, we introduce a new coding mode called the "texture mode" that reconstructs the blocks by warping the texture block from the reference frame towards the current frame using global motion parameters. The proposed method is implemented using AV1 codec and is tested on both standard test sequences and selected sequences from the YouTube UGC dataset [158].

4.2 Block-Based Texture Coding

We present in this section a block-based texture segmentation method using Convolutional Neural Network (CNN) we have developed for the AV1 codec. A deep learning based texture segmentation method was developed to detect texture regions in a frame that is "perceptually insignificant." The proposed method is implemented in the AV1 codec by enabling the global motion mode to ensure temporal consistency.

As mentioned in Section 4.1, deep learning based texture analyzer has the advantage of not requiring manual tuning of parameters for inference since these parameters are learned during training as opposed to hand craft features. The problem with using the texture analyzer alone to encode the texture region in the video is that if each frame is encoded separately, areas that are reconstructed using the texture models will be obvious when the video is displayed. This then requires the textures to be modeled both spatially and temporally. We propose a scheme that reconstructs the texture region differently for the bidirectional predicted frames (B-frames) and forward predicted frames (P-frames) which largely reduces the temporal visual artifacts. We introduce a new coding mode for AV1 called the "texture mode". The "texture mode" is completely an encoder side option, which in essence skips the coding of

the block entirely through leveraging the use of global motions provided by the AV1 codec. Specifically, the texture mode uses a modified version of the global motion coding tool [11] in the AV1 codec to ensure the temporal consistency of the texture regions between frames. Experimental results validate the efficacy of the texture mode with a consistent coding gain compared to the AV1 baseline over a variety of video test sets given a fixed perceptual quality level [159].

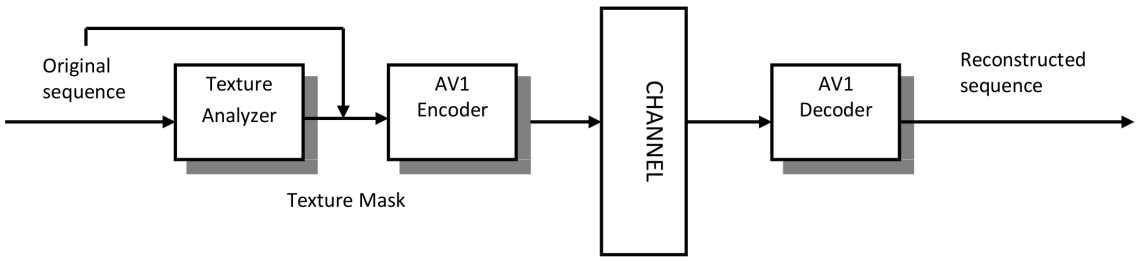


Fig. 4.1.: Block diagram of the proposed method

The general scheme for video coding using texture analysis and reconstruction is illustrated in Fig. 4.1. The texture analyzer identifies the texture regions in a frame. We use a classification convolutional neural network to label each block in a frame as textures or non-texture and generate a block-based texture mask for each frame. The texture mask and the original frame are fed into the AV1 video codec to enable the texture mode where the identified texture regions skip the encoding process. The texture region is reconstructed by warping texture region in a reference frame to the current frame. A modified version of the global motion tool in AV1 is used to obtain motion estimation and reconstructs the texture region without sending residues for the identified texture region.

4.2.1 Texture analysis using CNN[1]

To provide texture information for the AV1 encoder, we use a block-based deep learning texture detector to analyze video frames and produce segmentation masks. Our deep learning detector obtains texture segmentation masks by classifying each 32×32 block in a frame.

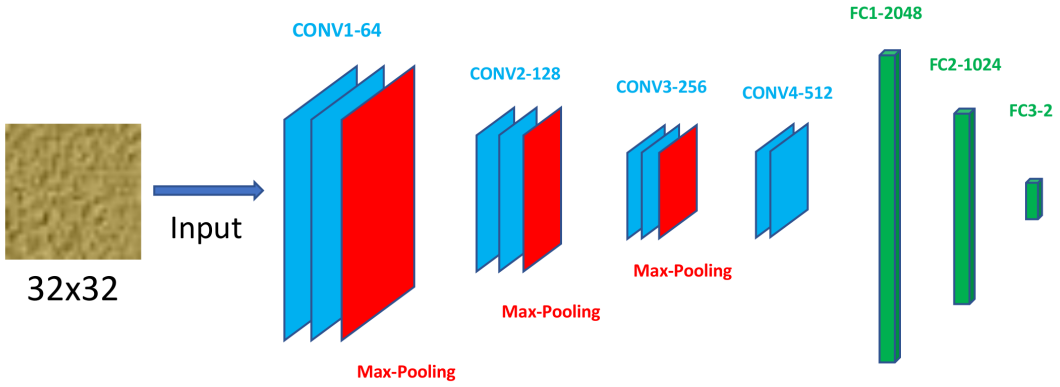


Fig. 4.2.: CNN architecture for block-based texture classification

The CNN architecture of our method is shown in Figure 4.2. Our CNN network is inspired by the VGG network architecture [160]. The input of our architecture is a 32×32 color image block. The architecture consists of convolutional layers followed by a batch normalization ReLU and a max pooling operation. Three fully connected layers with dropout operations and a softmax layer produces class probabilities. The output of our network is the probability that a 32×32 block is labeled as texture or non-texture, which indicates reliably of the texture/non-texture block label produced by the network. The kernel size of the convolutional layer is 3×3 and is padded by 1. The max pooling layer down samples the image by 2 and doubles the number of feature maps.

[1]Joint work with Chichen Fu

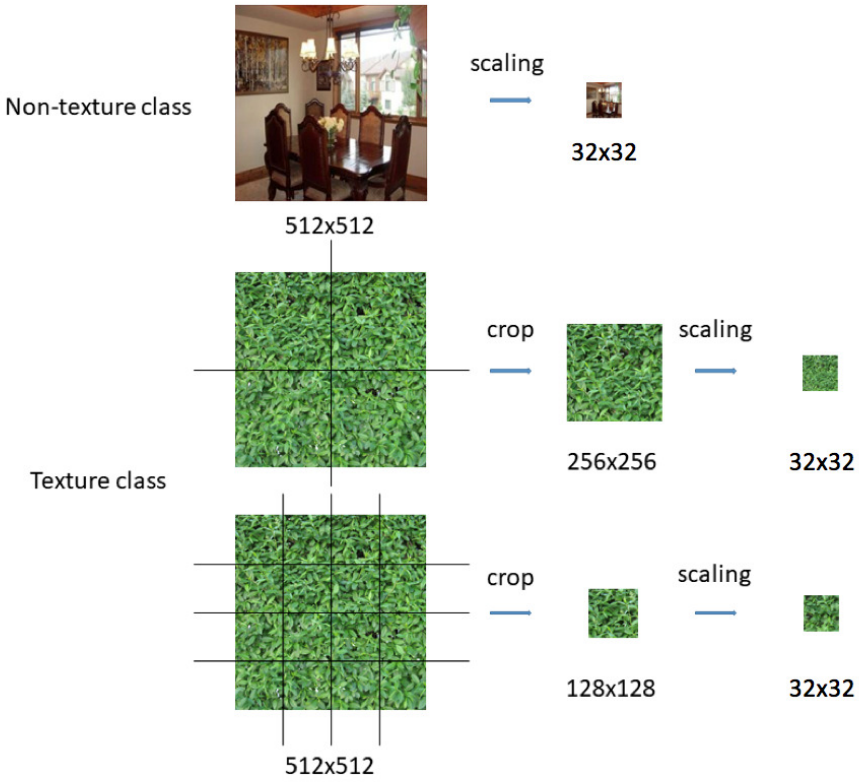


Fig. 4.3.: Training data preparation

Training

The Salzburg Texture Image Database (STex) and the Places365 [161] were used for training the CNN. Images in the STex dataset are "pure" texture images. Images in the Place365 dataset are of general scenes. The texture class samples were created by cropping 512 × 512 STex images into 256 × 256 and 128 × 128 image patches and resizing them to 32×32 patches. ThePlace365 images were resized into 32×32 image patches. Since the texture content in the Places365 images were lost during resizing operation, we can use resized Places365 images as non-texture class samples. In total, 1740 texture class images and 36148 non-texture class images were generated. The CNN training data preparation procedure is shown in Figure 2.

The proposed CNN architecture was implemented in Torch [162]. Mini-batch gradient descent is used with a fixed learning rate of 0.01, a momentum of 0.9 and a weight decay of 0.0005. The batch size of 512 image patches was trained in each iteration. For each epoch, 74 iterations were performed to cover the entire training set. The training set was shuffled before each epoch. Since our training set is unbalanced for the texture class images and non-texture class images, the class weight of each class was set proportion to the inverse of class frequency. The error of training was calculated using cross entropy criterion and was converted to probability score using softmax regression. A total of 100 epochs were trained using one NVIDIA GTX Titan GPU.

Inference

After training the CNN, texture segmentation is performed using test video frames. Each frame is divided into 32×32 adjacent non-overlapping blocks. Each block in the video frames is classified as either texture or non-texture block. The segmentation mask for each frame is formed by grouping the classified blocks in the frame.

Texture segmentation mask refinement

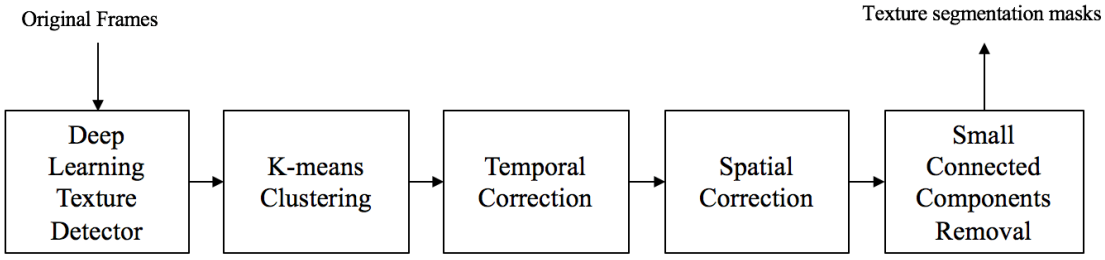


Fig. 4.4.: Flowchart of texture analyzer

In order to fit the texture segmentation mask in the AV1 codec and minimize the artifacts produced by encoding, we add a series of post-processes for block-based

masks. As shown in Fig. 4.4, an adaptive K-means clustering [163] is used to group different types of texture in the texture segmentation mask. A temporal and spatial correction is then used to maintain the consistency of texture segmentation mask throughout the entire video sequence. The temporal correction uses a majority voting of three consecutive frames to determine whether a block in the current frame should be labeled as a texture block. The spatial correction uses 4-connectivity neighborhood voting to fill holes in the texture segmentation mask. Finally, a small connected labeling marks the connected components that are less than 5 blocks as non-texture blocks.

4.2.2 A new AV1 coding tool - texture mode

In this section, we describe how we modified the AV1 codec by introducing a texture mode to encode the identified texture blocks in a video frame.

Texture mode encoder design

The texture analyzer is integrated into the AV1 encoder as illustrated in Fig. 4.11. At the encoder, for each frame that contains texture area, we first fetch the texture masks for the current frame and the selected reference frames from the texture analyzer. Based on the texture region in the current frame, a set of texture motion parameter that represents the global motion of the texture area is estimated for each reference frame. For each block larger than 32×32, we use a two-step method to check if a block is a texture block as described in section 4.2.2. A texture block is reconstructed using global motion parameters and no motion compensation residuals will be coded and transmitted for the texture block. We call this new coding paradigm the *texture mode*. At the decoder, since there is no syntax change to the AV1 bitstream, the bitstream is decoded the same as AV1 baseline.

In general, a texture block in the current frame is reconstructed by warping the texture block from the reference frame towards the current frame. We use a modified

Table 4.1.: Configuration of different texture mode implementation

tex-all	*tex-sp*	*tex-cp*
Disable texture mode for GOLDEN / ALTREF frame		
Original GF group interval (4-16)	Fixed 8 GF group interval	
single-layer coding structure	multi-layer coding structure	
Use texture mode for all frames except GOLDEN /ALTREF frame	Use texture mode for every other frame (frame1,3,5,7 in the GF group)	
Use single-prediction (forward or backward)	Use single forward prediction	Use compound prediction

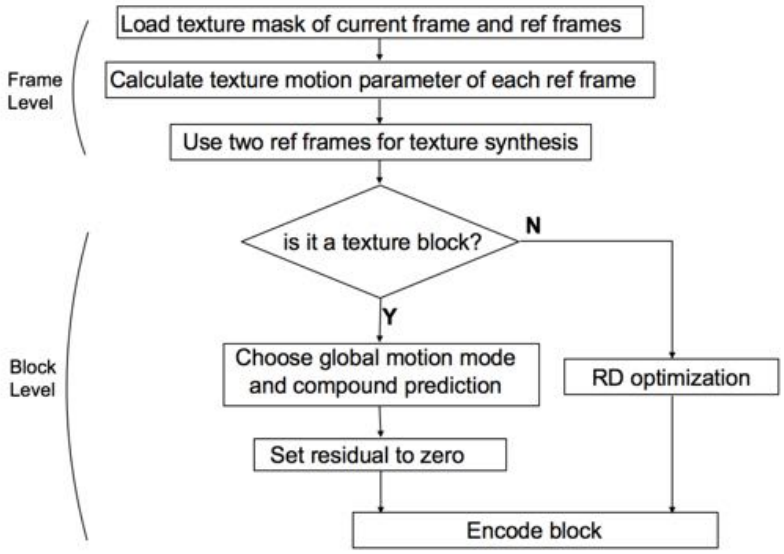


Fig. 4.5.: Texture mode encoder implementation

version of the global motion coding tool in the AV1 codec to perform block warping as described in Section 4.2.2. Based on the selection of coding structures and choices of reference frames for texture reconstruction, we investigated three different implementations, namely *tex-all*, *tex-sp*, and *tex-cp* of the texture mode in terms of data rate savings and perceived quality. Configuration of the three implementations are described in Table 4.1 and can be visualized in Fig. 4.6. Encoders of AV1 codec consider an input video sequence as a succession of frames grouped in Golden-Frame (GF) groups. GF groups may have 4-16 frames with the first frame being the GOLDEN

frame and the last frame being ALTREF frame. For *tex-sp* and *tex-cp*, a multi-layer hierarchical coding structure [140] is used for each GF group.

