Texture analysis for Mobiles Phones in Data Compression

Sabri KOÇER

Necmettin Erbakan University

Özgür DÜNDAR

Necmettin Erbakan University

To Cite This Chapter

Koçer, S., & Dündar, Ö. (2024). Texture analysis for Mobiles Phones in Data Compression. In S. Koçer & Ö. Dündar (Eds.), *Intelligent Systems and Optimization in Engineering* (pp. 115-126). ISRES Publishing.

Introduction

Storing and transmitting information are frequently important issues for businesses, states and different organizations. These organizations can handle additional information by packaging it, which likewise reduces the expected space and cost for storage. In the event that you record or send information as part of your responsibilities, it can be valuable to be aware of compression capabilities and the advantages it can offer you and your association (Kocer vd., 2022).

Compression for mobile phones is the most important area because we take care of almost all our needs over the phone. 2D, 3D images are used and in graphics 2D, 3D images can be called textures with different compression methods than normal images. Mostly all techniques are enhanced for human eyes brightness, not only the color in texture compression for mobile phones.

Nowadays when everything becomes computerized, the need for storage of huge amount of data becomes more crucial. The problem is that even if the capacity of storage increases the demand for extra storage will keep growing even twice (Jayasankar, 2021). Therefore, in order to deal with possible storage problems some data compression methods were introduced. The main thing to know before using some compression method is the level of compression the one wants to apply which is divided into categories such as Tuple level, Page level, Column Level, Block Level and Table Level compression. The data compression started its beginning from the compression of messages in telegraphs by Morse Code which was taken as origin for Huffman coding (Anuradha, D.,2016, Said, A.2003, Moffat, A., 2019)

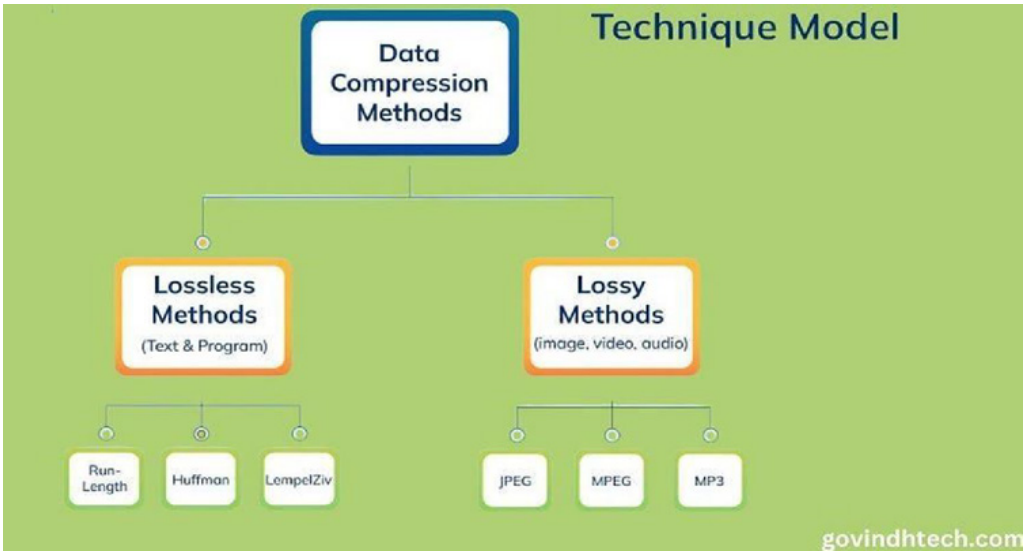
Data Compression Techniques

Usually, compression methods based on data quality are classified as lossless and lossy. However, a hybrid type of compression which is called near-lossless compression exists in practice. Usually, Lossy compression methods are not used in traditional database systems but only at the application level, since in database systems both data integrity and accuracy are principal because when the data loss happens the database can't

deduce what has to be thrown away and what does not (Pavlo, 2024). Nevertheless, lossy compression still can be applied in particular database-related contexts. In NearLossless technique the difference between the original and decompressed data differ by no more than a precise amount known as maximum absolute distortion (MAD) in Fig 1

