PEOPLE'S DEMOCRATIC REPUBLIC OF ALGERIA

MINISTRY OF HIGHER EDUCATION

AND SCIENTIFIC RESEARCH

YAHIA FARES UNIVERSITY OF MEDEA

Faculty of **T**echnology

Department of **E**lectrical **E**ngineering

Master's thesis

Field of study: Telecommunications

Speciality: Telecommunications systems

Study and Implementation of Inter Prediction in the latest video compression standard VVC/H.266

Presented by:

Boudali Ahmed Amine

Daoudi Ihab Abdennour

Proposed and Directed by:

Dr. Menasri Wahiba

Academic year 2023/2024

Thanks

We would like to thank "Allah" the Almighty for giving us the courage, strength and will to complete this dissertation.

Presenting this work, we offer to express my gratitude to the people who helped and advised us in invaluable ways during the preparation of this dissertation.

We would like to express our highest esteem to Miss. MENASRI Wahiba my promoter for the assistance, encouragement and guidance that she gave.

We would like to thank all the people who directly or indirectly contributed to the production of this dissertation.

We hope that this memory will be useful for all future generations.

Dedication

To my happiness to the one who watered me with tenderness and hope at the source of love "my mother"
To my support in my life who taught me how to support me and directed me towards glory "my father"
To my dear sisters.
To everyone in my extended family
To all my friends who I knew.
To my fellow students in my class of 2024.
And to everyone helped me prepare this work.

Ihab.

Dedication

This project is dedicated to my parents, whose unwavering support and encouragement have been my guiding light throughout this journey.

To my professors and mentors, thank you for your invaluable guidance and expertise that have shaped my academic and professional growth.

To my friends and loved ones, your patience, understanding, and words of encouragement have kept me going during the challenging times.

Lastly, to all the people who got inspired with this project, may this project empower you to achieve even greater success and realizing your dreams.

Ahmed

Abstract

Inter prediction in Versatile Video Coding (VVC) enhances video compression efficiency by predicting frames based on temporal redundancy, leveraging previously encoded frames to forecast future frames. This process uses motion estimation and compensation techniques to model the motion of objects across consecutive frames. VVC introduces advanced tools like affine motion compensation, sub-block motion refinement, and merged motion prediction, improving accuracy and reducing bitrate. These innovations enable higher compression ratios without significant loss of quality, optimizing storage and transmission of high-resolution video content. The problematic is how to implement the inter prediction in software. In our work we were given the implementation software of inter prediction derivation mode under matlab and python environments and we had the same results in the majority of cases.

Key words: Inter prediction, Versatile Video Coding (VVC), video compression, motion estimation, motion compensation, motion vectors, high resolution.

Résumé

L’inter prédiction dans Codage vidéo polyvalent (VVC) améliore l'efficacité de la compression vidéo en prédisant les images sur la base de la redondance temporelle, en exploitant les images précédemment codées pour prévoir les images futures. Ce processus utilise des techniques d'estimation et de compensation de mouvement pour modéliser le mouvement des objets sur des images consécutives. VVC introduit des outils avancés tels que la compensation de mouvement affine, l'affinement du mouvement des sous-blocs et la prédiction de mouvement fusionnée, améliorant ainsi la précision et réduisant le débit binaire. Ces innovations permettent des taux de compression plus élevés sans perte significative de qualité, optimisant ainsi le stockage et la transmission du contenu vidéo haute résolution. La problématique est de savoir comment implémenter l’inter-prédiction dans un logiciel. Dans notre travail, nous avons reçu le logiciel d'implémentation du mode de

dérivation inter-prédiction sous les environnements matlab et python et nous avons eu les mêmes résultats dans la majorité des cas.

Mots clés : Inter-prédiction, Codage vidéo polyvalent (VVC), compression vidéo, estimation de mouvement, compensation de mouvement, vecteurs de mouvement, haute résolution.

ملخص:

يعمل التنبؤ الداخلي في تشفير الفيديو متعدد الاستخدامات (VVC) على تحسين كفاءة ضغط الفيديو من خلال التنبؤ بالإطارات بناءً على التكرار الزمني، والاستفادة من الإطارات المشفرة مسبقًا للتنبؤ بالإطارات المستقبلية. تستخدم هذه العملية تقنيات تقدير الحركة والتعويض لنمذجة حركة الكائنات عبر إطارات متتالية. يقدم VVC أدوات متقدمة مثل تعويض الحركة المتقاربة، وتحسين حركة الكتلة الفرعية، والتنبؤ بالحركة المدمجة، مما يؤدي إلى تحسين الدقة وتقليل معدل البت. تتيح هذه الابتكارات نسب ضغط أعلى دون فقدان كبير للجودة، وتحسين تخزين ونقل محتوى الفيديو عالي الدقة. تكمن المشكلة في كيفية تنفيذ التنبؤ الداخلي في البرنامج. في عملنا، حصلنا على برنامج تنفيذ لوضع اشتقاق التنبؤ المشترك في بيئات matlab وpython وحصلنا على نفس النتائج في معظم الحالات.

الكلمات المفتاحية: التنبؤ البيني، ترميز الفيديو متعدد الاستخدامات (VVC)، ضغط الفيديو، تقدير الحركة، تعويض الحركة، ناقلات الحركة، الدقة العالية.

List of acronyms

A

AVC: Advanced video coding
AMVP: Motion Vector Predictor Algorithm
AMVR: Adaptive Motion Vector Resolution

B

B: Bytes
BAC: Binary Arithmetic Coding

C

CABAC: Context Adavptive Binary Arithmetic Coding
CAVLC: Context Adaptive Variable Length Coding
CNNs: Convolutional neural networks
CSM: Context Selection Modling
CTU: Coding Tree Unit
CU: Coding Unit

D

DCT: Discrete Cosine Transform
DST: Discrete Sine Transform

F

FLV: Flash Video

G

GB: Gigabyte
GIF: Graphics Interchange Formats
GPM: Geometric Partitioning mode

H

HD: High Definition
HDR: High Dynamic Range
HEVC: High Efficiency Video Coding

I

IEC: International Electrotechnical Commission
ISE: Integrated Software Environment
ISO: International Organization for Standardization
ITU-T: International Union of Telecommunication

J

JCT-VC: Joint Collaborative Team on Video Coding
JPEG: Joint Photographic Experts Group
JVET: Joint Video Experts Team

K

KB: Kilobytes

M

MAD: Mean Absolute Difference

ME: Motion Estimation
MPEG: Moving Picture Experts Group
MRL: multiple reference line
MTS: Multiple Transform Selection
MTT: Multi-Type Tree
MV: Motion Vectors
MVs: Several Motion Vectors
MC: Motion Compensation

P

PNG: Portable Network Graphics
POC: Picture Order Count

Q

QP: Quantization Parameter

Qstep: No Quantize

R

RD: rate-distortion
RDO: Rate Distortion Optimization
RGB: Red Green Blue
Re: Error Rate

S

SMVD: Symmetric Motion Vector Difference
SIF: Switchable Interpolation Filter

T

TB: Terabytes
TB: Transform Block
TMVP: Temporal Motion Vector Predictor

V

VVC: Versatile Video Coding
VCEG: Video Coding Experts Group

List of Figures

Figure I. 1: How an analog-to-digital conversion work. 4
Figure I. 2: Schematic screen structure. 5
Figure I. 3: Decomposition of the image into RGB components. 5
Figure I. 4: Decomposition of the image into YCbCr components. 6
Figure I. 5: Block diagram of the VVC/H.266 video codec. 11
Figure I. 6: Concept of selecting 2D separable transforms in VVC. 12
Figure I. 7: Intra and inter prediction of a PU (Prediction Unit) block in a video sequence. 13
Figure I. 8: 65 intra-prediction directions used for VVC. 14
Figure I. 9: Simplified example of Advanced Motion Vector Prediction. 14
Figure II. 1: Illustration of the principle of the corresponding macroblock and its Motion vector (a: reference frame, b: cible frame). 20
Figure II. 2: Explanation of 2D logarithmic search. 22
Figure II. 3: Example of the prediction process of GPM. 26

Figure III. 1: Software tools used in our project. 27
Figure III. 2: Derivation process for block vector components for IBC blocks (general). 30
Figure III. 3: Implementation of derivation process for block vector components for IBC blocks (general) under matlab. 31
Figure III. 4: Simulation result of derivation process for block vector components for IBC blocks (general). 31
Figure III. 5: Flowchart of Picture reconstruction process. 34
Figure III. 6: Implementation and simulation result of picture reconstruction process under matlab. 35
Figure III. 7: Flowchart of Derivation process for chroma block vectors. 36
Figure III. 8: Implementation and simulation result of derivation process for chroma block vectors. 37
Figure III. 9: Flowchart of Residual modification process for blocks using color space conversion. 38
Figure III. 10: Part of implementation Matlab of Residual modification process for blocks using color space conversion. 39
Figure III. 11: Simulation result of Residual modification process for blocks using color space conversion. 39
Figure III. 12: Flowchart of Decoding process for IBC blocks. 41
Figure III. 13: Part of implementation Matlab for decoding process for IBC blocks. 42
Figure III. 14: Simulation result of decoding process for IBC blocks. 42
Figure III. 15: Flowchart of Decoding process for the reconstructed signal of chroma coding blocks coded in inter prediction mode. 46
Figure III. 16: Matlab implementation and simulation result of decoding process for the reconstructed signal of chroma coding blocks coded in inter prediction mode. 47

Figure III. 17: Flowchart of decoding process for the residual signal of coding blocks coded in inter prediction mode. 51
Figure III. 18: Matlab implementation and simulation result of decoding process for the residual signal of coding blocks coded in inter prediction mode. 52
Figure III. 19: Flowchart of derivation process for quantization parameters. 57
Figure III. 20: Matlab implementation and simulation result of derivation process for quantization parameters. 58

Figure IV. 1: Implementation and simulation result of derivation process for block vector components for IBC block (general) under python. 59
Figure IV. 2: Implementation and simulation result of picture reconstruction process under python. 60
Figure IV. 3: Implementation and simulation for chroma block vectors. 60
Figure IV. 4: Implementation and simulation result of residual modification process for blocks using color space conversion under python. 61
Figure IV. 5: Implementation and simulation result of decoding process for IBC blocks under python. 62
Figure IV. 6: Implementation and simulation result of decoding process for the reconstructed signal of chroma coding blocks coded inter prediction mode. 63
Figure IV. 7: Implementation and simulation result of decoding process for the residual signal of coding blocks in inter prediction mode. 64
Figure IV. 8: Implementation and simulation result of derivation process for quantization parameters. 65
Figure IV. 9: Comparison results obtained by Matlab and python for derivation process for block vector camponents for IBC blocks (general). 66
Figure IV. 10: Comparison results obtained by Matlab and python for the reconstructed samples derivation. 66
Figure IV. 11: Comparison results obtained by Matlab and python for derivation process for chroma block vectors. 67
Figure IV. 12: Comparison results obtained by Matlab and python for residual modification derivation of blocks using color space conversion. 67
Figure IV. 13: Comparison results obtained by Matlab and python for quantization parameters derivation. 68

List of tables

TableI.1: Comparison of H.264, H.265 and H.266 standards 10

Table II. 1 Major differences of functionality between H.264 ,H.265 and H.266 decodecs.....19

Table of contents

General Introduction 1

ChapterI: General information about video compression.

I.1 Introduction 3
I.2 General information of the video 3
I.2.1 Definition of digital video 3
I.2.2 Simple Expression of Digital Video 3
I.2.3. Types of video 3
I.2.4 Formats video 4
I.2.5 Video characteristics 4
I.2.6 Definition of Color Space 5
I.2.7 Video File Size 6
I.3 General information on video compression 7
I.3.1 Definition of video compression 7
I.3.2 Compression Types 7
I.3.3 Image Compression Types 7
I.3.4 Video Compression 7
I.3.4.1 Technics of Video Compression 8
I.3.4.2 The Most Widely Used Codecs 9
I.4 Codec H.266 10
I.4.1 Definition of VVC/H.266 codec 10
I.4.2 Characteristics of the H.266 standard 11
I.4.2.1Transformation 11
I.4.2.2 Loop filtering 12
I.4.2.3 Quantization 12
I.4.2.4 CABAC 13
I.4.2.5 Prediction 13
I.5 Conclusion 15

ChapterII: Inter prediction in VVC (versatile video coding).

II.1 Introduction 16
II.2 State of the art 16
II.2.1 Analysis of H.264 Inter Prediction Mode Decision Variants 16
II.2.1.1Challenges and Motivations 16

II.2.1.2 Advantages and disadvantages of Inter Prediction in H.264 17
II.2.2 Analysis of H.265 Inter Prediction Mode Decision Variants 17
II.2.2.1 Challenges and variants 17
II.2.2.2Advantages and Disadvantages of Inter Prediction in H.265 18
II.2.3 Analysis of H.266 Inter Prediction mode decision variants 18
II.2.3.1Challenges and variants 18
II.2.3.2 Advantages and Disadvantages of Inter Prediction in H.266 19
II.2.4 Major Differences of performances between H.266, H.265 and H.264 standards 19
II.3 Inter prediction of VVC standard 20
II.3.1 Inter prediction definition 20
II.3.2 Compression based on the motion compensation 20
II.3.2.1 Search for motion vectors 21
II.3.3 Motion Vector Coding in VVC 22
II.3.3.1 Motion Vector Predictor List Generation Algorithm 22
II.3.3.2 Symmetric Motion Vector Difference 23
II.3.3.3 Adaptive Motion Vector Resolution 24
II.3.4 Block merging in VVC 25
II.3.4.1 Block merging candidate list generation algorithm 25
II.3.4.2 Merge mode with motion vector difference 25
II.3.4.3 Geometric partitioning mode GPM 26
II.4 Conclusion 26
ChapterIII: Implementation of inter prediction algoritms in VVC under Matlab.
III.1 Introduction 27
III.2 Used tools 27
III.2.1 MATLAB 27
III.2.2 Python 28
III.3 Inter prediction derivation algorithm 28
III.3.1 Derivation process for block vector components for IBC blocks (general) .. 28
III.3.1.1 Matlab implementation and simulation result 28
II.3.2 Picture reconstruction process (general) 28
III.3.2.1 Matlab implementation and simulation result 35
III.3.3 Derivation process for chroma block vectors 35
III.3.3.1 Matlab implementation and simulation result 36
III.3.4 Residual modification process for blocks using color space conversion 37

III.3.4.1 Matlab implementation and simulation result 38
III.3.5 Decoding process for IBC blocks 40
III.3.5.1: Matlab implementation and simulation result 41
III.3.6 Decoding process for the reconstructed signal of chroma coding blocks coded in inter prediction mode 42
III.3.6.1: Matlab implementation and simulation result 46
III.3.7 Decoding process for the residual signal of coding blocks coded in inter prediction mode 47
III.3.7.1: Matlab implementation and simulation result 52
III.3.8 Derivation process for quantization parameters: 52
III.3.8.1: Matlab implementation and simulation result 57
III.3.9 Conclusion 58

ChapterIV: Implementation of inter prediction algorithms in VVC under python.

IV.1 Introduction 59
IV.2 Python implementation and simulation result 59
IV.2.1 Derivation process for block vector components for IBC block (general) 59
IV.2.2 Picture reconstruction process (general) 59
IV.2.3 Derivation process for chroma block vectors 60
IV.2.4 Residual modification process for blocks using color space conversion 61
IV.2.5 Decoding process for IBC blocks 62
IV.2.6 Decoding process for the reconstructed signal of chroma coding blocks coded inter prediction mode 62
IV.2.7 Decoding process for the residual signal of coding blocks in inter prediction mode 63
IV.2.8 Derivation process for quantization parameters 64
IV.3 Comparison and validation of the results obtained 65
IV.4 Conclusion 68
General Conclusion 69

General Introduction

General Introduction

The Area and band width of video consumption that rapidly grows has made video compression techniques that are efficient become the key to enable continuous storing and transmitting of massive visual data. Video coding standards, e.g. H.264 /AVC (Advanced video coding), H.265/ HEVC (High efficiency video coding) and VVC (Versatile video coding)/ H.266 codecs, are continuously changing to meet the challenge of high-quality video content transmitting and storage. Prediction is a crucial technique among the wide range of methods adopted in these standards and is the key foundation for achieving enhanced compression performance [1].

As the amount of video data is increasing significantly, more efficient video compression is needed to transmit and store this video data with limited available bandwidth and storage space [2]. Therefore, Joint Video Experts Team (JVET) of ITU-T and ISO standardization organizations developed Versatile Video Coding (VVC) standard in 2020. VVC provides 50% higher compression efficiency than its predecessor High Efficiency Video Coding (HEVC) standard developed in 2013 [3]. VVC is designed to encode diverse video content such as high dynamic range, 360º video and virtual reality [4].

VVC achieves higher compression efficiency than HEVC at the cost of significant increase in computational complexity. VVC encoder is 5 times and 31 times more complex than HEVC encoder under Low-Delay and All-Intra configurations, respectively [5]. The encoding time of VVC reference software encoder (VTM) is about 10 times more than the encoding time of HEVC reference software encoder (HM). Therefore, dedicated hardware implementations are needed for processing high resolution videos in real-time [6].

One of the basic tools used in video compression standards VVC is the inter prediction, which we are interested in this work. It plays a crucial role in reducing the redundancy within video frames, resulting in efficient video compression and improved coding efficiency.

Inter prediction which is also referred to as motion compensation, captures the temporal redundancy in video by using a predicted frame that is estimated from the previously decoded frames. Through identifying and exploiting the similarities between an adjacent frame, inter prediction can achieve considerable reduction in the amount of data needed to represent subsequent frames and saves a lot of bitrate without compromising on perceptual quality [7], video coding based on transform coding is the basis of today's video compression, which allows for the transmission of high-resolution video over networks with low bandwidths and the storage of large video libraries on devices with limited capacity.

Nevertheless, the effectiveness of inter prediction is also decided by the subtleties of its implementation within video coding standards besides the theories behind this method.

The implementation of video coding algorithms, especially in hardware, can present several challenges due to the complexity of the algorithms and the demanding real-time processing requirements. Here are some common problems encountered in hardware implementation and the complexity of video coding: high computational complexity, memory band width requirements, parallelism and pipelining resource constraints, error resilience and robustness, standard compliance and compatibility.

This thesis is organized into four chapters:

- The first chapter provides a general overview of video compression standards. In the first step, we give video compression technics. The second step is to describe the video compression standards. Then, we detail the H.266/VVC standard.
- In the second chapter, we talk about our object which is the inter prediction in AVC, HEVC, VVC and his advantages/ disadvantages in the three standards, then we give a small comparison between them.
- The third chapter contains a presentation of the software tools used in our work, followed by the representation of the proposed flowcharts of inter prediction modes with their simulations in Matlab.
- The fourth chapter is reserved for the simulation by python and comparison between the results of Matlab and python codes.

Finally, we summarize the conclusions and future work.

Chapter I

General information about video compression

I.1 Introduction

The domain of video processing is extremely wide and this technology is present in many fields of everyday and professional life. Compression is often call encoding as it is a more general naming than compression where encoding includes any special that data representation is needed. Like file compression, media compression aims at lowering a file's size and reducing disk space. Nevertheless, the media compression algorithms are media-specific, i.e., targeting image, audio, and video files. We will introduce the basics of video and later provide details for the H.266/VVC standard.

I.2 General information of the video

I.2.1 Definition of digital video

Digital video is audio/visual content recorded and stored in binary data and represented as a sequence of digital weaves of data instead of a continuous signal like analog information. The natural information received by human by their five senses is all alike. This thus symbolizes that it is infinitary. However, the digital photos and audio information are discrete data units that are so close to each other it takes human’s perception to distinguish them as a continuous stream of data.

Analog data, such as video recorded on tape, is transmitted as electronic signals with variable frequency or amplitude that are added to carriers of a specific frequency. For example, when you shoot video on your phone, it is converted from analog media to digital as we show in the Figure I.1 below, and stored in memory, and when you play it, it is converted from digital to analog. This allows you to stream your video over the network, store it on your hard drive, and edit and compress it [8].

I.2.2 Simple Expression of Digital Video

A digital video is like a series of pictures that play one after another to create a moving image on a screen.

I.2.3. Types of video

The video contain is the signal that allows a sequence of images to be transported from the source to a display device in electrical form. Depending on how the signal are processed, we can distinguish two modes.

A) Analog Video

It describes the analog signal as an electrical signal whose intensity varies continuously over time. The quality of the final signal in this mode is lower because the noise added to the signal during processing alters its quality.

B) Digital Video

It is a signal that carries information represented by a series of minimum or maximum values corresponding respectively to 0 and 1. One of the factors that advantage the digital signal compared to the analog signal is the ease of distinguishing the transmitted information from the noise [8].

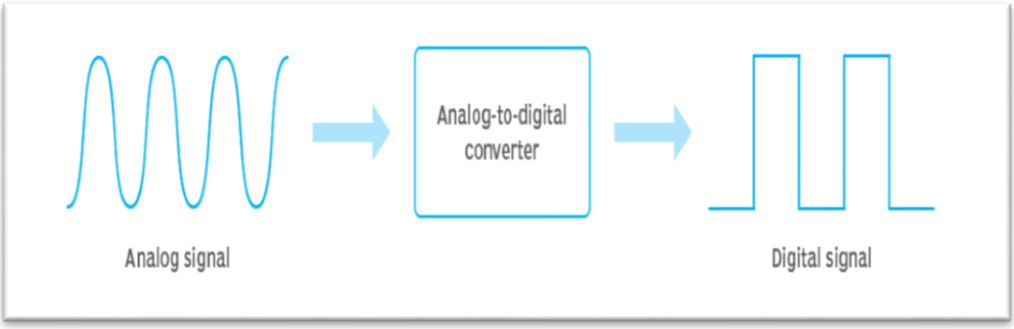


Figure I. 1: How an analog-to-digital conversion work.

I.2.4 Formats video

Video formats involve two distinct, and very different technology concepts: containers (sometimes called wrappers) & codecs (short for coder/decoder).
Codecs are used inside of a container and because of this video formats can be confusing.

There is a lot of video formats but the most common video formats are:

- AVI (Audio Video Interleave): a Windows' standard multimedia container.
- MPEG-4 Part 14 (known as .mp4): is the standardized container for MPEG-4.
- FLV (Flash Video): the format used to deliver MPEG video through Flash Player.
- MOV: Apple's QuickTime container format.
- OGG, OGM & OGV: open-standard containers.
- MKV (Mastroska): another open-specification container that you've seen if you've ever downloaded anime.
- VOB (DVD Video Object): It's DVD's standard container.
- ASF: a Microsoft format designed for WMV and WMA—files can end in .wmv or .asf [9]

I.2.5 Video characteristics

A) Pixel definition

A pixel, short for "picture element," is the smallest building block of a digital image or display. Imagine it like a tiny tile in a mosaic; many pixels come together to form the complete picture you see on your screen.
Here are some key points about pixels:

- **Size**: They are very small, usually measured in micrometers.
- **Color**: Each pixel holds information about its color and brightness, creating the overall image.
- **Resolution**: The number of pixels in a given area determines the image's resolution and sharpness. More pixels generally mean higher quality and sharper details.

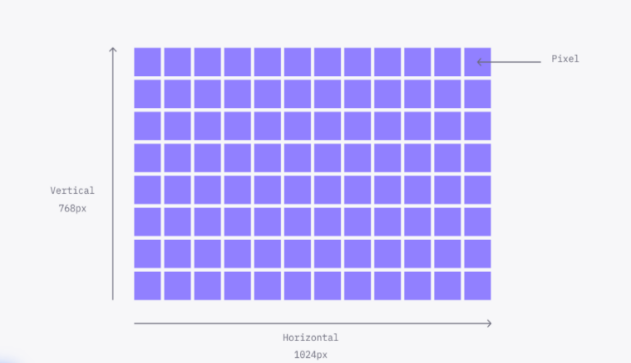


Figure I. 2: Schematic screen structure.

I.2.6 Definition of Color Space

A) RGB Color Space

A Color Space in digital videos is how colors are mapped mathematically, it is very unusual for it to be driven by two different devices in the same way, except that it allows devices to produce and display colors correctly. Visualize it as a map with actually defined values for each color known.

The classic representation of a digital image is based on a so-called RGB (Red Green Blue) color space Figure I.3.

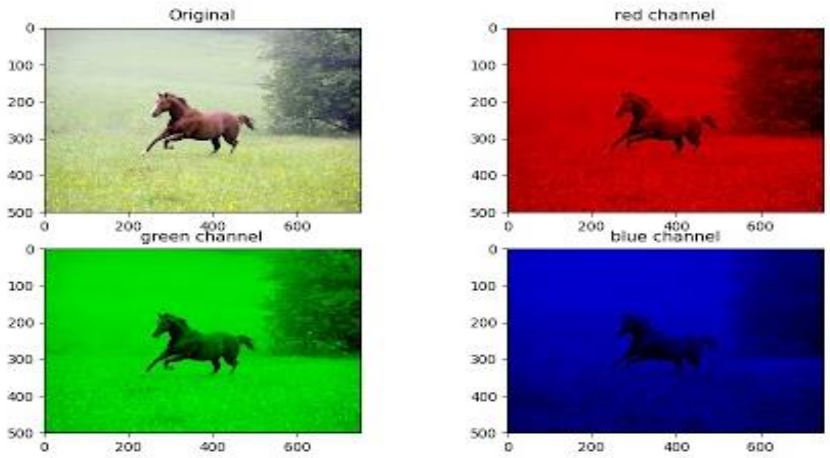


Figure I. 3: Decomposition of the image into RGB components.

Here's a breakdown of what makes up a color space:

B) YCbCr Color Space

YUV/YCbCr Isolates luma (brightness) from chroma (color) information, they efficient for storage and transmission also this space is the most widely used because it offers the advantage of separating luminous intensity and color information in the image see Figure I.4.



Figure I. 4: Decomposition of the image into YCbCr components.

I.2.7 Video File Size

Video file size is referring to the amount of storage space a video file occupies on a device, measured in bytes (B), kilobytes (KB), megabytes (MB), gigabytes (GB), or terabytes (TB). It's determined by factors like:

A) Video resolution: Higher resolution videos (e.g., 4K, 8K) have more pixels, leading to larger file sizes.
B) Video length: Longer videos naturally have more data and are larger.
C) Bitrate: This determines the amount of data used per second of video, impacting quality and size. Higher bitrates mean better quality but larger files.
D) Video codec: Different codecs compress data differently, affecting file size for similar quality.
E) Audio codec: Audio quality and format influence file size, though typically less than video data.

However, using high resolution video and high quality is always leading to larger file sizes so the scientist thought about video compression techniques to improve storage demand as well as file transfer and transfer speed. This facilitates video uploading and sharing, allowing viewers to have enjoy.

I.3 General information on video compression

I.3.1 Definition of video compression

In general, data compression involves taking a stream of symbols and transforming them into codes [10]. Compression is performed by a program that uses a formula or algorithm to determine how to reduce the size of the data to save space in the case of storage and bandwidth in the case of transmission.

Data compression is also called source coding or bitrate reduction. It is a process of modifying, encoding or converting the structure of data bits so as to consume less disk space. The goal of data compression is to represent a source of information as accurately as possible using the smallest storage space.