Fig. 4.6.: Coding structure of texture mode: (a) GF Group Coding Structure Using *tex-all* Configuration. (b) GF Group Coding Structure Using *tex-sp* Configuration. (c) GF Group Coding Structure Using *tex-cp* Configuration.

The *tex-all* implementation has the best data rate savings since the number of frames with texture mode enabled is approximately twice as many as the other two implementations. However, we noticed some visual artifacts in the reconstructed

videos in several test sequences due to the accumulated errors from warping displacement. If a block contains both texture and non-texture region and is classified as a texture block, there will be visual artifacts in the non-texture region. These artifacts can be passed on to other frames that use this frame as a reference frame, and causing the artifacts to propagate. The artifacts are most prominent in videos with high motion or complex global motion.

The *tex-sp* implementation solves the accumulation error by only enabling texture mode for every B-frame and disabling texture mode for every I and P frame. It only uses the immediate previous frame as the reference frame for texture warping to estimate a more accurate global motion model. As a result, the data rate savings are reduced to approximately half the data rate savings of the *tex-all* configuration. Some flickering artifacts can still be observed between frames for some test videos.

The *tex-cp* further reduces the flickering artifacts by using compound prediction from the previous frame and the next frame. The data rate savings are only slightly lower than that of *tex-sp*. The improvement in visual quality is most prominent in low-mid resolution videos. In practice, we use *tex-cp* as the configuration for the texture mode which ensures good visual quality.

Texture motion parameters

The global motion coding tool in AV1 is used primarily to handle camera motion. A motion model is explicitly conveyed at the frame level for the motion between a current frame and any one or more of its reference frames. The motion model can be applied to any block in the current frame to generate a predictor. The affine transformation is selected as the motion model. The motion model is estimated using a FAST feature matching scheme followed by a robust model fitting using RANSAC. The estimated global motion parameter is added to the compressed header of each inter-frame.

Since the motion model parameters of the global motion coding tool in AV1 is estimated at the frame level between the current frame and the reference frame,

these parameters may not accurately reflect the motion model for the texture regions within a frame. We modified the global motion tool to design a new set of motion modal parameters, called texture motion parameters. The texture motion parameters are estimated based on the texture region of the current frame and the reference frame using the same feature extraction and model fitting method as in the global motion coding tool. A more accurate motion model for texture region may reduce the artifacts on the block edges between the texture blocks and non-texture blocks in the reconstructed video. In order to keep the syntax of AV1 bitstream consistent, the texture motion parameters are sent to the decoder in the compressed header of the inter frames by replacing the global motion parameters of the reference frames.

Texture block decision

The minimum size of a texture block is 32×32 as described in Section 4.2.1. For all blocks larger than or equal to 32×32, we use a two-step approach to check if a block should be encoded using the novel texture mode scheme we proposed to AV1. First, we overlap the texture mask generated by the texture analyzer and the current frame to check if the entire block is inside the texture region of the current frame. We also need to ensure that the pixels used for texture reconstruction in the reference frames are within the texture regions identified by the texture analyzer. In order to maintain temporal consistency of the texture regions, in the second step, we warp the blocks inside the texture region towards the two reference frames, i.e., the previous frame and the next frame in the *tex-cp* configuration. If the two warped blocks are within the texture regions of both corresponding reference frames, the block is considered a texture block and texture mode is enabled.

Block Splitting Decision

Like most modern video coding standards, the baseline AV1 codec uses rate-distortion (RD) optimization to make block splitting decision and choose the best coding units. The problem of finding the block splitting decision that minimizes the distortion

between the original block and its reconstruction subject to a constrained rate can be described as

$$\underset{p \in P_k}{\text{minimize}}\, D_k(p) + \lambda R_k(p) \tag{4.1}$$

where $R_k(p)$ represents the number of bits that are required for signaling the block splitting and prediction decision p and the actual bit cost by an entropy coder in the bitstream. $D_k(p)$ represents the distortion measure. $\lambda \geq 0$ denotes the Lagrange multiplier.

In our proposed method, however, the position of the texture regions inside a block has higher priority than the RD values of different block splitting methods for this block. If the block is a texture block, we do not further split it into smaller sub-blocks. If the block contains no texture region, RD optimization is performed for block partitioning and mode decision. If part of a block contains texture region, we split it into sub-blocks regardless of the RD value as long as the size of the sub-blocks is equal or larger than 32×32. In general, there is no block that is part texture and part non-texture. The use of texture mode also largely increases the encoding speed, since no RD optimization is performed for a texture block which reduces the need for different prediction modes, reference frames selection, and the block splitting recursion.

Texture Reconstruction

We use AV1 codec's global motion tool and compound prediction to reconstruct texture for texture blocks at the decoder. The previous frame and the next frame of the current frame are chosen to be the reference frames for the texture block reconstruction. The texture regions in the two reference frames are warped towards the texture blocks in the current frame using the corresponding texture motion parameters. We used compound prediction to reconstruct the texture block from the two reference frames. As discussed earlier, the use of compound prediction for texture blocks reduces flickering artifacts between frames. The residual of the texture blocks is set to zero. Since all the texture blocks in one frame use the same reference frames and the

same set of motion parameters, there is no displacement on the block edges of the texture blocks within the texture region.

4.2.3 Experimental results

Texture analysis results:

Nine representative video sequences contain different texture types were tested using the CNN based texture analyzer. Sample texture segmentation results are shown in Fig. 4.7. Our texture analyzer can successfully identify and locate most texture regions. Currently, the texture analyzer uses a block-based texture classification method with fixed block size. Texture within smaller blocks is not included in the segmentation mask. For example, in the sequence bridgefar, some parts of the water, as well as the sky, are not included in the texture mask because they cannot cover a 32×32 block. Small non-texture parts can also be included in the segmentation mask when the majority of the 32×32 block is in a texture region, such as the bowing of the white boat in the sequence coastguard.

Coding performance:

To evaluate the performance of the proposed texture-based method, data rate savings at four quantization levels (QP = 16, 24, 32, 40) are calculated for each test sequence using the *tex-cp* configuration and compared to the AV1 baseline. The AV1 baseline is the original codec with a fixed group interval of eight frames and using hierarchical multilayer coding structure [140]. The data rate is computed by dividing the output WebM file size by the number of frames. The WebM file is the output bitstream from the AV1 encoder. The data rate saving is calculated as

$$P_{bit} = (R - R_b)/R_b \times 100\% \tag{4.2}$$

where P_{bit} represents data rate saving, R represents the bitstream size of the proposed method, R_b represents the bitstream size of the AV1 baseline method. A negative value indicates a reduction in the codec's bitstream data rate compared to the AV1 baselines.

Results for several test videos are shown in Table 4.2. The test videos include large texture areas. We also include the average percentage of pixels that uses the texture mode in a frame in the table.

As shown in Table 4.2, at low QP, most of the videos show large data rate savings. However, as the QP increases, the data rate savings decreases. At high QP, texture-based method tends to have worse coding performance than AV1 baseline for some test videos, such as football, waterfall and netflix_aerial. This is because at high QP, many non-texture blocks also have zero residual and our method requires a few extra bits for the texture motion parameters and for using two reference frames in compound prediction.

Result for subjective visual quality test:

Established quality assessment measures, such as the PSNR cannot be applied to evaluate our method because there can be a large pixel-wise difference in the expected MSE between texture regions reconstructed using motion compensation versus using the proposed method. However, an observer will not notice any difference without observing the original video sequence. A similar argument holds for the SSIM as it is based on the sample cross-covariance. Therefore, we performed a subjective visual quality study on 20 participants. The study received Purdue Institutional Review Board approval under protocol #1802020229. All subjects have signed the consent forms prior to participating in the research.

In the study, each participant is asked to watch two versions of a test video. One is the reconstructed video using the original AV1 codec with QP=16. The other is the reconstructed video using our proposed method (*tex-cp*) with QP=16. The participants are asked to choose the video that has better visual quality without

Table 4.2.: Data rate savings at different QP level with block-based texture mask

Video	Resolution	Number of Frames	Data Rate Saving (%)				Texture Region (%)
			QP=16	QP=24	QP=32	QP=40	
bridgeclose	cif	300	-13.77	-9.78	-3.20	1.67	33.12
bridgefar	cif	300	-8.80	-5.90	-4.44	-4.38	21.19
coastguard	cif	300	-7.80	-6.99	-4.70	-1.90	37.42
flower	cif	150	-10.55	-8.66	-5.96	-3.38	58.00
football	cif	150	-0.35	0.02	0.01	0.02	10.32
waterfall	cif	150	-4.63	-3.96	0.33	3.74	61.22
netflix_aerial	512x270	300	-8.59	-2.15	0.68	4.59	29.94
netflix_rollercoaster	512x270	300	-3.44	-2.39	-1.02	0.41	35.55
intotree	1280x720	300	-5.32	-4.23	-1.99	2.83	43.25
tractor	1280x270	300	-3.32	-2.25	-1.68	0.30	20.17
crowd_run	1920x1080	300	-0.16	-0.10	-0.05	0.10	9.45

Table 4.3.: Result for subjective visual quality test of *tex-cp*

Video	Quality of reconstructed video from *tex-cp* is **better** than the original codec	Quality of reconstructed video from *tex-cp* is **equal** to the original codec	Quality of reconstructed video from *tex-cp* is **worse** than the original codec
bridgeclose	15%	60%	25%
bridgefar	10%	65%	25%
coastguard	40%	40%	20%
flower	20%	55%	25%
football	0%	65%	35%
waterfall	25%	60%	15%
netflix_aerial	20%	75%	5%
netflix_rollercoaster	25%	60%	15%
intotree	15%	55%	30%

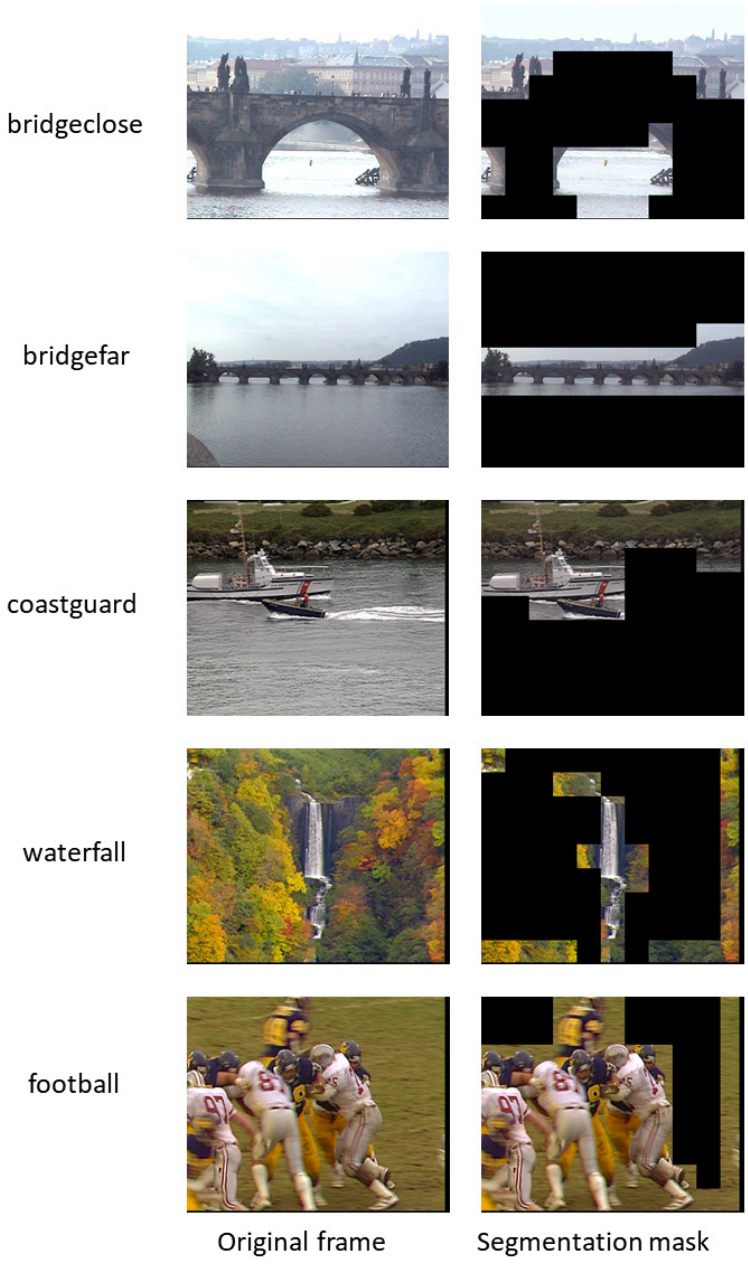


Fig. 4.7.: Texture segmentation examples

knowing which method is used to encoding the video. They were asked to choose

among three options: the first video has better visual quality, the second video has better visual quality, or there is no difference between the two versions of the video. The two versions of a test video are shown one at a time on a monitor. The viewing distance is about 50cm. The participants can choose to watch the videos multiple times. The result of this study is summarized in Table 4.3.

Results show that on average 59% of the time participants cannot tell the difference between the reconstructed video by the original codec and the proposed method. 19% of the time participants prefer the visual quality of the reconstructed video by the proposed method. 22% of the time the visual quality of the reconstructed video using baseline AV1 is preferred. This indicates that the visual quality degradation due to our video coding models is minimal and acceptable. The texture region does not use residual for reconstruction. Although they have quality degradation with respect to PSNR, an observer will not notice any difference without observing the original video sequence. The main artifacts mainly come from the inaccurate block-based texture mask. For example, the football sequence has flickering artifacts on some of the frames which is noticeable to the participant. In the proposed texture analyzer, a block is considered a texture block if it contains a large percentage of texture area, although it may also contain a small amount of structural objects. In the football sequence, some parts of the player are also included in the texture mask and reconstructed using texture mode. Since the motion of the player is different from the motion of the texture area, i.e. the grass, there are noticeable flickering artifacts around those parts of the frame. This phenomenon also happens to some frames in the coastguard sequence. We plan to improve the texture analyzer in our future work by generating a more accurate segmentation mask of the texture area to reduce these artifacts. Another visual artifact we noticed comes from the inaccurate motion model. For example, the intotree sequence has the most complex global motion among all the test sequences which may be better presented using the planar perspective motion model instead of the affine motion model used in the global motion model.