Figure.1

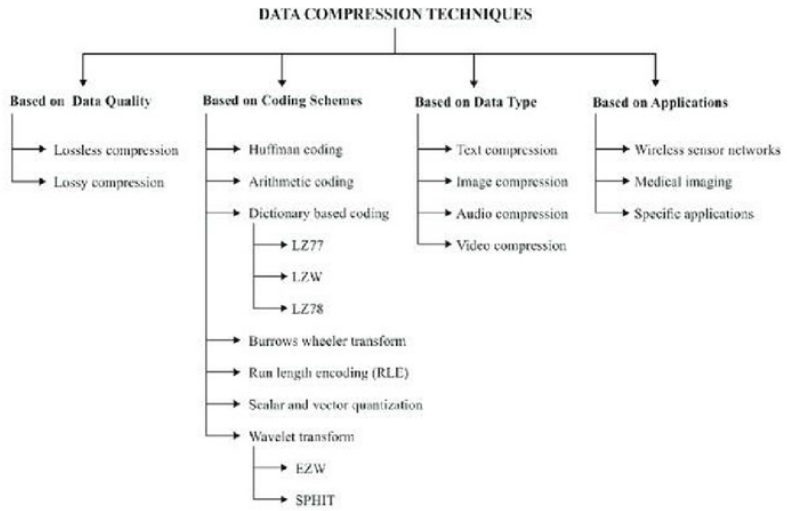
Data Compression Techniques' main Classification (Khalid A, 2006)



Depending on what is needed to be compressed in database system, distinct compression schemes can be applied (Sayood, K., 2017). As was stated in abstract part of this review paper the main rule is to have some goals such as producing fixed-length values which means to have the predetermined length of compressed output regardless of the original size or complexity of the input data which is crucial for Storage Efficiency, Simplified Data Access, Improved Query Performance in Fig 2. Another goal is to have a lossless scheme which means that input data and output data have to correspond to each other. In addition, another solution for improving query performance is decompressing data until necessary during query execution to reduce I/O operations, optimize memory and CPU usage (Salomon, D. 2007).

Figure 2

Data Compression Techniques (Jayasankar, 2021).



Copyright © 2024 by ISRES Publishing

Levels of compression

As discussed before there are several compression levels in database systems. Compression granularity refers to the size of data to be compressed in the database and can vary depending on specific requirements (Jayasankar, 2021).

Tuple or row level compression is compression which covers individual rows in database. It is usually selected when there are rows that have repetitive terms. Besides inconsiderable performance impact, another positive thing about it is the fact that when the data is updated in row level it is much simpler than decompression and recompression of higher-level data compression.

Page level compression is held on the database pages the size of each is $2n$KB usually 8KB. Pages are categorized into 3 types Data Pages, Index pages, Overflow pages. When a table is created each row is stored in tables. Here the balance between performance and effective storage usage is achieved. Since This level of compression covers more data than tuple level it is easier to reduce redundancy (Severance, D. G. 1983).

Block level compression is the technique when database management system compresses the data before transferring to the disc and decompress when there is need to read it. It stands for reducing I/O operations and saving space for more data to be stored in physical space. Depending on different factors and DBMS capabilities, algorithms such as LZ77, Huffman, RLE can be applied to blocks.

Columnar level compression is another level of compression when the data is kept in the form of columns. It is effective when columns have similar items and algorithms such as RLE, dictionary Encoding are good for achieving great ratio of compression.