I.3.2 Compression Types

A compression algorithm can be categorized into two types: lossless and lossy. Lossless compression implies that all bits of data that are originally contained in the file are preserved after decompression, and the information is restored. Rumplessness takes away most of the information, especially the redundant, from a file [11]. It is different from the second one in that no one ever wants to have the source reconstructed. This lossy compression technique seeks to achieve the optimal balance between the visual quality of the reconstructed image and the compression rate accomplished.

I.3.3 Image Compression Types

Image compression is a data compression approach which uses few bits in encoding the original image. The three widely used compression formats belong to JPEG (Joint Photography Experts Group), GIF (Graphics Interchange Format) and PNG (portable network graphic).

A) JPEG compression: which is widely used in digital photos, makes use of a lossy compression algorithm that simply approximates the color and then removes the nuance of color that are imperceptible to the human eye.

B) GIF compression: reduces the color palette of a picture to maximum of 256 colors or less, which provides a good way to represent each color in the image.

C) Compression PNG : compression is based on lossless data filtering algorithm that predicts pixel colors from other frequently appearing neighboring pixels. Even though the methods have the underlying common concept of reducing the uncompressed image size, they all use different approaches.

I.3.4 Video Compression

Video compression technologies are cutting and removing repeated videos so that the digital video file can be transmitted over a network in a speedy way. Thus, the video file can be easily stored on disks. Using advanced techniques of compression, it

is possible to reach a big size reduction in file size and at the same time, deliver visually acceptable quality. However, if the file size is further reduced by solely focusing on optimizing the compression level for a particular compression technique, then the video quality may be affected. The main target of compressing digital video is to save the storage space and transmission bandwidth.

I.3.4.1 Technics of Video Compression

Video compression utilizes several key principles to reduce file size while maintaining acceptable image quality:

A) Exploiting redundancy

- Temporal Redundancy: Consecutive video frames often contain significant similarities. By predicting upcoming frames based on previous ones and encoding only the differences, redundancy is eliminated.
- Spatial Redundancy: Within a single frame, neighboring pixels often share similar colors or patterns. Techniques like transform coding analyze pixel groups and represent them more efficiently.

B) Entropy Coding

After redundancy reduction, remaining data may still have uneven information distribution. Entropy coding assigns shorter codes to more frequent symbols, further reducing file size. This includes techniques like Huffman coding and arithmetic coding.

C) Quantization

Lossy compression discards some information to achieve greater size reduction. Quantization reduces the number of possible values for each pixel or audio sample, leading to controlled quality loss that is often imperceptible to the viewer.

D) Psychovisual Optimization

Human perception is not equally sensitive to all frequencies and colors. By focusing on visually significant data and discarding less noticeable details, compression can achieve better quality at the same bitrate.

E) Adaptive Coding

Different video scenes exhibit varying complexities. Adaptive coding adjusts compression parameters based on scene content, optimizing efficiency for different regions within the video.

I.3.4.2 The Most Widely Used Codecs

H.264/AVC: The most widely used codec currently, offering good balance between compression and quality. Used in streaming (YouTube, Netflix), Blu-ray discs, and various online platforms.

H.265/HEVC: Successor to H.264, offers significant efficiency improvements with higher compression ratios for similar quality. However, it requires more processing power for encoding and decoding. Used in some streaming services and 4K content.

H.266/VVC (Versatile Video Coding) is a state-of-the-art video compression standard designed to be the successor to H.265/HEVC. It aims to significantly improve compression efficiency, offering up to 50% bitrate reduction compared to its predecessor while maintaining comparable video quality

A) H.264/AVC

The H.264 or MPEG-4 AVC, a compression format for video, developed in 2003, is used for recording and distributing full HD video and audio. It was designed and maintained by the ITU-T (ITU-T International Telecommunication Union), VCEG (Video Coding Experts Group) in partnership with the ISO (the International Organization for Standardization) and the IEC (the International Electrotechnical Commission), and MPEG (Moving Picture Experts Group).

H.264 code is broadly used for recording, compressing, and distributing video materials among many other things. It follows an approach that allows for good quality images, as well as low bandwidth requirements.

B) H.265/HEVC

Developed in 2013, HEVC is developed video compression standard pioneered by the Joint Collaborative Team on Video Coding (JCT-VC). This HEVC compression standard was developed to double the compression efficiency of the previous standard, H.264/AVC, and at typical consumer video distribution bitrates, HEVC has been largely successful in doing so and is the supported coded for the emerging Ultra HD format. However, compression efficiency results will always vary according to the type of content and the encoder settings [12]

C) H.266 /VVC

Developed in 2020, H.266/VVC is the brainchild of the Fraunhofer Heinrich Hertz Institute, the same group that developed the famous MP3 audio format, as well as AVC and HEVC. VVC can reduce the bandwidth needed for a given video by a further 50% over HEVC as an example by using HEVC, you'd need 10GB of data to transmit a 90-minute 4K video. But with VVC, only 5GB of data are required to achieve the same quality [13]. In addition, H.266 may seem like an efficient and

flexible standard, but it will take years to replace current standards such as H.265 [14].

To better understand the encoding process, get the basic ideas first: The main part of video compression is split into two processes: inter compression and intra compression. Inter means the temporal redundancy, while the purpose of Intra is the reducing of spatial redundancy. The subsequent table depicts the distinctions between H.264 and H.265 and H.266

TableI.1: Comparison of H.264, H.265 and H.266 standards.

	H.264/AVC	**H.265/HEVC**	**H.266/VVC**
Publishing date	2003	2013	2020
Entropy coding	CABAC or CAVLC	CABAC	CABAC with parallelоperations
intra-frame prediction	9 modes	35 modes	81 modes: 65 angular + planar + DC + Sub
Inter-frame prediction	**ME**	16×16, 8×8, 16×8,8×16,8×16,4×4	Max 64×64
	MV	16×16,8×8,16×8,8×16, 8×16,4×4	Half pixel:8 or 7 tap filter quad pixel:8 tap filter
Coding unit	16 x 16 mocroblock	8x8 to 64x64	4x4 to 128x128
Transform	8×8, or 4×4 integer DCT transfom	square DCT-based transforms • 1 DST4x4-based transform	DCT2 (up to 64x64) + MTS (<=32x32, DST7, DCT8, independent(horizontal / vertical))
In-loop filtering	Adaptive deblockingfilter	Adaptive deblocking filter, sample adaptive offset filter	Deblocking • SAO • ALF • LMCS

In our work we are interested to the study, design and implementation of one technique of the compression chain in the new h.266/VVC standard.

I.4 Codec H.266

I.4.1 Definition of VVC/H.266 codec

H.266/VVC is a versatile video encoder that should cut video sizes in half compared to its predecessor H.265/HEVC for equivalent video quality. The Fraunhofer Heinrich-Hertz Institute has just announced the finalization of this new

standard called H.266 /VVC. It is a format adapted to the arrival of videos in 8K definition, in the same way as H.264 with the deployment of Full HD definition and H.265 with 4K definition [15]

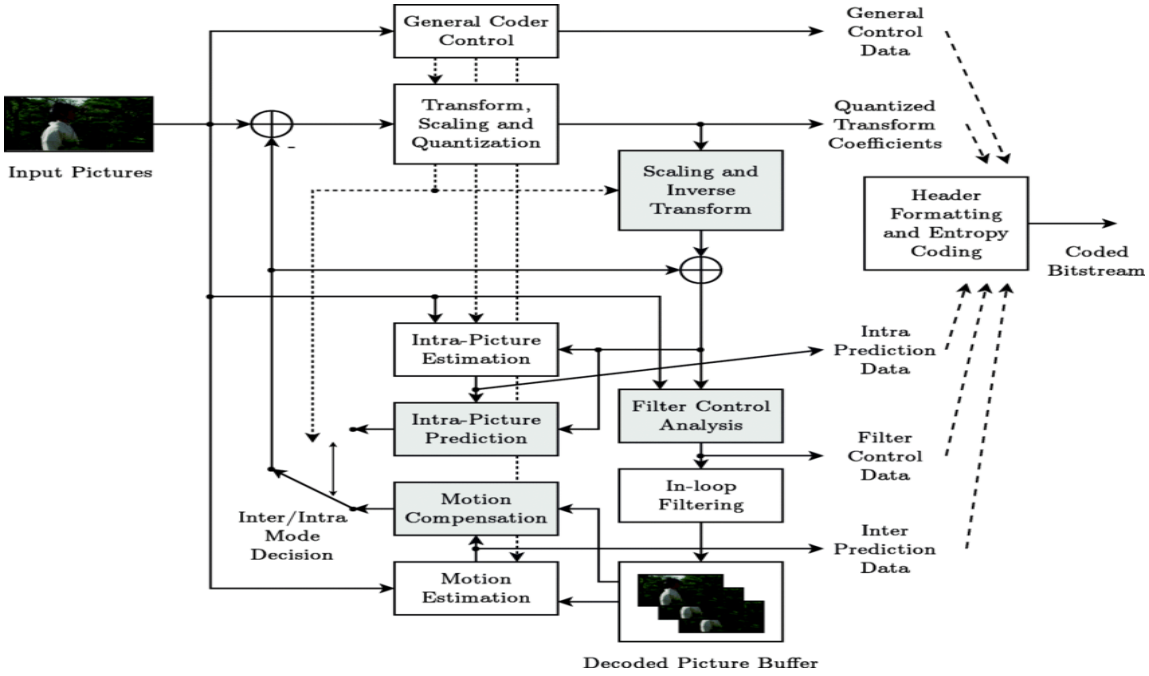


Figure I. 5: Block diagram of the VVC/H.266 video codec.

I.4.2 Characteristics of the H.266 standard

In addition to better management of high image definitions, the H.266 format is suitable for HDR (High Dynamic Range) videos but also for videos filmed at 360 degrees. The H.266 standard also supports 16K definition, variable refresh rates up to 120 Hz and panoramic formats.

According to the Fraunhofer Heinrich-Hertz Institute, the H.265 format requires 10 GB of data for a 90-minute 4K UHD video, but only 5 GB with the new H.266 standard [15]. H.266 a based on different compression techniques:

I.4.2.1Transformation

The concept of Multiple Transform Selection (MTS) of VVC defines three types of separable trigonometric transformations: Discrete Cosine Transform (DCT)-II, VIII and Discrete Sine Transform (DST)-VII.

As shown in Figure I.6, the MTS concept selects, for the following luma blocks of size less than 64, the transform set that minimizes the cost of rate distortion among five transform sets and block skipping [16].

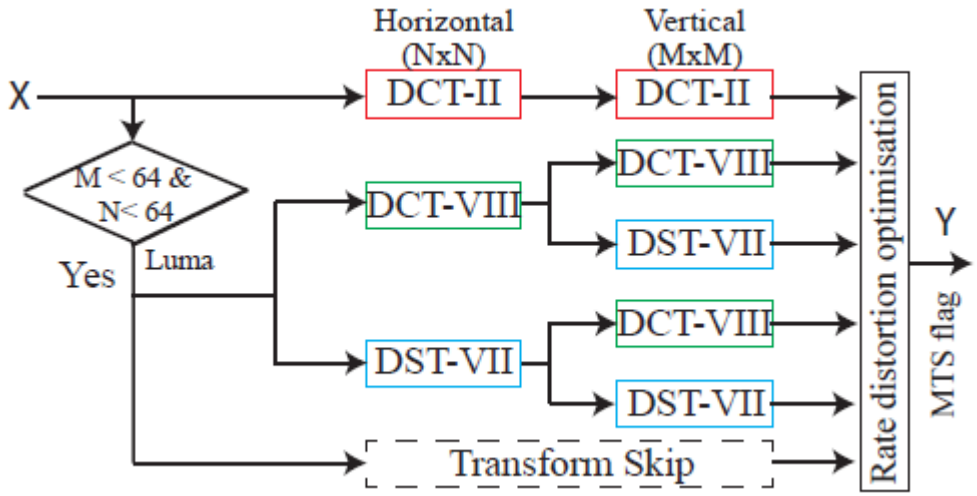


Figure I. 6: Concept of selecting 2D separable transforms in VVC.

I.4.2.2 Loop filtering

By the way of loop filtering, which is at the heart of the versatile video coding, the occurrence of artifacts is minimized, while video content is improved visually. The adaptive loop filter in the VVC standard has the following features:
a 7×7 diamond filter (13 different coefficients) is utilized for luma components and a 5x5 diamond filter (7 different coefficients) for chroma components, each 4x4 luma block can be assigned one of 25 different classes depending on the calculated gradient data and a corresponding to luma gradient three transforms: diagonal reflections, vertical reflections or rotation before they are applied [17].

I.4.2.3 Quantization

The principle of quantization is to reduce the dynamic range of the images after the operation performed by removing the small high frequency values or coefficients [12].

The VVC works using the same quantization algorithm like that of the HEVC standard (along with QP like controls). The range of QP values constitutes the set of values (51 to 63), a 6-times leap in quantization step size. Furthermore, QP also affects the visual appearance and rate of compression of images or video. If the QP is either the same as 1 or lower, the reconstruction is perfect and the compression ratio is very low, and the compression process is carried out through the multiplication by the Qstep (Quantize), so the inverse quantization uses a division by the Qstep. In fact, the Qstep is given by equation (I.1)

$$QStep=2(QP-4)/6(I.3) \quad (I.1)$$

I.4.2.4 CABAC

The last CABAC (Context-Based Adaptive Binary Arithmetic Coding) version that is represented in the new VVC technique is the leader in the area of entropy coding technologies. It results into further enhancement of the accuracy of symbols probabilities coding [18].

CABAC is applied in three steps: a representation of this, that is, converting all non–binary symbols into the bins (bins) because CABAC is based on binary (binomial) coding (binarization). Next, for each bit, CSM (context selection modeling) first makes the most suitable probability model selection according to the neighborhood data, and then this probability is estimated with the aim of enhancing the estimation. Moreover, Binary Arithmetic Coding (BAC) is applied at the final stage to compress the binary symbols depending on the coding mode. At the decoding level, the corresponding steps are simply the inverse form of coding.

I.4.2.5 Prediction

There is Two types of prediction are adopted by the H.266 codec: intra prediction and inter prediction. Figure I.7 shows that intra prediction is used to eliminate spatial redundancies between blocks of the same image and inter prediction to eliminate temporal redundancies between successive images.

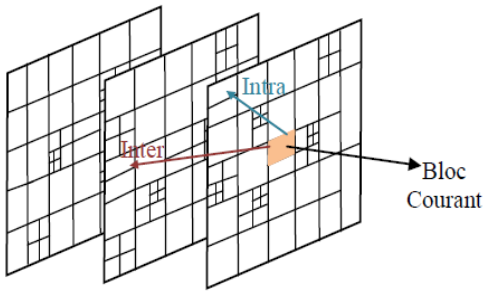


Figure I. 7: Intra and inter prediction of a PU (Prediction Unit) block in a video sequence.

A) Intra prediction

Angular intra prediction is a directional intra prediction method supported by both AVC and HEVC and is also part of VVC. Compared to HEVC, the intra angular prediction of VVC has been modified. By improving the prediction accuracy and adapting to the user's needs to the new partitioning framework. In addition, the number of directional modes available for a given block increases from 33 directions in the HEVC standard to say 65 in the VVC standard. Directions with index pairs between 2 and 66 are equivalent to the directions of the angular modes supported by the HEVC (Figure I.8).

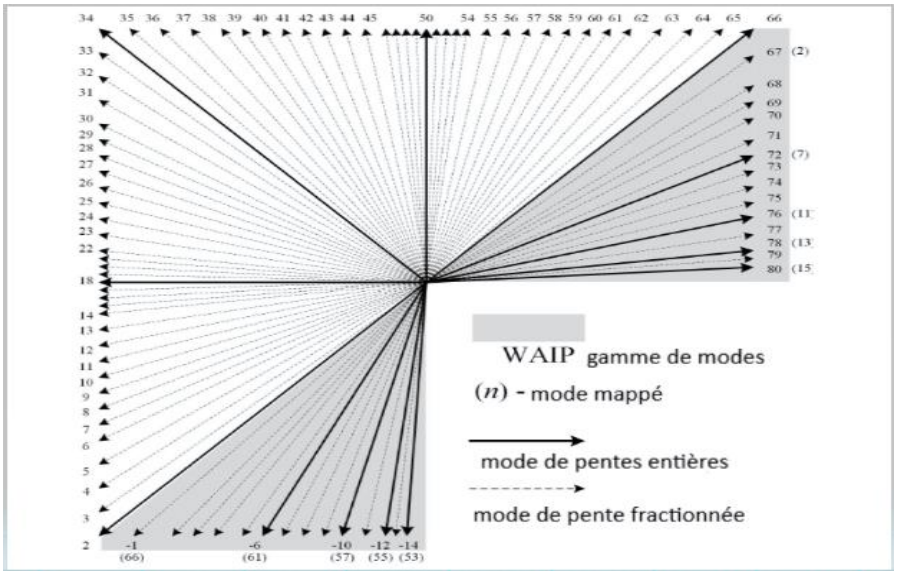


Figure I. 8: 65 intra-prediction directions used for VVC.

B) Inter prediction

H.266 has made some modifications, including the length of the Merge motion vector candidate list, the construction process, and the range of neighboring blocks. At the same time, some new prediction technologies are added, such as affine transformation technology and adaptive motion vector accuracy technology.

Inter prediction exploits temporal redundancy in the video sequence. It predicts the samples of a CU from samples of previously coded reference pictures. The process of finding the most similar block in the reference pictures is called ME (Motion estimation). The ME relies on block matching and is one of most computationally complex operations in the encoding process. It outputs one or several Motion Vectors (MVs), each composed of a vertical and an horizontal component, representing the translation of the CU from the reference frame to current frame [18].

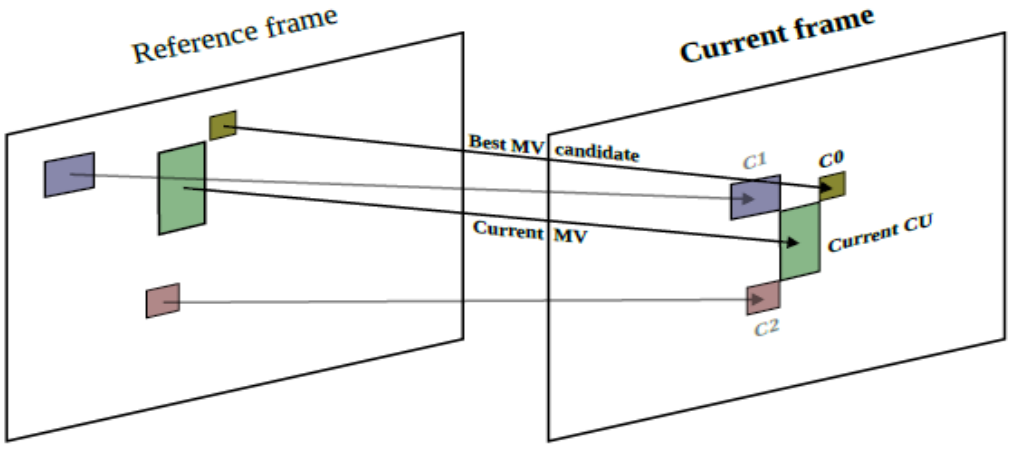


Figure I. 9: Simplified example of Advanced Motion Vector Prediction.

I.5 Conclusion

In this chapter, the basics of digital video and its compression are introduced. Then modern video compression standards were introduced and we focused on the latest H.266/VVC standard and the technologies associated with this standard are presented, namely discrete cosine transform, quantization and prediction, and loop filters. Finally, a detailed study of inter prediction algorithms will be the subject of the next chapter.

Chapter II

Inter prediction in VVC (versatile video coding)

II.1 Introduction

H266\VCC, is an emerging video coding standard. It supports up to 16K resolution and panoramic videos. The improvement of VVC in coding efficiency derives mainly from the more flexible encoding structure, both for intra-picture prediction and inter-picture prediction, which in turn requires more computations. In this chapter, we focus on inter prediction technique and its modes.

II.2 State of the art

II.2.1 Analysis of H.264 Inter Prediction Mode Decision Variants

The analysis of AVC inter prediction mode decision variants is a multifaceted topic, crucial for maximizing compression efficiency and encoding speed in AVC video encoding. Here's a comprehensive breakdown.

II.2.1.1Challenges and Motivations

- **HE Complexity:** AVC offers significantly more inter prediction modes than its predecessors, allowing for better prediction but increasing the complexity of choosing the optimal mode for each block [19].
- **RDO Bottleneck:** Evaluating all possible modes using Rate-Distortion Optimization (RDO) is computationally expensive, hindering real-time encoding applications [19].
- **Fast Mode Decision Algorithms:** These methods aim to quickly identify a smaller set of promising candidates for RDO, reducing computational cost. Techniques include [19].
- **Early termination:** Stop searching after a threshold number of evaluated modes.
- **Cost prediction:** Estimate RDO cost for each mode using heuristics, avoiding full calculations.
- **Machine learning**: Train models to predict best mode based on extracted video features.
- **Complexity Reduction Techniques:** Simplify RDO itself to decrease overall encoding complexity.
- **Simplified motion estimation:** Employ faster, less accurate motion estimation algorithms.
- **Reduced precision quantization**: Represent pixel values with fewer bits, making RDO calculations less demanding [19].
- **Hybrid Approaches:** Combine both strategies for optimal trade-off between speed, complexity, and quality [19].

II.2.1.2 Advantages and disadvantages of Inter Prediction in H.264

A) Advantages

- Wide choice of prediction modes, offering great flexibility.
- Well-established and widely used techniques.
- Good performance for low and medium complexity content.

B) Disadvantages

- High computational complexity for optimal mode selection.
- Less efficient than newer codecs for high complexity content.

II.2.2 Analysis of H.265 Inter Prediction Mode Decision Variants

H.265/(HEVC), offers significant advancements in video compression compared to its predecessor, H.264. One key element contributing to this improvement is the use of complex inter prediction techniques. These techniques rely on choosing the optimal mode to predict a block of pixels based on previously encoded information. However, this process can be computationally expensive due to the large number of potential prediction modes available.

II.2.2.1 Challenges and variants

- **High computational complexity**: Evaluating all possible prediction modes for every block is computationally expensive, especially for real-time applications [20].
- **Rate-Distortion (RD) trade-off:** Finding the optimal mode requires balancing the bitrate (compression efficiency) and distortion (visual quality) of the encoded video [20].
- **Early termination:** This approach stops evaluating modes after a certain threshold is reached, aiming to find a good enough mode quickly [20].
- **Skip mode:** Directly copy data from the reference frame without prediction, suitable for areas with little change [20].
- **Merge mode**: Combines motion vectors from previously encoded blocks, efficient for repetitive patterns [20].
- **Fast mode selection:** This approach uses heuristics or machine learning to narrow down the candidate modes before full evaluation, reducing the number of calculations [20].
- **Texture-based methods:** Analyze the texture of the block to predict which modes are likely to be effective [20].
- **Motion vector-based methods**: Utilize information from previously encoded motion vectors to guide mode selection [20].

- **Machine learning-based methods**: Employ techniques like decision trees or neural networks to learn optimal mode selection from training data [20].

II.2.2.2Advantages and Disadvantages of Inter Prediction in H.265

A) Advantages

- Significant improvement in compression efficiency compared to H.264.
- Better exploitation of temporal redundancy thanks to more precise prediction modes.
- Support for higher resolutions and more demanding video formats.

B) Disadvantages

- Increased complexity compared to H.264, impacting real-time encoding.
- Requires greater computing and memory power for decoding.

II.2.3 Analysis of H.266 Inter Prediction mode decision variants

H.266 also known as Versatile Video Coding (VVC), builds upon the successes of H.265 by offering even greater compression efficiency and visual quality. Similar to H.265, inter prediction plays a crucial role in achieving this improvement.

II.2.3.1Challenges and variants

- **Increased complexity:** H.266 introduces new features like multi-type tree (MTT) partitioning and advanced motion compensation, leading to a larger set of potential prediction modes compared to H.265. Evaluating all these modes remains computationally expensive [21].
- **Maintaining RD balance:** Achieving optimal bitrate-distortion trade-off requires careful selection among even more options [21].
- **Early termination:** Similar to H.265, methods like skip and merge modes are still effective in certain scenarios [21].
- **Fast mode selection:**

Texture-based methods are further refined to handle the complexities of H.266's prediction modes.

Motion vector-based methods leverage the improved motion compensation capabilities of H.266.

Machine learning-based methods are increasingly employed, with more sophisticated neural network architectures like convolutional neural networks (CNNs) being explored.

- **Hybrid approaches:** Combine early termination with machine learning for efficient selection within a narrowed candidate set [21].

II.2.3.2 Advantages and Disadvantages of Inter Prediction in H.266

A) Advantages

- New generation of video compression with significant efficiency gains compared to H.265.
- Advanced inter and intra prediction techniques for better exploitation of redundancy.
- Better adaptability to different types of content and new display technologies.

B) Disadvantages

- Even higher complexity than H.265, potentially limiting widespread adoption.
- More resource-intensive encoding and decoding, requiring more efficient hardware.

II.2.4 Major Differences of performances between H.266, H.265 and H.264 standards

H.266 offers better compression than H.265 and H.264, but at the cost of increased complexity. H.265 is a good compromise between efficiency and complexity, while H.264 is more suitable for less demanding applications. Additionally, the choice of codec depends on the specific needs in terms of compression, video quality and available resources. The following table shows us the major differences of functionality between the three standards.

Table II. 1 Major differences of functionality between H.264, H.265 and H.266 decodecs.

Functionality	H.264	H.265	H.266
Compression efficiency	Average	Good	Excellent
Encoding complexity	Average	High	Very high
Decoding complexity	Average	High	Very high

Video quality	Good	Excellent	Superior
Material support	Wide	Growing	Limit

II.3 Inter prediction of VVC standard

II.3.1 Inter prediction definition

Inter-picture prediction (also called motion compensated prediction) is used to find redundant information between adjacent picture frames in a video sequence and describe movement of textures as motion vectors. The inter prediction is typically performed only on the luminance component (Y in YCbCr/YUV) as it contains the grayscale texture details of the video. Inter prediction include motion estimation (ME) to find suitable motion vectors and motion compensation (MC) to apply the estimated motion vectors to reconstruct the video sequence. Video sequences typically have different types of movement involved, for example objects in motion and camera panning.

In our work, we will focus on Inter prediction technique and its modes.

II.3.2 Compression based on the motion compensation

- The motion compensation is a whole item in the video codec plug-in.
- Estimation of the motion (search motion vector).
- Prediction basic on the motion compensation.
- Calculation of the error of prediction (the difference).

The compensation method for moving the most used devices is the Motion compensation method by corresponding block (block matching). For more efficiency, the image is divided into macroblocks of N×N sizes. N = 16 for images luminance. and chrominance images, N=8 if the space 4: 2: 0 is adopted, the actual frame is applicable to the frame is available. This correspondence is sent to you the macroblock is installed in the frame and the larger macroblock in the frame is more suitable. The motion estimate function is shown in Figure II.1 [22].