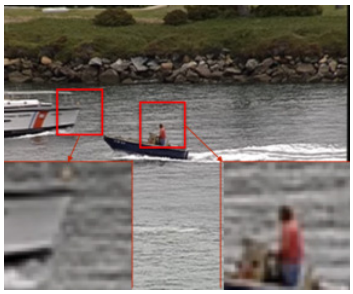
4.3 Pixel-level Texture Segmentation Based Video Coding With Switchable Scheme

4.3.1 Motivation

a) **Pixel-level texture mask based on semantic scene segmentation**

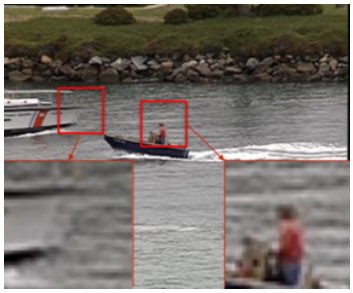
While the proposed approach in 4.2 can achieve a data rate saving of 1% to 13% compared to the baseline when implemented using AV1 with satisfactory visual quality, the block-based texture masks cannot always accurately represent the texture regions. The block-based texture masks can be seamlessly integrated into AV1 since the common coding units are blocks. However, it can sometimes cause noticeable visual artifacts when an identified texture block consists of small structural region. In addition, the smallest texture block size in [159] was 32×32 in order to avoid detecting small moving objects, but at the same time limits the size of identified texture regions and reduces potential data rate savings. To illustrate this, an example is shown in Figure 4.8 where part of the bow of the white boat and the person's head are identified as texture region in [159] since the majority of that block is texture. The bow of the white boat and the person's head show flickering artifacts since they have different motion trajectory than the river. There are also some texture regions in the river not identified due to the large block size used in the texture analyzer. The methods proposed in this paper address both of these issues.

Recent advances in deep neural networks have led to a renewed interest in semantic scene segmentation [104–106]. Large-scale datasest like ImageNet [102], COCO [103] and ADE20K [104] have enabled improved performance for these tasks. For example, the Fully Convolutional Network (FCN) [105] is one of the most commonly used network architectures for semantic scene segmentation. The major issue with FCN [105] is the lack of global contextual information to categories global scene which could lead to parsing error. The pyramid scene parsing network(PSPNet) [106] addresses this issue by adding a global pyramid pooling module to extract global information from the image.

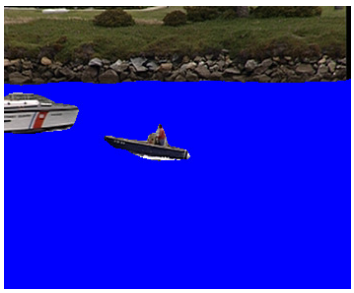


(a) reconstructed frame by AV1 original codec

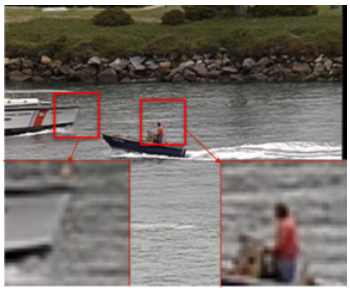
(b) block-based mask



(c) reconstructed frame using block-based mask



(d) pixel-level mask



(e) reconstructed frame using pixel-level mask

Fig. 4.8.: Comparison between block-based and pixel-level texture mask

In this section, we incorporate semantic scene segmentation into video compression by generating pixel-level texture segmentation masks to represent "perceptually insignificant" regions in a frame and use motion models to reconstruct the texture regions at the decoder to improve the coding efficiency.

b) **A switchable scheme for texture mode**

In our work [164], the texture mode is enabled for every B-frames. Experimental

results show that the method in [164] achieved significant data rate reductions in most standard test videos. However, some videos have worse coding performance than the AV1 baseline. One reason is that at high QP, the high compression ratio results in many zero residual blocks thus there is limited margin for data rate saving using texture based methods. In general, as QP increases, the data rate saving decreases. Another reason is that some videos have relatively small texture regions. Thus a limited number of blocks are using texture mode. In addition, the texture based method needs to transmit extra side information of the texture motion parameters, and some extra bits for using two reference frames in compound prediction for all the texture blocks.

We further improve this method by introducing a switchable scheme for the "texture mode" that automatically detects the Golden-Frame groups that have the largest potential data rate savings when texture mode is enabled. Our switchable region-based coding scheme leverages semantic segmentation to improve coding efficiency for video sequences containing texture-like areas.

4.3.2 Texture analysis using CNN[2]

A semantic scene segmentation method is used to generate masks for different semantic classes, and then group several semantic classes with similar texture into four texture classes to produce a single pixel-level segmentation mask for each texture class.

Stuff and objects are two high-level categories often used in semantic scene understanding [106, 165]. Stuff refers to background areas such as sky, grass, and they usually contain large texture areas, and objects are more likely to appear in the foreground.

In this work, we use a two-stream cascade network [165] to generate semantic scene segmentation shown in Figure 4.9. The stuff stream generates stuff segmentation

[2]Joint work with Qingshuang Chen

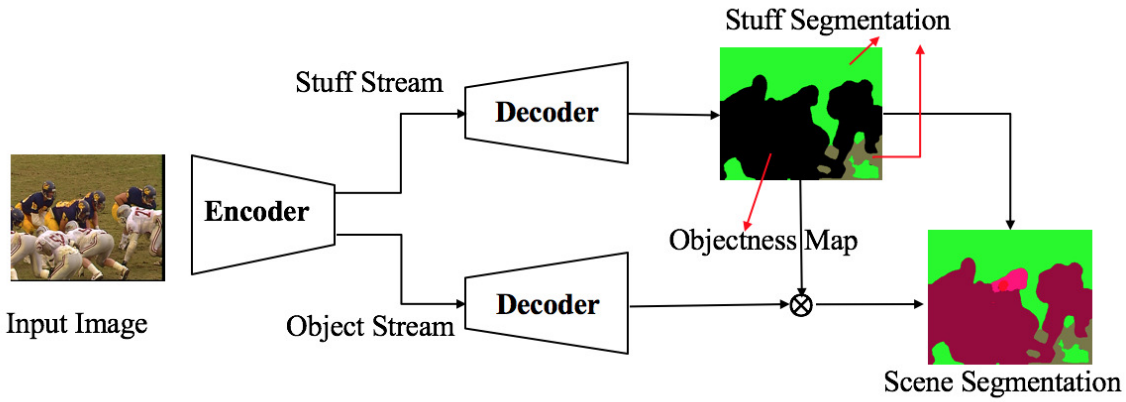


Fig. 4.9.: Two steam cascade framework

and objectness map, while the object stream generates object segmentation using the objectness map from the stuff stream. The final segmentation combines the results from stuff stream and object stream. We use ResNet50 [166] with dilated convolutions [167,168] as the encoder to extract feature map. For the decoder, we use the pyramid pooling module from PSPNet [106] followed by bilinear upsampling. The pyramid pooling module uses four different sizes of CNN receptive fields to represent global contextual information contained in four pyramid scales. The pyramid pooling module reduces the scene parsing errors by considering global contextual relationship in a scene. Cross-entropy loss is used at the end of each stream. The total training loss is the stuff steam loss plus the object stream loss as expressed in Equation 4.3, where for a given pixel x, N is the number of stuff classes and M is the number of object classes, $p_{x,i}$ is the predicted probability of pixel x for class i and $y_{x,i}$ is the binary indicator (0 or 1) for that class.

$$\mathcal{L}_{total} = -\sum_{i=1}^{N} y_{x,i} \log(p_{x,i}) - \sum_{j=1}^{M} y_{x,j} \log(p_{x,j}) \tag{4.3}$$

The pre-trained model is obtained from a scene parsing benchmark, MIT Scen-Parse150 [165], to generate semantic segmentation. The model is trained on a subset

of a densely annotated dataset, ADE20K [104], with the top 150 categories ranked by their pixel ratios in which 35 of them are stuff classes, 115 are object classes. The pixel accuracy of this model is 80.23% as reported on the benchmark [165].

After obtaining pixel-level semantic segmentation for each class, several semantic classes with similar texture are grouped into four texture classes to produce a single pixel-level segmentation mask for each texture class. We define four perceptually insignificant texture classes that are commonly observed in nature scenes. The four texture classes are based on groupings of different semantic classes defined in ADE20K dataset [104] that have similar textures. Texture class 1 includes earth and grass; texture class 2 includes water, sea and river; texture class 3 includes mountain and hill; texture class 4 includes tree. A single pixel-level mask for each texture class is generated by combining semantic segmentation within each group.

4.3.3 Switchable texture mode

We introduce a switchable scheme that is designed to identify GF groups that show potential data rate saving and only enables the texture mode for those GF groups. As shown in Figure 4.10, the first GF group of each scene is encoded with and without texture mode enabled to collect data rate savings. The texture region percentage is calculated as the average ratio of the total area of the texture blocks in the B-frames. If the texture region percentage of the first GF group is less than 10%, or if the GF group has no data rate saving when texture mode is enabled, we disable the texture mode for all the GF group in the scene. Otherwise, all the GF group in the scene are encoded with texture mode enabled.

The switchable scheme is designed under the assumption that the texture region percentage and the data rate saving performance of the first GF group in a scene can represent the rest of the GF groups in the same scene. In practice, we use the intra frames selected by AV1 as the scene change frames.

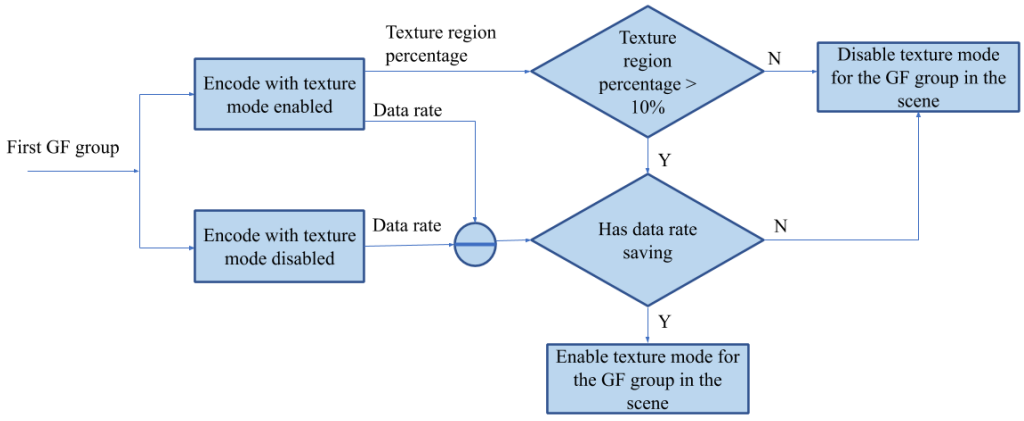


Fig. 4.10.: Switchable scheme of texture mode encoder implementation

The texture analyzer is then integrated into the AV1 encoder as illustrated in Figure 4.11. The configuration of *tex-cp* is used. At the encoder, We process the log of statistics of the first pass encoding to retrieve the scene change information. For each scene, we compare the size of the texture region of each texture class among the 4 texture classes and use the pixel-level texture mask of the largest one. The AV1 codec syntax allows only one set of motion parameters for each pair of the current frame and reference frame to be compressed and transmitted in the frame header. Therefore, for this study, only one class of texture is considered using texture mode in order to keep the original decoding syntax.

Then we use the switchable scheme to decide whether to enable the texture mode for each GF group in each scene. Within the GF groups whose texture mode are enabled, for each frame that contains texture area, we first fetch the texture masks for the current frame and the selected reference frames from the texture analyzer. Based on the texture region in the current frame, a set of motion parameter that represents the global motion of the texture area is estimated for each reference frame. For each block larger than 16×16, we use a two-step method to check if a block is a texture block as described in Section 4.2.2. The difference is that the texture analyzer produces pixel-level texture mask of arbitrary shape comparing with the

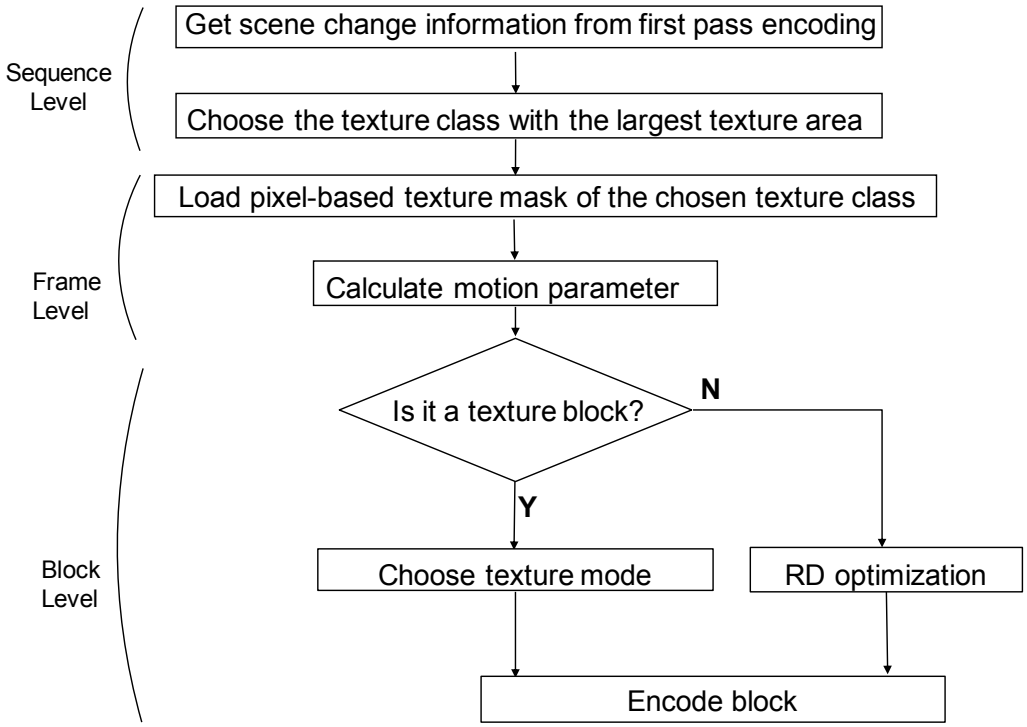


Fig. 4.11.: Switchable texture mode encoder implementation

block-based texture mask, as shown in Figure 4.12. A texture block is reconstructed using global motion parameters and no motion compensation residuals will be coded and transmitted for the texture block. At the decoder, since there is no syntax change to the AV1 bitstream, the bitstream is decoded the same as AV1 baseline.

