



**Miguel Oliveira
Inocêncio**

**Co-processador da Transformada e Quantização
para AV1**

AV1 Transform and Quantization Co-Processor

DOCUMENTO PROVISÓRIO



**Miguel Oliveira
Inocêncio**

**Co-processador da Transformada e Quantização
para AV1**

AV1 Transform and Quantization Co-Processor

Dissertação de Mestrado apresentada à Universidade de Aveiro, para
obtenção do grau de Mestre em Engenharia Electrónica e de Telecomu-
nicações, sob orientação do Professor Doutor António Navarro . . .

DOCUMENTO PROVISÓRIO

o júri / the jury

presidente / president

ABC

Professor Catedrático da Universidade de Aveiro (por delegação da Reitora da Universidade de Aveiro)

vogais / examiners committee

DEF

Professor Catedrático da Universidade de Aveiro (orientador)

GHI

Professor associado da Universidade J (co-orientador)

KLM

Professor Catedrático da Universidade N

agradecimentos / acknowledgements

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetur id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum...

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris...

Palavras-Chave

Resumo

HEVC, ...

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetur adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.

Keywords

Abstract

HEVC, ...

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetur adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.

Contents

| | |
|--|------------|
| List of Figures | iii |
| List of Tables | v |
| Acronyms | vii |
| Glossary | ix |
| 1 Introduction | 1 |
| 1.1 Background and Motivation | 1 |
| 1.2 Scope | 3 |
| 1.3 Outline | 3 |
| References | 4 |
| 2 Video Compression Systems | 5 |
| 2.1 Basic Functioning | 5 |
| 2.1.1 Human Visual System | 5 |
| 2.1.2 Redundancy Exploitation | 5 |
| 2.1.3 Basic Video Compression/Decompression System | 5 |
| 2.2 Previous Standards | 5 |
| 2.3 AV1 | 6 |
| 2.4 Performance Analysis | 6 |
| 3 Video Coding Transforms | 7 |
| 3.1 Introduction | 7 |
| 3.2 Background | 7 |
| 3.2.1 Basis vector/image interpretation | 7 |
| 3.3 Transformation Kernels | 9 |
| 3.3.1 Discrete Fourier Transform (DFT) | 9 |
| 3.3.2 Discrete Walsh-Hadamard Transform (WHT) | 10 |
| 3.3.3 Discrete Cosine Transform (DCT) | 11 |
| 3.3.4 Discrete Sine Transform (DST) | 13 |
| References | 14 |
| 4 Developed Architecture | 15 |
| 4.1 REEEEEEEEEEEEEEEEEEEEEEEEEEE | 15 |

List of Figures

| | | |
|-----|--|----|
| 3.1 | Sequences generated in the first step of table 3.1 | 12 |
|-----|--|----|

List of Tables

| | | |
|-----|---|----|
| 3.1 | Similarity between the processes of the DFT and the DCT | 11 |
|-----|---|----|

Acronyms

AOM Alliance for Open Media.

AV1 AOM Video 1.

CMOS Complementary metal–oxide–semiconductor.

Codec Encoder-Decoder.

CRT Cathode Ray Television.

DCT Discrete Cosine Transform.

DFT Discrete Fourier Transform.

FFT Fast Fourier Transform.

fps frames per second.

GPU Graphical Processing Unit.

HEVC High Efficiency Video Coding.

IC Integrated Circuit.

MPEG Motion Picture Experts Group.

TV Television.

UHD Ultra-High-Definition.

WHT Walsh-Hadamard Transform.

Glossary

Codec Encoder-Decoder. Also referred to the method of compressing and decompressing a video sequence.

Interlaced scanning Technique used by televisions for broadcasting and displaying, where only odd or even numbered lines of a frame are transmitted/displayed at a time, alternately.

JPEG Still image compression format, developed by the Joint Photographic Experts Group (JPEG).

libaom Reference software for AV1, released by Google in June 2018.

Progressive scanning Technique used by more recent screens, where each frame is displayed as a whole, from top to bottom, and left to right.

RGB Color space based on the addition of Red, Green and Blue components for complex color representation.