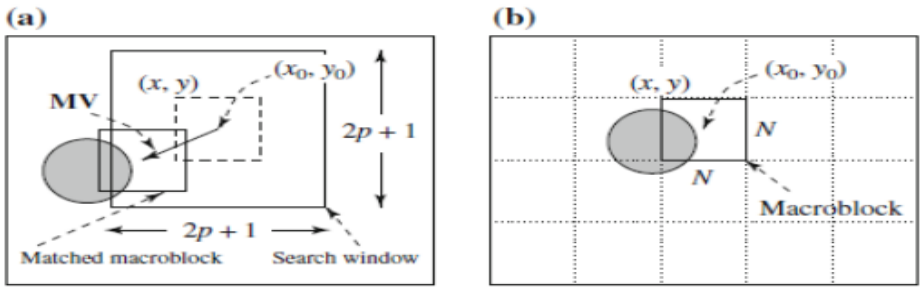


Figure II. 1: Illustration of the principle of the corresponding macroblock and its Motion vector (a: reference frame, b: target frame).

If the reference frame is a pre-operation frame, it will be applied early (forward prediction). If the reference frame is a new frame (future), the operation is applied for prediction and arrival (back word prediction). The difference between the macroblock of the frame is clear and its corresponding in the reference frame is used to refer to the prediction [22].

II.3.2.1 Search for motion vectors

Searching for motion vectors MV (u, v) is called matching problem. Since the search for MVs is costly in terms of the number of operations, it is generally limited to a small immediate neighborhood. The horizontal and vertical displacements i and j are included in the range [-p, p], where p is a positive integer of relatively small value. This creates a search window of size (2p+1)×(2p+1), as shown in Figure II.1.

We use the upper left corner (x, y) as the origin of the macroblock in the target frame. Let C(x+k, y+l) be pixels in the macroblock of the target frame (current) and R(x+i+k, y+ j+l) be pixels of the macroblock of the reference frame, where k and l are indices for the pixels of the macroblock and i and j are the horizontal and vertical displacements, respectively. The difference between two macroblocks can be measured by their average absolute difference (MAD: Mean Absolute Difference) [22].

A) Sequential search for motion vectors

The simplest way to find motion vectors is to perform a sequential search in the entire window (2p+1)×(2p+1) of the reference frame (this method is also called full search). A macroblock centered at each position in the window is compared to the macroblock in the target frame, pixel by pixel, and their respective MAD is then derived using one previous expressions. The vector (i, j) which offers the minimum MAD is designated as a motion vector (MV (u, v)) for the current macroblock of the target frame [22].

B) 2D logarithmic search

A more efficient version, suboptimal but still effective, is called search logarithmic. This method is based on searching by region. The search procedure 2D logarithm of motion vectors takes several iterations as illustrated in Figure I.2. For each iteration we define a new search region more restricted than the iteration previous with a new center of the zone [22].

- In the first step, we observe the eight positions at the distance of p/2 pixels at from the central point (x0, y0). The metrics are calculated for the nine points (denoted 1 in Figure II.2). The position that provides the minimum metric becomes the position center for the next step.
- In the second step, we consider locations located at a distance p/4 of the new central position. Again, the metrics are calculated for the 9 surrounding locations. The position linked to the lowest metric is a new central position.

- The operation is repeated until reaching a distance between the points of the region equal to 1.

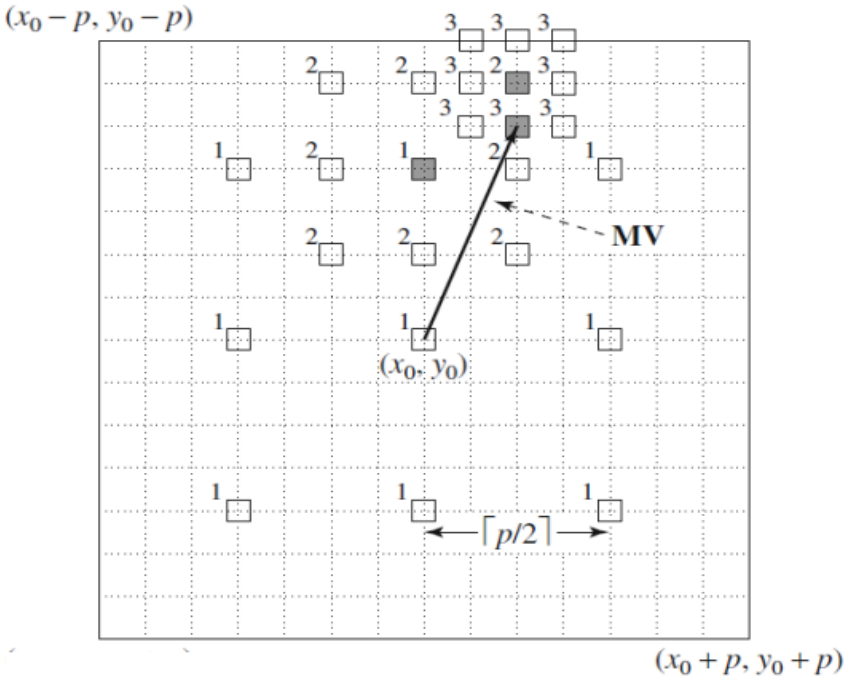


Figure II. 2: Explanation of 2D logarithmic search.

II.3.3 Motion Vector Coding in VVC

As known, inter prediction is a key part of every video coding standard. The compression ratio heavily relies on efficient representation of motion data. The motion vector coding in the new VVC standard is based on the well established concepts of HEVC. However, there are several refinements, including the revised AMVP (Advanced Motion Vector Prediction) candidate list construction process, SMVD (Symmetric Motion Vector Difference) and AMVR (Adaptive Motion Vector Resolution), which lead to an improved coding efficiency. The detailed descriptions and theories behind those tools will be described in the following subsections.

II.3.3.1 Motion Vector Predictor List Generation Algorithm

The motion vector prediction algorithm of VVC is based on the AMVP of HEVC. It is applied in case of explicit motion vector signaling, i.e., for inter predicted blocks that do not use merge mode. For each motion vector, a list of exactly two motion vector predictor candidates is generated and a flag indicates which of the two is used. The following candidates are checked for availability:

- Up to two spatial candidates (similar to that in HEVC).
- Up to one temporal candidate (similar to that in HEVC).
- Up to four HMVP candidates (new in VVC).

- Zero motion vectors, if not enough other candidates are available (similar to that in HEVC).

For the spatial and the temporal (co-located) candidates, the same spatial locations as in HEVC are used. Unlike in HEVC, when the Picture Order Count (POC) of the candidate's reference picture does not match the POC of the current reference picture, the candidate is considered unavailable in this case and the scaling of spatial candidates is removed in VVC to save the computational complexity. As in HEVC, in order to limit memory bandwidth requirements, the Temporal Motion Vector Predictor (TMVP) is restricted to only use co-located candidates that belong to the same Coding Tree Unit (CTU) row as the current block. Also, as in HEVC, the TMVP candidate can be disabled at sequence level or at picture level. The TMVP is also not available for coding blocks of size 8×4 and 4×8 in VVC in order to improve the MV throughput since TMVP derivation process typically requires a scaling process which is relatively more complex compared to other merge candidates and MV determination process for small blocks is in a critical path for hardware implementation. In case that after checking the spatial and temporal candidates, still less than two candidates have been found, up to two candidates derived from History-based Motion Vector Predictor HMVP table can be used as a new feature of VVC [23].

Finally, if there are still less than two candidates after checking HMVP, the motion vector candidate list is filled with zero motion vectors as in HEVC.

II.3.3.2 Symmetric Motion Vector Difference

In the bi-prediction mode, lots of bits are used to code the motion information which includes the reference picture indices, MVP indices and MVDs for reference picture list 0 and list 1. To code the motion information of bi-prediction mode in a more effective way, the SMVD technology is adopted. The SMVD mode is an inter bi-prediction mode in which part of motion information is derived with an assumption of linear motion. Therefore, the number of bits for motion information coding can be reduced. For a CU coded with a non-affine bi prediction mode, a SMVD flag is signaled to indicate whether the SMVD mode is selected or not. When the flag is true, only the MVP indices of list 0 and list 1 and the MVD of list 0 are signaled, and other motion information (i.e., reference pictures and MVD of list 1) is not signaled but derived at the decoder side.

First, the MVD of list 1 is symmetrically derived from the MVD of list 0. That is, the MVD of list 1 ($mvdxL1$, $mvdyL1$) is symmetric to the MVD of list 0 ($mvdxL0$, $mvdyL0$), as shown in below: ($mvdxL1$, $mvdyL1$) = ($-mvdxL0$,$-mvdyL0$).

Second, the reference pictures of list 0 and list 1 are determined at slice-level. These two reference pictures can only be short-term reference pictures and may be as in either case 1 or case 2, as follows.

- Case 1: Reference picture of list 0 is the nearest picture among all the pictures preceding the current picture in output order in list 0. Reference picture of list 1 is the nearest picture among all the pictures following the current picture in output order in list 1.
- Case 2: Reference picture of list 0 is the nearest picture among all the pictures following the current picture in output order in list 0. Reference picture of list 1 is the nearest picture among all the pictures preceding the current picture in output order in list 1 If none of the reference pictures can be found as in case 1 or case 2, the SMVD mode is marked as unavailable and the SMVD flag is not sent [24].

II.3.3.3 Adaptive Motion Vector Resolution

The video coding standards HEVC and AVC/H.264 use a fixed motion vector resolution of quarter luma sample. However, it is a well-known fact that in order to achieve overall rate-distortion optimality, an optimum trade-off between displacement vector rate (Rd) and prediction error rate (Re) has to be chosen [25].

In particular, at optimality the partial derivatives of the multivariate distortion rate function with respect to Rd and Re have to be equal. Previously, this aspect has only been considered at the encoder-side in the motion estimation stage. The new video coding standard VVC allows to select the motion vector resolution at coding block level and, therefore, to trade-off bit rate versus fidelity for the signaling of the motion parameters. [26]

For translational inter predicted blocks, the following motion vector resolutions are which is out of scope for this paper. The AMVR mode is signaled at the coding block level if at least one component of an MVD is unequal to zero. The motion vector predictor is rounded to the given resolution such that the resulting motion vector is guaranteed to fall on a grid of the given resolution. Note that in case the half luma sample resolution is selected for a coding block, also an alternative luma interpolation filter is used for the half-sample position in this block. This aspect of AMVR is also known as switchable interpolation filter (SIF). It can be seen that the alternative filter has a strong low pass characteristic which can be beneficial for attenuating high frequency noise components. More details about SIF can be found in [27].

The application of the alternative interpolation filter is also further propagated in merge mode, i.e., if a coding block in merge mode references a neighboring block that uses the alternative interpolation filter, the referencing block will use it as well.

II.3.4 Block merging in VVC

Block merging is an efficient coding tool in HEVC. In VVC, more merge modes are introduced for higher coding efficiency. In this section, regular merge mode is first introduced. Subsequently, new merge modes, including geometric partitioning mode, merge mode with motion vector difference, combined inter-intra prediction are discussed in order. Finally, merge estimation region for low encoding complexity is described. [24].

II.3.4.1 Block merging candidate list generation algorithm

There are five types of motion vector predictor candidates in regular merge mode, i.e., spatial candidates, temporal candidates, HMVP candidates, pairwise average candidate, and zero MV candidates. The spatial and temporal candidates are the same as those in HEVC except that the order of the first two spatial candidates is swapped for higher coding efficiency. HMVP candidates, derived from an HMVP table, are inserted into merge list after spatial and temporal candidates until the merge list reaches the maximum allowed size minus one. To avoid duplicated candidates while keeping a relatively low complexity, redundancy check is applied. Only when any of the following three conditions is met, an HMVP candidate is inserted into the merge list.

The pairwise average candidate is generated by averaging a pre-defined candidate pair in the existing merge candidate list. There is up to one pairwise candidate which averages the first two existing candidates in merge candidate list. The averaged motion vectors are calculated separately for each reference list. If both motion vectors are available in one list, these two motion vectors are averaged even when they point to different reference pictures; if only one motion vector is available, use the one directly; if no motion vector is available, keep this list invalid. If the merge candidate list is not full after inserting the pairwise average candidate, zero MV candidates will be added until the candidate list is full, which is the same as in HEVC.

II.3.4.2 Merge mode with motion vector difference

In addition to merge mode wherein motion information is directly derived from neighboring, historic or zero motion information, the MMVD technology is adopted to allow further encoding an MVD as refinement of the derived motion information. Roughly speaking, MMVD mode is between AMVP mode and merge mode, and it provides a new trade-off between accuracy of motion information and bitrate. In MMVD, one of the first two candidates in the merge candidate list is selected as base motion, and an MVD represented by a direction and a distance is encoded as refinement of the base motion. Four directions including {0, 90, 180, 270}-degrees are allowed for an MVD and a direction index is signaled to indicate the selected direction. Meanwhile, two distance tables, each of which has eight distance entries as illustrated in Table I are designed for the MVD. The encoder can select a

distance table at picture level. A distance index is signaled for an MVD to indicate the selected entry of the distance table [28].

Only one MVD is signaled for both unidirectional and bidirectional base motion. When the base motion is bidirectional, the signaled MVD is directly used for one reference picture list and is scaled according to POC distances from the current picture to the two reference pictures before being used for the other reference picture list. When the current picture is with shorter absolute POC distance to the reference picture in list 1 than to the reference picture in list 0, the signaled MVD is directly used for list 1, otherwise, it is directly used for list 0.

II.3.4.3 Geometric partitioning mode GPM

The GPM aims to increase the partition precision and to better fit the moving objects boundaries using a geometrical partition of a coding tree leaf node CU. Because of the more flexible partitioning and the blending process, the GPM is benefit to video contents that include rigid moving objects relative to static background or other moving objects. Furthermore, the newly design GPM algorithm significantly reduces the encoder and decoder complexity yet reserves the coding gain comparing with the prior methods proposed to HEVC. Therefore, the presented algorithm has been adopted in VVC [29].

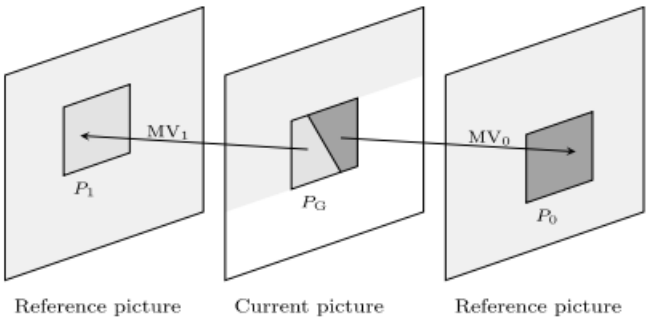


Figure II. 3: Example of the prediction process of GPM.

II.4 Conclusion

In this chapter we present the different papers that studied inter prediction technique in HEVC and VVC. Next we give a detailed study of all the inter prediction modes used in VVC standard. In the next chapter, we focused on the software implementation of inter prediction algorithms in VVC standard by using Matlab environment.

Chapter III

Implementation of inter prediction algorithms in VVC standard under MATLAB

III.1 Introduction

The VVC standard defines many derivations of inter prediction: we will start this chapter with the tools used in our work. Then, we will present the proposed flowcharts of inter prediction derivations with their simulations under Matlab.

III.2 Used tools

It is essential to present the different software tools and their versions used in this work because each tool has its specific characteristics and functions. We have chosen the software description language Matlab for the software description of the inter prediction modes in the latest VVC video compression standard.

We used also python software implementation of our algorithm in order to compare and validate the results obtained from Matlab.

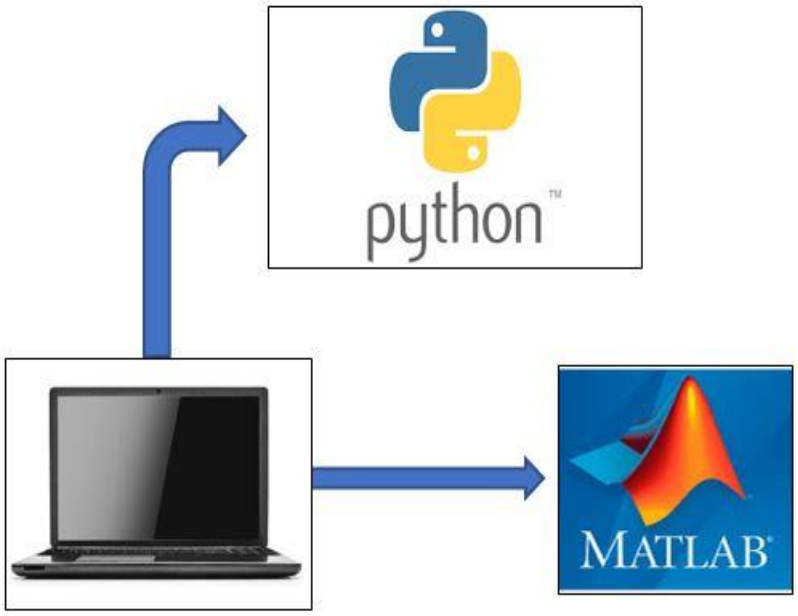


Figure III. 1: Software tools used in our project.

III.2.1 MATLAB

Matlab is a numerical calculation software marketed by the Math Works company to allow students to work from a high-level programming tool. It stands for Matrix laboratory. Moreover, it is a language for scientific computing of data analysis visualization and algorithm development. Its interface offers, on the one hand, an interactive console-type window for executing commands, and on the other hand, IDE (integrated development environment) [30].

III.2.2 Python

Python is a widely-used programming language known for its simplicity and readability. It allows developers to write clear, logical code for both small and large-scale projects. Python is versatile, supporting various programming paradigms such as procedural, object-oriented, and functional programming. It is commonly used in web development, data analysis, artificial intelligence, scientific computing, and automation due to its extensive libraries and frameworks. Created by Guido van Rossum and first released in 1991, Python has become one of the most popular programming languages in the world.

III.3 Inter prediction derivation algorithm

In this section, we give the proposed flowcharts, and its simulation on MATLAB software for each inter prediction derivation modes in VVC standard.

III.3.1 Derivation process for block vector components for IBC blocks (general)

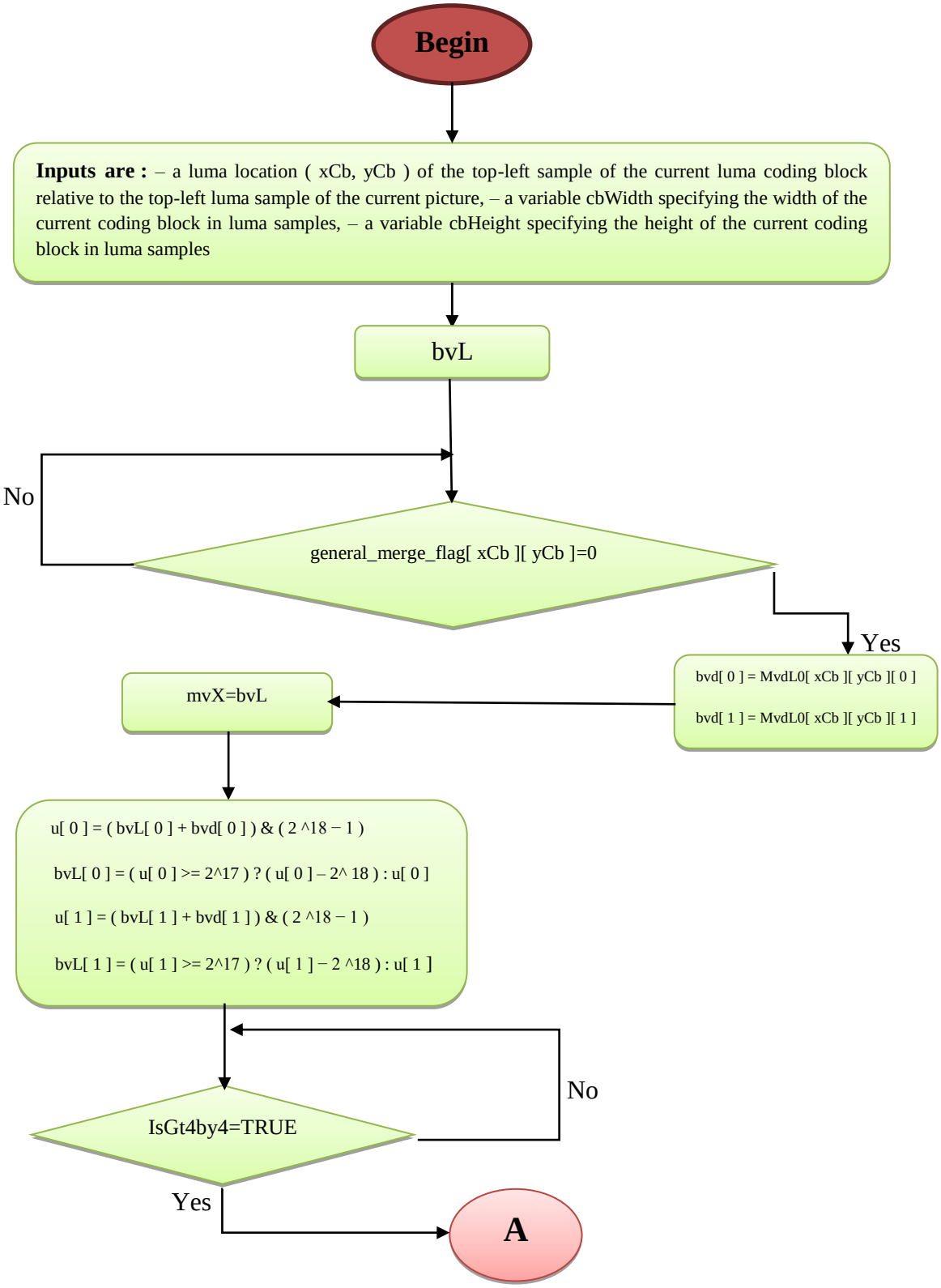
Inputs to this process are: a luma location (xCb, yCb) of the top-left sample of the current luma coding block relative to the top-left luma sample of the current picture, a variable cbWidth specifying the width of the current coding block in luma samples, a variable cbHeight specifying the height of the current coding block in luma samples [31]. The flowchart of this process presented in the Figure III.2.

III.3.1.1 Matlab implementation and simulation result

The implementation and simulation results under Matlab of derivation process for block vector components for IBC blocks (general) are shown in the Figure III.3, and Figure III.4 respectively.

II.3.2 Picture reconstruction process (general)

Inputs to this process are: – a location (xCurr, yCurr) specifying the top-left sample of the current block relative to the top-left sample of the current picture component, – the variables nCurrSw and nCurrSh specifying the width and height, respectively, of the current block, – a variable cIdx specifying the colour component of the current block, – an nCurrSw x nCurrSh array predSamples specifying the predicted samples of the current block, – an nCurrSw x nCurrSh array resSamples specifying the residual samples of the current block [31]. The flowchart of this process presented in the Figure III.5.


Begin
Inputs are : – a luma location (xCb, yCb) of the top-left sample of the current luma coding block relative to the top-left luma sample of the current picture, – a variable cbWidth specifying the width of the current coding block in luma samples, – a variable cbHeight specifying the height of the current coding block in luma samples
bvL
No
general_merge_flag[xCb][yCb]=0
Yes
bvd[0] = MvdL0[xCb][yCb][0]
bvd[1] = MvdL0[xCb][yCb][1]
mvX=bvL
u[0] = (bvL[0] + bvd[0]) & (2 ^18 − 1)
bvL[0] = (u[0] >= 2^17) ? (u[0] – 2^ 18) : u[0]
u[1] = (bvL[1] + bvd[1]) & (2 ^18 − 1)
bvL[1] = (u[1] >= 2^17) ? (u[1] − 2 ^18) : u[1]
No
IsGt4by4=TRUE
Yes
A

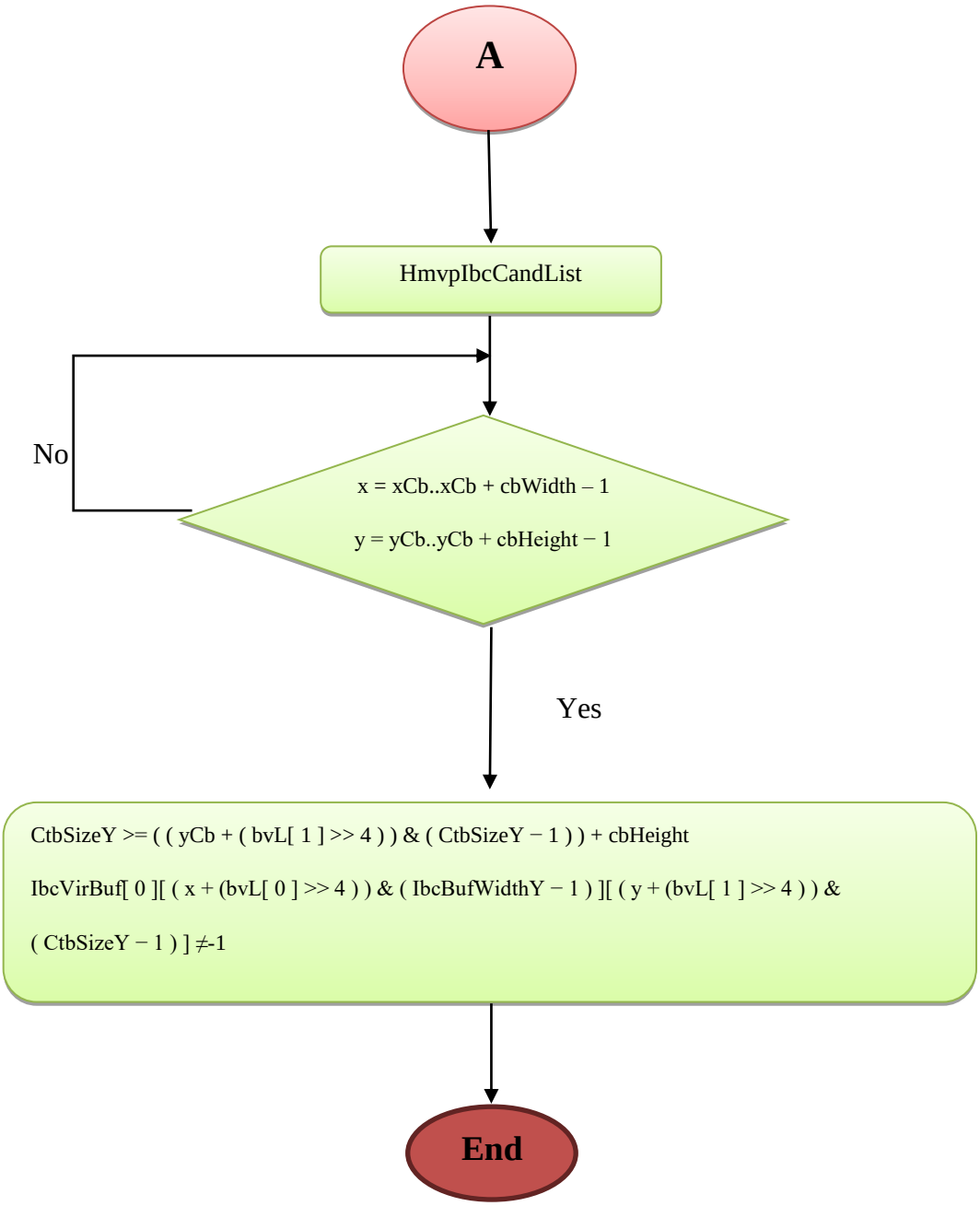


Figure III. 2: Derivation process for block vector components for IBC blocks (general).