We call this new coding paradigm the *switchable texture mode*.

4.3.4 Visual quality assessment

As mentioned in section 4.1.1, for texture synthesis based video compression, existing distortion-based metrics, such as PSNR and SSIM, are not suitable for evaluating visual quality since their calculations involve pixel-wise difference comparison rather than perceptually based metrics, which may not correspond with the human subjec-

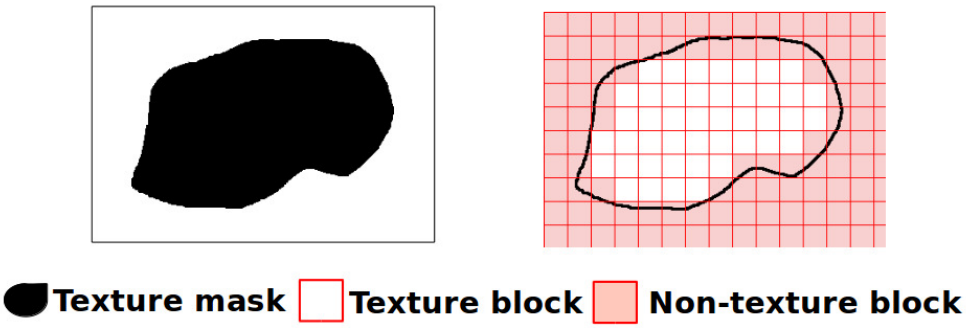


Fig. 4.12.: Texture block decision

tive evaluation. Inspired by [148, 169], we propose an objective video metric (Q_{tex}) to conduct visual quality assessment of the reconstructed video frames.

Q_{tex} consists of four components: blur artifact metric, edge artifacts metric, STSIM2 [169] and the temporal variance of STSIM2. The proposed metric is the average of the four components.

Blur Distortion Metric. In global motion transform and texture warping based reconstruction method, warping tends to cause blur. The level of blur depends on the type of transform used and the motion parameters. In [148], the authors proposed a blur estimation method by comparing subband coefficients (excluding DC) between the reference and test frames for the six high-frequency subbands in a single stage DT-CWT decomposition. We use it to estimate the blur artifacts in the texture area on the reconstructed frames and the corresponding area on the original frames. The blur artifact metric is described as:

$$A_b(x,y) = \sum_{i=1}^{6} |{B_i}^r(x,y)| - \sum_{i=1}^{6} \left|{B_i}^t(x,y)\right| \tag{4.4}$$

$$A_b = MEAN(A_b(x,y)) \tag{4.5}$$

$$A_{bn} = 1 - \frac{CLAMP(A_b, 0, 5)}{5} \tag{4.6}$$

For location (x, y), $|B_i{}^r(x, y)|$ and $\left|B_i{}^t(x, y)\right|$ are the amplitude of one of the six subband coefficients in the original frame and reconstructed frames, respectively. A_b is the average of $A_b(x, y)$ in the texture area. A_{bn} is the normalized A_b within range $[0, 1]$. The clamping threshold is empirically derived.

Edge Artifact Metric. Edge artifacts can occur at the boundary of the texture block and the non-texture block. Edge artifacts are most visible when motion parameters are not accurate. AV1 and most block-based codec use loop filters to reduce the block artifacts on the block edges. However, loop filters are not suitable for effectively reducing the edge artifact between a conventionally encoded block and a synthesized block. We employ a metric to quantify edge artifact similar to that presented in [148]. The edge artifact metric is given by:

$$A_e(x, y) = (E_v(x, y) + E_h(x, y)) \tag{4.7}$$

$$E_v(x, y) = (I_t(x, y) - I_t(x - 1, y)) - (I_r(x, y) - I_r(x - 1, y)) \tag{4.8}$$

$$E_h(x, y) = (I_t(x, y) - I_t(x, y - 1)) - (I_r(x, y) - I_r(x, y - 1)) \tag{4.9}$$

$$A_{en} = 1 - \frac{CLAMP(A_e, 0, 50)}{50} \tag{4.10}$$

For location (x, y), $I_r(x, y)$ and $I_t(x, y)$ are the pixel value in the original frame and reconstructed frames, respectively. A_e is the average of $A_e(x, y)$ among pixel locations on the boundary of texture and non-texture area. A_{en} is the normalized A_e within range $[0, 1]$. The clamping threshold is empirically derived.

STSIM2_mean. To compare the structural texture similarity between the reconstructed frames and the original frames, we employ the STSIM2 metric from [169]. STSIM2 is derived from SSIM. It decouples from the point-to-point measurement in SSIM and the structure is characterized by the cross-correlation between corresponding windows in the two frames being compared.

$$STSIM2\_mean = \frac{1}{N}\sum_{i=1}^{N}(STSIM2(F_i)) \tag{4.11}$$

where N is the number of frames and F_i is the i^{th} frame.

STSIM2_var. We use *tex-cp* implementation in our proposed method which only utilizes texture mode on the B-frames in the GF groups. This may cause flickering artifact if there exists a large visual quality difference between the frames that use texture mode and neighboring frames that do not use texture mode. The STSIM2_var characterizes the temporal artifact with the variance of STSIM2 of sequential frames.

$$STSIM2\_var = \frac{1}{N-1}\sum_{i=2}^{N}(STSIM2(F_i) - STSIM2(F_{i-1})) \tag{4.12}$$

$$STSIM2\_var_n = 1 - \frac{CLAMP(STSIM2\_var, 0, 0.05)}{0.05} \tag{4.13}$$

where N is the number of frames and F_i is the i^{th} frame. The clamping threshold is empirically derived.

The proposed perceptual texture metric Q_{tex} is the average of the four metrics:

$$Q_{tex} = \frac{1}{4}(A_{bn} + A_{en} + STSIM2\_mean + STSIM2\_var_n) \tag{4.14}$$

where Q_{tex} is within the range of $[0, 1]$.

$$Q_{tex}(dB) = 10log(\frac{1}{1 - Q_{tex}}) \tag{4.15}$$

Figure 4.13 shows a sample frame from the test sequence *NewsClip_480P-7a0d* reconstructed by AV1 and by proposed texture mode. Figure 4.14 shows the RD curve for PSNR and Q_{tex} of the reconstructed frame at four QP values. Although the visual quality difference between the two reconstructed frames is unnoticeable, there is a large PSNR gap. However, the Q_{tex} shows better consistency with the perceptual quality of the frame.

Fig. 4.13.: Visual comparison of a sample reconstructed frame using AV1 baseline and our proposed texture mode, along with the texture mask used.

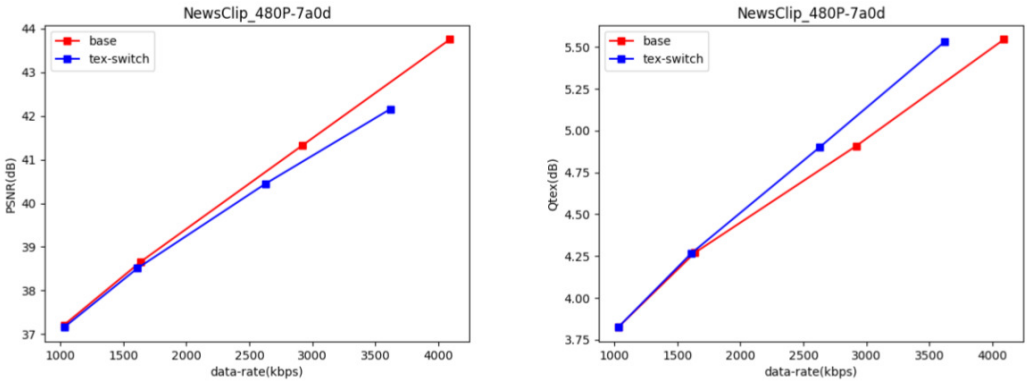


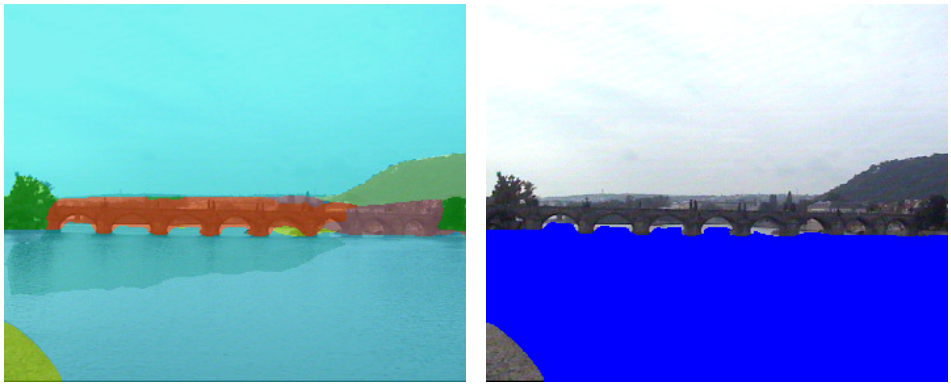
Fig. 4.14.: PSNR and Q_{tex} comparison for the sample reconstructed frame in Figure 4.13.

4.3.5 Experimental results

We selected sequences with texture regions from standard test sequences and the YouTube UGC data set [158]. YouTube UGC dataset is a sampling from thousands of User Generated Content (UGC) videos uploaded to YouTube. We calculate the data rate savings at different quantization levels. The Q_{tex} perceptual quality metric introduced in Section 4.3.4 is used to estimate the BD-RATE savings. We also conducted a subjective video quality test on the reconstructed videos.

Texture analysis results:

Figure 4.15 shows an example of pixel-level texture segmentation mask for texture class 2, which combines semantic segmentation of water and river.



(a) Semantic segmentation (b) Texture mask for class 2

Fig. 4.15.: An example of pixel-level texture segmentation for video sequence *bridgefar*. Texture mask for class 2 contains semantic segmentation of water and river in this example.

Figure 4.16 shows a comparison between the texture segmentation results of the first five frames of *coastguard* by a feature based texture analyzer and two proposed CNN based texture analyzers. Figure 4.16(a) is the result of the segmentation method proposed in [170] that uses color and edge based features. Figure 4.16(b) is the result of the proposed block-based CNN method described in section 4.2.1. The texture mask generated by block-based CNN is more accurate and shows more spatial and temporal consistency. As shown in 4.16(c), the result of the block-based CNN method is further improved by a pixel-level texture segmentation method as described in 4.3.2.

Coding performance:

To evaluate the performance of the proposed switchable texture mode method, data rate savings at four quantization levels (QP = 16, 24, 32, 40) are calculated for each test sequence using the *tex-cp* configuration and compared to the AV1 baseline. We compared the data rate saving result of encoding with the color-edge feature based texture mask, blocked-based texture mask and the pixel-level texture mask. We also

(a) Texture mask for *coastguard* with color and edge features

(b) Texture mask for *coastguard* with block-based CNN

(c) Texture mask for *coastguard* with pixel-level CNN

Fig. 4.16.: Texture segmentation example with CNN method and color-edge feature based method

compared the data rate saving result of encoding with the proposed switchable texture mode scheme and with texture mode enabled for all GF groups.

Table 4.4.: AV1 data rate savings comparison between color-edge feature based (FM), block-level (BM) and pixel-level (PM) texture segmentation. A negative value indicates a reduction in the codec's bitstream data rate compared to the AV1 baselines.

Video		**Data rate saving(%)**											
		QP=16			QP=24			QP=32			QP=40		
		FM [170]	BM [159]	PM [164]	FM	BM	PM	FM	BM	PM	FM	BM	PM
low res	*coastguard*	0.17	-7.8	-9.14	0.36	-6.99	-8.01	0.43	-4.7	-5.72	0.62	-1.9	-2.13
	flower	-7.42	-10.55	-13	-5.42	-8.66	-10.78	-2.51	-5.96	-4.95	-0.19	-4.95	-4.95
	football	-0.61	-0.35	-0.63	-0.41	0.02	-0.08	-0.04	0.01	0.01	0.02	0.02	0
	waterfall	-3.65	-4.63	-13.11	-1.58	-3.96	-7.21	0.14	0.33	-1.3	3.00	3.74	3.48
	netflix_aerial	-1.15	-8.59	-9.15	0.26	-2.15	-5.59	1.32	0.68	-1.05	2.10	4.59	4.01
high res	*intotree*	-0.88	-5.32	-9.71	-0.15	-4.32	-9.42	0.14	-1.99	-8.46	0.26	2.83	-4.92

a) **FM vs. BM vs. PM**

As shown in Table 4.4, compared to the FM method, the BM and PM methods show larger data rate saving for the most of the test videos. The texture mask generated by color and edge features is less accurate and less consistent both spatially and temporally. So the number of pixels that are reconstructed using texture mode is usually much smaller than that of BM and PM method. The parameters in the FM method need to be tuned for each video to get better texture segmentation result.

compared to the BM method, the PM method shows larger data rate savings. For the BM method, the fixed size blocks for CNN based texture analyzer need to be large enough to ensure classification accuracy. While for the PM method, there is no such limitation so we use 16×16 as the minimum size for texture blocks instead of 32×32 in the texture mode. Therefore, there are more pixels in a frame that are reconstructed using the texture mode in the PM method leading to larger data rate savings. We did not using smaller texture blocks because further block splitting will require extra bits to send the motion information for these blocks. Table 4.5 shows the texture region percentage, defined in equation 4.16 as average percentage of the regions that are reconstructed using texture mode within the frames where texture mode is enabled.

$$P_{tex} = (\sum_{j=1}^{F_{tex}} (\frac{\sum_{i=1}^{N_j} B_{ij}}{W \times H}))/F_{tex} \times 100\% \tag{4.16}$$

where F_{tex} is the number of frames that enables texture mode, N_j is the number of texture blocks in the j^{th} frame, B_{ij} is the block size of texture block i in frame j, W and H are frame width and height.