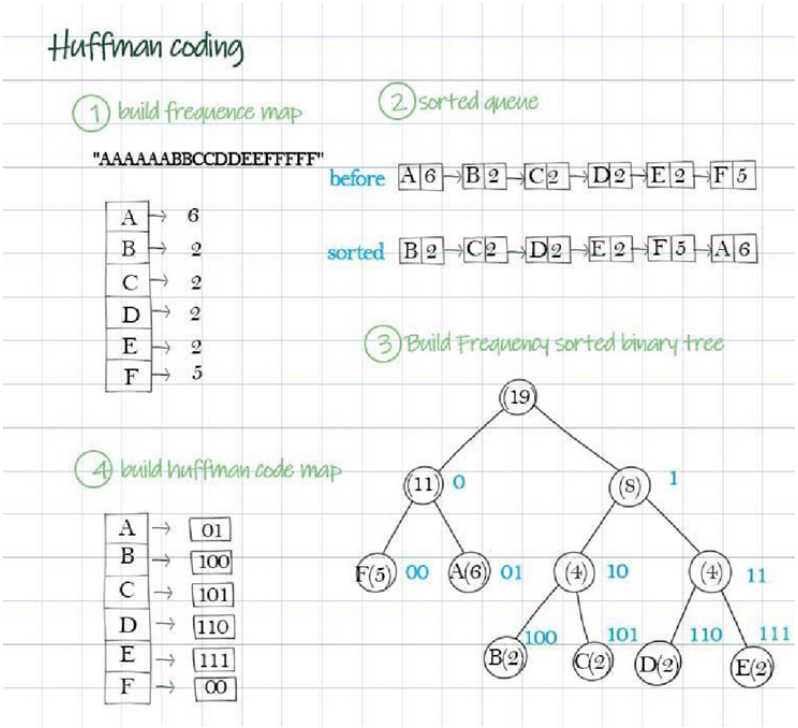
Tablespace level compression is the most significant storage reduction method which ameliorates performance, since tablespace contains all the tables, indexes and other objects.

Huffman Coding

Huffman Coding is one of the essential techniques used in lossless compression methods first introduced in 1952 (Moffat, A. 2019). Even so, its simplicity continues to make it popular. primary concept involves allocating tables with codes according to each character's frequency of appearance, which is determined by $log\ p$, where p is the probability of occurrence, that's why shorter codes represent most frequently appearing character and are on the upper branch of the binary tree while longer codes identify less frequently seen characters in Fig 3.

In the creation of tree, the thing that should be considered is the fact that elements with high frequency have to stand on the upper branches. And starting from the root node which is the overall number of all occurrences of all elements in the original data, left and right branches represent 0s and 1s and this process keep going until leaf node. Eventually Huffman code map is built.

Figure 3
Example Of Huffman Coding (Lavivienpost,2024)

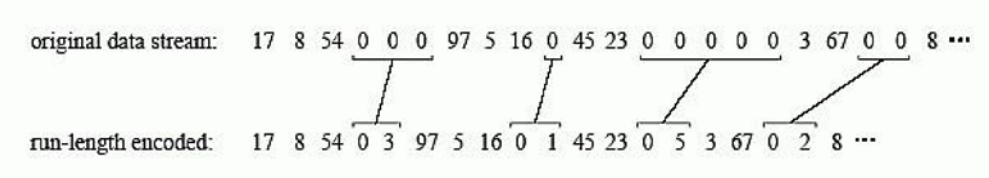


There are multiple variants of Huffman coding, including Minimum Variance Huffman Code, Length-Limited Huffman Code, Non-Binary Huffman Code, Canonical Huffman Code Adaptive Huffman Code and others. Many compression algorithms, such as Deflate, JPEG, and MP3, utilize Huffman coding as their underlying technique (Pigeon, S., 2003).

Run Length Encoding

One of the simplest forms of compression is Run Length Encoding (Fig 4) which was found by Capon in 1959. It is used particularly in cases when the data brings significant redundancy and repetitive, otherwise it can even increase the size of the compressed file. In database systems it is applicable in columnar database where for instance column identifying the gender of people and terms female and male repeat (Severance, D. G. 1983).