deriveChromaBlocksssssssssssss.m | untitled.m

/MATLAB Drive/untitled.m

```
function bvL = derive_bvL(xCb, yCb, cbWidth, cbHeight, general_merge_flag, MvdL0, AmvrS
    % Initialize bvL with zeros
    bvL = [0, 0];

    % Check if general_merge_flag is 0
    if general_merge_flag(xCb, yCb) == 0
        % Derive bvd
        bvd = [MvdL0{xCb, yCb}(1), MvdL0{xCb, yCb}(2)];

        % Rounding process for motion vectors
        bvL = round_mv(bvL, AmvrShift);

        % Modify the luma block vector bvL
        u = [bitand(bvL(1) + bvd(1), 2 ^ 18 - 1), bitand(bvL(2) + bvd(2), 2 ^ 18 - 1)];
        bvL = [(u(1) - 2 ^ 18) * (u(1) >= 2 ^ 17) + u(1) * (u(1) < 2 ^ 17), ...
               (u(2) - 2 ^ 18) * (u(2) >= 2 ^ 17) + u(2) * (u(2) < 2 ^ 17)];

        % Check constraints
        if IsGt4by4
            % Update block vector predictor list based on history
            update_history(bvL);
        end

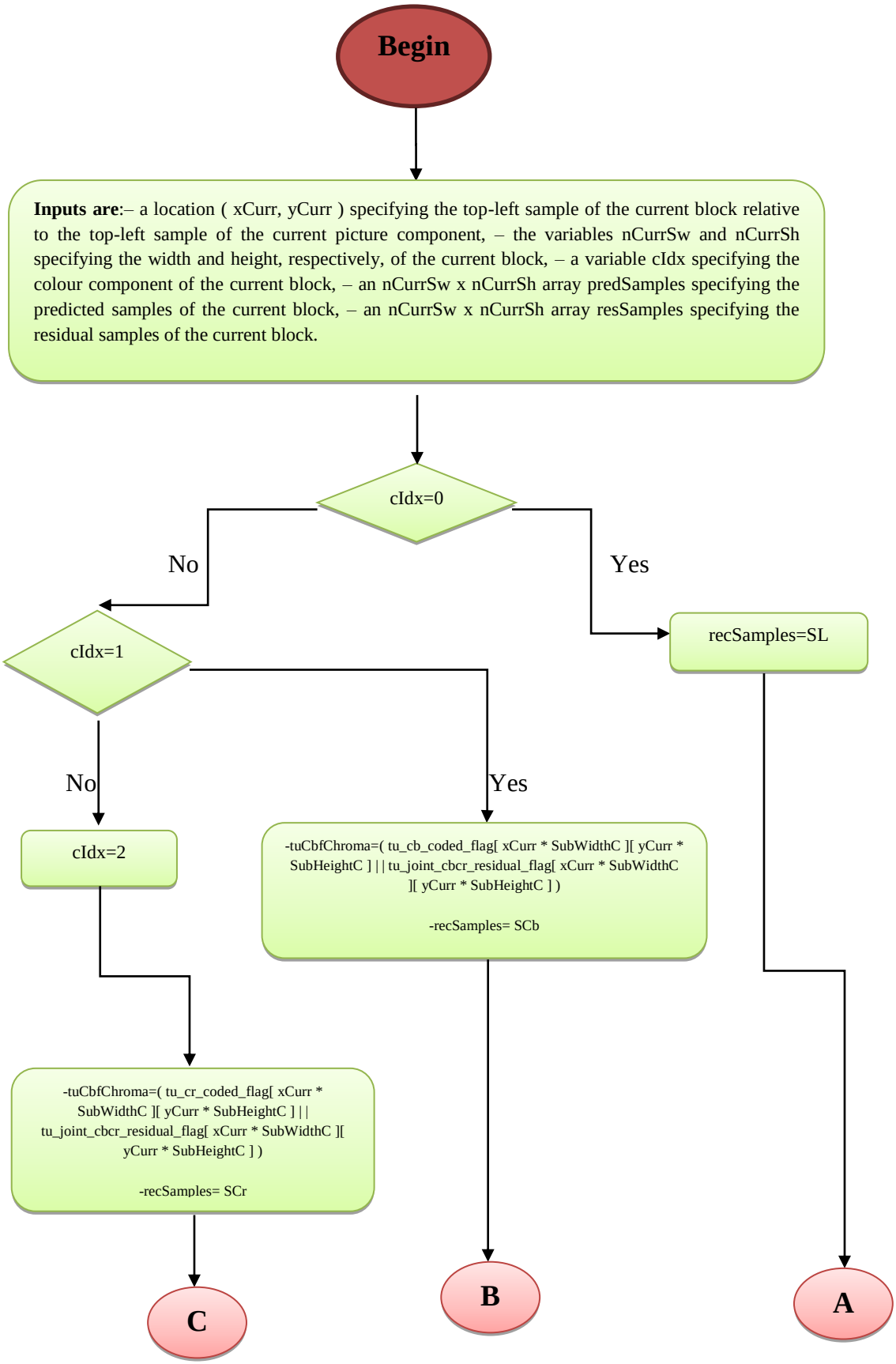
        % Additional constraints
        assert(CtbSizeY >= bitand((yCb + bitshift(bvL(2), -4)), (CtbSizeY - 1)) + cbHei
```

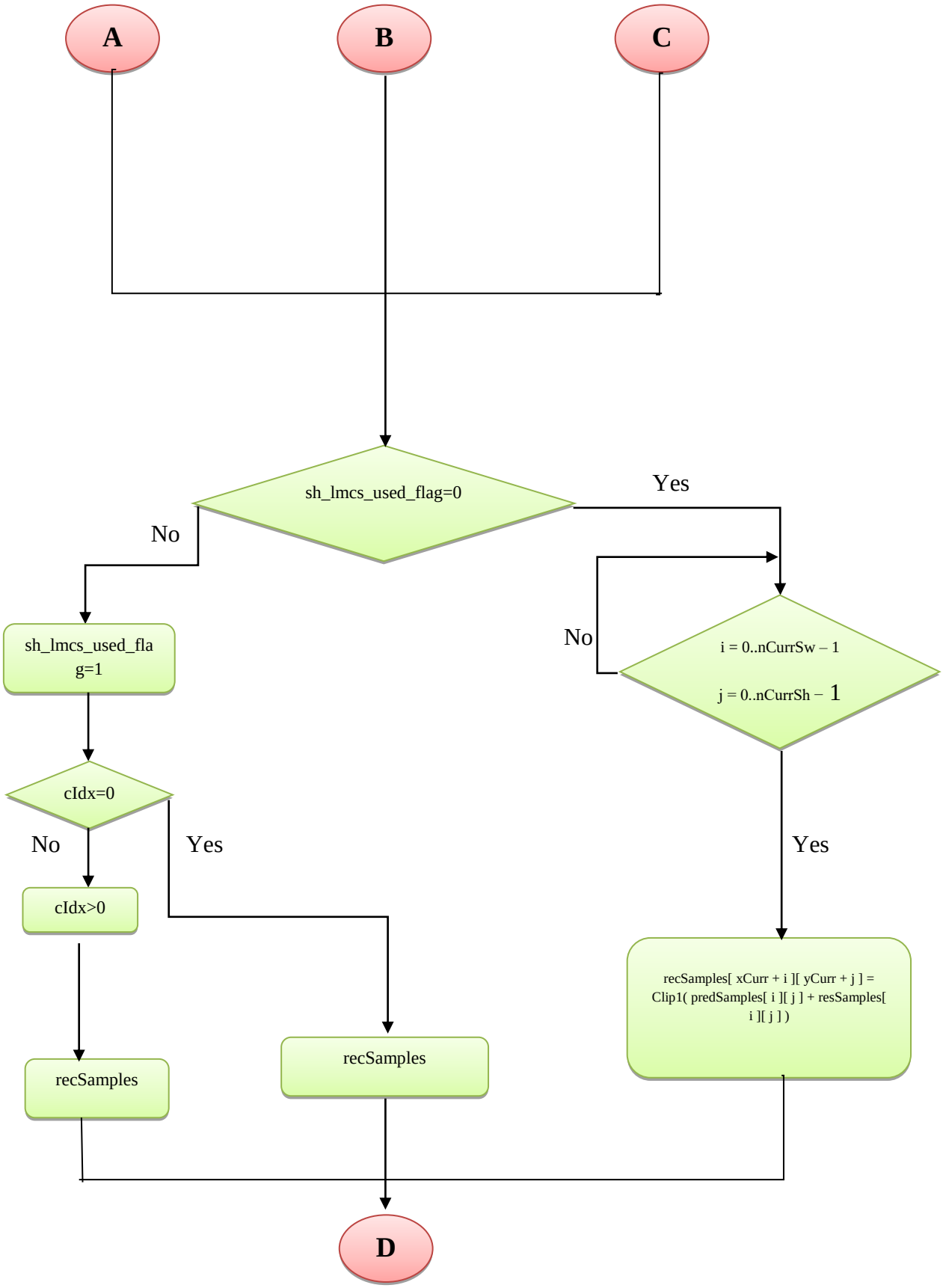
Figure III. 3: Implementation of derivation process for block vector components for IBC blocks (general) under matlab.

```
>> untitled
Derived luma block vector bvL:
     0     0

>>
```

Figure III. 4: Simulation result of derivation process for block vector components for IBC blocks (general).


Begin
Inputs are:– a location (xCurr, yCurr) specifying the top-left sample of the current block relative to the top-left sample of the current picture component, – the variables nCurrSw and nCurrSh specifying the width and height, respectively, of the current block, – a variable cIdx specifying the colour component of the current block, – an nCurrSw x nCurrSh array predSamples specifying the predicted samples of the current block, – an nCurrSw x nCurrSh array resSamples specifying the residual samples of the current block.
cIdx=0
No
Yes
recSamples=SL
cIdx=1
No
Yes
cIdx=2
-tuCbfChroma=(tu_cb_coded_flag[xCurr * SubWidthC][yCurr * SubHeightC] | | tu_joint_cbcr_residual_flag[xCurr * SubWidthC][yCurr * SubHeightC])
-recSamples= SCb
-tuCbfChroma=(tu_cr_coded_flag[xCurr * SubWidthC][yCurr * SubHeightC] | | tu_joint_cbcr_residual_flag[xCurr * SubWidthC][yCurr * SubHeightC])
-recSamples= SCr
C
B
A


A
B
C
sh_lmcs_used_flag=0
Yes
No
sh_lmcs_used_fla
g=1
No
i = 0..nCurrSw – 1
j = 0..nCurrSh – 1
cIdx=0
No
Yes
Yes
cIdx>0
recSamples[xCurr + i][yCurr + j] =
Clip1(predSamples[i][j] + resSamples[
i][j])
recSamples
recSamples
D

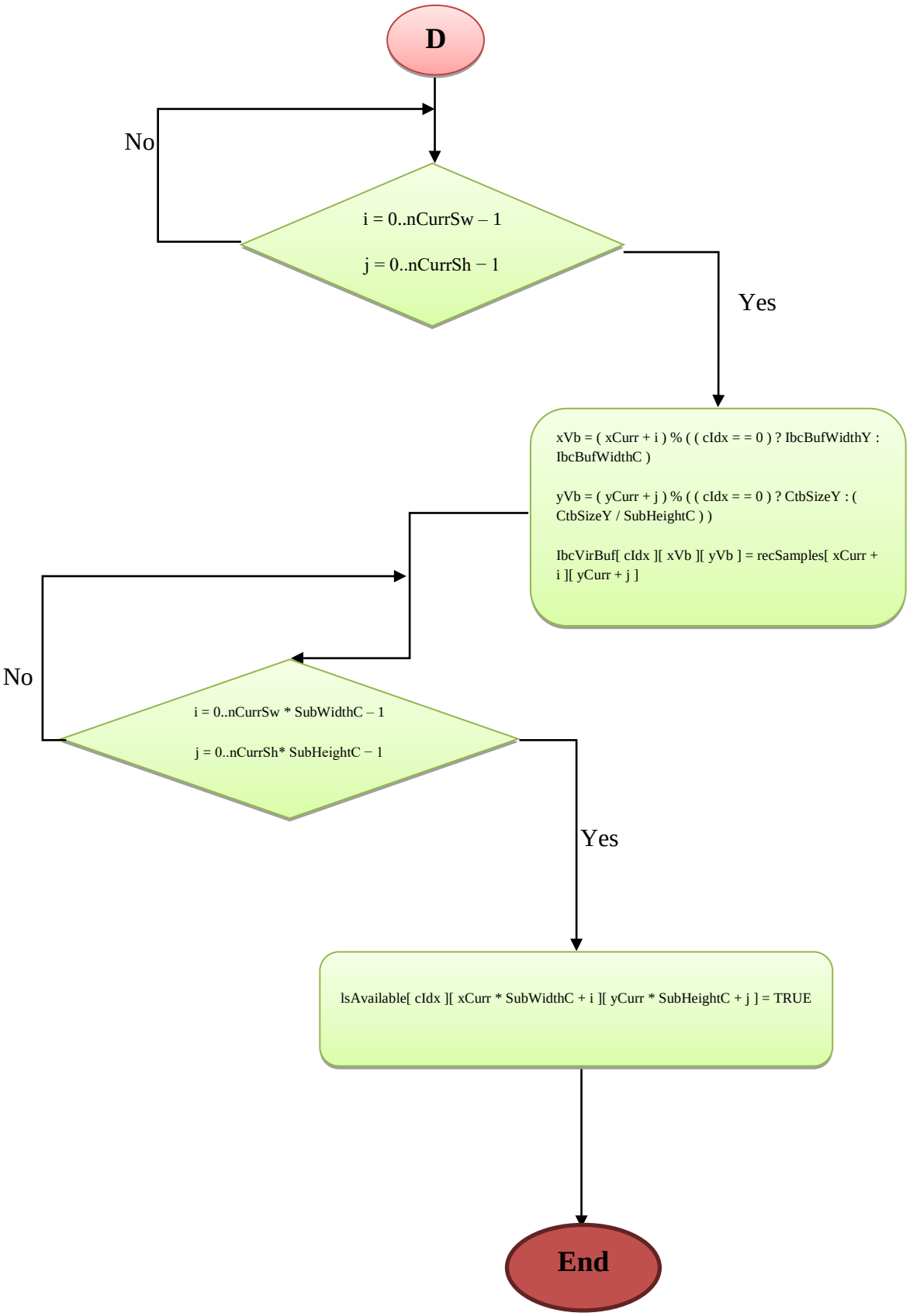


Figure III. 5: Flowchart of Picture reconstruction process.

III.3.2.1 Matlab implementation and simulation result

Matlab implementation and simulation result of picture reconstruction process (general) are given in Figure III.6.

```
            yVb = mod(yCurr + j, (cIdx == 0) * CtbSizeY + (cIdx ~= 0) * (CtbSizeY / SubHeightC));
            % Assuming IbcVirBuf is defined elsewhere in the code
            IbcVirBuf{cIdx + 1}(xVb + 1, yVb + 1) = recSamples(xCurr + i + 1, yCurr + j + 1);
        end
    end

    % Assuming IsAvailable is a global variable
    global IsAvailable;
    for i = 0:nCurrSw*SubWidthC-1
        for j = 0:nCurrSh*SubHeightC-1
            IsAvailable{cIdx + 1}(xCurr*SubWidthC + i + 1, yCurr*SubHeightC + j + 1) = true;
        end
    end

    % Display the generated matrices
    disp('Prediction Samples (predSamples):');
    disp(predSamples);
    disp('Residual Samples (resSamples):');
    disp(resSamples);
    disp('Reconstructed Samples (recSamples):');
    disp(recSamples);
end
```

Command Window

New to MATLAB? See resources for Getting Started.

```
Reconstructed Samples (recSamples):
    86   115
     0     0
```

Figure III. 6: Implementation and simulation result of picture reconstruction process under matlab.

III.3.3 Derivation process for chroma block vectors

Input to this process is a luma block vector in 1/16 fractional-sample accuracy bvL [31]. At the output we obtained two vectors bvc (0) and bvc (1). The Figure III.7 represent the flowchart of derivation process for chroma block vectors.

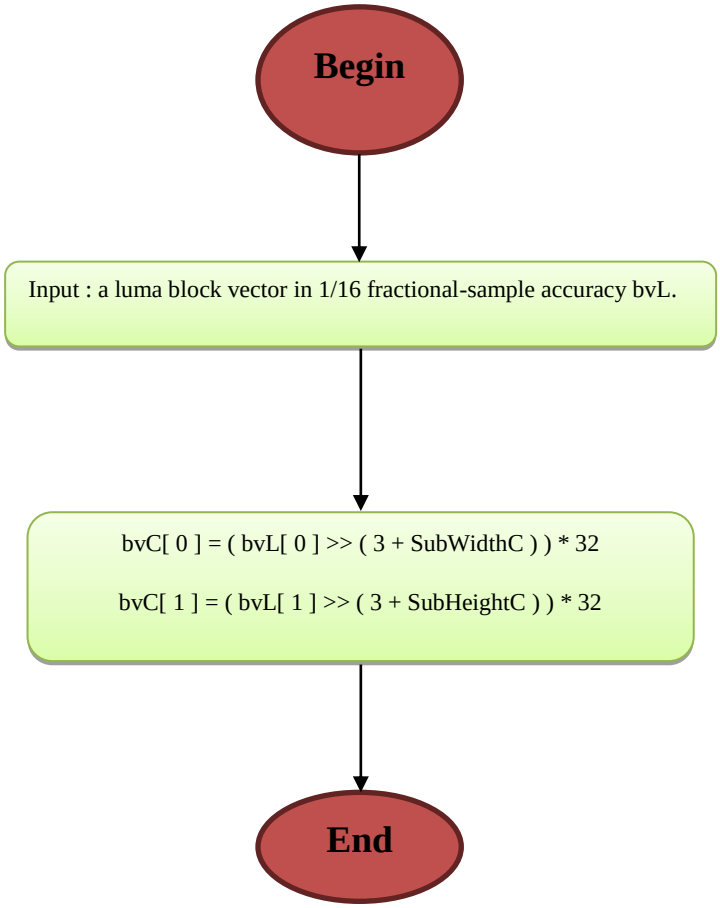


Figure III. 7: Flowchart of Derivation process for chroma block vectors.

III.3.3.1 Matlab implementation and simulation result

The implementation and simulation result of Derivation process for chroma block vectors are given in the Figure III.8. The bvc (0) and bvc (1) variables are calculating by using the shifter, addition and multiplication operations.

deriveChromaBlocksssssssssssss.m ×

/MATLAB Drive/deriveChromaBlocksssssssssssss.m

```
1   function bvC = bvC(~, ~, ~)
2   bvL = [16, 4];
3   SubWidthC = 2;
4   SubHeightC = 2;
5       % bvL: Input luma block vector
6       % SubWidthC: Subsampling width factor
7       % SubHeightC: Subsampling height factor
8
9       % Initialize the chroma block vector
10
11
12      % Derive chroma block vector
13      bvC(1) = bitshift(bvL(1), -(3 + SubWidthC)) * 32;
14      bvC(2) = bitshift(bvL(2), -(3 + SubHeightC)) * 32;
15
16  end
```

Command Window

```
>> deriveChromaBlocksssssssssssss
bvC(0)
     0     0

bvC(1)
     0     0

bvC
     0     0
```

Figure III. 8: Implementation and simulation result of derivation process for chroma block vectors.

III.3.4 Residual modification process for blocks using color space conversion

Inputs to this process are: a variable nTbW specifying the block width, a variable nTbH specifying the block height, an (nTbW)x(nTbH) array of luma residual samples rY with elements rY[x][y], an (nTbW)x(nTbH) array of chroma residual samples rCb with elements rCb[x][y], an (nTbW)x(nTbH) array of chroma residual samples rCr with elements rCr[x][y] [31]. The flowchart of residual modification process for blocks using color space conversion in the Figure III.9.

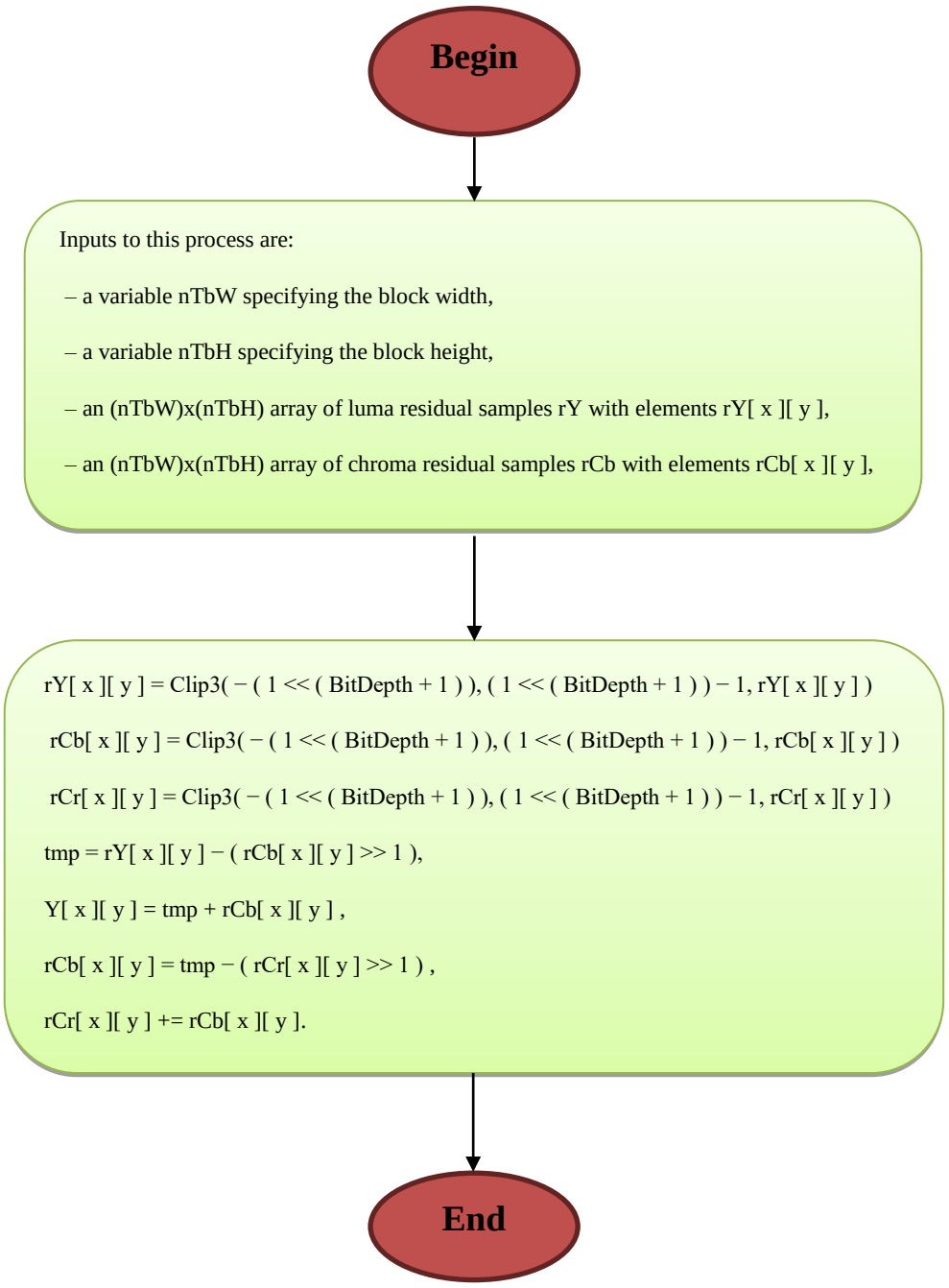


Figure III. 9: flowchart of Residual modification process for blocks using color space conversion.

III.3.4.1 Matlab implementation and simulation result

The implementation and simulation result of Residual modification process for blocks using color space conversion are given in Figure III.10 and Figure III.11 respectively. we obtain three matrix of 4*4 sizes for luma Y and chroma components rCb, rCr.

```
% Clip3 function
Clip3 = @(x) max(min(x, (bitshift(1, (BitDepth + 1)) - 1)), -bitshift(1, (BitDepth + 1)));

for x = 1:min(nTbW, size(rY, 1))
    for y = 1:min(nTbH, size(rY, 2))
        % Clip values
        rY_clip = Clip3(rY(x, y));
        rCb_clip = Clip3(rCb(x, y));
        rCr_clip = Clip3(rCr(x, y));

        % Compute new values using temporary variables
        tmp = rY_clip - bitshift(rCb_clip, -1);
        new_rY = tmp + rCb_clip;
        new_rCb = tmp - bitshift(rCr_clip, -1);
        new_rCr = rCr_clip + new_rCb;

        % Update matrices with new values
        rY(x, y) = new_rY;
        rCb(x, y) = new_rCb;
        rCr(x, y) = new_rCr;
    end
end

% Display the modified matrices
disp('Modified rY:');
disp(rY);
disp('Modified rCb:');
```

Figure III. 10: Part of implementation Matlab of Residual modification process for blocks using color space conversion.

```
Modified rY:
   1   3   2   1
   1   0   2   1
   0   2   3   2
   3   1   2   2

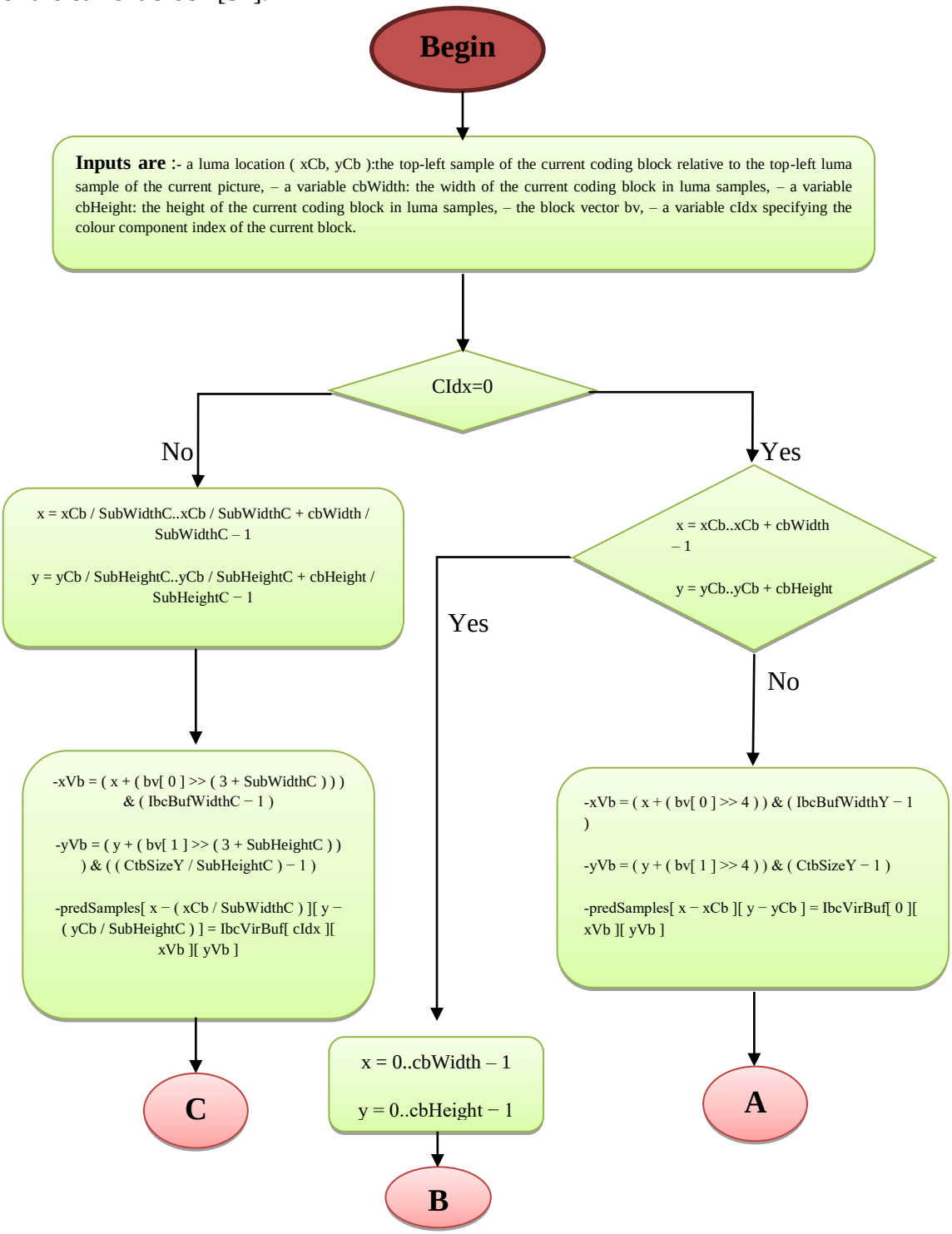
Modified rCb:
   -2    2   -1   -1
   -2    0    1    1
    0    1    1    1
    1    0    2    0

Modified rCr:
    0    2    1   -1
    0    0    1    1
    1    3    3    2
    1    0    3    1
```

Figure III. 11: Simulation result of Residual modification process for blocks using color space conversion.