Table 4.5.: Texture region percentage

Texture region encoded	FM (%)	BM (%)	PM (%)
coastguard	12	37	41
flower	19	58	24
football	16	10	22
waterfall	56	61	77
netflix_aerial	24	37	53
intotree	5	43	52

The texture percentage of the FM method is smaller than that of the BM and PM method for most of the test videos. The texture masks generated by FM are less consistent. So there are larger number of blocks in the texture region on the current frame cannot be identified as texture blocks because When warped onto the reference frame they are not in the texture region of the reference frame. One exception is the texture region percentage of FM for *football* which is larger than that of BM. Because the color and edge features for the texture and non-texture region of *football* are easy to be separated and FM uses 16×16 texture block size while BM uses 32×32 texture block size. The texture region percentage of FM for *intotree* is much smaller than the other two. Because the FM identifies the smaller sky area in the *intotree* as the texture region but not the larger tree area identified by BM and PM.

In general the texture region percentage of the PM method is larger than that of the BM method, thus the increase in data rate saving. The texture region percentage of PM for *flower* is smaller because the texture mask of BM contains sky and flowerbed area as it fails to identify them as two different classes of texture. Although texture mask of PM only contains flowerbed area, the sky area is very homogeneous which has small residual using AV1 baseline. Therefore, we still achieve more data rate saving using PM than BM. The PM method also reduces flickering artifacts in some of the reconstructed video when using the BM method. The pixel-level texture mask can more accurately represent the perceptually insignificant pixels. An example is illustrated in Figure 4.8 and discussed in Section 3.1.

b) ***tex-allgf* vs. *tex-switch***

Compare to the AV1 baseline, the coding performance of *tex-allgf* shows larger data rate savings at lower QP. However, as QP increases, the data rate saving decreases. In some cases, *tex-allgf* has worse coding performance than the AV1 baseline at high QP.

Compare to the *tex-allgf* method, the proposed *tex-switch* method shows larger or equal data rate savings in most cases. When the switchable scheme enables the

Table 4.6.: Data rate saving comparison between *tex-allgf* and *tex-switch* methods on UGC dataset videos. A negative value indicates a reduction in the bitstream data rate compared to the AV1 baseline. The green blocks indicate more data rate saving when switchable scheme is applied while the red blocks indicate the opposite.

Video Sequence	**Resolution**	**QP=16**		**QP=24**		**QP=32**		**QP=40**	
		tex-allgf	**tex-switch**	**tex-allgf**	**tex-switch**	**tex-allgf**	**tex-switch**	**tex-allgf**	**tex-switch**
NewsClip_360P-1b1c	360P	-0.75	-0.75	-0.49	-0.55	0.34	0.09	1.45	0.00
NewsClip_360P-1e1c	360P	-10.77	-10.77	-9.27	-9.27	-5.23	-5.23	-1.54	-1.54
NewsClip_360P-22ce	360P	-17.37	-17.37	-15.79	-15.79	-16.37	-16.37	-17.98	-17.98
NewsClip_360P-66ae	360P	0.03	0.00	0.06	0.00	0.05	0.00	0.08	0.00
Sports_360P-0c66	360P	-0.71	-0.71	0.05	0.00	0.67	0.00	1.32	0.00
TelevisionClip_360P-3b9a	360P	-1.45	-1.45	-0.48	-0.48	1.09	0.00	3.26	0.00
TelevisionClip_360P-74dd	360P	-1.66	-1.66	-1.17	-1.17	-0.36	-0.36	0.37	0.00
HowTo_480P-04f1	480P	-3.81	-3.81	-2.57	-2.57	-0.93	-0.93	-0.06	-0.36
HowTo_480P-4c99	480P	-2.36	-2.36	-1.67	-1.67	-0.37	0.00	1.16	0.00
MusicVideo_480P-1eee	480P	-3.31	-3.31	-3.29	-3.29	-2.53	-2.53	0.30	0.30
MusicVideo_480P-2de0	480P	0.35	0.00	0.40	0.00	0.74	0.00	1.16	0.00
MusicVideo_480P-41ce	480P	0.32	-0.11	0.63	0.01	1.14	0.07	1.06	-0.10
NewsClip_480P-15fa	480P	-6.31	-6.31	-6.05	-5.79	-0.53	-0.11	0.79	-0.03
NewsClip_480P-7a0d	480P	-11.54	-11.54	-10.03	-10.03	-1.53	-1.53	-0.08	0.00
TelevisionClip_480P-19d3	480P	-3.13	-3.13	-2.86	-2.86	-1.66	-1.66	-0.58	0.00
HowTo_720P-0b01	720P	-12.72	-12.72	-11.84	-11.84	-9.31	-9.31	-6.35	-6.35
MusicVideo_720P-3698	720P	-1.76	-1.76	-1.07	-1.07	-0.30	-0.30	0.17	0.00
MusicVideo_720P-4ad2	720P	-6.93	-6.93	-3.81	-3.81	-1.87	-1.87	-0.60	-0.11
NewsClip_720P-4603	720P	-0.01	0.00	0.52	0.00	0.80	0.00	0.45	0.00
HowTo_1080P-4d7b	1080P	-7.31	-7.31	-6.07	-6.07	-3.21	-3.21	-0.72	-0.72
MusicVideo_1080P-55af	1080P	-3.88	-3.88	-1.78	-1.78	-0.31	-0.33	0.99	0.68
NewsClip_1080P-1db0	1080P	-0.24	-0.24	-0.17	0.00	0.01	0.00	0.08	0.00
average		-4.33	-4.37	-3.49	-3.55	-1.80	-1.98	-0.69	-1.19

Table 4.7.: Data rate savings comparison between *tex-allgf* and *tex-switch* methods on standard test sequences. A negative value indicates a reduction in the bitstream data rate compared to the AV1 baseline. The green blocks indicate more data rate saving when switchable scheme is applied.

Video Sequence	**Resolution**	**QP=16**		**QP=24**		**QP=32**		**QP=40**	
		tex-allgf	**tex-switch**	**tex-allgf**	**tex-switch**	**tex-allgf**	**tex-switch**	**tex-allgf**	**tex-switch**
bridgeclose	cif	-15.78	-15.78	-10.87	-10.87	-4.21	-4.21	-2.77	-2.77
bridgefar	cif	-10.68	-10.68	-8.56	-8.56	-6.34	-6.34	-6.01	-6.01
coastguard	cif	-9.14	-9.14	-8.01	-8.01	-4.70	-4.70	-2.13	-2.13
flower	cif	-13.00	-13.00	-10.78	-10.78	-4.95	-4.95	-3.20	-3.20
waterfall	cif	-13.11	-13.11	-7.21	-7.21	-1.30	-1.30	3.48	0.00
netflix_ariel	512x270	-9.15	-9.15	-5.59	-5.59	-1.05	-1.05	4.59	0.00
intotree	1080P	-9.71	-9.71	-9.42	-9.42	-8.46	-8.46	-4.92	-4.92
average		-11.55	-11.55	-8.63	-8.63	-4.43	-4.43	-1.57	-2.72

texture mode for all the GF group in the test video, data rate saving of *tex-allgf* and *tex-switch* are the same. The *tex-switch* method disables texture mode for some of the GF groups in the test video to avoid using more data rate than the AV1 baseline. In the cases where *tex-switch* has zero data rate saving compared to the AV1 baseline, the texture mode is disabled for all the GF groups. The *tex-switch* method has improved data rate savings on average for all the four QPs, especially for larger QPs where *tex-allgf* has worse coding performance than the AV1 baseline in many cases. In a few cases, however, *tex-switch* has less data rate saving than *tex-allgf*. This is because the data rate saving performance of the first GF group in the scene fails to accurately represent the whole scene.

Quality assessment:

Table 4.8 and Table 4.9 show the BD-RATE(Q_{tex}) of *tex-switch* method compared to AV1 baseline. A negative value indicates a reduction in bitstream data rate with the same Q_{tex} as the quality metric. The average data rate saving in the tables is the average data rate saving of different QPs from Table 4.8 and Table 4.9.

The texture type in the tables is either time-static (e.g. ground, grass) or dynamic (e.g. water). We observe that videos with large data rate saving and static texture type have more consistency BD-RATE(Q_{tex}) improvement compared to dynamic texture, e.g. *HowTo_480P-04f1*, *MusicVideo_480P-1eee*, *NewsClip_480P-15fa*, *TelevisionClip_480P-19d3*, *MusicVideo_720P-4ad2*, *Howto_1080P-4d7b*, *flower*, and *netflix_aerial*. The correlation between the Q_{tex} and the subjective visual quality for the dynamic texture is not as high as that of static texture type. As a result, the Q_{tex} may be low for videos with dynamic texture area reconstructed by *tex-switch* method even when the reconstructed video does not have noticeable visual artifact. In these cases, there is little or no BD-RATE(Q_{tex}) improvement by the proposed method but the data rate saving is high, e.g. *NewsClip_360P-22ce* in the UGC dataset, and *bridgefar, coastguard, waterfall* in the standard test sequences. In the cases where

Table 4.8.: BD-RATE(Q_{tex}) of *tex-switch* on UGC dataset videos.

Video Sequence	**BD-RATE (Qtex)**	**Average Data Rate Saving(%)**	**Texture Type**
NewsClip_360P-1b1c	4.53	-0.30	static
NewsClip_360P-1e1c	-5.12	-6.70	dynamic
NewsClip_360P-22ce	2.72	-16.88	dynamic
NewsClip_360P-66ae	0.00	0.00	static
Sports_360P-0c66	3.52	-0.18	static
TelevisionClip_360P-3b9a	-0.06	-0.48	static
TelevisionClip_360P-74dd	2.36	-0.80	static
HowTo_480P-04f1	-1.35	-1.92	static
HowTo_480P-4c99	-0.43	-1.01	static
MusicVideo_480P-1eee	-1.20	-2.21	static
MusicVideo_480P-2de0	0.00	0.00	static
MusicVideo_480P-41ce	-0.01	-0.03	static
NewsClip_480P-15fa	-3.33	-3.06	static
NewsClip_480P-7a0d	-6.31	-5.77	dynamic
TelevisionClip_480P-19d3	-1.12	-1.91	static
HowTo_720P-0b01	-7.03	-10.05	dynamic
MusicVideo_720P-3698	-0.58	-0.78	static
MusicVideo_720P-4ad2	-2.20	-3.18	static
NewsClip_720P-4603	0.00	0.00	static
HowTo_1080P-4d7b	-2.32	-4.33	static
MusicVideo_1080P-55af	1.52	-1.33	dynamic
NewsClip_1080P-1db0	-0.01	-0.06	static

Table 4.9.: BD-RATE(Q_{tex}) of *tex-switch* on standard test sequences.

Video Sequence	**BD-RATE (Qtex)**	**Average Data Rate Saving(%)**	**Texture Type**
bridgeclose	-1.65	-8.41	dynamic
bridgefar	5.06	-7.90	dynamic
coastguard	10.12	-6.06	dynamic
flower	-1.41	-7.98	static
waterfall	5.29	-5.41	dynamic
netflix_aerial	-1.14	-3.95	static
intotree	3.88	-8.13	dynamic

BD-RATE and data rate saving are zero, the texture mode is disabled for all the GF groups.

Result for subjective visual quality test:

We also performed a subjective visual quality study on 20 participants in addition to evaluating using objective video quality metrics, Q_{tex}.

In this study, each participant is asked to watch two versions of a test video sequence. One is the reconstructed video using the original AV1 codec with QP=16. The other is the reconstructed video using our proposed method (*tex-switch*) with QP=16. The subjective test setting is as described in section 4.2.3.

The result of this study is summarized in Table 4.10. Results show that on average 42.59% of the time participants cannot tell the difference between the reconstructed video by the original codec and the proposed method. 28.62% of the time participants prefer the visual quality of the reconstructed video by the proposed method. 28.79% of the time the visual quality of the reconstructed video using baseline AV1 is preferred.

The main artifacts come from the inaccurate pixel-based texture mask. For example, in some frames of *MusicVideo_720P-3698* and *TelevisionClip_360P-74dd* sequence, the texture masks include parts of the moving objects in the foreground, which are reconstructed using texture mode. Since the motion of the moving objects is different from the motion of the texture area, there are noticeable artifacts around those parts of the frame. Another visual artifact we noticed comes from the inaccurate motion model. For example, the *intotree* sequence has the most complex global motion among all the test sequences which may be better presented using the planar perspective motion model instead of the affine motion model used in the global motion model.

Figure 4.17, Figure 4.18 and Figure 4.19 show some sample reconstructed frames using *tex-switch* and AV1 baseline method, as well as the texture blocks in those frames.

Table 4.10.: Result for subjective visual quality test of *tex-switch*

Video	Quality of reconstructed video from *tex-switch* is **better** than the original codec	Quality of reconstructed video from *tex-switch* is **equal** to the original codec	Quality of reconstructed video from *tex-switch* is **worse** than the original codec
NewsClip_360P-1b1c	50%	25%	25%
NewsClip_360P-1e1c	50%	10%	40%
NewsClip_360P-22ce	35%	25%	40%
NewsClip_360P-66ae	15%	45%	40%
Sports_360P-0c66	65%	30%	5%
TelevisionClip_360P-3b9a	50%	20%	30%
TelevisionClip_360P-74dd	20%	15%	65%
HowTo_480P-04f1	40%	20%	40%
HowTo_480P-4c99	25%	55%	20%
MusicVideo_480P-1eee	35%	55%	10%
MusicVideo_480P-2de0	70%	10%	20%
MusicVideo_480P-41ce	45%	30%	25%
NewsClip_480P-15fa	60%	25%	15%
NewsClip_480P-7a0d	40%	55%	5%
TelevisionClip_480P-19d3	35%	30%	35%
HowTo_720P-0b01	55%	10%	35%
MusicVideo_720P-3698	15%	5%	80%
MusicVideo_720P-4ad2	55%	5%	40%
NewsClip_720P-4603	50%	30%	20%
HowTo_1080P-4d7b	60%	30%	10%
MusicVideo_1080P-55af	40%	15%	45%
NewsClip_1080P-1db0	20%	75%	5%
bridgeclose	65%	5%	30%
bridgefar	50%	45%	5%
coastguard	40%	45%	15%
flower	70%	0%	30%
waterfall	40%	45%	15%
netflix_aerial	30%	55%	15%
intotree	10%	15%	75%
average	42.59%	28.62%	28.79%

(a) Frame reconstructed by AV1 baseline method

(b) Frame reconstructed by tex-switch

(c) Corresponding texture area

Fig. 4.17.: Sample reconstructed video frame for *NewsClip_480P-15fa*, QP=16

(a) Frame reconstructed by AV1 baseline method

(b) Frame reconstructed by tex-switch

(c) Corresponding texture area

Fig. 4.18.: Sample reconstructed video frames for *MusicVideo_720P-3698*, QP=16

(a) Frame reconstructed by AV1 baseline method

(b) Frame reconstructed by tex-switch

(c) Corresponding texture area

Fig. 4.19.: Sample reconstructed video frames for *MusicVideo_720P-4ad2*, QP=16

5. CONCLUSION AND FUTURE WORK

5.1 VP9 Video Coding For Lossy Transmission Channels Using Error Resilience Packets

5.1.1 Conclusion

In chapter 2, we presented a VP9-based error resilient video coding method that uses error resilience packets that consist of the frame-level macroblock prediction information and encoded keyframe information. Experimental results exhibit that our method performs well in terms of both PSNR and image quality under typical lossy network conditions.