Figure 4
Run Length Encoding



Arithmetic Coding

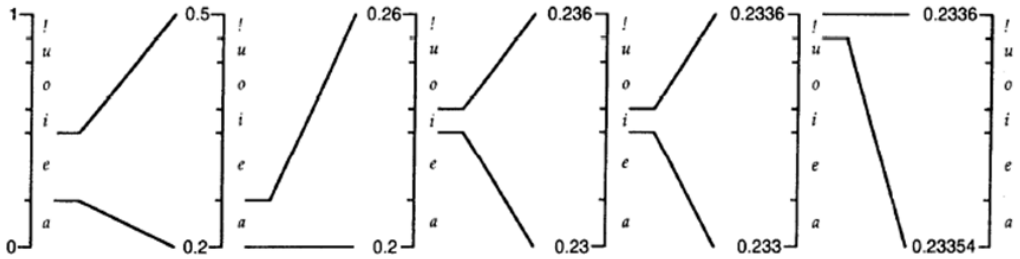
Arithmetic coding, introduced by Langdon in 1984, is a notable technique and differs from traditional coding methods since it represents an entire message as a single number that is a fraction between 0 and 1 which allows compression rates to approach the theoretical limits defined by the entropy of the input data in Fig 5. Despite its high effectiveness, it is more challenging to implement than some other methods. Arithmetic coding is especially advantageous over Huffman coding when handling sources with small alphabets and imbalanced probability distributions. A key advantage of arithmetic

Copyright © 2024 by ISRES Publishing

coding is its ability to separate the modeling and coding phases of the compression process, providing greater flexibility and efficiency. Here with increasing the frequency of symbol the bits encoded are decreasing. Arithmetic Coding is categorized in two types – adaptive and binary (Said, A. 2023).

Figure 5

Arithmetic coding (Witten,1986)



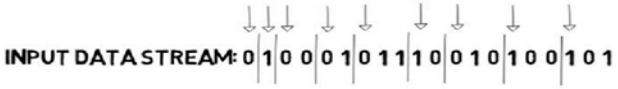
Dictionary based coding

Dictionary based coding is used when we deal with long text or any other file format with repetitive components inside it. This method uses a dictionary where most frequently sequences of characters are kept. The stored sequences now have indexes and each time when new character or sequence of characters which has to be encoded is the same as the one in the dictionary encoder indexes them from dictionary in Fig 6. Otherwise, when the character is not repetitive the character or sequence of characters is sent in uncompressed form to dictionary. The dictionary exists in two forms-static which is built before process of encoding and is left unchanged, meanwhile the dynamic dictionary is adjustable can change in the process, leading to enhanced compression ratios. Also, dictionaries's scope can be chosen I block level, table level and multi table level or so-called database level. One of the most renowned and effective methods for larger files is Lempel–Ziv algorithm (LZ) is lossless compression method which replaces frequent characters by indexes in the dictionary (Das, D, 2005).

It has two versions developed in 1977 and 1978, first employs a sliding window technique to look for matches within a specified range of past data from the current position. It encodes sequences by referencing earlier occurrences within the window, using (distance, length) pairs, occasionally including a literal character. For instance, Open-source MySQL employs the LZ77 algorithm for compressing its InnoDB tables.

Figure 6

Example of Lempel Ziv Compression Method



NUMERICAL POSITION	1	2	3	4	5	6	7	8	9
SUBSEQUENCES	0	1	00	01	011	10	010	100	101
NUMERICAL REPRESENTATION	—	—	1 1	1 2	4 2	2 1	4 1	6 1	6 2
BINARY ENCODED BLOCKS	0	1	0010	0011	1001	0100	1000	1100	1101

LZ78 on the other hand, uses a dictionary-based approach. Here characters or ssequence of characters are appended to the dictionary dynamically and as an output references to the built dictionary created. Lempel–Ziv-Welch (LZW) is the most recent version which is used very often and developed by Terry Welch inspired by LZ77 and LZ78 (Subathra, S.,2005).