III.3.5 Decoding process for IBC blocks

This process is invoked when decoding a coding unit coded in IBC prediction mode. Inputs to this process are: a luma location (xCb, yCb) specifying the top-left sample of the current coding block relative to the top-left luma sample of the current picture, a variable cbWidth specifying the width of the current coding block in luma samples, a variable cbHeight specifying the height of the current coding block in luma samples, the block vector bv, a variable cIdx specifying the colour component index of the current block [31].



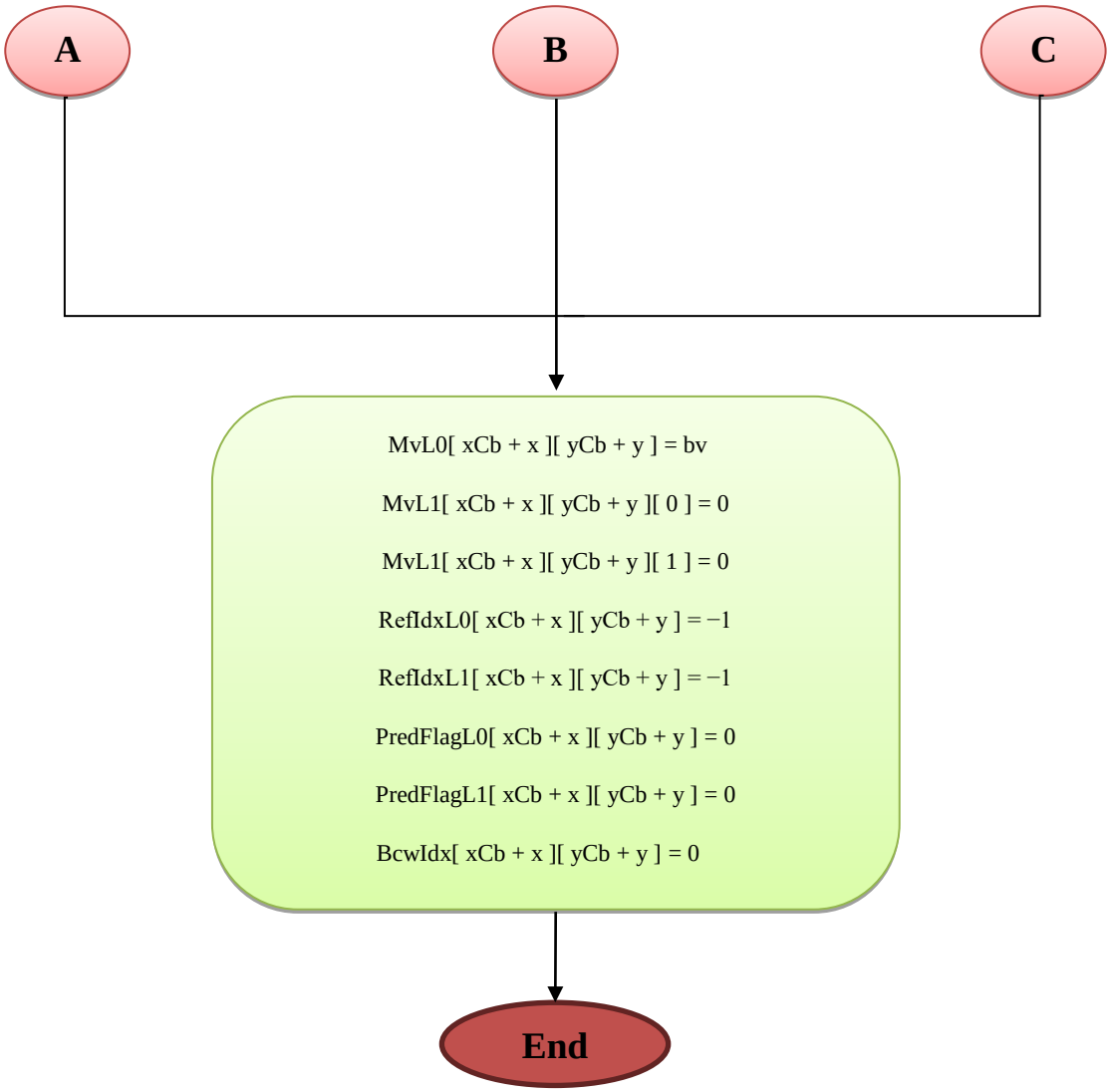


Figure III. 12: Flowchart of Decoding process for IBC blocks.

III.3.5.1: Matlab implementation and simulation result

The implementation and simulation result of decoding process for IBC blocks are given in the Figure III.13 and Figure III.14 respectively.

```
    for x = xCb:(xCb + cbWidth - 1)
        for y = yCb:(yCb + cbHeight - 1)
            if cIdx == 0
                xVb = bitand(x + bitshift(bv(1), -4), IbcBufWidthY - 1);
                yVb = bitand(y + bitshift(bv(2), -4), CtbSizeY - 1);
                predSamples(x - xCb + 1, y - yCb + 1) = IbcVirBuf(1, xVb + 1, yVb + 1);

                % Assignments when cIdx is equal to 0
                MvL0(x - xCb + 1, y - yCb + 1) = bv;
                MLv1(x - xCb + 1, y - yCb + 1, :) = [0, 0];
                RefIdxL0(x - xCb + 1, y - yCb + 1) = -1;
                RefIdxL1(x - xCb + 1, y - yCb + 1) = -1;
                PredFlagL0(x - xCb + 1, y - yCb + 1) = 0;
                PredFlagL1(x - xCb + 1, y - yCb + 1) = 0;
                BcwIdx(x - xCb + 1, y - yCb + 1) = 0;
            else
                xVb = bitand(x + bitshift(bv(1), -(3 + SubWidthC)), IbcBufWidthC - 1);
                yVb = bitand(y + bitshift(bv(2), -(3 + SubHeightC)), floor((CtbSizeY / Sub
                predSamples(x - floor(xCb / SubWidthC) + 1, y - floor(yCb / SubHeightC) +
            end
        end
```

Figure III. 13: Part of implementation Matlab for decoding process for IBC blocks.

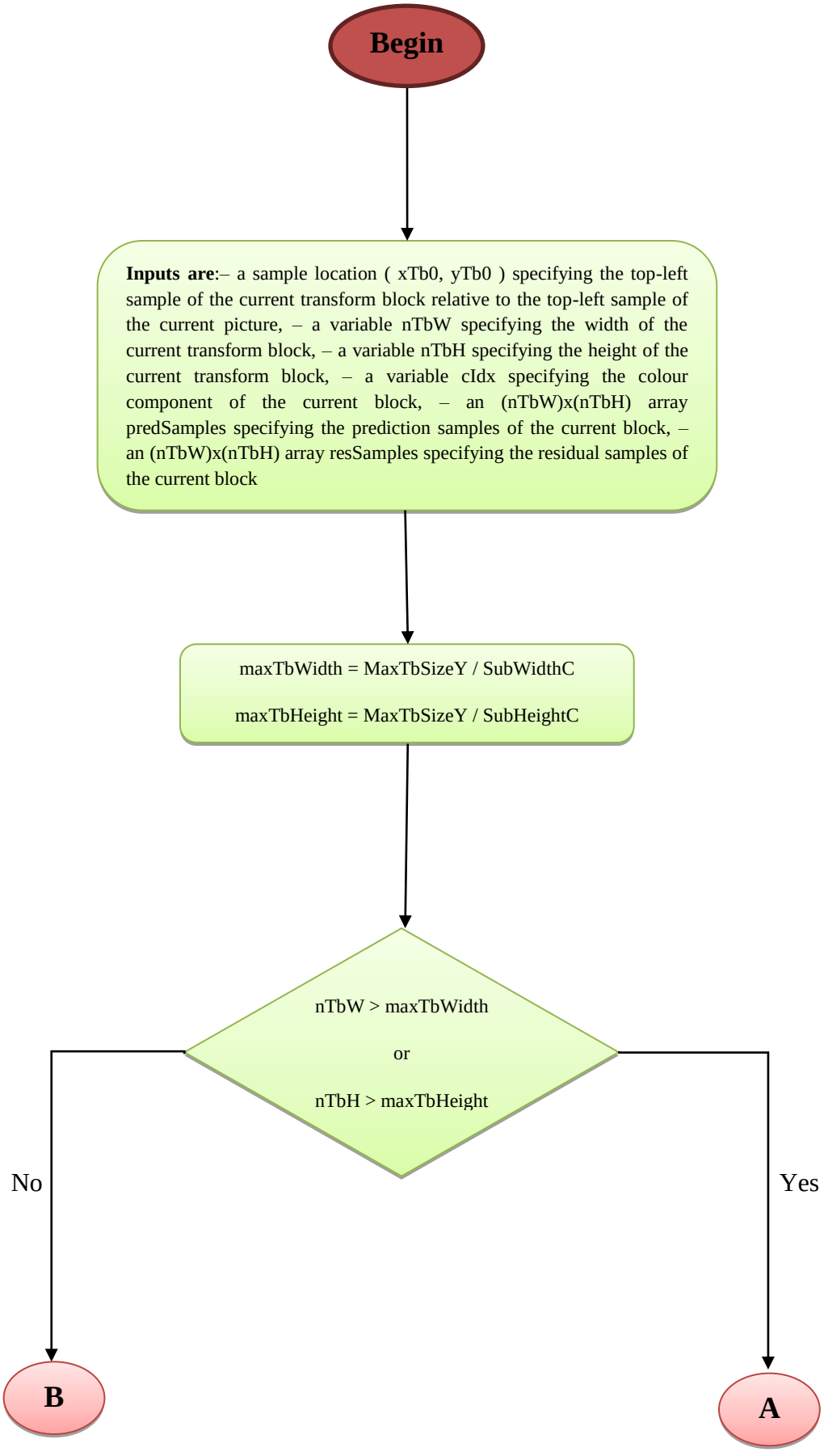
```
predSamples =

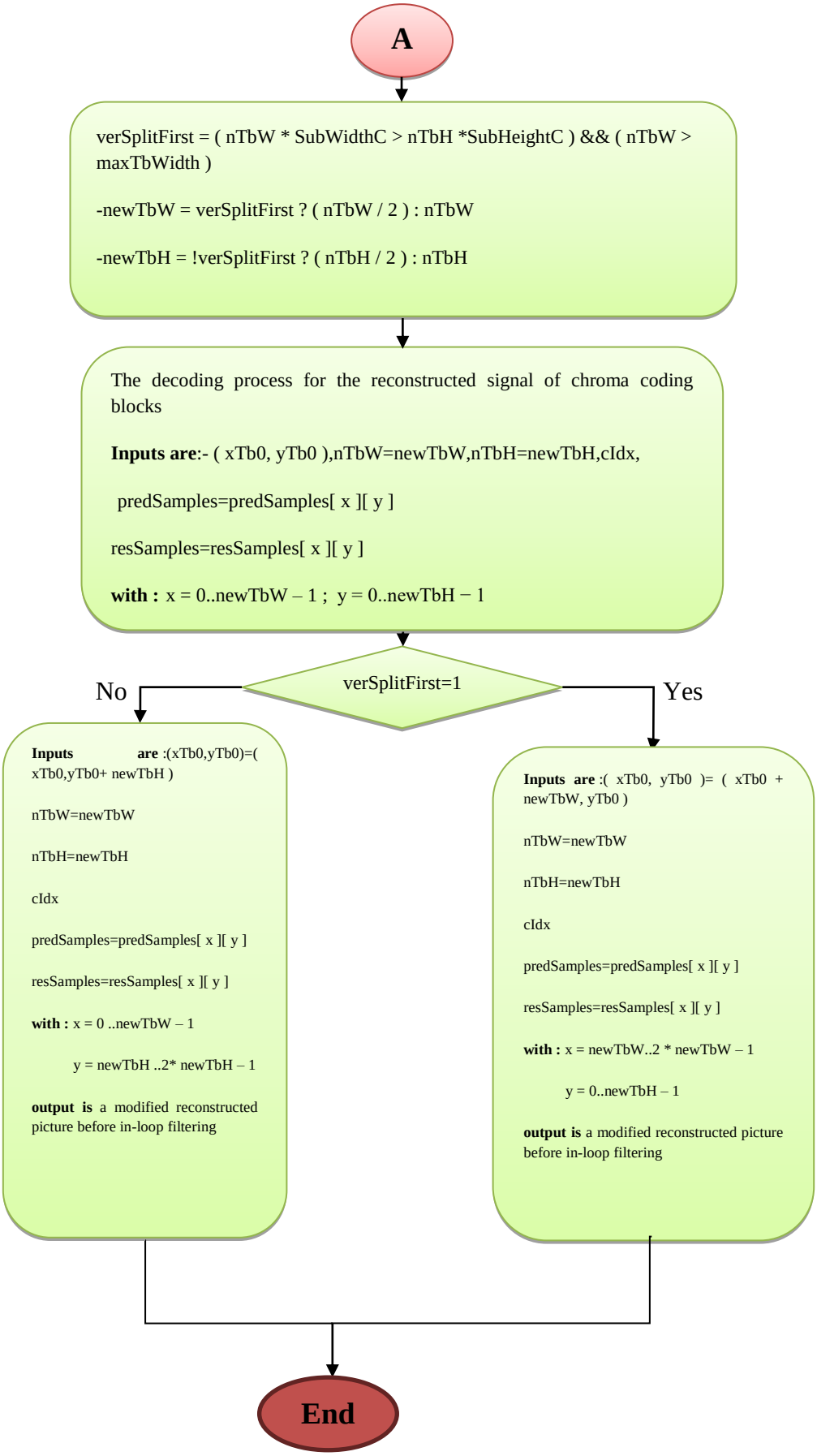
  -115    81
  -115    81
```

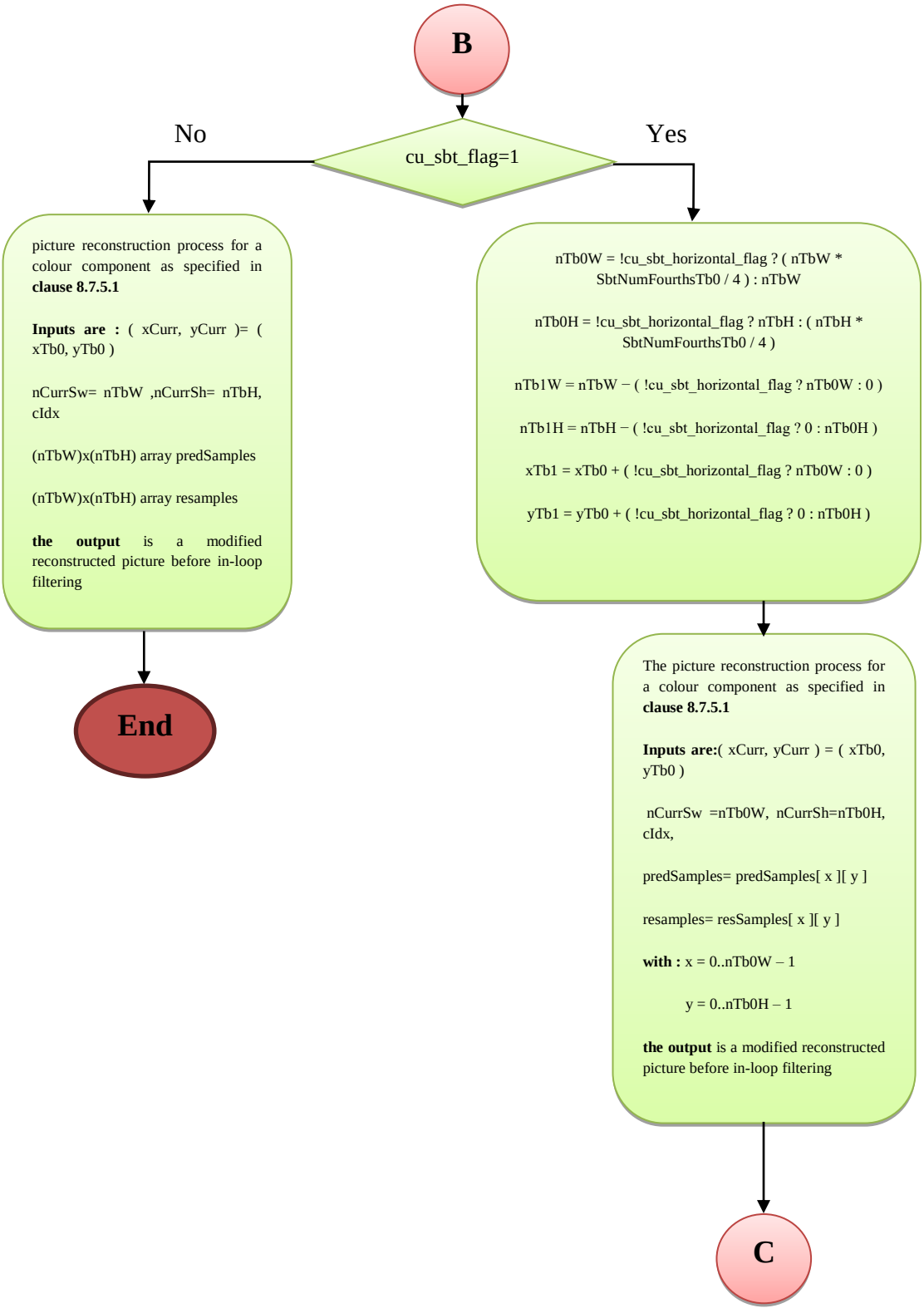
Figure III. 14: Simulation result of decoding process for IBC blocks.

III.3.6 Decoding process for the reconstructed signal of chroma coding blocks coded in inter prediction mode

Inputs to this process are: a sample location (xTb0, yTb0) specifying the top-left sample of the current transform block relative to the top-left sample of the current picture, a variable nTbW specifying the width of the current transform block, a variable nTbH specifying the height of the current transform block, a variable cIdx specifying the colour component of the current block, an (nTbW)x(nTbH) array predSamples specifying the prediction samples of the current block, an (nTbW)x(nTbH) array resSamples specifying the residual samples of the current block [31].


Begin
Inputs are:– a sample location (xTb0, yTb0) specifying the top-left sample of the current transform block relative to the top-left sample of the current picture, – a variable nTbW specifying the width of the current transform block, – a variable nTbH specifying the height of the current transform block, – a variable cIdx specifying the colour component of the current block, – an (nTbW)x(nTbH) array predSamples specifying the prediction samples of the current block, – an (nTbW)x(nTbH) array resSamples specifying the residual samples of the current block
maxTbWidth = MaxTbSizeY / SubWidthC
maxTbHeight = MaxTbSizeY / SubHeightC
nTbW > maxTbWidth
or
nTbH > maxTbHeight
No
Yes
B
A


A
verSplitFirst = (nTbW * SubWidthC > nTbH *SubHeightC) && (nTbW > maxTbWidth)
-newTbW = verSplitFirst ? (nTbW / 2) : nTbW
-newTbH = !verSplitFirst ? (nTbH / 2) : nTbH
The decoding process for the reconstructed signal of chroma coding blocks
Inputs are:- (xTb0, yTb0),nTbW=newTbW,nTbH=newTbH,cIdx,
predSamples=predSamples[x][y]
resSamples=resSamples[x][y]
with : x = 0..newTbW – 1 ; y = 0..newTbH − 1
verSplitFirst=1
No
Yes
Inputs are :(xTb0,yTb0)=(xTb0,yTb0+ newTbH)
nTbW=newTbW
nTbH=newTbH
cIdx
predSamples=predSamples[x][y]
resSamples=resSamples[x][y]
with : x = 0 ..newTbW – 1
y = newTbH ..2* newTbH – 1
output is a modified reconstructed picture before in-loop filtering
Inputs are :(xTb0, yTb0)= (xTb0 + newTbW, yTb0)
nTbW=newTbW
nTbH=newTbH
cIdx
predSamples=predSamples[x][y]
resSamples=resSamples[x][y]
with : x = newTbW..2 * newTbW – 1
y = 0..newTbH – 1
output is a modified reconstructed picture before in-loop filtering
End


B
cu_sbt_flag=1
No
Yes
picture reconstruction process for a colour component as specified in clause 8.7.5.1
Inputs are : (xCurr, yCurr)= (xTb0, yTb0)
nCurrSw= nTbW ,nCurrSh= nTbH, cIdx
(nTbW)x(nTbH) array predSamples
(nTbW)x(nTbH) array resamples
the output is a modified reconstructed picture before in-loop filtering
End
nTb0W = !cu_sbt_horizontal_flag ? (nTbW * SbtNumFourthsTb0 / 4) : nTbW
nTb0H = !cu_sbt_horizontal_flag ? nTbH : (nTbH * SbtNumFourthsTb0 / 4)
nTb1W = nTbW − (!cu_sbt_horizontal_flag ? nTb0W : 0)
nTb1H = nTbH − (!cu_sbt_horizontal_flag ? 0 : nTb0H)
xTb1 = xTb0 + (!cu_sbt_horizontal_flag ? nTb0W : 0)
yTb1 = yTb0 + (!cu_sbt_horizontal_flag ? 0 : nTb0H)
The picture reconstruction process for a colour component as specified in clause 8.7.5.1
Inputs are:(xCurr, yCurr) = (xTb0, yTb0)
nCurrSw =nTb0W, nCurrSh=nTb0H, cIdx,
predSamples= predSamples[x][y]
resamples= resSamples[x][y]
with : x = 0..nTb0W – 1
y = 0..nTb0H – 1
the output is a modified reconstructed picture before in-loop filtering
C

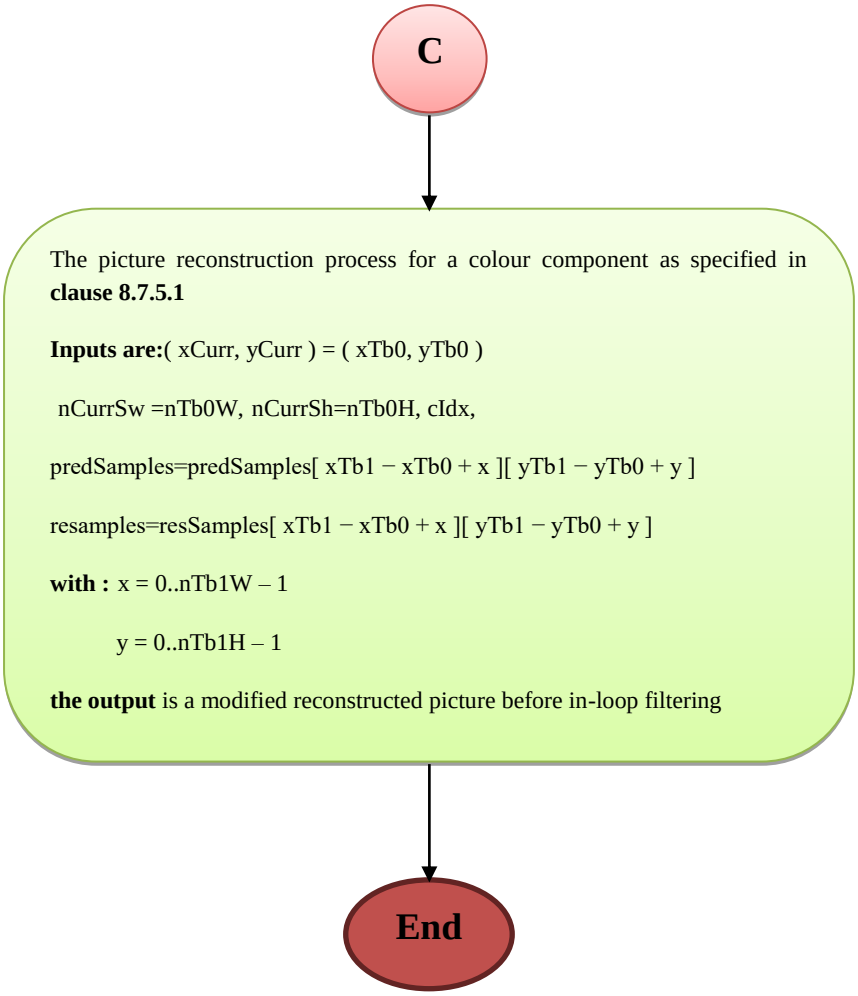


Figure III. 15: Flowchart of Decoding process for the reconstructed signal of chroma coding blocks coded in inter prediction mode.