VP8/VP9 Open-format video codecs developed by Google, released in 2008 and 2013, respectively.

Todo list

| | |
|--|----|
| Falar dos color spaces mais à frente? | 1 |
| referência para seção mais à frente? | 3 |
| *Thesis objectives | 3 |
| *General outline of the different chapters Last section to do | 3 |
| *Essency of video compression relies on making changes the image without serious perception by the user | 5 |
| *Eye Functioning | 5 |
| *"Known Issues" (lower perception to chroma, high frequencies, etc) | 5 |
| *Opportunity to explore various types of redundancies to the image | 5 |
| *Types of redundancies (Temporal, Statistical and Coding) | 5 |
| *Color subsampling | 5 |
| *Intra-prediction | 5 |
| *Inter-prediction | 5 |
| *Transform and Quantization | 5 |
| *Entropy Coding | 5 |
| *Encoder Model | 5 |
| *Decoder Model | 5 |
| *Previous generations | 5 |
| Review <i>AV1 Bitstream and Decoding Process</i> | 6 |
| *Development Process | 6 |
| *AOMedia companies | 6 |
| *Comparison with past generations | 6 |
| *Introduction of modules not present on other video codecs | 6 |
| *Block diagram | 6 |
| *Compression gains | 6 |
| *Quality assessment | 6 |
| *Complexity (general/modules) and timing issues | 6 |
| as seen in | 7 |
| Verify accordance with previous chapter | 7 |
| JPEG example? | 9 |
| Change image to TIKZ | 11 |

CHAPTER 1

Introduction

1.1 | Background and Motivation

Since the spark of television research in 1887, a tremendous investment has been put into increasing the quality of images, cameras and screens that display them [1].

In the early years of mechanical television, this desire was pursued by making changes to the *Nipkow* disks, up to the decline of the mechanical TV, around the 1930's. The consequential rise of all-electronic TVs started with the capture of images with the same cathode tubes put into Cathode Ray Televisions (CRTs), with broadcasts of the live analog recordings, since there were no available methods of storing images, up to 1955, with the development of the open-reel magnetic tape [2].

The evolution of Complementary metal-oxide-semiconductor (CMOS) technologies however, led to the downfall of cathode ray tubes, and to the rise of image capture to a digital sensor, that allowed better image captures and lower demands in terms of physical storage space. However, with the desire for higher fidelity video, the quantity of information captured also increased. Whether by increasing the sensor resolution, color bit depth or frame rate, the captured video sequences have increased its size throughout the years. For instance, for a video of 640×360 (considered as a low resolution), at 30 frames per second (fps), considering each captured color (RGB) is represented with 8 bits, there is approximately 166 Million bits per second (Mbps) of captured information. This means that a short 5 minute video would occupy more than 6 Giga Bytes (GB) of memory. This aspect gets more severe once higher resolutions are considered. For newer standards such as 4K Ultra-High-Definition (UHD) (3840×2160) or 8K UHD (7680×4320), under the same conditions, a ten minute video would occupy 448 GB and 1792 GB of raw data, respectively.

Falar dos color spaces mais à frente?

To further aggravate the situation, video consumption got massively adopted on the average consumer level, and continues to grow, both in the average number of hours watched by users and in the resolutions of the video, making the bandwidth used on the visualization of video footage the highest between all other application. With the development of higher video sizes, increase of the average number of connected devices per user and overall market expansion through the number of consumers, this margin will continue to grow [3, Trends 1 & 4].

This problem has led to the introduction of a new concept: *Video Compression*¹, which is the process of reducing the size of a video sequence, while still maintaining its playback capabilities. The Codec takes advantage of redundant information present on the raw data to reduce the size of the video, without heavily modifying the original picture or its quality.

The first form of video compression, Interlaced scanning, dates from 1940, and was purely

¹Also called *Video Coding*.

analog. This solution was introduced with the intent of reducing the necessary broadcasting bandwidth for old CRTs, without decreasing the displayed fps. And even though this technique has been implemented over more than seventy years, it has proven to be so efficient that most TV channels today still use interlaced broadcasting.