5.1.2 Future work

In the future, we could use downsampled keyframes to improve the compression efficiency of our current method. The downsampled keyframes could be reconstructed using image super-resolution methods. We could also develop a backward frame prediction that uses future frames to improve the concealment performance.

5.2 Multi-Reference Video Coding Using Stillness Detection

5.2.1 Conclusion

In chapter 3, we proposed an automatic GF group stillness feature detection method. Each GF group is classified into still GF group and non-still GF group based on three metrics and the encoder adaptively chooses the coding structure based on optimized coding efficiency. Experimental results showed coding gain for videos containing still GF group.

5.2.2 Future work

We also observed that GF groups containing other features, such as fast zoom-out and high motion, may also benefit from the single layer coding structure. We could explore the possibility to obtain other GF group features from the AV1 first coding pass to help choose the coding structure for the second coding pass.

5.3 Advances In Region-Based Video Coding Using Deep Neural Network

5.3.1 Conclusion

In chapter 4, we proposed a new video coding paradigm that integrates texture modeling into a video codec. The texture modeling method uses deep learning based approaches to detect texture regions in a frame that is perceptually insignificant to the human visual system. We developed a scheme to integrate the texture analyzer into the video codec that reconstructs the texture region differently for B/P frames which largely reduces the temporal visual artifacts. We introduced a new perceptual video quality metric to assess the quality of reconstructed videosThe proposed method is implemented and tested using the AV1 codec by introducing a new coding mode - texture mode, to the AV1 encoder. The texture mode uses the multi-layer coding structure, a modified global motion tool and the compound prediction mode.

5.3.2 Future work

We envision that region-based coding will continue to improve in terms of both quality and data rate, especially by leveraging advances of deep learning methods. However, there remain several open challenges that require further investigation.

Accurate region analysis: The accuracy of region analysis or segmentation is one of the major challenges of region-based video coding. Recent advances in scene under-

standing have significantly improved the performance of region analysis. For example, in the case study, the proposed pixel-level deep learning based segmentation approach has improved texture segmentation accuracy, leading to improved data rate savings and visual quality. However, visual artifacts are still noticeable when non-texture region is incorrectly included in the texture mask, particularly if the analysis/synthesis coding system is open loop. One potential solution is to incorporate the perceptual visual quality measure in-loop in the texture region reconstruction.

Memory and computation efficiency of deep tools: In the case study, we have shown that the analysis/synthesis approach using texture masks generated by neural networks can result in significant data rate savings while maintaining visual quality. This result reveals the huge potential of utilizing neural networks in region based video coding to improve code efficiency. However, the computational overhead of obtaining the segmentation masks is still a big obstacle for practical systems. Generating pixel-based mask is more computationally complex and requires more memory for model parameters compared to earlier block-based texture analyzer [171]. There exists a trade-off between the accuracy of the region representation, e.g., texture, and computational complexity. To address this challenge, future work needs to explore more compact and efficient networks structures, and to design hardware that is efficient to deploy deep neural networks.

Proper visual quality metric: A proper perceptual visual quality metric for video with synthesized region still remains a challenge. Most research on analysis/synthesis approaches conduct subjective test for visual quality measurement. In our case study, the proposed metric does not work well for dynamic texture. One solution could be to add a thresholding mask for the STSIM2 part of the proposed metric to reduce the impact of the high frequency component in the image as adopted in [45].

In addition, the blur distortion metric in our proposed quality metric is designed to be the small the better. Because in general, a clearer image is more preferable and

blurring is an artifact the viewer would like to avoid. However, sometimes an over-sharpened video is also not desirable by the viewer. We need to quantitatively verify the effectiveness of the proposed quality metric and its fitness with the subjective visual quality sense of a human, for example through subjective MOS (Mean Opinion Score).

Lack of annotated datasets: Video segmentation benchmark datasets with high quality annotation are important for developing machine learning methods for region-based video coding. There are several texture segmentation datasets for images [172]. However, the images in these dataset are not segmented into "perceptually significant" and "perceptually insignificant" regions, but rather objects with homogeneous texture. The lack of temporal information in these datasets would result in poor temporal consistency when directly applied to texture analysis in video, therefore post-processing is required. There are few datasets for general video object segmentation [173, 174], but they do not include texture segmentation. A publicly available dataset is provided in [175] which contains test sequences with a wide range of static and dynamic textures, as well as subjective opinion scores, but it does not provide texture segmentation labels.

Data dependency robustness: The current test videos are picked from standard test video dataset and Youtube UGC dataset that contain texture regions which have potential data rate savings using texture analysis/synthesis method. The proposed method needs to be further tested on more general test video datasets that contain large categories of video content.

REFERENCES

REFERENCES

[1] D. Mukherjee, J. Bankoski, R. S. Bultje, A. Grange, J. Han, J. Koleszar, P. Wilkins, and Y. Xu, "The latest open-source video codec VP9 - an overview and preliminary results," *Proceedings of the IEEE Picture Coding Symposium*, pp. 390–393, December 2013, San Jose, CA.

[2] ——, "A technical overview of VP9 - the latest open-source video codec," *SMPTE Annual Techical Conference & Exhibition*, October 2013.

[3] U. Joshi, D. Mukherjee, J. Han, Y. Chen, S. Parker, H. Su, A. Chiang, Y. Xu, Z. Liu, Y. Wang, J. Bankoski, C. Wang, and E. Keyder, "Novel inter and intra prediction tools under consideration for the emerging AV1 video codec," *Applications of Digital Image Processing XL*, vol. 10396, pp. 54–66, 2017.

[4] Z. Liu, D. Mukherjee, W. Lin, P. Wilkins, J. Han, and Y. Xu, "Adaptive multi-reference prediction using a symmetric framework," *Electronic Imaging*, vol. 2017, no. 2, pp. 65–72, 2017.

[5] Y. Chen, D. Murherjee, J. Han, A. Grange, Y. Xu, Z. Liu, S. Parker, C. Chen, H. Su, U. Joshi, C. Chiang, Y. Wang, P. Wilkins, J. Bankoski, L. Trudeau, N. Egge, J. Valin, T. Davies, S. Midtskogen, A. Norkin, and P. de Rivaz, "An overview of core coding tools in the AV1 video codec," *Proceedings of Picture Coding Symposium*, pp. 41–45, June 2018, San Francisco, CA.

[6] P. Tudor, "MPEG-2 video compression," *Electronics Communication Engineering Journal*, vol. 7, no. 6, pp. 257–264, December 1995.

[7] G. Sullivan, J. Ohm, W. Han, and T. Wiegand, "Overview of the high efficiency video coding (HEVC) standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, no. 12, pp. 1649–1668, December 2012.

[8] M. Wien, *High efficiency video coding: coding tools and specification.* New York, NY: Springer Publishing Company, Incorporated, 2014.

[9] "AOM - alliance for open media," http://www.aomedia.org/.

[10] D. Mukherjee, S. Li, Y. Chen, A. Anis, S. Parker, and J. Bankoski, "A switchable loop-restoration with side-information framework for the emerging AV1 video codec," *Proceedings of the IEEE International Conference on Image Processing*, pp. 265–269, September 2017, Beijing, China.

[11] S. Parker, Y. Chen, D. Barker, P. de Rivaz, and D. Mukherjee, "Global and locally adaptive warped motion compensation in video compression," *Proceedings of the IEEE International Conference on Image Processing*, pp. 275–279, September 2017, Beijing, China.

[12] Y. Chen and D. Mukherjee, "Variable block-size overlapped block motion compensation in the next generation open-source video codec," *Proceedings of the IEEE International Conference on Image Processing*, pp. 938–942, September 2017, Beijing, China.

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

[14] G. Bjøntegaard, "Calculation of average PSNR differences between RD-curves," *13th VCEG meeting*, March 2001, Austin, Texas.

[15] N. Gadgil, "Error resilient video coding using bitstream syntax and iterative microscopy image segmentation," Ph.D. dissertation, Purdue University, West Lafayette, 2016.

[16] E. Rosten and T. Drummond, "Fusing points and lines for high performance tracking," *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, October 2005, Beijing, China.

[17] M. A. Fischler and R. C. B. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the Association for Computing Machinery*, vol. 24, no. 6, pp. 381–395, June 1981. [Online]. Available: http://doi.acm.org/10.1145/358669.358692

[18] G. J. Sullivan and T. Wiegand, "Rate-distortion optimization for video compression," *IEEE Signal Processing Magazine*, vol. 15, no. 6, pp. 74–90, Nov 1998.

[19] "Cisco visual networking index: Forecast and methodology, 2016-2021, white paper," Cisco Systems Inc., Tech. Rep., 2017.

[20] D. Tse and P. Viswanath, *Fundamentals of wireless communication*. Cambridge University Press, 2005.

[21] Y. Wang and Q.-F. Zhu, "Error control and concealment for video communication: A review," *Proceedings of the IEEE*, vol. 86, no. 5, pp. 974–997, May 1998.

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

[23] I. E. Richardson, *The H.264 advanced video compression standard*, 2nd ed. San Francisco, CA: Wiley Publishing, 2010.

[24] J. Bankoski, J. Koleszar, L. Quillio, J. Salonen, P. Wilkins, and Y. Xu, "VP8 data format and decoding guide, rfc 6386," http://datatracker.ietf.org/doc/rfc6386/.

[25] M. Yang, "Multiple description video coding with adaptive error concealment," Ph.D. dissertation, Purdue University, West Lafayette, 2012.

[26] R. Zhang, "Efficient inter-layer motion compensation and error resilience for spatially scalable video coding," Ph.D. dissertation, Purdue University, West Lafayette, 2009.

[27] U. Horn, K. Stuhlmller, M. Link, and B. Girod, "Robust Internet video transmission based on scalable coding and unequal error protection," *Image Communication*, vol. 15, pp. 77–94, September 1999.

[28] V. Dang, A. Mansouri, and J. Konrad, "Motion estimation for region-based video coding," *Proceedings of International Conference on Image Processing*, vol. 2, pp. 189–192, October 1995, Washington D.C.

[29] T. E. Slowe and I. Marsic, "Saliency-based visual representation for compression," *Proceedings of International Conference on Image Processing*, vol. 2, pp. 554–557, October 1997, Santa Barbara, CA.

[30] A. Vetro, T. Haga, K. Sumi, and H. Sun, "Object-based coding for long-term archive of surveillance video," *Proceedings of International Conference on Multimedia and Expo*, vol. 2, p. 417, July 2003, Baltimore, MD.

[31] T. Nishi and H. Fujiyoshi, "Object-based video coding using pixel state analysis," *Proceedings of the 17th International Conference on Pattern Recognition*, vol. 3, pp. 306–309, August 2004, Cambridge, UK.

[32] D. Grois, E. Kaminsky, and O. Hadar, "Adaptive bit-rate control for region-of-interest scalable video coding," *Proceedings of IEEE 26th Convention of Electrical and Electronics Engineers*, pp. 761–765, November 2010, Eilat, Israel.

[33] H. Hu, B. Li, W. Lin, W. Li, and M. Sun, "Region-based rate control for H.264/AVC for low bit-rate applications," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, no. 11, pp. 1564–1576, November 2012.

[34] M. Meddeb, M. Cagnazzo, and B. Pesquet-Popescu, "Region-of-interest-based rate control scheme for high efficiency video coding," *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 7338–7342, May 2014, Florence, Italy.

[35] S. Kattula and M. Dasari, "On the adaptive motion estimation in video coding based on video content analysis," *Proceedings of International Conference on Advances in Computing, Communications and Informatics*, pp. 132–136, September 2016, Jaipur, Indian.

[36] G. Qiang, L. Yue, and F. Yu, "An region of interest based video compression for indoor surveillance," *Proceedings of 2nd International Conference on Information Technology and Electronic Commerce*, pp. 157–160, December 2014, Dalian, China.

[37] X. K. Yang, W. S. Lin, Z. K. Lu, X. Lin, S. Rahardja, E. P. Ong, and S. S. Yao, "Local visual perceptual clues and its use in videophone rate control," *Proceedings of IEEE International Symposium on Circuits and Systems*, vol. 3, pp. III–805, May 2004, Vancouver, Canada.

[38] H. Wang and K. El-Maleh, "Joint adaptive background skipping and weighted bit allocation for wireless video telephony," *Proceedings of International Conference on Wireless Networks, Communications and Mobile Computing*, vol. 2, pp. 1243–1248, June 2005, Wuhan, China.

[39] P. Ndjiki-Nya, D. Doshkov, H. Kaprykowsky, F. Zhang, D. Bull, and T. Wiegand, "Perception-oriented video coding based on image analysis and completion: A review," *Signal Processing: Image Communication*, vol. 27, no. 6, pp. 579–594, 2012.

[40] P. Ndjiki-Nya, T. Hinz, and T. Wiegand, "Generic and robust video coding with texture analysis and synthesis," *Proceedings of IEEE International Conference on Multimedia and Expo*, pp. 1447–1450, July 2007, Beijing, China.

[41] P. Ndjiki-Nya, T. Hinz, A. Smolic, and T. Wiegand, "A generic and automatic content-based approach for improved H.264/MPEG4-AVC video coding," *Proceedings of IEEE International Conference on Image Processing*, vol. 2, p. 874, September 2005, Genoa, Italy.

[42] P. Ndjiki-Nya, T. Hinz, C. Stuber, and T. Wiegand, "A content-based video coding approach for rigid and non-rigid textures," *Proceedings of International Conference on Image Processing*, pp. 3169–3172, October 2006, Atlanta, Georgia.