III.3.6.1: Matlab implementation and simulation result

The implementation and simulation result of the decodig process for reconstructed signal of chroma coding blocks coded in inter prediction mode are given on the Figure III.16. The simulation of this process represent a part of picture of 4*4 size.

```
function  reconPic = reconstructBlock(~, ~, ~, ~, ~, ~, ~, ~)
    % Example usage and input values
    xTb0 = 2; % Sample location x-coordinate
    yTb0 = 4; % Sample location y-coordinate
    nTbW = 4; % Transform block width
    nTbH = 4; % Transform block height
    cIdx = 1; % Color component index (0 for luma, 1 or 2 for chroma components)
    MaxTbSizeY = 8; % Example value, set according to your context
    SubWidthC = 2; % Example value for chroma sub-sampling
    SubHeightC = 2; % Example value for chroma sub-sampling

    % Initialize prediction and residual samples
    predSamples = randi(255, nTbW, nTbH) - 128; % Example prediction samples
    resSamples = randi(255, nTbW, nTbH) - 128; % Example residual samples
    reconPic = predSamples + resSamples;
    % Display the reconstructed picture
    disp('Reconstructed Picture:');
    disp(reconPic);
end
```

Command Window

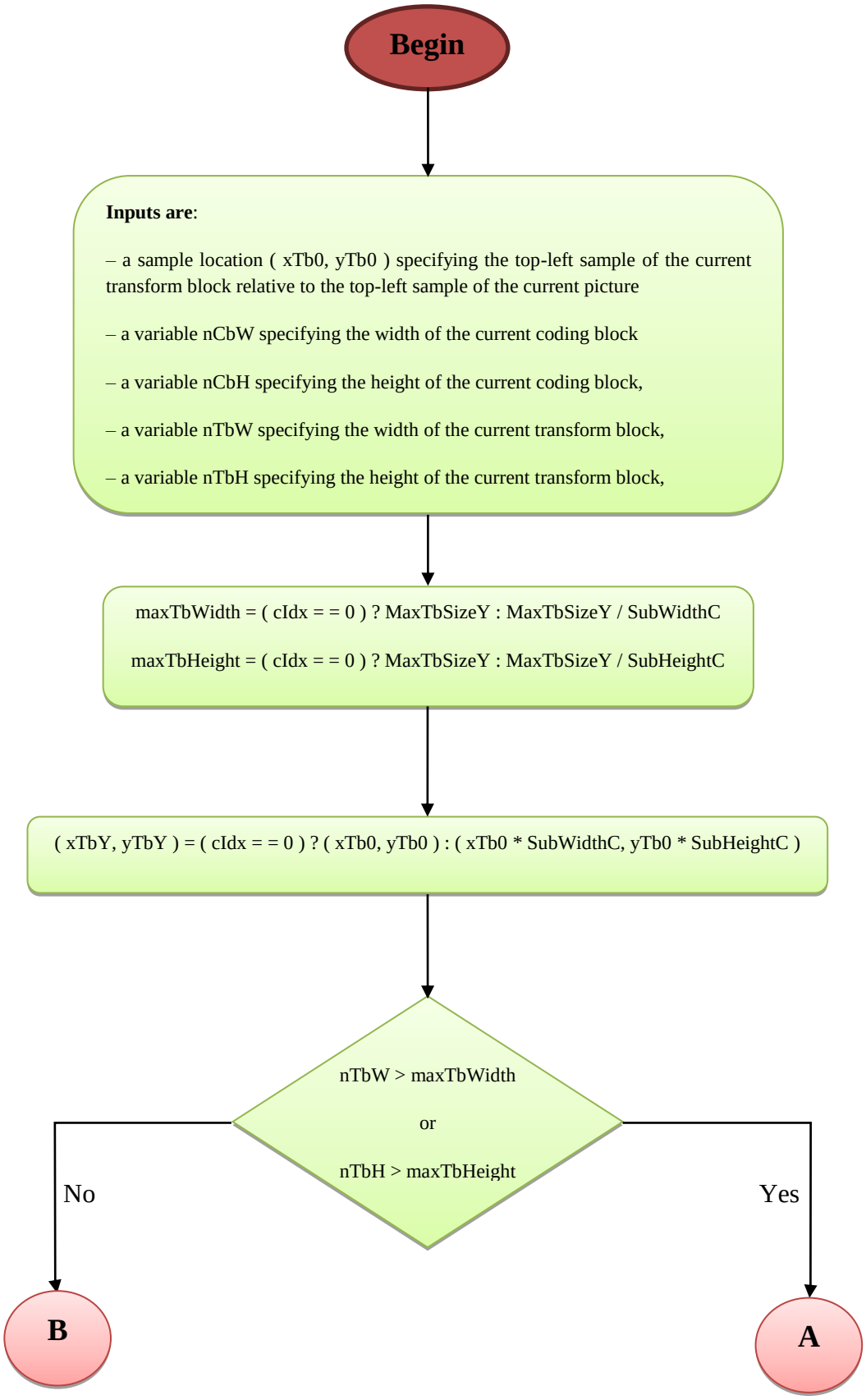
New to MATLAB? See resources for Getting Started.

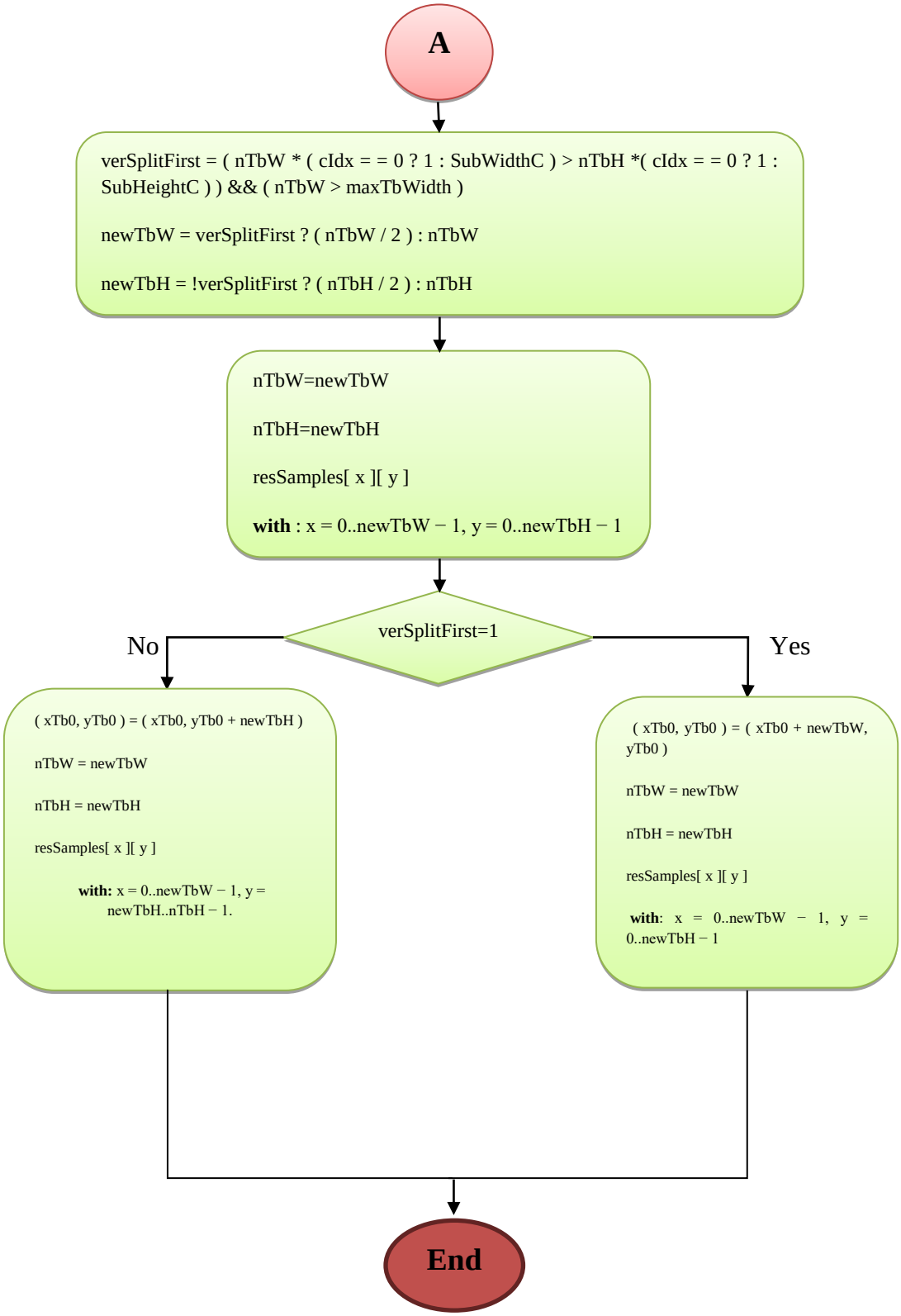
```
>> reconstructBlock
Reconstructed Picture:
    98   -51   -70    28
   141   -91  -110   105
   -56    -6    78   121
   155  -171   104   159
```

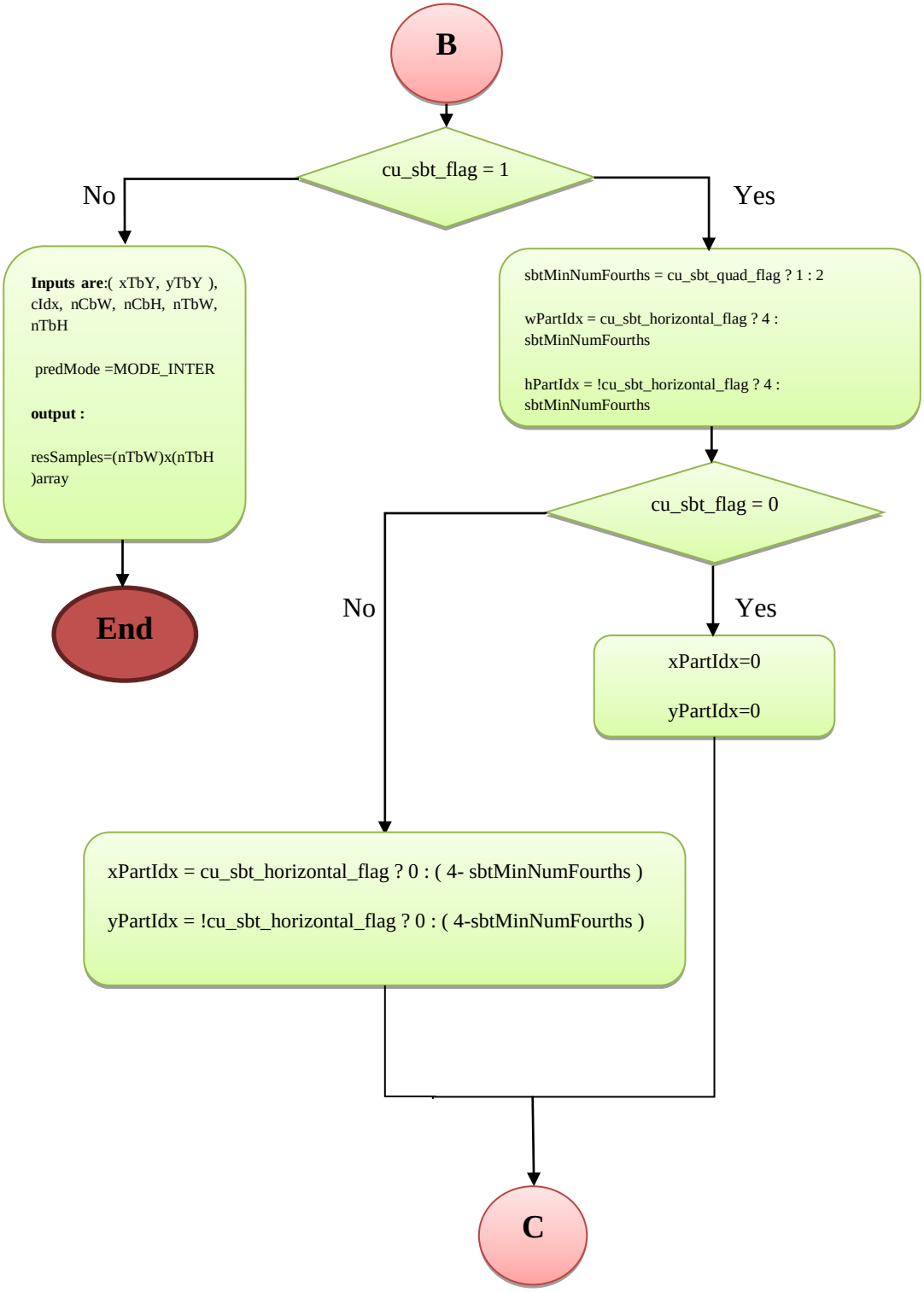
Figure III. 16: Matlab implementation and simulation result of decoding process for the reconstructed signal of chroma coding blocks coded in inter prediction mode.

III.3.7 Decoding process for the residual signal of coding blocks coded in inter prediction mode

Inputs to this process are: a sample location (xTb0, yTb0) specifying the top-left sample of the current transform block relative to the top-left sample of the current picture, a variable nCbW specifying the width of the current coding block, a variable nCbH specifying the height of the current coding block, a variable nTbW specifying the width of the current transform block, a variable nTbH specifying the height of the current transform block, a variable cIdx specifying the colour component of the current block [31].


Begin
Inputs are:
– a sample location (xTb0, yTb0) specifying the top-left sample of the current transform block relative to the top-left sample of the current picture
– a variable nCbW specifying the width of the current coding block
– a variable nCbH specifying the height of the current coding block,
– a variable nTbW specifying the width of the current transform block,
– a variable nTbH specifying the height of the current transform block,
maxTbWidth = (cIdx = = 0) ? MaxTbSizeY : MaxTbSizeY / SubWidthC
maxTbHeight = (cIdx = = 0) ? MaxTbSizeY : MaxTbSizeY / SubHeightC
(xTbY, yTbY) = (cIdx = = 0) ? (xTb0, yTb0) : (xTb0 * SubWidthC, yTb0 * SubHeightC)
nTbW > maxTbWidth
or
nTbH > maxTbHeight
No
Yes
B
A


A
verSplitFirst = (nTbW * (cIdx = = 0 ? 1 : SubWidthC) > nTbH *(cIdx = = 0 ? 1 : SubHeightC)) && (nTbW > maxTbWidth)
newTbW = verSplitFirst ? (nTbW / 2) : nTbW
newTbH = !verSplitFirst ? (nTbH / 2) : nTbH
nTbW=newTbW
nTbH=newTbH
resSamples[x][y]
with : x = 0..newTbW − 1, y = 0..newTbH − 1
verSplitFirst=1
No
Yes
(xTb0, yTb0) = (xTb0, yTb0 + newTbH)
nTbW = newTbW
nTbH = newTbH
resSamples[x][y]
with: x = 0..newTbW − 1, y = newTbH..nTbH − 1.
(xTb0, yTb0) = (xTb0 + newTbW, yTb0)
nTbW = newTbW
nTbH = newTbH
resSamples[x][y]
with: x = 0..newTbW − 1, y = 0..newTbH − 1
End


B
cu_sbt_flag = 1
No
Yes
Inputs are:(xTbY, yTbY), cIdx, nCbW, nCbH, nTbW, nTbH
predMode =MODE_INTER
output :
resSamples=(nTbW)x(nTbH)array
End
sbtMinNumFourths = cu_sbt_quad_flag ? 1 : 2
wPartIdx = cu_sbt_horizontal_flag ? 4 : sbtMinNumFourths
hPartIdx = !cu_sbt_horizontal_flag ? 4 : sbtMinNumFourths
cu_sbt_flag = 0
No
Yes
xPartIdx=0
yPartIdx=0
xPartIdx = cu_sbt_horizontal_flag ? 0 : (4- sbtMinNumFourths)
yPartIdx = !cu_sbt_horizontal_flag ? 0 : (4-sbtMinNumFourths)
C

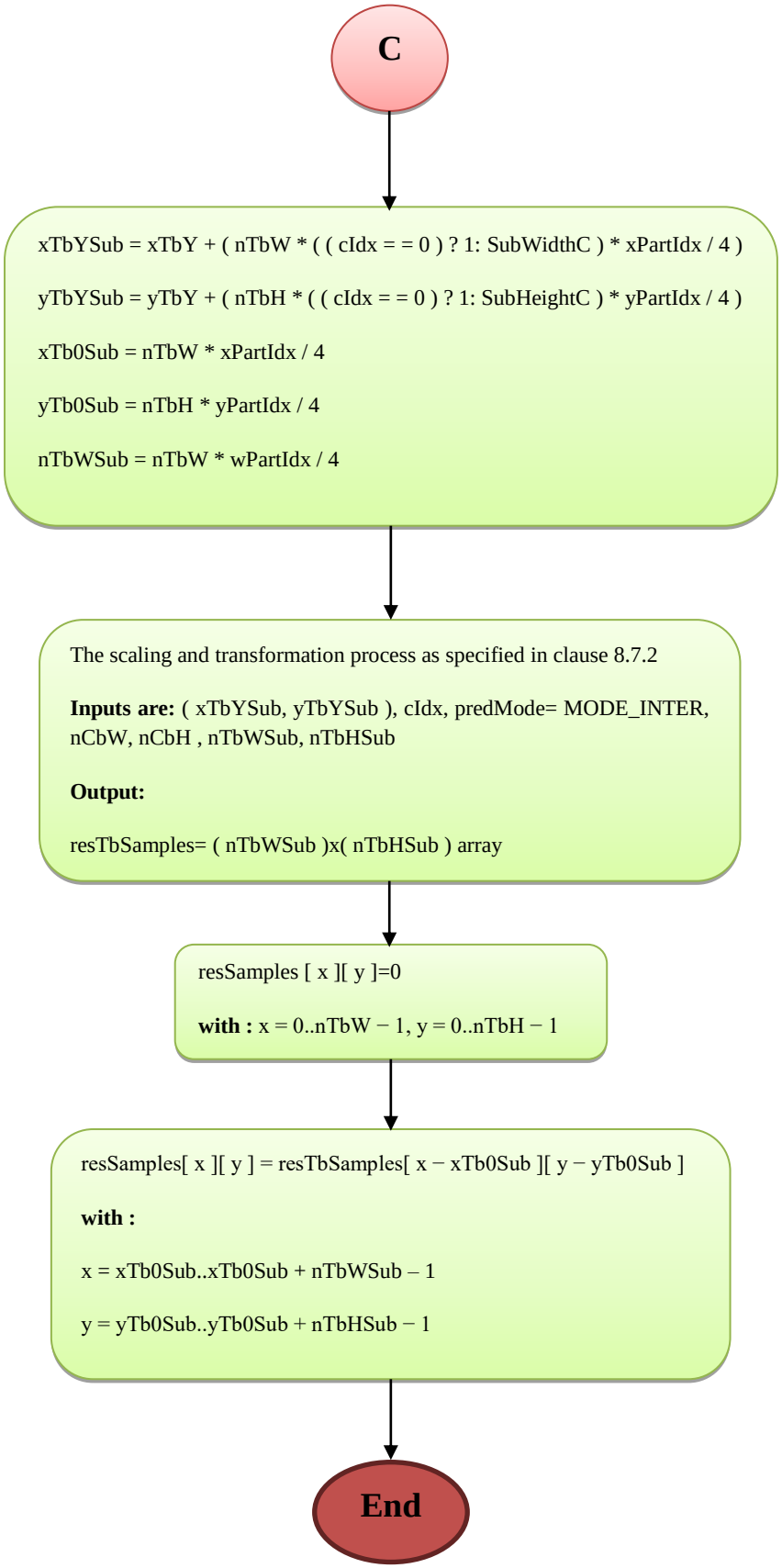


Figure III. 17: Flowchart of decoding process for the residual signal of coding blocks coded in inter prediction mode.

III.3.7.1: Matlab implementation and simulation result

The implementation and simulation result of Decoding process for the residual signal of coding blocks coded in inter prediction mode are given in Figure III.18.

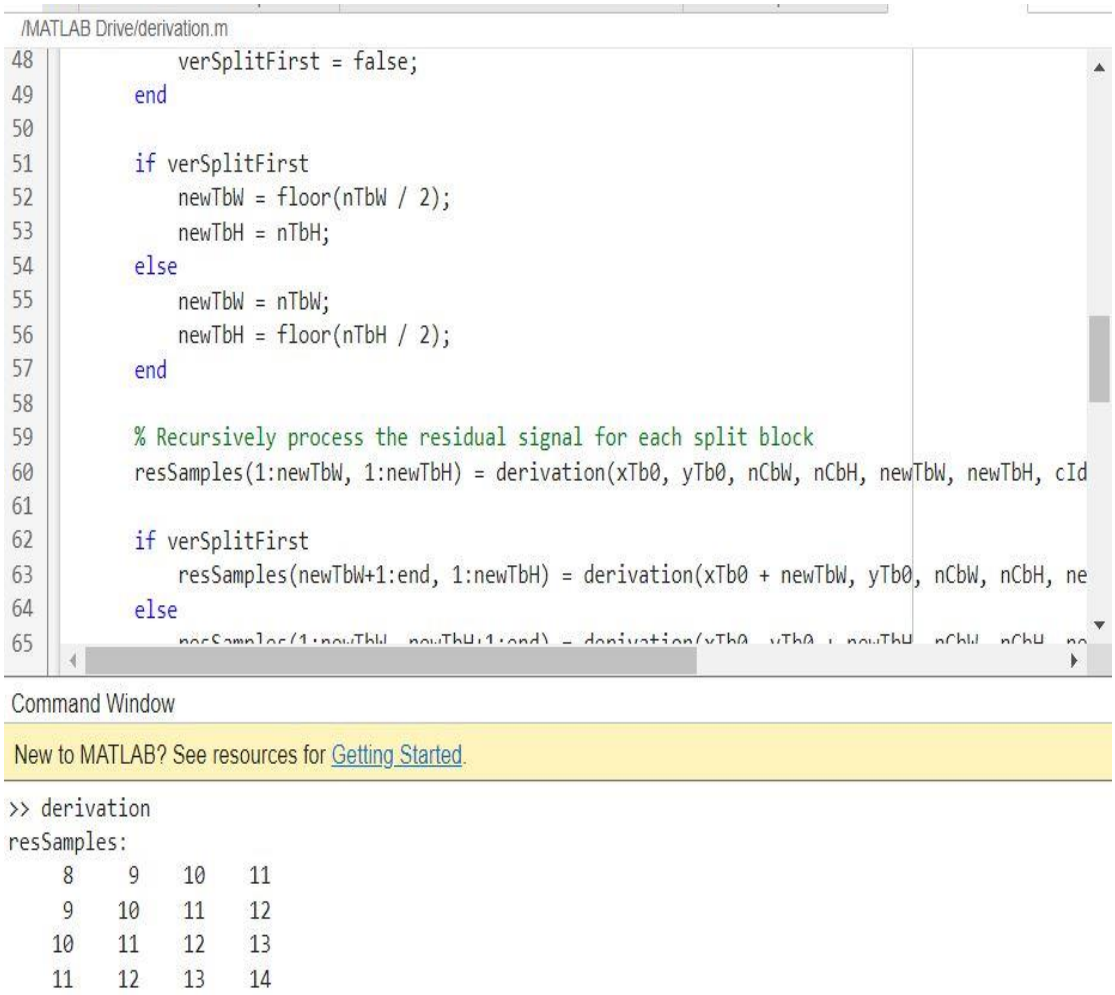
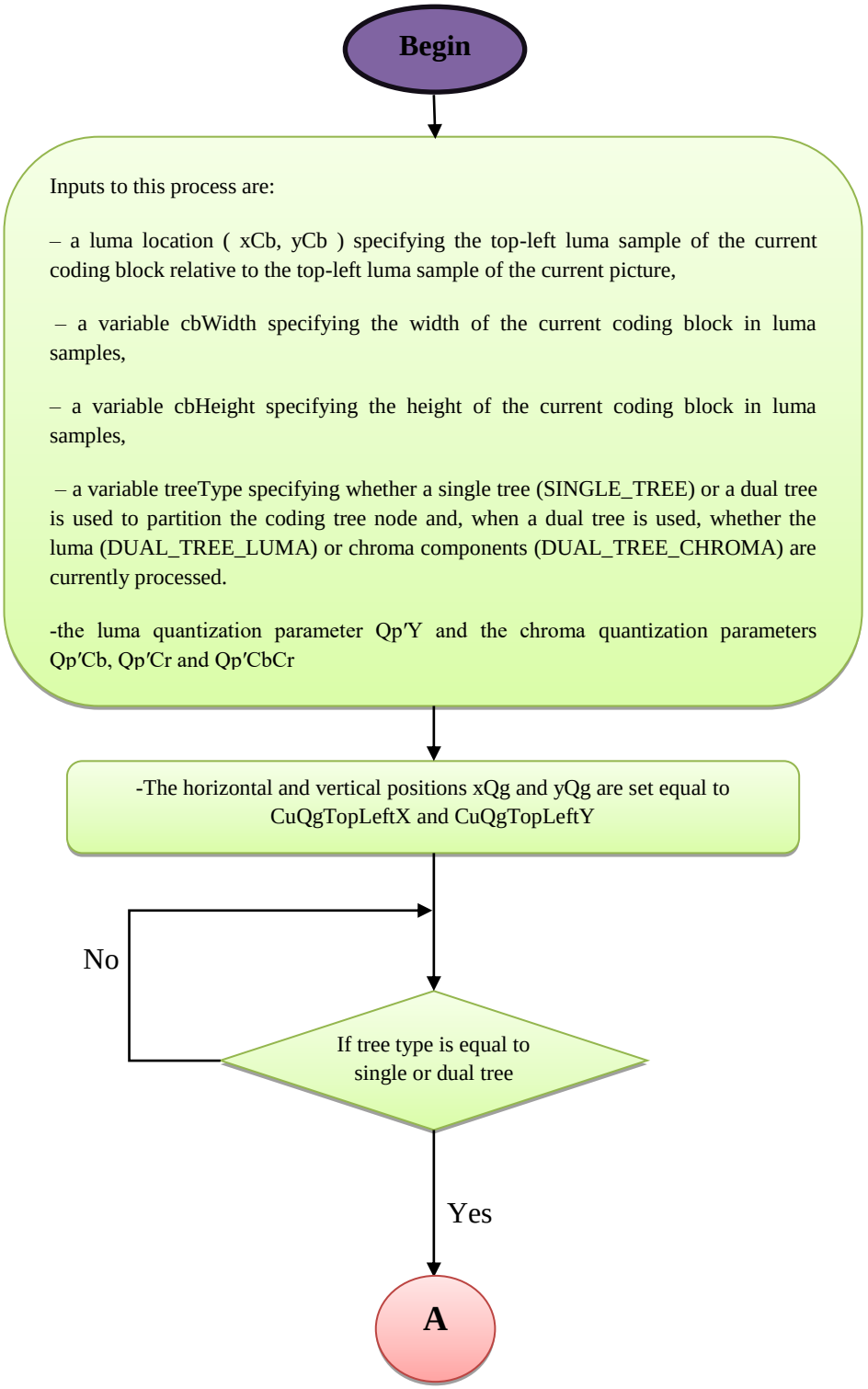
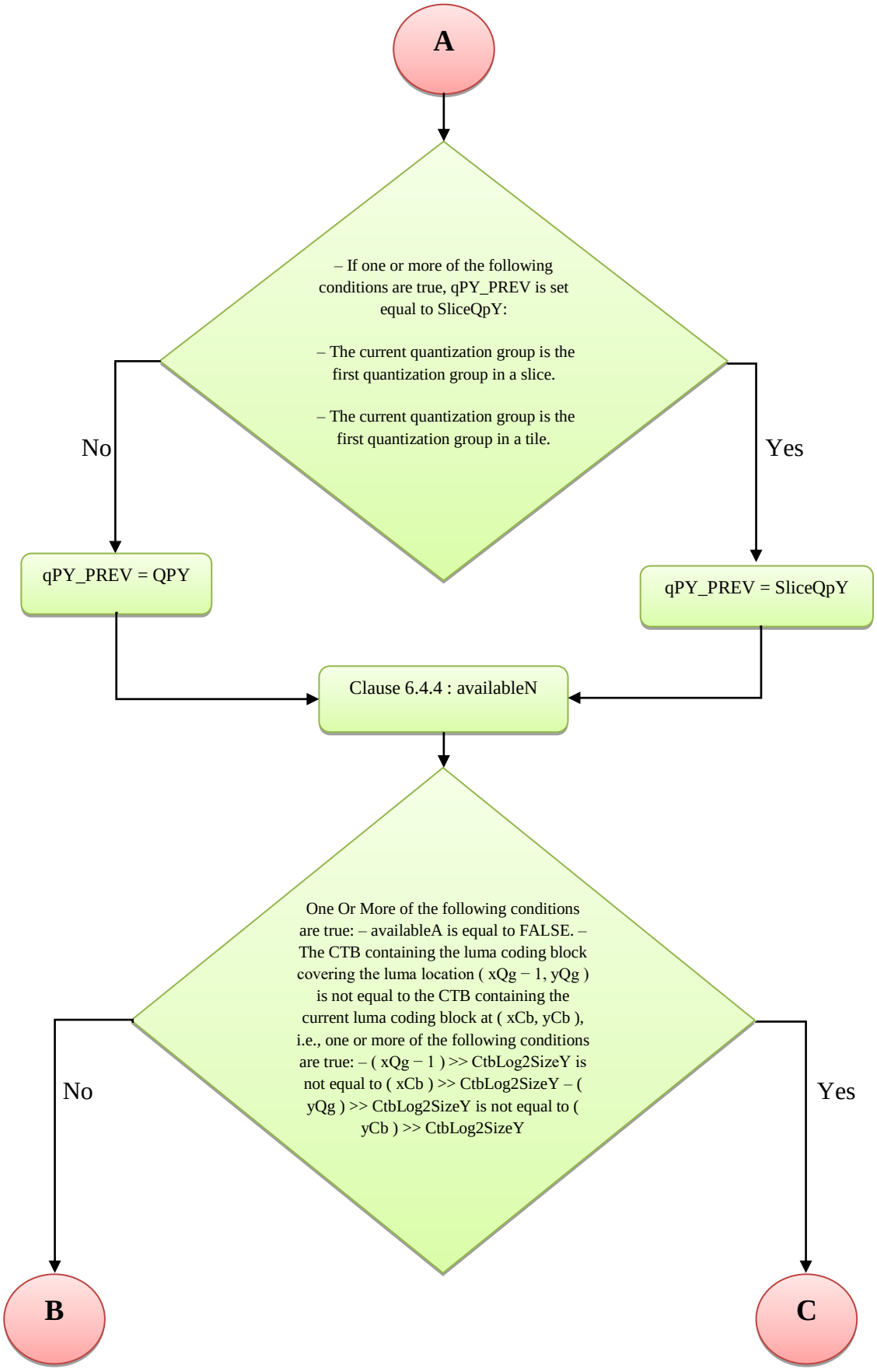