However, analog television is now obsolete, as well as CRTs. The massive developments in Integrated Circuit (IC) fabrication led to the rise of the digital era we now live in. Therefore, most screens (be it televisions, monitors or cellphones) use digital, Progressive scanning. As such, the use of analog compression techniques wasn't applicable. Accordingly, the evolution of digital video led to the development of digital compression techniques, such as the one presented in this work.

Being purely digital, these methodologies rely on computers and other processors to analyze data and apply the compression algorithms, making them very demanding processes from a computational standpoint. As expected, a high compression ratio is only obtainable by a high complexity algorithm, which also increases with the size of the video (more data leads to more analysis). Since in the early days of digital video, the used resolutions were lower as to the ones used in the present days, the compression algorithms used were not very demanding. However as the pursuit for higher quality video continued, so did the necessity for better compression ratios, and therefore the computational needs also increased. Such complex softwares lead to a high power consumption from the processor executing it, making such implementations unsuitable for portable, battery limited applications, such as cellphones or laptops. Besides this huge factor, such softwares tend to be very slow, specially when a real time compression or decompression is desired.

To amend for these factors, and to increase the reachability of high quality video to as many users as possible, these applications needed to have a viable solution that didn't compromise its usability. Accordingly, a new approach has been implemented on the most recent codec's. Besides the optimization of pure software compression/decompression solutions, there has been a great focus on the development of specialized hardware for such codecs. This solution could redress many of the problems presented previously, making them viable on a mobile implementation, as well as other specialized appliances, since such co-processors usually present a better performance than generic ones. This tendency has already been verified on the implementation choices on recent smartphones [4, p. 14], as well as recent *Nvidia* Graphical Processing Unit (GPU) lineups [5].

Since each compression algorithm tend to be very different from its predecessors, either by making changes to its bitstream or functioning principles, each time a new codec is released, there is a need to backup its development with a new set of hardware implementations. This makes the improvement of video compression techniques a continuous effort, in many engineering branches, as the technology needs to keep up with the demands of consumers, in a variety of applications.

Due to the broad access to video, and its influence in a variety of markets (besides video consumption itself), big companies have made investments on the improvement of video quality, and respective compression algorithms. These investments have provoked somewhat of a "*Codec War*". Since 2010, several video codecs have been deployed, and quickly replaced by a newer version, which presents better compression gains, at a lower quality degradation, such as the replace of *VP8* (released in 2008) with *VP9* (2013).

1.2 | Scope

AOM Video 1 (AV1) is the most recently released² video codec. It was developed as a Joint Development Foundation [6] project, under the name of Alliance for Open Media (AOM)³. This codec took the same objective as its main predecessor, *VP9*, which was to be an open source, royalty free alternative to Motion Picture Experts Group (MPEG)'s state of the art video codec, *High Efficiency Video Coding (HEVC)*.

Upon release, *VP9* rivaled *HEVC*'s performance. However, soon after, the market demanded higher compression performance, giving origin to consortium of enterprises that now represent AOM, and to the development of *AV1*, in 2015. The first release of this coding format was made in March 2018, with the first release of its reference software, *libaom*, being made three months later, in June 2018.

Besides its main objectives, *AV1* was also developed with the intent of being implementable in hardware. Therefore, various design choices were made to make the algorithm low memory consuming, and highly parallelizable.

The desired compression performance was obtained at the cost of a highly complex algorithm (and reference software), that severely outperforms *VP9*, at the cost of much higher compression times [7].

Taken these factors, there is a high demand for dedicated hardware architectures, that can speed up the compression/decompression times and reach real-time usability on live-streaming applications, such as video-conferencing, live-content visualization, etc.

*Thesis objectives

referência
para
secção
mais à
frente?