[43] M. Bosch, F. Zhu, and E. J. Delp, "Segmentation-based video compression using texture and motion models," *IEEE Journal of Selected Topics in Signal Processing*, vol. 5, no. 7, pp. 1366–1377, 2011.

[44] F. Zhang and D. R. Bull, "Enhanced video compression with region-based texture models," *Proceedings of 28th Picture Coding Symposium*, pp. 54–57, December 2010, Nagoya, Japan.

[45] F. Zhang and D. R. Bull, "A parametric framework for video compression using region-based texture models," *IEEE Journal of Selected Topics in Signal Processing*, vol. 5, no. 7, pp. 1378–1392, November 2011.

[46] U. S. Thakur and O. Chubach, "Texture analysis and synthesis using steerable pyramid decomposition for video coding," *Proceedings of International Conference on Systems, Signals and Image Processing*, pp. 204–207, September 2015, London, UK.

[47] J. Portilla and E. P. Simoncelli, "A parametric texture model based on joint statistics of complex wavelet coefficients," *International Journal of Computer Vision*, vol. 40, no. 1, pp. 49–70, 2000.

[48] A. Dumitras and B. G. Haskell, "An encoder-decoder texture replacement method with application to content-based movie coding," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 14, no. 6, pp. 825–840, June 2004.

[49] S. Bansal, S. Chaudhury, and B. Lall, "Dynamic texture synthesis for video compression," *Proceeding of National Conference on Communications*, pp. 1–5, Feb 2013, New Delhi, India.

[50] S. Ma, X. Zhang, C. Jia, Z. Zhao, S. Wang, and S. Wanga, "Image and video compression with neural networks: A review," *IEEE Transactions on Circuits and Systems for Video Technology*, 2019.

[51] D. Liu, Y. Li, J. Lin, H. Li, and F. Wu, "Deep learning-based video coding: A review and a case study," *arXiv preprint arXiv:1904.12462*, 2019.

[52] W. Cui, T. Zhang, S. Zhang, F. Jiang, W. Zuo, and D. Zhao, "Convolutional neural networks based intra prediction for HEVC," *arXiv preprint arXiv:1808.05734*, 2018.

[53] J. Li, B. Li, J. Xu, R. Xiong, and W. Gao, "Fully connected network-based intra prediction for image coding," *IEEE Transactions on Image Processing*, vol. 27, no. 7, pp. 3236–3247, 2018.

[54] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification," *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1026–1034, 2015, Santiago, Chile.

[55] K. Sueh, "HEVC test model, 16.9," 2016. [Online]. Available: https://hevc.hhi.fraunhofer.de/

[56] J. Pfaff, P. Helle, D. Maniry, S. Kaltenstadler, W. Samek, H. Schwarz, D. Marpe, and T. Wiegand, "Neural network based intra prediction for video coding," *Applications of Digital Image Processing XLI*, vol. 10752, p. 1075213, 2018.

[57] Y. Hu, W. Yang, M. Li, and J. Liu, "Progressive spatial recurrent neural network for intra prediction," *IEEE Transactions on Multimedia*, vol. 21, no. 12, pp. 3024–3037, 2019.

[58] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," *Advances in Neural Information Processing Systems*, pp. 2672–2680, 2014, Montreal, Canada.

[59] Z. Jin, P. An, and L. Shen, "Video intra prediction using convolutional encoder decoder network," *Neurocomputing*, 2019.

[60] Y. Seki, Y. Shishikui, and S. Iwamura, "Two-stage neural network for intra prediction mode decision," *Proceedings of the International Workshop on Advanced Image Technology*, vol. 11049, p. 1104924, 2019, Singapore.

[61] Y. Li, D. Liu, H. Li, L. Li, F. Wu, H. Zhang, and H. Yang, "Convolutional neural network-based block up-sampling for intra frame coding," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 28, no. 9, pp. 2316–2330, 2018.

[62] N. Yan, D. Liu, H. Li, and F. Wu, "A convolutional neural network approach for half-pel interpolation in video coding," *Proceedings of the IEEE International Symposium on Circuits and Systems*, pp. 1–4, 2017, Baltimore, MD.

[63] H. Zhang, L. Song, Z. Luo, and X. Yang, "Learning a convolutional neural network for fractional interpolation in HEVC inter coding," *Proceedings of the IEEE Conference on Visual Communications and Image Processing*, pp. 1–4, 2017, St. Petersburg, FL.

[64] J. Liu, S. Xia, W. Yang, M. Li, and D. Liu, "One-for-all: Grouped variation network-based fractional interpolation in video coding," *IEEE Transactions on Image Processing*, vol. 28, no. 5, pp. 2140–2151, 2018.

[65] L. Zhao, S. Wang, X. Zhang, S. Wang, S. Ma, and W. Gao, "Enhanced motion-compensated video coding with deep virtual reference frame generation," *IEEE Transactions on Image Processing*, vol. 28, no. 10, pp. 4832–4844, 2019.

[66] S. Xia, W. Yang, Y. Hu, and J. Liu, "Deep inter prediction via pixel-wise motion oriented reference generation," *Proceedings of the IEEE International Conference on Image Processing*, pp. 1710–1774, 2019, Taipei, Taiwan.

[67] S. Niklaus, L. Mai, and F. Liu, "Video frame interpolation via adaptive separable convolution," *Proceedings of the IEEE International Conference on Computer Vision*, pp. 261–270, 2017, Venice, Italy.

[68] S. Xia, Y. Hu, and J. Liu, "Deep integer-position samples refinement for motion compensation of video coding," *Proceedings of the International Forum on Digital TV and Wireless Multimedia Communications*, pp. 391–400, 2018, Shanghai, China.

[69] S. Huo, D. Liu, F. Wu, and H. Li, "Convolutional neural network-based motion compensation refinement for video coding," *Proceedings of the IEEE International Symposium on Circuits and Systems*, pp. 1–4, 2018, Florence, Italy.

[70] Y. Dai, D. Liu, and F. Wu, "A convolutional neural network approach for post-processing in HEVC intra coding," *Proceedings of the International Conference on Multimedia Modeling*, pp. 28–39, 2017, Reykjavik, Iceland.

[71] M. M. Alam, T. D. Nguyen, M. T. Hagan, and D. M. Chandler, "A perceptual quantization strategy for HEVC based on a convolutional neural network trained on natural images," *Applications of Digital Image Processing XXXVIII*, vol. 9599, p. 959918, 2015.

[72] R. Song, D. Liu, H. Li, and F. Wu, "Neural network-based arithmetic coding of intra prediction modes in HEVC," *Proceedings of the IEEE Visual Communications and Image Processing*, pp. 1–4, 2017, St. Petersburg, FL.

[73] S. Puri, S. Lasserre, and P. Le Callet, "CNN-based transform index prediction in multiple transforms framework to assist entropy coding," *Proceedings of the European Signal Processing Conference*, pp. 798–802, 2017, Kos island, Greece.

[74] C. Ma, D. Liu, X. Peng, and F. Wu, "Convolutional neural network-based arithmetic coding of dc coefficients for HEVC intra coding," *Proceedings of the IEEE International Conference on Image Processing*, pp. 1772–1776, 2018, Athens, Greece.

[75] S. Midtskogen, A. Fuldseth, and M. Zanaty, "Constrained low pass filter," *Network Working Group*, 2016.

[76] D. Mukherjee, S. Li, Y. Chen, A. Anis, S. Parker, and J. Bankoski, "A switchable loop-restoration with side-information framework for the emerging AV1 video codec," *Proceedings of the IEEE International Conference on Image Processing*, pp. 265–269, 2017, Beijing, China.

[77] D. Ding, G. Chen, D. Mukherjee, U. Joshi, and Y. Chen, "A CNN-based in-loop filtering approach for AV1 video codec," *Proceedings of the Picture Coding Symposium*, pp. 1–5, 2019, Ningbo, China.

[78] G. Chen, D. Ding, D. Mukherjee, U. Joshi, and Y. Chen, "AV1 in-loop filtering using a wide-activation structured residual network," *Proceedings of the IEEE International Conference on Image Processing*, pp. 1725–1729, 2019, Taipei, Taiwan.

[79] Y. Fan, J. Yu, and T. S. Huang, "Wide-activated deep residual networks based restoration for bpg-compressed images." *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 2621–2624, 2018, Salt Lake City, UT.

[80] C. Dong, Y. Deng, C. Change Loy, and X. Tang, "Compression artifacts reduction by a deep convolutional network," *Proceedings of the IEEE International Conference on Computer Vision*, pp. 576–584, 2015, Santiago, Chile.

[81] L. Cavigelli, P. Hager, and L. Benini, "CAS-CNN: A deep convolutional neural network for image compression artifact suppression," *Proceedings of the IEEE International Joint Conference on Neural Networks*, pp. 752–759, 2017, Anchorage, Alaska.

[82] J. Guo and H. Chao, "Building dual-domain representations for compression artifacts reduction," *Proceedings of the European Conference on Computer Vision*, pp. 628–644, 2016, Amsterdam, The Netherlands.

[83] L. Galteri, L. Seidenari, M. Bertini, and A. Del Bimbo, "Deep generative adversarial compression artifact removal," *Proceedings of the IEEE International Conference on Computer Vision*, pp. 4826–4835, 2017, Venice, Italy.

[84] P. Liu, H. Zhang, K. Zhang, L. Lin, and W. Zuo, "Multi-level wavelet-cnn for image restoration," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 773–782, 2018, Salt Lake City, UT.

[85] Y. Zhang, L. Sun, C. Yan, X. Ji, and Q. Dai, "Adaptive residual networks for high-quality image restoration," *IEEE Transactions on Image Processing*, vol. 27, no. 7, pp. 3150–3163, 2018.

[86] T. Wang, M. Chen, and H. Chao, "A novel deep learning-based method of improving coding efficiency from the decoder-end for HEVC," *Proceedings of the Data Compression Conference*, pp. 410–419, 2017, Snowbird, Utah.

[87] R. Yang, M. Xu, Z. Wang, and T. Li, "Multi-frame quality enhancement for compressed video," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 6664–6673, 2018, Salt Lake City, UT.

[88] Y.-H. Tsai, M.-Y. Liu, D. Sun, M.-H. Yang, and J. Kautz, "Learning binary residual representations for domain-specific video streaming," *Proceedings of the AAAI Conference on Artificial Intelligence*, 2018, New Orleans, Louisiana.

[89] J. Tong, X. Wu, D. Ding, Z. Zhu, and Z. Liu, "Learning-based multi-frame video quality enhancement," *Proceedings of the IEEE International Conference on Image Processing*, pp. 929–933, 2019, Taipei, Taiwan.

[90] G. Toderici, D. Vincent, N. Johnston, S. Jin Hwang, D. Minnen, J. Shor, and M. Covell, "Full resolution image compression with recurrent neural networks," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5306–5314, 2017, Honolulu, Hawaii.

[91] J. Ballé, V. Laparra, and E. P. Simoncelli, "End-to-end optimized image compression," *arXiv preprint arXiv:1611.01704*, 2016.

[92] L. Theis, W. Shi, A. Cunningham, and F. Huszár, "Lossy image compression with compressive autoencoders," *arXiv preprint arXiv:1703.00395*, 2017.

[93] T. Chen, H. Liu, Q. Shen, T. Yue, X. Cao, and Z. Ma, "Deepcoder: A deep neural network based video compression," *Proceedings of the IEEE Visual Communications and Image Processing*, pp. 1–4, 2017, St. Petersburg, FL.

[94] Z. Chen, T. He, X. Jin, and F. Wu, "Learning for video compression," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 30, no. 2, pp. 566–576, Feb 2020.

[95] A. v. d. Oord, N. Kalchbrenner, and K. Kavukcuoglu, "Pixel recurrent neural networks," *arXiv preprint arXiv:1601.06759*, 2016.

[96] G. Lu, W. Ouyang, D. Xu, X. Zhang, C. Cai, and Z. Gao, "DVC: An end-to-end deep video compression framework," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 11 006–11 015, 2019, Long Beach, CA.

[97] O. Rippel, S. Nair, C. Lew, S. Branson, A. G. Anderson, and L. Bourdev, "Learned video compression," *Proceedings of the IEEE International Conference on Computer Vision*, pp. 3454–3463, 2019, South Korea.

[98] C.-Y. Wu, N. Singhal, and P. Krahenbuhl, "Video compression through image interpolation," *Proceedings of the European Conference on Computer Vision*, pp. 416–431, 2018, Munich, Germany.

[99] J. Han, S. Lombardo, C. Schroers, and S. Mandt, "Deep probabilistic video compression," *arXiv preprint arXiv:1810.02845*, 2018.

[100] S. Lombardo, J. HAN, C. Schroers, and S. Mandt, "Deep generative video compression," *Advances in Neural Information Processing Systems*, pp. 9283–9294, 2019.

[101] C. Vondrick, H. Pirsiavash, and A. Torralba, "Generating videos with scene dynamics," *Proceedings of Advances in Neural Information Processing Systems*, pp. 613–621, December 2016, Barcelona, Spain.

[102] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein *et al.*, "ImageNet large scale visual recognition challenge," *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, 2015.

[103] T. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: Common objects in context," *Proceedings of the IEEE European Conference on Computer Vision*, pp. 740–755, September 2014, Zürich, Switzerland.

[104] B. Zhou, H. Zhao, X. Puig, S. Fidler, A. Barriuso, and A. Torralba, "Semantic understanding of scenes through the ADE20K dataset," *arXiv preprint arXiv:1608.05442*, 2016.

[105] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3431–3440, June 2015, Boston, MA.

[106] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, "Pyramid scene parsing network," *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pp. 2881–2890, July 2017, Honolulu, HI.

[107] A. K. Jain and F. Farrokhnia, "Unsupervised texture segmentation using gabor filters," *Proceedings of the International Conference on Systems, Man, and Cybernetics Conference proceedings*, pp. 14–19, 1990, Los Angeles, CA.

[108] A. C. Bovik, M. Clark, and W. S. Geisler, "Multichannel texture analysis using localized spatial filters," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 12, no. 1, pp. 55–73, 1990.

[109] G. R. Cross and A. K. Jain, "Markov random field texture models," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, no. 1, pp. 25–39, 1983.

[110] R. Chellappa and S. Chatterjee, "Classification of textures using Gaussian Markov random fields," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 33, no. 4, pp. 959–963, 1985.

[111] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.