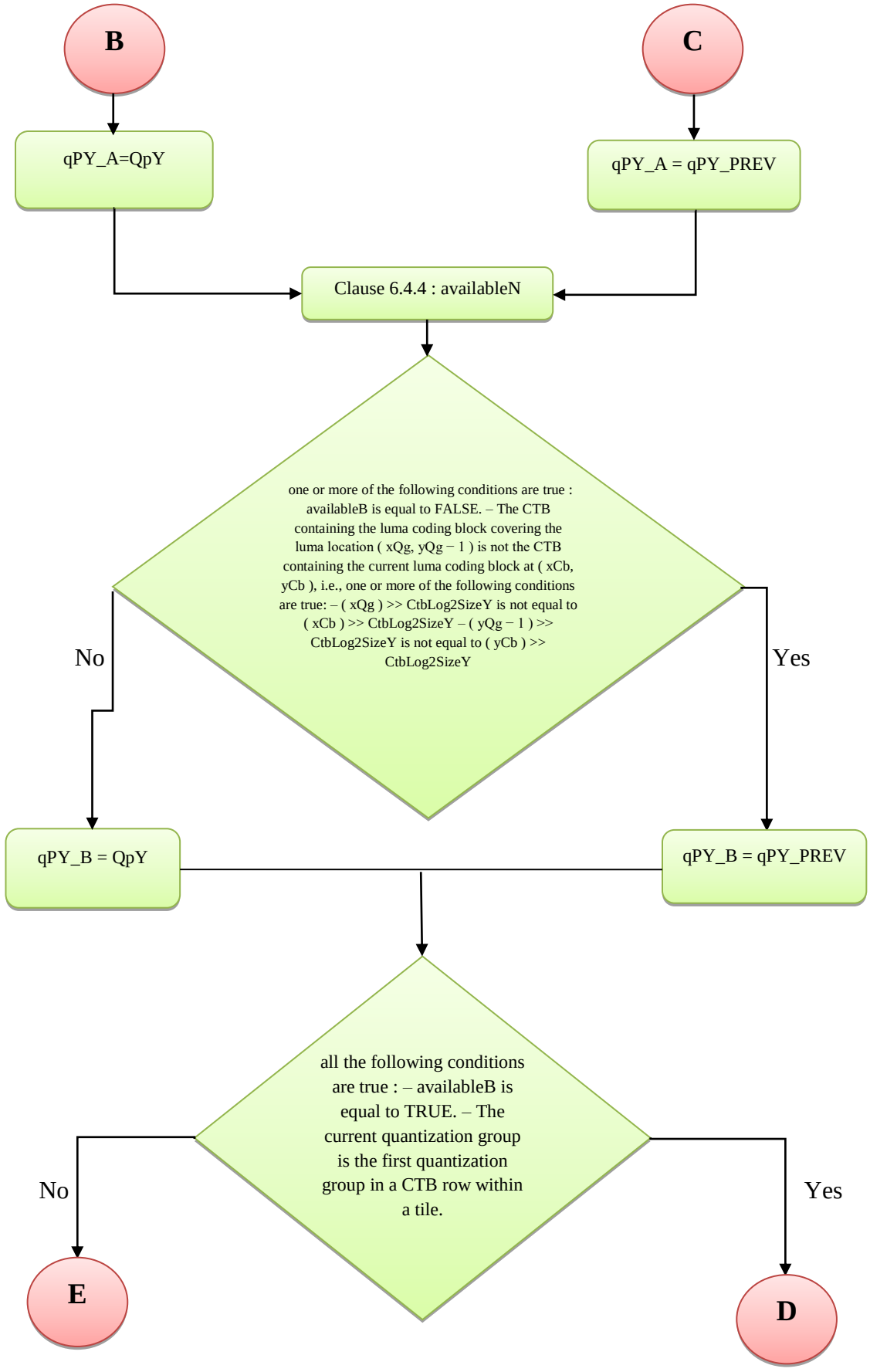
Figure III. 18: Matlab implementation and simulation result of decoding process for the residual signal of coding blocks coded in inter prediction mode.

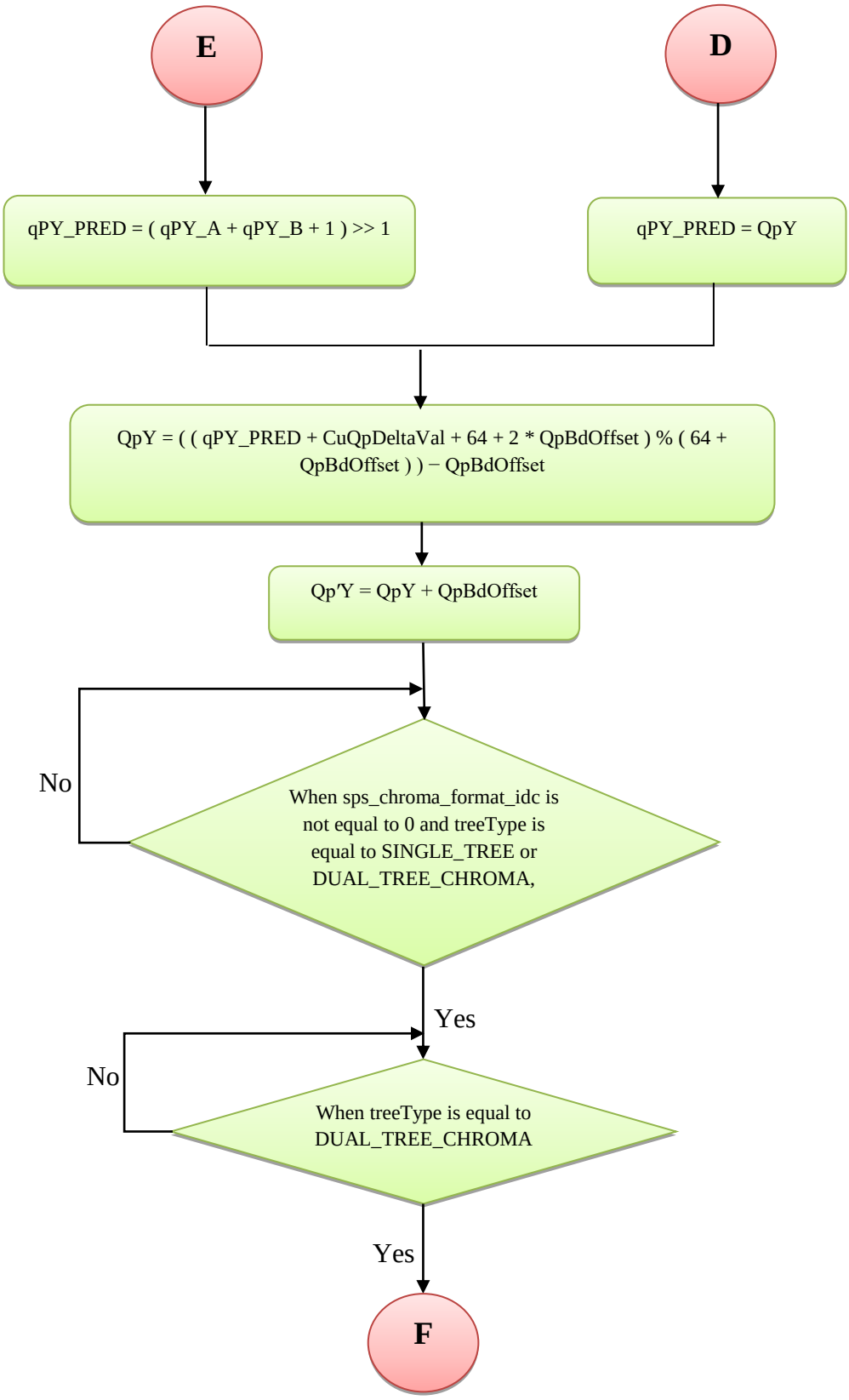
III.3.8 Derivation process for quantization parameters:

Inputs to this process are: a luma location (xCb, yCb) specifying the top-left luma sample of the current coding block relative to the top-leftluma sample of the current picture, cbWidth specifying the width of the current coding block in luma samples, e cbHeight specifying the height of the current coding block in luma samples, a variable treeType specifying whether a single tree (SINGLE_TREE) or a dual tree is used to partition the coding tree node and, when a dual tree is used, whether the luma (DUAL_TREE_LUMA) or chroma components (DUAL_TREE_CHROMA) are currently processed [31].


Begin
Inputs to this process are:
– a luma location (xCb, yCb) specifying the top-left luma sample of the current coding block relative to the top-left luma sample of the current picture,
– a variable cbWidth specifying the width of the current coding block in luma samples,
– a variable cbHeight specifying the height of the current coding block in luma samples,
– a variable treeType specifying whether a single tree (SINGLE_TREE) or a dual tree is used to partition the coding tree node and, when a dual tree is used, whether the luma (DUAL_TREE_LUMA) or chroma components (DUAL_TREE_CHROMA) are currently processed.
-the luma quantization parameter Qp′Y and the chroma quantization parameters Qp′Cb, Qp′Cr and Qp′CbCr
-The horizontal and vertical positions xQg and yQg are set equal to CuQgTopLeftX and CuQgTopLeftY
No
If tree type is equal to single or dual tree
Yes
A


A
– If one or more of the following conditions are true, qPY_PREV is set equal to SliceQpY:
– The current quantization group is the first quantization group in a slice.
– The current quantization group is the first quantization group in a tile.
No
Yes
qPY_PREV = QPY
qPY_PREV = SliceQpY
Clause 6.4.4 : availableN
One Or More of the following conditions are true: – availableA is equal to FALSE. – The CTB containing the luma coding block covering the luma location (xQg − 1, yQg) is not equal to the CTB containing the current luma coding block at (xCb, yCb), i.e., one or more of the following conditions are true: – (xQg − 1) >> CtbLog2SizeY is not equal to (xCb) >> CtbLog2SizeY – (yQg) >> CtbLog2SizeY is not equal to (yCb) >> CtbLog2SizeY
No
Yes
B
C


B
C
qPY_A=QpY
qPY_A = qPY_PREV
Clause 6.4.4 : availableN
one or more of the following conditions are true : availableB is equal to FALSE. – The CTB containing the luma coding block covering the luma location (xQg, yQg − 1) is not the CTB containing the current luma coding block at (xCb, yCb), i.e., one or more of the following conditions are true: – (xQg) >> CtbLog2SizeY is not equal to (xCb) >> CtbLog2SizeY – (yQg − 1) >> CtbLog2SizeY is not equal to (yCb) >> CtbLog2SizeY
No
Yes
qPY_B = QpY
qPY_B = qPY_PREV
all the following conditions are true : – availableB is equal to TRUE. – The current quantization group is the first quantization group in a CTB row within a tile.
No
Yes
E
D


E
D
qPY_PRED = (qPY_A + qPY_B + 1) >> 1
qPY_PRED = QpY
QpY = ((qPY_PRED + CuQpDeltaVal + 64 + 2 * QpBdOffset) % (64 + QpBdOffset)) − QpBdOffset
Qp′Y = QpY + QpBdOffset
No
When sps_chroma_format_idc is not equal to 0 and treeType is equal to SINGLE_TREE or DUAL_TREE_CHROMA,
Yes
No
When treeType is equal to DUAL_TREE_CHROMA
Yes
F

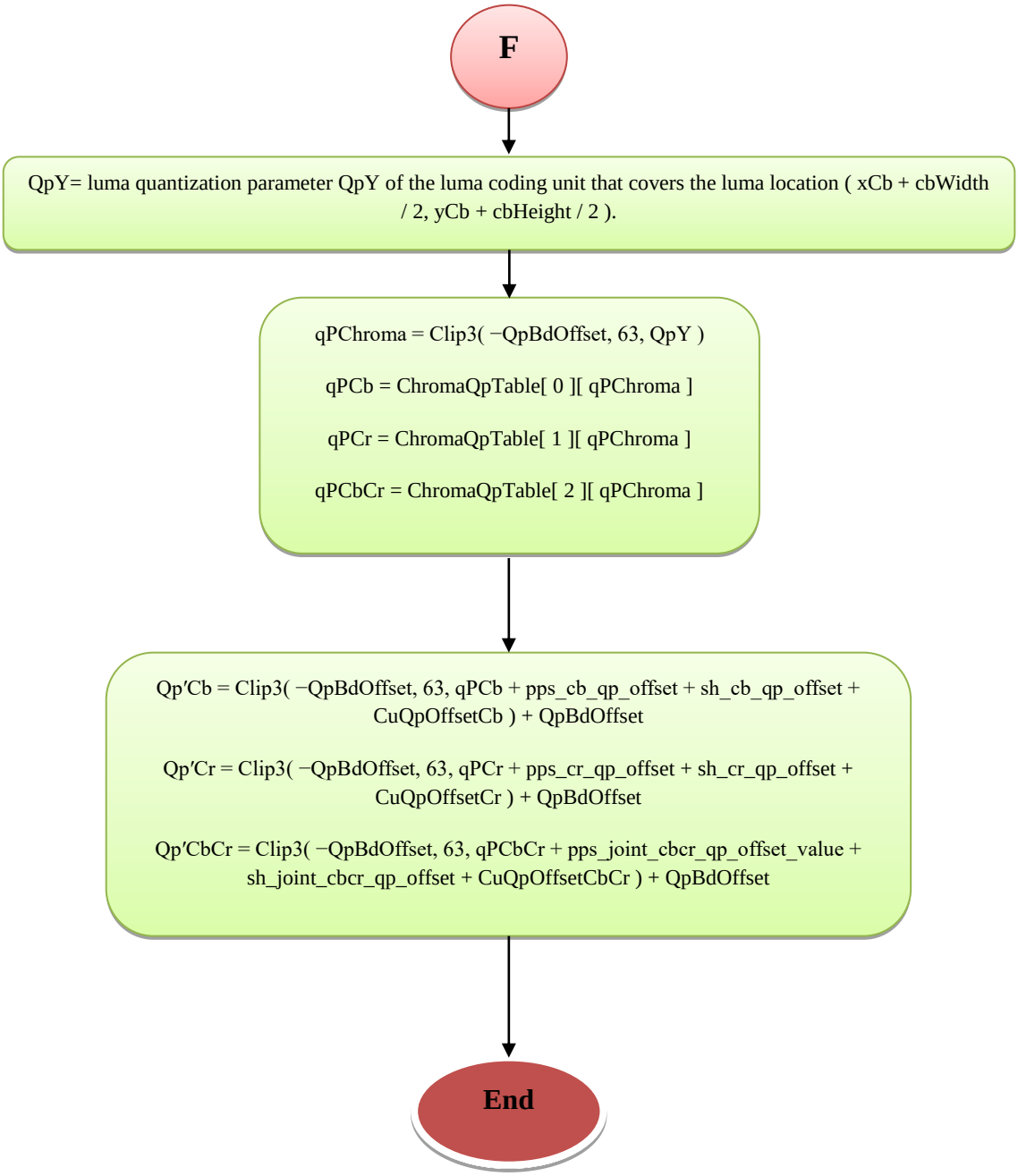


Figure III. 19: Flowchart of derivation process for quantization parameters.

III.3.8.1: Matlab implementation and simulation result

The implementation and simulation result of derivation process for quantization parameters are given in the Figure III.20.

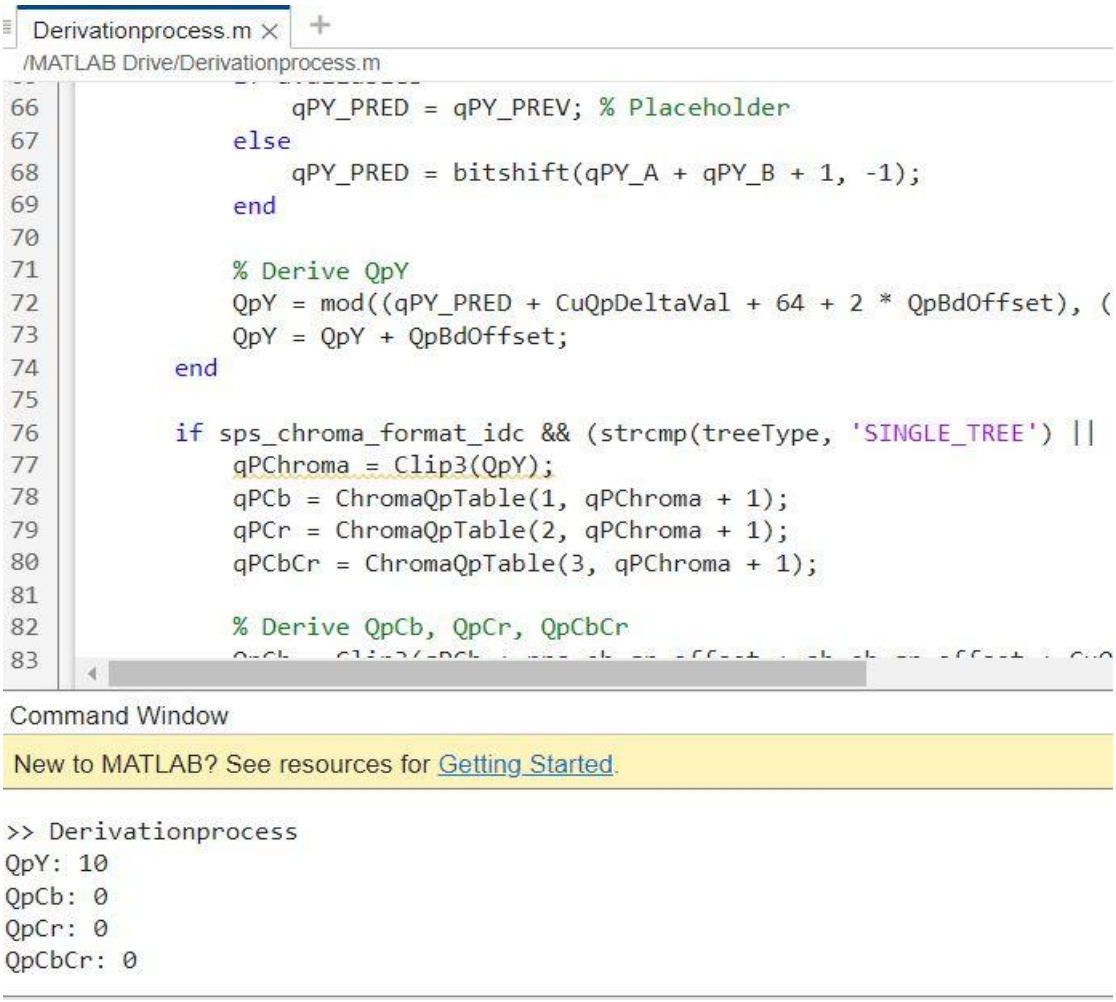


Figure III. 20: Matlab implementation and simulation result of derivation process for quantization parameters.

III.3.9 Conclusion

In this chapter, the software tools used have been presented. Then we also designed flowcharts for inter prediction modes with their implementations under Matlab. This passage allowed us to master the modes algorithms for inter prediction used in the VVC video compression standard before moving on to software description under python. For that, the next chapter will be devoted to the software implementation under python in order to validate the results obtained by Matlab.

Chapter IV

Implementation of Inter prediction in VVC under python

IV.1 Introduction

This chapter is the last one which reserved for the implementation of inter prediction derivations in versatile video coding (VVC) and simulation by python. After that, we will end the chapter with a comparison between the Matlab results and the python ones.

IV.2 Python implementation and simulation result

IV.2.1 Derivation process for block vector components for IBC block (general)

The implementation and simulation result of this process under python are given in the Figure IV.4.

```
            for y in range(yCb, yCb + cbHeight):
                assert IbcVirBuf[0][(x + (bvL[0] >> 4)) & (IbcBufWidthY - 1)][(y + (bvL

    return bvL

# Hypothetical values for parameters
xCb = 2
yCb = 2
cbWidth = 6
cbHeight = 6
general_merge_flag = [[0 for _ in range(30)] for _ in range(30)]  # 30x30 grid initiali
MvdL0 = [[[0, 0] for _ in range(30)] for _ in range(30)]  # 30x30 grid of 2D vectors in
AmvrShift = 4
CtbSizeY = 8
IbcVirBuf = [[[0 for _ in range(CtbSizeY)] for _ in range(CtbSizeY)] for _ in range(2)]
IbcBufWidthY = CtbSizeY  # Assuming IbcBufWidthY is equal to CtbSizeY
IsGt4by4 = True  # Assuming IsGt4by4 is true

# Example usage of derive_bvL function
bvL = derive_bvL(xCb, yCb, cbWidth, cbHeight, general_merge_flag, MvdL0, AmvrShift, Ctb

# Print the result

print("Derived luma block vector bvL:", bvL)
```

```
Derived luma block vector bvL: [0, 0]
```

Figure IV. 1: Implementation and simulation result of derivation process for block vector components for IBC block (general) under python.

IV.2.2 Picture reconstruction process (general)

The implementation and simulation result of picture reconstruction process (general) under python are given in the Figure IV.2.

```
    for i in range(nCurrSw):
        for j in range(nCurrSh):
            if sh_lmcs_used_flag == 0:
                recSamples[xCurr + i, yCurr + j] = Clip1(predSamples[i, j] + resSa
            else:
                recSamples = np.array([1, 2, 3, 5])  # Placeholder for other recon

            xVb = (xCurr + i) % (IbcBufWidthY if cIdx == 0 else IbcBufWidthC)
            yVb = (yCurr + j) % (CtbSizeY if cIdx == 0 else (CtbSizeY // SubHeight
            # Assuming IbcVirBuf is defined elsewhere in the code
            # IbcVirBuf[cIdx][xVb, yVb] = recSamples[xCurr + i, yCurr + j]

    # Assuming IsAvailable is a global variable
    # global IsAvailable
    # for i in range(nCurrSw * SubWidthC):
    #     for j in range(nCurrSh * SubHeightC):
    #         IsAvailable[cIdx][xCurr * SubWidthC + i, yCurr * SubHeightC + j] = T

    # Display the generated matrices

    print('Reconstructed Samples (recSamples):')
    print(recSamples)

# Call the function to display the matrices
reconstructBlock()
```

```
Reconstructed Samples (recSamples):
[[ 86 115]
 [  0   0]]
```

Figure IV. 2: Implementation and simulation result of picture reconstruction process under python.

IV.2.3 Derivation process for chroma block vectors

The implementation and simulation of the derivation process for chroma block vectors are presented in the Figure IV.3.

```
import numpy as np

def bvC():
    bvL = [16, 4]
    SubWidthC = 2
    SubHeightC = 2

    # Initialize the chroma block vector
    bvC = np.zeros(2, dtype=int)

    # Derive chroma block vector
    bvC[0] = (bvL[0] >> (3 + SubWidthC)) * 32
    bvC[1] = (bvL[1] >> (3 + SubHeightC)) * 32

    return bvC

bvC_result = bvC()

print('bvC[0]:', bvC_result[0])
print('bvC[1]:', bvC_result[1])
print('bvC:', bvC_result)
```

```
bvC[0]: 0
bvC[1]: 0
bvC: [0 0]
```

Figure IV. 3: Implementation and simulation for chroma block vectors.

IV.2.4 Residual modification process for blocks using color space conversion

The Figure IV.4 represent the implementation of derivation process for blocks using color space conversion in the first step. Otherwise, in the second part the simulation of this implementation is given which is presented by three matrix ry, rcb and rcr respectively.

```python
import numpy as np

def ahmed():
    nTbW = 4
    nTbH = 4
    BitDepth = 8

    # Specified input matrices
    rY = np.array([[0, 2, 1, 0], [0, 0, 1, 1], [0, 2, 2, 1], [2, 0, 2, 1]], dtype=np.int32)
    rCb = np.array([[2, 1, 2, 2], [2, 0, 1, 0], [0, 0, 1, 1], [2, 1, 0, 2]], dtype=np.int32)
    rCr = np.array([[2, 0, 2, 0], [2, 0, 0, 0], [1, 2, 2, 1], [0, 0, 1, 1]], dtype=np.int32)

    # Display initial matrices


    # Clip3 function
    def Clip3(x):
        return max(min(x, (1 << (BitDepth + 1)) - 1), -(1 << (BitDepth + 1)))

    for x in range(min(nTbW, rY.shape[0])):
        for y in range(min(nTbH, rY.shape[1])):
            # Clip values
            rY_clip = Clip3(rY[x, y])
            rCb_clip = Clip3(rCb[x, y])
            rCr_clip = Clip3(rCr[x, y])

            # Compute new values using temporary variables
            tmp = rY_clip - (rCb_clip >> 1)
```

```
Modified rY:
[[1 3 2 1]
 [1 0 2 1]
 [0 2 3 2]
 [3 1 2 2]]
Modified rCb:
[[-2  2 -1 -1]
 [-2  0  1  1]
 [ 0  1  1  1]
 [ 1  0  2  0]]
Modified rCr:
[[ 0  2  1 -1]
 [ 0  0  1  1]
 [ 1  3  3  2]
 [ 1  0  3  1]]
```

Figure IV. 4: Implementation and simulation result of residual modification process for blocks using color space conversion under python.