1.3 | Outline

*General outline of the different chapters **Last section to do**

References

- [1] Mark Schubin. "What Sparked Video Research in 1877? The Overlooked Role of the Siemens Artificial Eye [Scanning Our Past]". In: *Proceedings of the IEEE* 105.3 (Mar. 2017), pp. 568–576. ISSN: 0018-9219, 1558-2256.
- [2] Marco Jacobs and Jonah Probell. "A Brief History of Video Coding". en. In: (), p. 6.
- [3] *Cisco Visual Networking Index: Forecast and Trends, 2017–2022 White Paper*. en. <https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white-paper-c11-741490.html>.
- [4] Scientiamobile. *Mobile Overview Report April – June 2018*.
- [5] *Video Encode and Decode GPU Support Matrix*. en. <https://developer.nvidia.com/video-encode-decode-gpu-support-matrix>. Nov. 2016.

²Currently, there are other codecs being developed, without official bitstream release

³Further explained in Chapter 2

- [6] *Joint Development Foundation*. en. <http://www.jointdevelopment.org/>.
- [7] Dan Grois, Tung Nguyen, and Detlev Marpe. “Performance Comparison of AV1, JEM, VP9, and HEVC Encoders”. In: *Applications of Digital Image Processing XL*. Ed. by Andrew G. Tescher. San Diego, United States: SPIE, Feb. 2018, p. 120. ISBN: 978-1-5106-1249-5 978-1-5106-1250-1.

CHAPTER 2

Video Compression Systems

2.1 | Basic Functioning

2.1.1 | Human Visual System

*Essency of video compression relies on making changes the image without serious perception by the user

*Eye Functioning

*"Known Issues" (lower perception to chroma, high frequencies, etc)

*Opportunity to explore various types of redundancies to the image

2.1.2 | Redundancy Exploitation

*Types of redundancies (Temporal, Statistical and Coding)

*Color subsampling

*Intra-prediction

*Inter-prediction

*Transform and Quantization

*Entropy Coding

2.1.3 | Basic Video Compression/Decompression System

*Encoder Model

*Decoder Model

2.2 | Previous Standards

*Previous generations

2.3 | AV1

Review *AV1 Bitstream and Decoding Process*

*Development Process

*AOMedia companies

*Comparison with past generations

*Introduction of modules not present on other video codecs

*Block diagram

2.4 | Performance Analysis

*Compression gains

*Quality assessment

*Complexity (general/modules) and timing issues

CHAPTER 3

Video Coding Transforms

3.1 | Introduction

As mentioned previously, the basic principle behind the compression of video, is the reduction of inter-pixel/inter-symbol correlation. The various integral blocks of a video compression system try to accomplish this objective through different strategies. The *Intra-frame* and *Inter-frame Prediction* exploit spatial and temporal correlation, respectively. Through the subtraction of the input by the output of one of these blocks, and the attainment of the *residue*, the next compression stage is made in the *Transform* block, which is the focus of this work.

as seen
in ...

Verify accordance with previous chapter

The technique implemented by this process relies on the energy compaction in the frequency domain to reduce the correlation within a frame block, i.e. the input of the Transform block is evaluated on its main frequencies — the *transform coefficients* — on a spatial and/or temporal domain, similarly to the process executed on an Fast Fourier Transform (FFT). Once each block is quantized on these coefficients, the compression is made with the removal of the least significant ones, on the *Quantization* stage. The intent of the *transform* is to split the image into a set of predefined coefficients, that get transmitted instead of the pixel values.

The objective of this chapter is to give the reader a basic understanding of the theoretical basis behind said Transformations, as well as to introduce the most commonly used ones.

3.2 | Background

3.2.1 | Basis vector/image interpretation

A useful interpretation, and a good starting point to the study of this process, is to see it as the decomposition of the input as a set of basis vectors (1D transforms) or images/matrices (2D transforms). The transformation outputs, T_i , can be seen as the weights of each basis vector/image, \vec{e}_i , that summed return the input, \vec{g} , i.e.

$$\vec{g} = \sum_{i=1}^N T_i \vec{e}_i \quad (3.1)$$

which means that the coefficients are related to the amount of correlation between the input and each basis component, and can be obtained with the *inner product* of the input and each

basis vector.

$$T_i = \vec{e}_i^T \vec{g} \quad (3.2)$$

Since each input vector will have different correlation values between the various basis vectors, this operation accomplishes two main objectives:

- De-correlation of the input values
- Signaling of the most important basis vectors.