[112] H. Bay, T. Tuytelaars, and L. Van Gool, "Surf: Speeded up robust features," *Proceedings of the European Conference on Computer Vision*, pp. 404–417, 2006, Graz, Austria.

[113] T. Ojala, M. Pietikainen, and T. Maenpaa, "Multiresolution gray-scale and rotation invariant texture classification with local binary patterns," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 7, pp. 971–987, 2002.

[114] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Advances in Neural Information Processing Systems*, pp. 1097–1105, 2012.

[115] M. Cimpoi, S. Maji, and A. Vedaldi, "Deep filter banks for texture recognition and segmentation," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3828–3836, 2015, Boston, Massachusetts.

[116] F. Perronnin, J. Sánchez, and T. Mensink, "Improving the fisher kernel for large-scale image classification," *Proceedings of the European Conference on Computer Vision*, pp. 143–156, 2010, Crete, Greece.

[117] A. A. Efros and T. K. Leung, "Texture synthesis by non-parametric sampling," *Proceedings of the Seventh IEEE International Conference on Computer Vision*, vol. 2, pp. 1033–1038, 1999, Kerkyra, Greece.

[118] L.-Y. Wei and M. Levoy, "Fast texture synthesis using tree-structured vector quantization," *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques*, pp. 479–488, 2000, New Orleans, LA.

[119] M. Ashikhmin, "Synthesizing natural textures," *Proceedings of the Symposium on Interactive 3D Graphics*, pp. 217–226, 2001, New York, NY.

[120] H. Derin and H. Elliott, "Modeling and segmentation of noisy and textured images using Gibbs random fields," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, no. 1, pp. 39–55, 1987.

[121] D. J. Heeger and J. R. Bergen, "Pyramid-based texture analysis/synthesis," *Proceedings of the 22nd annual Conference on Computer Graphics and Interactive Techniques*, pp. 229–238, 1995, Los Angeles, CA.

[122] L. Gatys, A. S. Ecker, and M. Bethge, "Texture synthesis using convolutional neural networks," *Advances in neural information processing systems*, pp. 262–270, 2015.

[123] K. Gregor, I. Danihelka, A. Graves, D. J. Rezende, and D. Wierstra, "Draw: A recurrent neural network for image generation," *arXiv preprint arXiv:1502.04623*, 2015.

[124] C. Li and M. Wand, "Precomputed real-time texture synthesis with markovian generative adversarial networks," *Proceedings of the European Conference on Computer Vision*, pp. 702–716, 2016, Amsterdam, The Netherlands.

[125] J. Zhang, J. Arnold, and M. Frater, "A cell-loss concealment technique for MPEG-2 coded video," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 10, no. 4, pp. 659–665, June 2000.

[126] W. Lam, A. Reibman, and B. Liu, "Recovery of lost or erroneously received motion vectors," *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 5, no. 12, pp. 417–420, April 1993, Minneapolis, MN.

[127] S. Belfiore, M. Grangetto, E. Magli, and G. Olmo, "An error concealment algorithm for streaming video," *Proceedings of the IEEE International Conference on Image Processing*, vol. 3, pp. 649–652, September 2003, Barcelona, Spain.

[128] Q. Peng, T. Yang, and C. Zhu, "Block-based temporal error concealment for video packet using motion vector extrapolation," *Proceedings of the IEEE International Conference on Communications, Circuits and Systems and West Sino Expositions*, vol. 1, pp. 10–14, June 2002, Chengdu, China.

[129] B. Yan and H. Gharavi, "A hybrid frame concealment algorithm for H.264/AVC," *IEEE Transactions on Image Processing*, vol. 19, no. 1, pp. 98–107, January 2010.

[130] M. Dissanayake, C. Hewage, S. Worrall, W. Fernando, and A. Kondoz, "Redundant motion vectors for improved error resilience in H.264/AVC coded video," *Proceedings of the IEEE International Conference on Multimedia and Expo*, no. 7, pp. 25–28, June 2008, Hannover, Germany.

[131] X. Fan, O. Au, D. Zhao, and W. Gao, "Joint forward backward error concealment of redundantly coded video," *Proceedings of the IEEE International Packet Video Workshop*, no. 7, pp. 25–32, December 2010, Hong Kong.

[132] H. Schwarz, D. Marpe, and T. Wiegand, "Overview of the scalable video coding extension of the H.264/AVC standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 17, no. 9, pp. 1103–1120, September 2007.

[133] N. Gadgil, M. Yang, M. L. Comer, and E. J. Delp, *E-Signal processing*. Oxford, UK: Elsevier, vol. 4, ch. Multiple Description Coding, p. to appear.

[134] T. Stockhammer, M. Hannuksela, and T. Wiegand, "H.264/AVC in wireless environments," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, no. 7, pp. 657–673, July 2003.

[135] N. Gadgil and E. Delp, "VPx error resilient video coding using duplicated prediction information," *Proceedings of the IS&T Electronic Imaging: Conference on Visual Information Processing and Communication VII*, February 2016, San Francisco, CA.

[136] http://www.webmproject.org/.

[137] "VP8 Encoder Parameter Guide," URL: http://www.webmproject.org/docs/encoder-parameters/, Last accessed: 05/27/2016.

[138] M. Yang, M. Comer, and E. Delp, "A four-description MDC for high loss-rate channels," *Proceedings of the Picture Coding Symposium*, pp. 418–421, December 2010, Nagoya, Japan.

[139] P. Seeling, F. Fitzek, and M. Reisslein, "Incorporating transmission errors into simulations using video traces," in *Video Traces for Network Perforamance Evaluation*. Dordrecht, Netherlands: Springer, 2007, pp. 195–228.

[140] Z. Liu, D. Mukherjee, W. Lin, P. Wilkins, J. Han, Y. Xu, and J. Bankoski, "Adaptive multireference prediction using a symmetric framework," *Proceedings of the IS&T International Symposium on Electronic Imaging, Visual Information Processing and Communication VIII*, pp. 65–72(8), January 2017, Burlingame, CA.

[141] S. C. Hsia, "An adaptive video coding control scheme for real-time MPEG applications," *EURASIP Journal on Advances in Signal Processing*, vol. 2003, no. 3, p. 161846, Mar 2003. [Online]. Available: https://doi.org/10.1155/S1110865703210040

[142] B. Zatt, M. S. Porto, J. Scharcanski, and S. Bampi, "Gop structure adaptive to the video content for efficient h.264/avc encoding," *Proceedings of the International Conference on Image Processing*, pp. 3053–3056, September 2010.

[143] 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*, vol. 13, no. 4, pp. 600–612, April 2004.

[144] Z. Chen, W. Lin, and K. N. Ngan, "Perceptual video coding: challenges and approaches," *Proceedings of IEEE International Conference on Multimedia and Expo*, pp. 784–789, July 2010, Singapore.

[145] D. Doshkov and P. Ndjiki-Nya, "Chapter 6 - how to use texture analysis and synthesis methods for video compression," in *Academic Press Library in signal Processing*, ser. Academic Press Library in Signal Processing, S. Theodoridis and R. Chellappa, Eds. Oxford, UK: Elsevier, 2014, vol. 5, pp. 197–225.

[146] J. Balle, A. Stojanovic, and J. Ohm, "Models for static and dynamic texture synthesis in image and video compression," *IEEE Journal of Selected Topics in Signal Processing*, vol. 5, no. 7, pp. 1353–1365, Nov 2011.

[147] K. Naser, V. Ricordel, and P. L. Callet, "Local texture synthesis: A static texture coding algorithm fully compatible with HEVC," *2015 International Conference on Systems, Signals and Image Processing (IWSSIP)*, pp. 37–40, Sept 2015.

[148] F. Zhang and D. R. Bull, "A parametric framework for video compression using region-based texture models," *IEEE Journal of Selected Topics in Signal Processing*, vol. 5, no. 7, pp. 1378–1392, November 2011.

[149] 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*, vol. 13, no. 4, pp. 600–612, April 2004.

[150] Z. Wang, E. P. Simoncelli, and A. C. Bovik, "Multiscale structural similarity for image quality assessment," *Proceedings of The Thrity-Seventh Asilomar Conference on Signals, Systems Computers*, vol. 2, pp. 1398–1402, Nov 2003, pacific Grove, CA.

[151] Zhou Wang and E. P. Simoncelli, "Translation insensitive image similarity in complex wavelet domain," *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 2, pp. 573–576, March 2005, Philadelphia, PA.

[152] A. C. Brooks, X. Zhao, and T. N. Pappas, "Structural similarity quality metrics in a coding context: exploring the space of realistic distortions," *IEEE Transactions on Image Processing*, vol. 17, no. 8, pp. 1261–1273, August 2008.

[153] S. Varadarajan and L. J. Karam, "A reduced-reference perceptual quality metric for texture synthesis," *Proceedings of IEEE International Conference on Image Processing*, pp. 531–535, October 2014, Paris, France.

[154] X. Zhao, M. G. Reyes, T. N. Pappas, and D. L. Neuhoff, "Structural texture similarity metrics for retrieval applications," *Proceedings of 15th IEEE International Conference on Image Processing*, pp. 1196–1199, October 2008, San Diego, CA.

[155] J. Zujovic, T. N. Pappas, and D. L. Neuhoff, "Structural similarity metrics for texture analysis and retrieval," *Proceedings of 16th IEEE International Conference on Image Processing*, pp. 2225–2228, Novebmer 2009, Cairo, Egypt.

[156] Netflix, Inc., "VMAF: Perceptual video quality assessment based on multimethod fusion," https://github.com/Netflix/vmaf, 2017.

[157] F. Zhang, F. M. Moss, R. Baddeley, and D. R. Bull, "BVI-HD: A video quality database for HEVC compressed and texture synthesized content," *IEEE Transactions on Multimedia*, vol. 20, no. 10, pp. 2620–2630, Oct 2018.

[158] Y. Wang, S. Inguva, and B. Adsumilli, "YouTube UGC dataset for video compression research," *IEEE International Workshop on Multimedia Signal Processing*, September 2019, Kuala Lumpur, Malaysia.

[159] D. Chen, C. Fu, and F. Zhu, "AV1 video coding using texture analysis with convolutional neural networks," *ArXiv e-prints*, April 2018.

[160] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint*, p. arXiv:1409.1556, 2014.

[161] B. Zhou, A. Khosla, A. Lapedriza, A. Torralba, and A. Oliva, "Places: An image database for deep scene understanding," *arXiv preprint*, p. arXiv:1610.02055, 2016.

[162] R. Collobert, K. Kavukcuoglu, and C. Farabet, "Torch7: A matlab-like environment for machine learning," *Proceedings of the BigLearn workshop at the Neural Information Processing Systems*, pp. 1–6, December 2011, Granada, Spain.

[163] C. Chen, J. Luo, and K. J. Parker, "Image segmentation via adaptive k-mean clustering and knowledge-based morphological operations with biomedical applications," *IEEE transactions on image processing*, vol. 7, no. 12, pp. 1673–1683, December 1998.

[164] D. Chen, Q. Chen, and F. Zhu, "Pixel-level texture segmentation based AV1 video compression," *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 1622–1626, May 2019, Brighton, UK.

[165] B. Zhou, H. Zhao, X. Puig, S. Fidler, A. Barriuso, and A. Torralba, "Scene parsing through ADE20k dataset," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, vol. 1, no. 2, p. 4, July 2017, Honolulu, HI.

[166] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778, June 2016, Las Vegas, NV.

[167] L. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Semantic image segmentation with deep convolutional nets and fully connected crfs," *arXiv preprint arXiv:1412.7062*, 2014.

[168] F. Yu and V. Koltun, "Multi-scale context aggregation by dilated convolutions," *arXiv preprint arXiv:1511.07122*, 2015.

[169] T. N. Pappas, J. Zujovic, and D. L. Neuhoff, "Image analysis and compression: renewed focus on texture," *Visual Information Processing and Communication*, vol. 7543, pp. 178 – 189, 2010. [Online]. Available: https://doi.org/10.1117/12.851682

[170] M. Bosch, F. Zhu, and E. J. Delp, "Spatial texture models for video compression," *Proceedings of IEEE International Conference on Image Processing*, vol. 1, pp. 93–96, September 2007, San Antonio, TX.

[171] C. Fu, D. Chen, Z. Liu, E. J. Delp, and F. Zhu, "Texture segmentation based video compression using convolutional neural networks," *Proceedings of Electronic Imaging*, February 2018, Burlingame, CA, USA.

[172] M. Haindl and S. Mike, "Texture segmentation benchmark," *Proceedings of the 19th International Conference on Pattern Recognition*, pp. 1–4, December 2008, Tampa, FL.

[173] N. Xu, L. Yang, Y. Fan, D. Yue, Y. Liang, J. Yang, and T. Huang, "YouTube-VOS: A large-scale video object segmentation benchmark," *arXiv preprint*, p. arXiv:1809.03327, 2018.

[174] F. Perazzi, J. Pont-Tuset, B. McWilliams, L. V. Gool, M. Gross, and A. Sorkine-Hornung, "A benchmark dataset and evaluation methodology for video object segmentation," *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pp. 724–732, June 2016, Las Vegas, NV.

[175] M. A. Papadopoulos, F. Zhang, D. Agrafiotis, and D. Bull, "A video texture database for perceptual compression and quality assessment," *Proceedings of IEEE International Conference on Image Processing*, pp. 2781–2785, September 2015, Quebec City, QC.

VITA

VITA

Di Chen was born in China. She receives the B.E. in Electrical Engineering from Huazhong University of Science and Technology, China.

Di Chen joined the graduate program at the School of Electrical, Computer and Systems Engineering of Rensselaer Polytechnic Institute, Troy, NY in August 2012. She worked under the supervision of Professor John W. Woods.

Di Chen joined the graduate program at the School of Electrical and Computer Engineering, Purdue University, West Lafayette, Indiana in August 2015. She worked at the Video and Image Processing Laboratory (VIPER) under the supervision of Professor Fengqing Zhu. She was an intern at the Chrome Media team of Google Inc., Mountain View, CA in the summer of 2017. She was an intern at the Media Algorithm team at Youtube of Google Inc., Mountain View, CA in the summer of 2018.

Di Chen is a recipient of the National First-class Scholarship awarded by the Ministry of Education of China and Cha Chi Ming & Liu Bie Ju Undergraduate Student Scholarship awarded by Qiu Shi Science & Technologies Foundation. Her research focuses on video analysis and compression.