IV.2.5 Decoding process for IBC blocks

The implementation and simulation result of decoding process for IBC blocks in python software are presented in the following Figure IV.5. It gives at the output the prediction samples matrix (predSamples).

```
# Suggested values for testing
xCb = 0             # X coordinate of the top-left corner of the current block
yCb = 0             # Y coordinate of the top-left corner of the current block
cbWidth = 2        # Width of the current block
cbHeight = 2       # Height of the current block
bv = [4, 8]         # Motion vector for the current block
cIdx = 1            # Chroma index (0 for luma, 1 for chroma)
SubWidthC = 2       # Chroma subsampling ratio for width
SubHeightC = 2      # Chroma subsampling ratio for height
IbcBufWidthY = 6   # Width of the buffer for luma
IbcBufWidthC = 3   # Width of the buffer for chroma
CtbSizeY = 4       # Size of the coding tree block for luma

# Example buffer for IbcVirBuf
IbcVirBuf = np.random.randint(0, 255, (2, IbcBufWidthC, CtbSizeY // SubHeightC)) - 128

# Call the function
predSamples = derivePredictionSamples(xCb, yCb, cbWidth, cbHeight, bv, cIdx, SubWidthC, S

# Display the prediction samples
print('PredSamples')
print(predSamples)
```

```
PredSamples
[[-102   66]
 [-102   66]]
```

Figure IV. 5: Implementation and simulation result of decoding process for IBC blocks under python.

IV.2.6 Decoding process for the reconstructed signal of chroma coding blocks coded inter prediction mode

The implementation and simulation result of the decoding process for the reconstructed signal of chroma coding blocks coded inter prediction mode are presented in the Figure IV.6 which based on different variables and fixed values according to the VVC standard. The simulation results give at the output the reconstructed block which is a part of the picture.

```
    SubWidthC = 2  # Example value for chroma sub-sampling
    SubHeightC = 2  # Example value for chroma sub-sampling

    # Initialize prediction and residual samples
    predSamples = np.random.randint(0, 255, (nTbW, nTbH)) - 128  # Example prediction samples
    resSamples = np.random.randint(0, 255, (nTbW, nTbH)) - 128  # Example residual samples

    # Perform the reconstruction
    reconPic = predSamples + resSamples

    # Display the reconstructed picture
    print('Reconstructed Picture:')
    print(reconPic)

    return reconPic

# Call the function to reconstruct the block
reconstructed_block = reconstruct_block()
```

```
Reconstructed Picture:
[[ 130   76 -186    6]
 [ -20    9  -50   17]
 [ -85  -24   77   20]
 [ -23  -10  -89  -12]]
```

Figure IV. 6: Implementation and simulation result of decoding process for the reconstructed signal of chroma coding blocks coded inter prediction mode.

IV.2.7 Decoding process for the residual signal of coding blocks in inter prediction mode

The implementation and simulation result of the decoding process for the residual signal of coding blocks in inter prediction mode are represented in Figure IV.7.

The output of this figure shows the matrix of resSamples obtained by using different variables such as nTbw and nTbh…etc.

```
    np.ndarray: Residual samples.
    """
    # Placeholder function for the scaling and transformation process
    resSamples = np.zeros((nTbW, nTbH), dtype=int)

    # Example processing: fill the array with some values for demonstration
    for x in range(nTbW):
        for y in range(nTbH):
            resSamples[x, y] = (xTbY + x + yTbY + y) % 256  # Dummy transformation

    return resSamples

# Example usage
xTb0 = 2
yTb0 = 4
nCbW = 4
nCbH = 4
nTbW = 4
nTbH = 4
cIdx = 0
resSamples = derivation(xTb0, yTb0, nCbW, nCbH, nTbW, nTbH, cIdx)
print("resSamples")
print(resSamples)
```

```
resSamples
[[ 6  7  8  9]
 [ 7  8  9 10]
 [ 8  9 10 11]
 [ 9 10 11 12]]
```

Figure IV. 7: Implementation and simulation result of decoding process for the residual signal of coding blocks in inter prediction mode.

IV.2.8 Derivation process for quantization parameters

The quantization parameters process's implementation and simulation results are depicted in the Figure IV.8, we obtained at the output the Qp values for luminance and chrominance color representation.

```
        if availableB:
            qPY_PRED = qPY_PREV  # Placeholder for actual value
        else:
            qPY_PRED = (qPY_A + qPY_B + 1) >> 1

        QpY = ((qPY_PRED + CuQpDeltaVal + 64 + 2 * QpBdOffset) % (64 + QpBdOffset)) - QpBdOffset
        QpY += QpBdOffset

    if sps_chroma_format_idc and treeType in ["SINGLE_TREE", "DUAL_TREE_CHROMA"]:
        qPChroma = Clip3(QpY, QpBdOffset)
        ChromaQpTable = np.arange(64).reshape((3, 64))  # Example ChromaQpTable
        qPCb = ChromaQpTable[0, qPChroma]
        qPCr = ChromaQpTable[1, qPChroma]
        qPCbCr = ChromaQpTable[2, qPChroma]

        QpCb = Clip3(qPCb + pps_cb_qp_offset + sh_cb_qp_offset + CuQpOffsetCb, QpBdOffset) + QpBdOffset
        QpCr = Clip3(qPCr + pps_cr_qp_offset + sh_cr_qp_offset + CuQpOffsetCr, QpBdOffset) + QpBdOffset
        QpCbCr = Clip3(qPCbCr + pps_cb_qp_offset + sh_cb_qp_offset + CuQpOffsetCb, QpBdOffset) + QpBdOffset

    return QpY, QpCb, QpCr, QpCbCr

# Example usage
QpY, QpCb, QpCr, QpCbCr = Derivationprocess()
print(f"QpY: {QpY}, QpCb: {QpCb}, QpCr: {QpCr}, QpCbCr: {QpCbCr}")
```

```
QpY: 10, QpCb: 0, QpCr: 0, QpCbCr: 0
```

Figure IV. 8: Implementation and simulation result of derivation process for quantization parameters.

IV.3 Comparison and validation of the results obtained

The following figures show us that we got the same results in the two languages codes (Matlab and Python) for four inter prediction modes (picture reconstruction process, derivation process for chroma block vectors, residual modification process for blocks using color space conversion, and derivation process for quantization parameters) respectively.

In the rest of derivation process (decoding process for IBC blocks, decoding process for the reconstructed signal of chroma coding blocks coded in inter prediction mode, and decoding process for the residual signal of coding blocks coded in inter prediction mode) we have not got the same results because we have generated some data at the inputs with random functions.

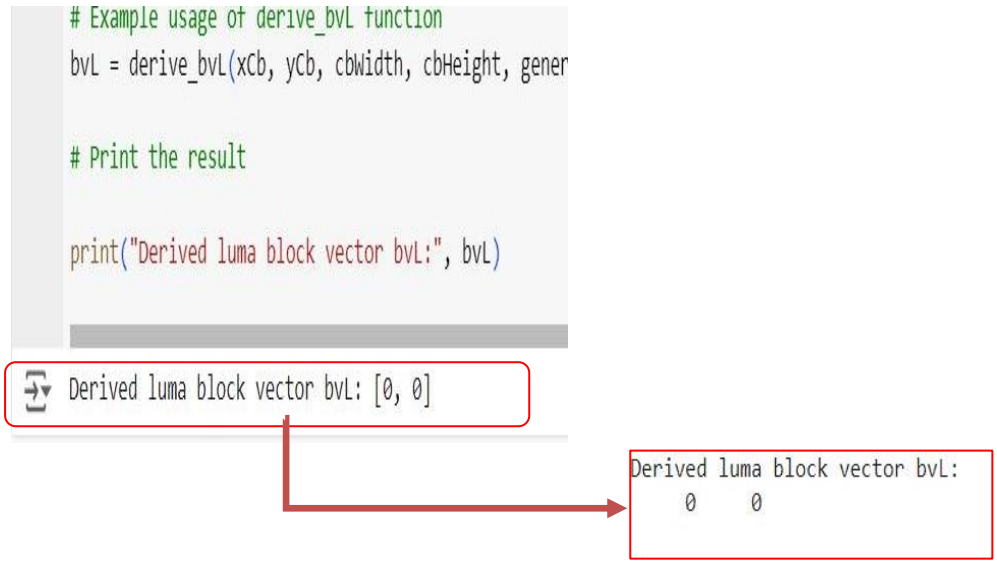


Figure IV. 9: Comparison results obtained by Matlab and python for derivation process for block vector camponents for IBC blocks (general).

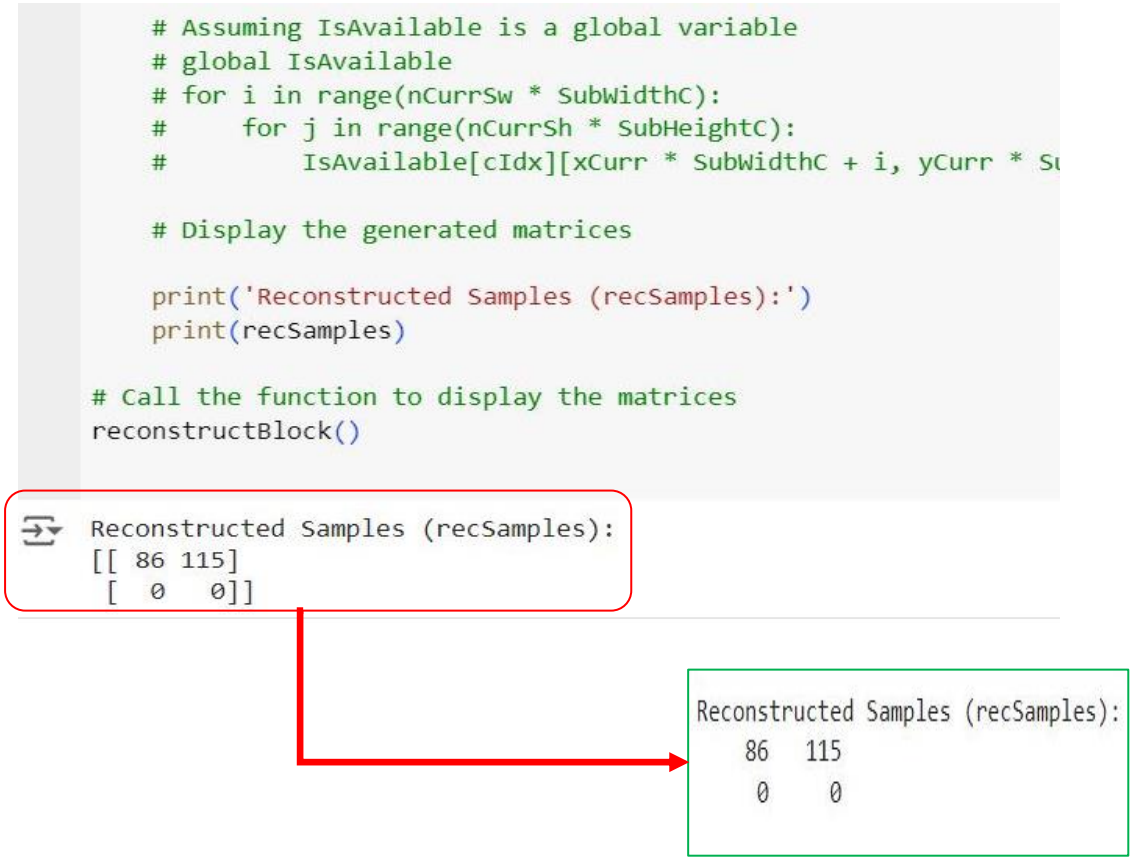


Figure IV. 10: Comparison results obtained by Matlab and python for the reconstructed samples derivation.

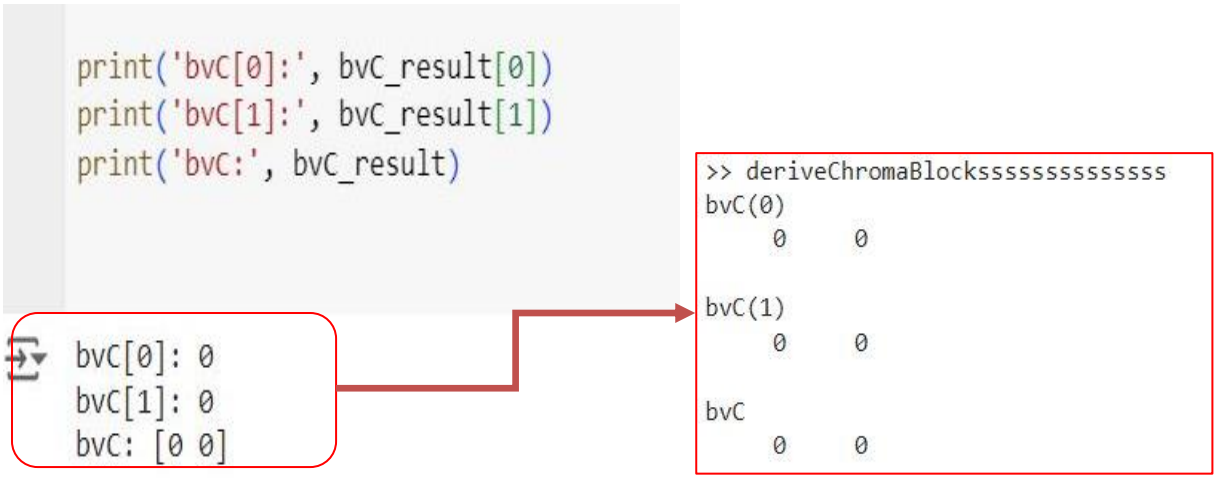


Figure IV. 11: Comparison results obtained by Matlab and python for derivation process for chroma block vectors.

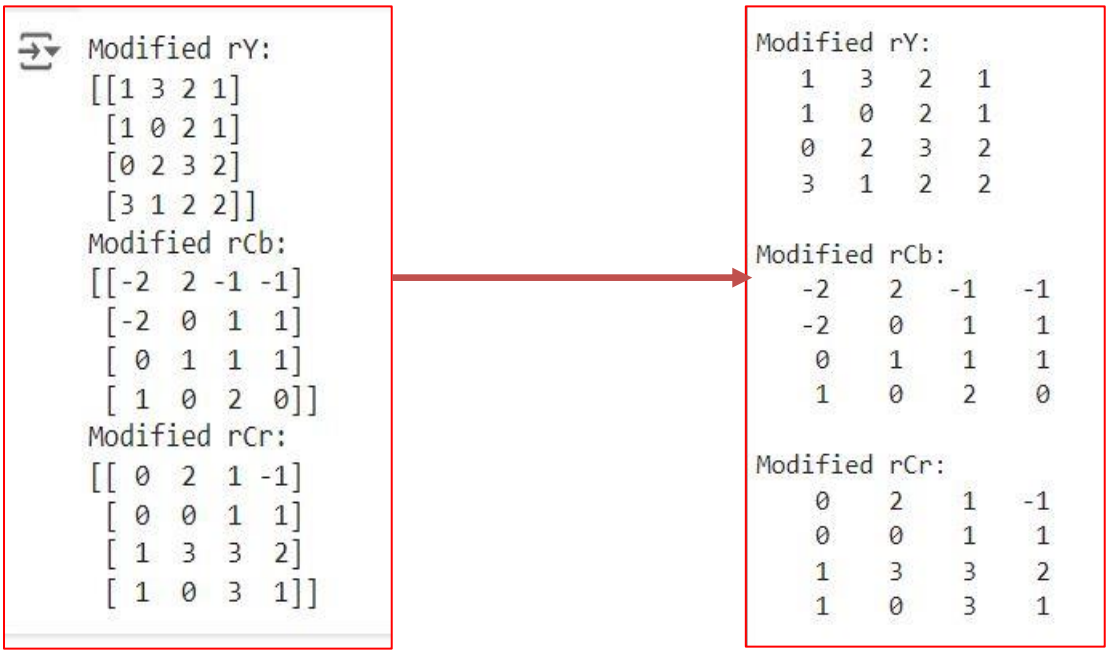


Figure IV. 12: Comparison results obtained by Matlab and python for residual modification derivation of blocks using color space conversion.

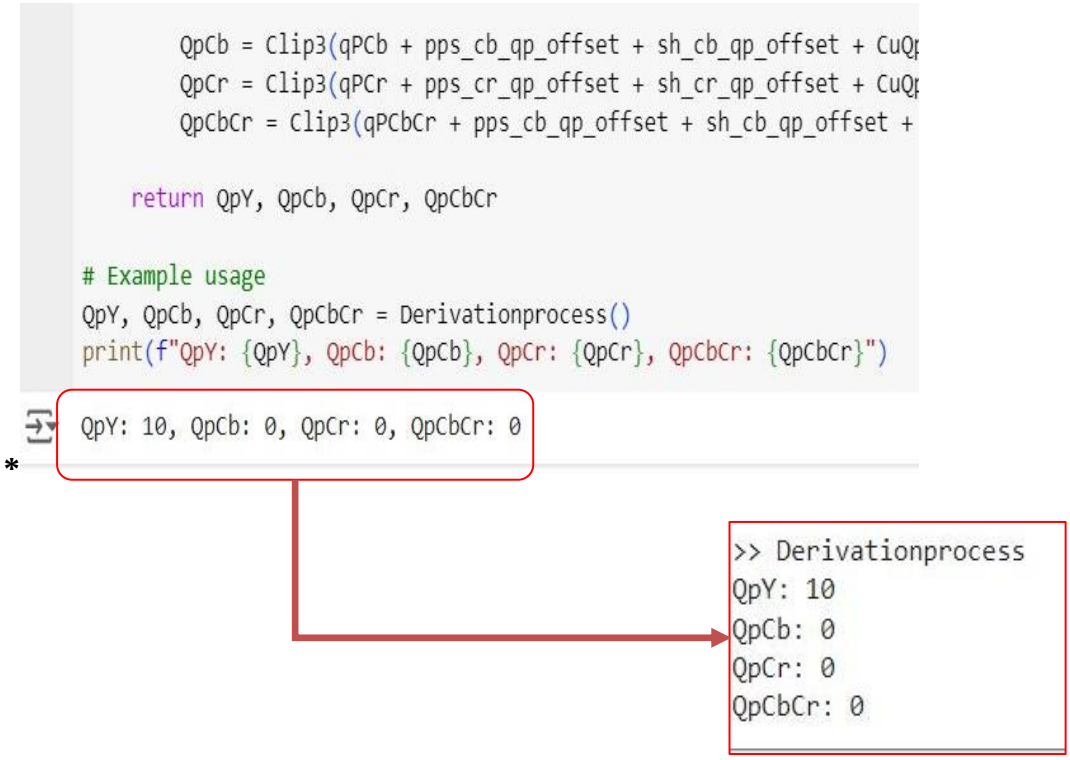


Figure IV. 13: Comparison results obtained by Matlab and python for quantization parameters derivation.

We can see that the results of both Matlab and python environments obtained for the five derivation algorithms:

- Derivation process for block vector components for IBC block (general).
- Picture reconstruction process (general).
- Derivation process for chroma block vectors.
- Residual modification process for blocks using color space conversion.
- Derivation process for quantization parameters.

are the same and we confirm that our implementations of these algorithms are exact.

IV.4 Conclusion

In this chapter, we used python language for validate our results obtained by Matlab for each inter prediction mode derivation. So, we have been compare the results obtained for the both software codes. The same results are obtained, so our proposed work validate.

General conclusion

General Conclusion

Videos compression is an essential process before they can be transmitted or broadcast on the internet. The objective of our work was to study and implement on Matlab and python the inter prediction modes algorithms in the VVC/H.166 video compression standard. Inter prediction is based on different derivation modes to improve compression efficiency.

Our work is organized mainly as follows:

Firstly, a general explanation of the concept of video compression technology. Then, recent video compression standards have been given and followed by VVC compression technics details. Next, we highlight the different modes of inter prediction introduced by VVC. Some of these technics are developed, and others are new adopted by VVC codec to improve coding efficiency.

Secondly, we design the flowcharts of the modes of inter prediction with implementation under Matlab. This transition from the implementation in Matlab enabled us to master inter prediction algorithms used in the VVC video compression standard.

Thirdly, we used python software in the last step in order to validate the results obtained with Matlab.

In the major cases, the results are the same except in decoding process for IBC blocks, decoding process for the reconstreucted signal of chroma coding blocks coded in inter prediction, and decoding process for the residual signal of coding blocks coded in inter prediction because we used at the inputs a random vectors, so the results are not the same.

As future work, we propose to implement our algorithms of inter prediction on FPGA platform by using vivado description environment and optimize the power consumption of this algorithm because it based on hard calculation.

References

[1]

Video Compression Handbook" by Andy Beach, 2018

[2]

Cisco, V.: Cisco visual networking index: Forecast and trends, 2017–2022. In: Cisco Systems White Paper. 1(1), (2018)

[3]

Menasri, W., Skoudarli, A.: Performance comparison of throughput between AVC, HEVC and VVC hardware CABAC decoder. Journal of Real-Time Image Proc. 20(2), 26 (2023).

[4]

Bross, B., Chen, J., Ohm, J.R., Sullivan, G.J., Wang, Y.K: Developments in international video coding standardization after AVC, with an overview of versatile video coding (VVC). In: Proceedings of the IEEE 109(9), 1463–1493 (2021).

[5]

Pakdaman, F., Adelimanesh, M.A., Gabbouj, M., Hashemi, M.R.: Complexity Analysis of Next-Generation VVC Encoding And Decoding. In Proc. IEEE Int. Conf. on Image Proc. (ICIP), Abu Dhabi, UAE, (2020).

[6]

Saldanha, M., Corrêa, M.C., Palomino, D., Porto, M., Zatt, B., Agostini, L.: An overview of dedicated hardware designs for state-of-the-art AV1 and H.266/VVC video codecs. In Proc. IEEE Int. Conf. on Electronics, Circuits and Systems (ICECS), Glasgow, UK, (2020)

[7]

"Introduction to Video Compression: From Simple Concepts to Complex Systems" by K.R. Rao and Do Nyeon Kim. (2019).

[8]

Touati R, "Reconnaissance des actions humaines à partir d‟une séquence vidéo", Mémoire pour l‟obtention du grade de Maîtres sciences (M.Sc.) en informatique, Université de Montréal, (Janvier 2014).

[9]

Mark Nelson and Jean-loup Gailly, “The data compression book” second edition, League for Programming Freedom, Cambridge MA 02139, (1995)

[10]

https://www.techtarget.com/whatis/definition/lossless-and-lossy-compression

(visit: Feb,2024)

[11]

https://www.haivision.com/resources/streaming-video-definitions/hevc-h-265/

(visit: Feb,2024)

[12]

https://www.digitaltrends.com/home-theater/h266-av1-evc-three-new-video-codecs-fully-explained/ (visit: Feb,2024)

[13]

https://forum.huawei.com/enterprise/en/h-266-vs-h-265-what-is-the difference/thread/979683-881 (visit: Feb,2024)

[14]

https://www.frandroid.com/android/732339_format-video-h-266-vvc

(visit: Feb,2024)

[15]

Thomas Amestoy. Optimisation du Codec VVC basé sur la Réduction de Complexité et le Traitement Parallèle. Traitement des images [eess.IV]. INSA de Rennes, Français. NNT : 2021 ISAR0009. tel-03566101. (2021).

[16]

CHENNOUFI Sarah, DJABRI Manel, Implémentation sur FPGA d’un binarisateur / dé-binarisateur dans la norme de compression vidéo HEVC / H.265, Projet de Fin d'Etudes, université YAHIA Fares de MEDEA. (2021).

[17]

J. Pfaff et al., "Intra Prediction and Mode Coding in VVC," in IEEE Transactions on Circuits and Systems for Video Technology, vol. 31, no. 10, pp. 3834-3847, doi: 10.1109/TCSVT.2021.3072430, Oct. 2021

[18]

https://ottverse.com/what-is-vvc-h266-versatile-video-coding-compare-hevc/

(visit: Feb,2024)

[19]

Wiegand, T., Sullivan, G. J., Bjøntegaard, G., & Luthra, A. Overview of the H.264/AVC Video Coding Standard. IEEE Transactions on Circuits and Systems for Video Technology (2003).

[20]

Sullivan, G. J., Ohm, J.-R., Han, W.-J., & Wiegand, T. Overview of the High Efficiency Video Coding (HEVC) Standard. IEEE Transactions on Circuits and Systems for Video Technology (2012).

[21]

B. Bross, Y. Wang, Y. Ye, S. Liu, J. Chen, G.J. Sullivan and J. Ohm, "Overview of the Versatile Video Coding (VVC) Standard and its Applications," IEEE Trans. Circuits Sys. Video Technol. (2021).

[22]

Dali.M cours TPMM/telecommunication system, university of YAHIA Fares, Medea. (2023/2024).

[23]

L. Zhang, K. Zhang, H. Liu, Y. Wang, P. Zhao, D. Hong, "CE4: History-based Motion Vector Prediction (Test 4.4.7)," document JVET-L0266, ITU-T/ISO/IEC Joint Video Experts Team (JVET), Macao, CN, (Oct. 2018).

[24]

L. Zhang, K. Zhang, H. Liu, H. C. Chuang, Y. Wang, J. Xu, et al., "History-based Motion Vector Prediction in Versatile Video Coding," 2019 Data Compression Conference (DCC), Snowbird, UT, USA, pp. 43-52, doi: 10.1109/DCC.2019.00012. (2019).

[25]

B. Girod, "Rate-constrained Motion Estimation," Proc. SPIE 2308, Visual Communications and Image Processing, Chicago, IL, USA, (Sep. 1994).

[26]

J. Chen, W.-J. Chien, M. Karczewicz, X. Li, H. Liu, A. Said, L. Zhang, X. Zhao, "Further Improvements to HMKTA-1.0," document VCEG-AZ07, Warsaw, PL, (Jun. 2015).

[27]

A. Henkel et al., "Alternative Half-Sample Interpolation Filters for Versatile Video Coding," Proc. ICASSP 2020, Barcelona, ES, pp. 2053-2057, (May 2020)

[28]

S. Jeong, M. W. Park, Y. Piao, M. Park and K. Choi, "CE4 Ultimate Motion Vector Expression (Test 4.5.4)," document JVET-L0054, ITU-T/ISO/IEC Joint Video Experts Team (JVET), Macao, CN, (Oct. 2018).

[29]

H. Gao, S. Esenlik, E. Alshina and E. Steinbach, "Geometric Partitioning Mode in Versatile Video Coding: Algorithm Review and Analysis," in IEEE Transactions on Circuits and Systems for Video Technology, doi: 10.1109/TCSVT.2020.3040291. (2020)

[30]

SALEM Roufaida, IRES afrah, master thesis 'study and implementation of intra prediction in the latest video compression standard VVC/H266. University of YAHIA Fares, MEDEA. (2023).

[31]

ITU-T telecommunication standardization sector of ITU/series H: audiovisual and Multimedia Systems, Infrastructure of audiovisual services – coding of moving video, versatile video coding (04/2022).