Considering a 2D image, $g(x, y)$, and its corresponding transformed coefficients, $T(u, v)$, where (x, y) are the pixel coordinates, and (u, v) are the corresponding coordinates in the transform domain, we can obtain an analogous version of equation 3.2 as

$$T(u, v) = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} g(x, y) f(x, y, u, v) \quad (3.3)$$

Similarly, we can re-obtain the original picture

$$g(x, y) = \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} T(u, v) i(x, y, u, v) \quad (3.4)$$

where $f(x, y, u, v)$ and $i(x, y, u, v)$ are the *forward* and *inverse transformation kernels*. To better explain the concept of these, first it's needed to introduce the two following concepts.

3.2.1.1 | Separability

A useful characteristic of 2D Video Coding Transforms is its ability to be independently calculated between rows and columns. This means that given a 2D block as input, the transform coefficients can be calculated first with the *horizontal transform*, and then with the *vertical transform*, or vice-versa.

This aspect is applicable if the following conditions are applied

$$f(x, y, u, v) = f_1(x, u) f_2(y, v) \quad (3.5)$$

$$i(x, y, u, v) = i_1(x, u) i_2(y, v) \quad (3.6)$$

This means that the equation 3.3 is reconstructed as 2 independent and sequential operations

$$T_{temp}(x, v) = \sum_{y=0}^{N-1} g(x, y) f_2(y, v) \quad (3.7)$$

$$T(u, v) = \sum_{x=0}^{M-1} T_{temp}(x, v) f_1(x, u) \quad (3.8)$$

On AV1, due to the various implemented transformation kernels, this aspect is severely explored, since the only way of implementing the combination of different 1D kernels, is to calculate them independently. This aspect is further explained with the following concept.

3.2.1.2 | Symmetry

Taking equation 3.5, a transformation kernel is said to be symmetric if

$$f_1(y, v) = f_2(x, u) \quad (3.9)$$

This characteristic is particularly useful because it makes the forward and inverse transformations expressible as matrix multiplications. Therefore, the equations 3.3 and 3.4 are represented, respectively, as

$$T = F^T G F \quad (3.10)$$

$$G = I^T T I \quad (3.11)$$

where F and I are the forward and inverse transform matrices. This aspect is only possible for square matrix, i.e., input blocks with the same height and width.

This concept isn't exploited in AV1, since the use of different 1D transformation kernels, and rectangular block sizes ($M \neq N$) make the 2D transform asymmetric, and therefore, not executable as matrix multiplication.

Looking now at equation 3.4, we can interpret the inverse transformation kernel as a set of basis images, dependent of the (u, v) pair. By this, it is meant

$$g(x, y) = \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} T(u, v) I_{u,v} \quad (3.12)$$

where

$$I_{u,v} = \begin{bmatrix} i(0, 0, u, v) & i(0, 1, u, v) & \dots & i(0, M-1, u, v) \\ i(1, 0, u, v) & i(1, 1, u, v) & \dots & i(1, M-1, u, v) \\ \vdots & \vdots & \dots & \vdots \\ i(N-1, 0, u, v) & i(N-1, 1, u, v) & \dots & i(N-1, M-1, u, v) \end{bmatrix} \quad (3.13)$$

Therefore, the forward and inverse transformation process can be seen as the deconstruction of an input block, into a set of $M \cdot N$ basis images, dependent of the used transformation kernel. As expressed in equations 3.5 and 3.6, this analogy can be made on a 1D space .

JPEG example?

Given a general comprehension of the theoretical principles behind the *Transform* block, now the most common transformation kernels are introduced, with focus on the AV1 video codec.

3.3 | Transformation Kernels

3.3.1 | Discrete Fourier Transform (DFT)

Although it isn't implemented in video coding, it's widely used in digital signal processing, and many of the used transformation kernels are approximations of this function.

It has its roots on the *Fourier Transform*, whose forward and inverse transformations are expressed in equations 3.14 and 3.15, respectively.

$$T(u, v) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} g(x, y) e^{-j2\pi(ux+vy)} dx dy \quad (3.14)$$

$$g(x, y) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} T(u, v) e^{j2\pi(ux+vy)} du dv \quad (3.15)$$

Once we consider a finite number of points, the previous equations become

$$T(u, v) = \frac{1}{MN} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} g(x, y) e^{-j2\pi(\frac{ux}{M} + \frac{vy}{N})} \quad (3.16)$$

$$g(x, y) = \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} T(u, v) e^{j2\pi(\frac{ux}{M} + \frac{vy}{N})} \quad (3.17)$$

which corresponds to replacing the kernels in equations 3.3 and 3.4 with

$$f(x, y, u, v) = \frac{1}{MN} e^{-j2\pi(\frac{ux}{M} + \frac{vy}{N})} \quad (3.18)$$

$$i(x, y, u, v) = e^{j2\pi(\frac{ux}{M} + \frac{vy}{N})} \quad (3.19)$$

The position of the multiplication factor, $\frac{1}{MN}$, is irrelevant, and in some works is divided into two terms in the forward and inverse kernels, $\frac{1}{M}$ and $\frac{1}{N}$, or even $\frac{1}{\sqrt{MN}}$.

Because of the use of complex numbers, this operation tends require a high computational effort, whence its disuse in video coding.

3.3.2 | Discrete Walsh-Hadamard Transform (WHT)

This transformation replaces the sum of sines and cosines of the DFT, alternating of positive and negative 1's, depending on the binary representation of the inputs.

Considering the inputs of the transform to be represented with m bits, where $m-1$ is the most significant bit (b_{m-1}), the forward and inverse kernels are represented as

$$f(x, y, u, v) = i(x, y, u, v) = \frac{1}{\sqrt{MN}} (-1)^{\sum_{i=0}^{m-1} [b_i(x)p_i(u) + b_i(y)p_i(v)]} \quad (3.20)$$

where

$$\begin{aligned} p_0(u) &= b_{m-1}(u) \\ p_1(u) &= b_{m-1}(u) + b_{m-2}(u) \\ &\vdots \\ p_{m-1}(u) &= b_1(u) + b_0(u) \end{aligned} \quad (3.21)$$

3.3.3 | Discrete Cosine Transform (DCT)

The most commonly used transform, the *DCT*, was published by Ahmed et al. in 1974 [3]. Since then, it has been adopted in a wide range of applications, being the only transform used in the first generations of video codecs, as well as in *still image compression*, being the basis of the *JPEG* standard.

It is frequently compared to the *DFT*, due to the similarity of their operation. However, as the name implies, the *DCT* relies on the cosine function to create its basis images, which is a *periodic* and *symmetrically even* function. Therefore, as mentioned by [4, A. V. Oppenheim], "*Just as the DFT involves an implicit assumption of periodicity, the DCT involves implicit assumptions of both periodicity and even symmetry*". This is easily observable once considered the equivalent process of both algorithms. Taking an N -point sequence, $g(n)$, the calculation of the *DFT* and *DCT* of such sequence is equivalent to the processes presented at table 3.1.

| Step | <i>DFT</i> | <i>DCT</i> |
|------|---|--|
| 1 | Repeat $g(n)$ every N points, giving origin to $\tilde{g}_N(n)$ | Concatenate $g(n)$ with a flipped version of itself, creating a $2N$ sequence, $g_{2N}(n)$, and repeat it, giving origin to $\tilde{g}_{2N}(n)$ |
| 2 | Calculate the <i>Fourier</i> expansion of \tilde{g}_N | Calculate the <i>Fourier</i> expansion of \tilde{g}_{2N} |
| 3 | Keep the first N coefficients and set all others to 0 | Keep the first N coefficients, and set all others to 0 |

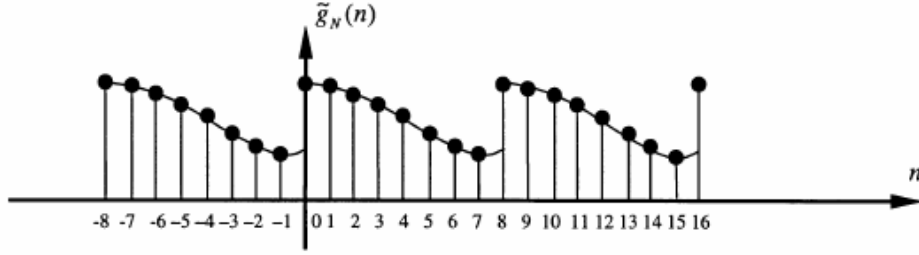
Table 3.1: Similarity between the processes of the *DFT* and the *DCT*

The main reason behind the heavy adoption of the *DCT* is its great energy compaction on the lower frequencies, where most of the energy in a picture is packed. If the the output of the first step of table 3.1 is observed, this aspect is more easily understood.

Due to the back-to-head repetition seen in figure 3.1a, there is a disruption every N points, which gives origin to high frequency components in the *Fourier* transform. Therefore, the more continuous behavior obtained with the back-to-back repetition of the *DCT* gives origin to more significant low frequency coefficients. However, there are many ways of creating a periodic and symmetric sequence from a finite number of points. This factor has led to the implementation of different versions of the *DCT*, which differ in minor details between themselves. These differences are consequence of the way the symmetry is obtained, which can be observed in figure 3.1b. The represented implementations are referred to as *DCT-I* to *DCT-IV*, but other possibilities exist. Their definition depends on the overlapping of points when repeating each sequence.

Change image to TIKZ

Since the *DCT* in *AV1* is implemented in one dimension, the description of the following kernels is also made in 1D. Therefore, the dimension of the transform, L , is referring either to the blocks' width or height, depending if the operation is made to the rows or columns, respectively (M or N , previously).



(a) DFT

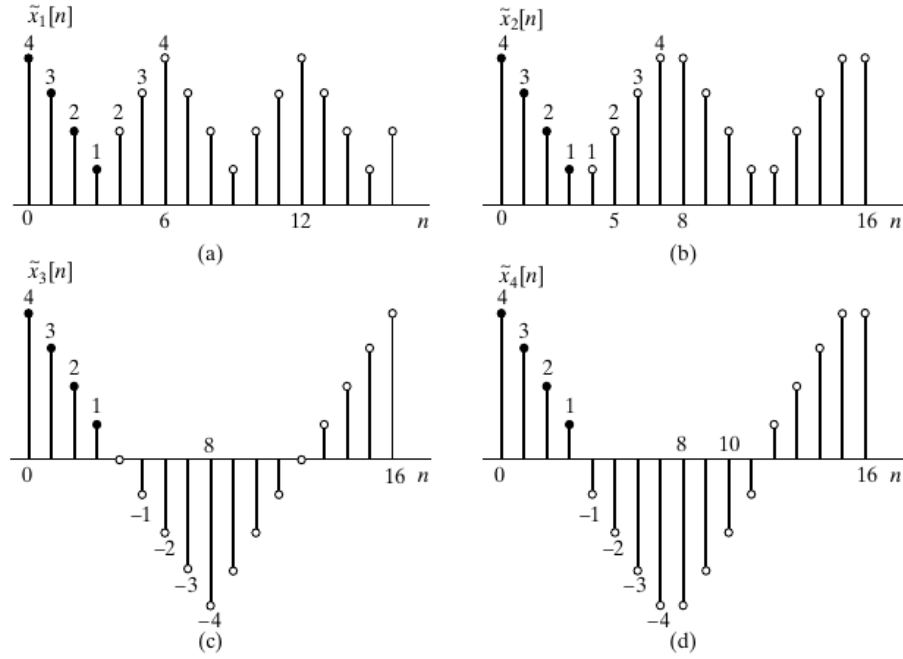


Figure 25 Four ways to extend a four-point sequence $x[n]$ both periodically and symmetrically. The finite-length sequence $x[n]$ is plotted with solid dots. (a) Type-1 periodic extension for DCT-1. (b) Type-2 periodic extension for DCT-2. (c) Type-3 periodic extension for DCT-3. (d) Type-4 periodic extension for DCT-4.

(b) DCT

Figure 3.1: Sequences generated in the first step of table 3.1 [1]

3.3.3.1 | DCT-I

The sequence created with first version of the DCT has overlapping points at $n = k(L - 1)$, $k = 0, 1, 2, \dots$, making the overall period of the final sequence $2L - 2$.

$$f(x, u) = \frac{2}{L-1} \alpha(x) \cos\left(\frac{\pi x u}{L-1}\right) \quad (3.22)$$

where

$$\alpha(x) = \begin{cases} \frac{1}{2}, & x = 0 \vee x = N - 1 \\ 1, & 1 \leq x \leq N - 2 \end{cases} \quad (3.23)$$

The inverse transform becomes

$$i(x, u) = \alpha(u) \cos\left(\frac{xu\pi}{L-1}\right) \quad (3.24)$$

3.3.3.2 | DCT-II

Usually referred to as "the *DCT*", it is by far the most implemented version, being the only one mentioned in many of the studied works.

As seen in figure 3.1b, this version has no overlap on the created sequence, making the period $2L$, and the points of symmetry $kL - \frac{1}{2}$.

$$f(x, u) = i(x, u) = \beta(u) \cos\left(\frac{(2x+1)u\pi}{2L}\right) \quad (3.25)$$

$$\beta(u) = \begin{cases} \sqrt{\frac{1}{L}}, & u = 0 \\ \sqrt{\frac{2}{L}}, & 1 \leq u \leq N-1 \end{cases} \quad (3.26)$$

3.3.3.3 | DCT-III

Named the *inverse* of DCT-II, due to the switch of the transform and pixel coordinates.

$$f(x, u) = i(x, u) = \beta(u) \cos\left(\frac{(2u+1)x\pi}{2L}\right) \quad (3.27)$$

$$\beta(u) = \begin{cases} \sqrt{\frac{1}{L}}, & u = 0 \\ \sqrt{\frac{2}{L}}, & 1 \leq u \leq N-1 \end{cases} \quad (3.28)$$


3.3.3.4 | DCT-IV

$$f(x, u) = i(x, u) = \sqrt{\frac{2}{L}} \cos\left(\frac{(2u+1)(2x+1)\pi}{4L}\right) \quad (3.29)$$

Is the basis of the *Modified Discrete Cosine Function (MDCT)*, where the input blocks overlap.

3.3.4 | Discrete Sine Transform (DST)

References

- [1] Yun Qing Shi and Huifang Sun. *Image and Video Compression for Multimedia Engineering: Fundamentals, Algorithms, and Standards*. eng. 2. ed. Image Processing Series. OCLC: 254564955. Boca Raton, Fla.: CRC Press, 2008. ISBN: 978-0-8493-7364-0.
- [2] Rafael C. Gonzalez and Richard E. Woods. *Digital Image Processing*. eng. Fourth edition. OCLC: 987436552. New York, NY: Pearson, 2018. ISBN: 978-0-13-335672-4.
- [3] N. Ahmed, T. Natarajan, and K. R. Rao. “Discrete Cosine Transform”. In: *IEEE Transactions on Computers* C-23.1 (Jan. 1974), pp. 90–93. ISSN: 0018-9340.
- [4] Alan V. Oppenheim, Ronald W. Schaffer, and John R. Buck. *Discrete-Time Signal Processing*. eng. 2nd ed. Upper Saddle River, NJ: Prentice Hall, 1998. ISBN: 978-0-13-754920-7.
- [5] *Discrete Cosine Transform - MATLAB Dct*. <https://www.mathworks.com/help/signal/ref/dct.html>. 

CHAPTER 4

Developed Architecture

4.1 | REEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEE

Sed commodo posuere pede. Mauris ut est. Ut quis purus. Sed ac odio. Sed vehicula hendrerit sem. Duis non odio. Morbi ut dui. Sed accumsan risus eget odio. In hac habitasse platea dictumst. Pellentesque non elit. Fusce sed justo eu urna porta tincidunt. Mauris felis odio, sollicitudin sed, volutpat a, ornare ac, erat. Morbi quis dolor. Donec pellentesque, erat ac sagittis semper, nunc dui lobortis purus, quis congue purus metus ultricies tellus. Proin et quam. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Praesent sapien turpis, fermentum vel, eleifend faucibus, vehicula eu, lacus.