Brief History of Texture Compression for Mobile Phones

In image pixilation, the bandwidth of the system is what limits performance. (Aila vd., 2003). One of the techniques that reduce bandwidth is texture compression. (Knittel vd., 1996). One of the must-have features in the system for texture compression is that textures can be stored compressed in the cache in order to make better use of the cache. Second, a constant compression ratio is needed for simple addressing and random access. Some losses occur due to fixed rate compression. Thirdly, texture-dependent lookup tables (LUT, Look Up Table) are avoided, eliminating the update that the LUT needs. (Hakura & Gupta, 1997).

While initially texture compression was targeted at mobile phones, it was later used in computer graphics cards and game consoles, using PACKMAN texture compression, which offers low complexity and reasonable image quality. (Ström & Akenine-Möller, 2004) .

With Block Truncation Coding (BTC), gray images were compressed into blocks of 4 × 4 pixels at a time. Two 8-bit grayscale values were stored for each block, and each pixel in the block was indexed to one of these grayscales, resulting in 2 bits per pixel (2 bpp – bit per bit).(Delp & Mitchell, 1979). Although BTC is not a tissue compression technique, the basis of other techniques is based on it.

The method called Color Cell Compression (CCC) uses an 8-bit color palette as an index instead of the grayscale value of BTC. The color palette is limited and requires memory searching (Campbell vd., 1986). Later CCC was implemented in hardware and was also used as texture cache (Knittel vd., 1996).

An evolution of CCC and one of the most popular is the S3TC texture compression method, which has been used in DirectX and has extensions for OpenGL. The block size is 4×4 and compressed to 64 bits. The two primary colors are stored in 16 bits. Additionally, it stores a two-bit index between two local base colors. All colors lie on a single line in the RGB space. The compression rate is 4 bpp. One disadvantage of S3TC is that only four colors can be used per block (Mccabe & Brothers, 1998) (Iourcha vd., 2003).

S3TC's problem was solved by using colors in neighboring blocks, but this increased the memory bandwidth (Ivanov & Kuzmin, 2000). The scheme, called POOMA, uses a variation of S3TC but compresses the 3×2 block size to 32 bits. Another difference from S3TC is that it uses fewer bits and only one intermediate color (Akenine-Moller vd., 2003). With the method called vector quantization, textures can be compressed up to 1 bpp and 2 bpp, and additional memory is required for access to vector quantization. This is not a very suitable solution for the high performance expected from computer graphics (Beers vd., 1996).

Later, a very different approach was taken. Two low-resolution images derived from the original texture are stored. During the process of decompressing textures, the textures are enlarged bidirectionally and a blend between the two is made to create the final color of the textures. It has two modes: 4 bpp and 2 bpp. In the 4 bpp version, two base colors are stored per 4x4 block. For bidirectional linear growth, 2x2 adjacent blocks are needed. Once textures are caught, decompression should be fast (Fenney, 2003).

In another method, mipmapping (the process of reducing large textures to increase the visibility of distant objects) and texture compression are combined. Each 4×4 block was compressed in YUV space and box filtering technique was used. Bitrate is approximately 4.6 bpp (Pereberin, 1999).

How Data Compression Works

3D computer graphics is whole with textures. This is a feature that increases the image quality and detail of 3D objects. Textures contain not only color but also information such as height and direction.

Copyright © 2024 by ISRES Publishing

Textures are two-dimensional images, and they are applied to 3D surfaces. Each pixel in the images is called a texture element. Standard compression algorithms (RLE, LZW, Deflate) and popular image formats (JPEG, PNG, TIFF) are not suitable for textures. Access to tissue is done randomly, and this process occurs on the tissues needed, not sequentially. Textures corresponding to pixels that are next to each other do not mean textures that should be next to each other. Therefore, the performance of graphics systems is highly dependent on texture access. Texture compression formats are characterized by random access (Paltashev1 & Perminov2,2014).

Four compression techniques are applied for real-time applications.

Decoding Speed: Since textures are presented in a compressed format, decoding speed is very important. The decoding algorithm should be simple so that the processor finds each texture quickly and easily when it searches for it.

Random Access: Quick access to any texture is important because it is unknown how to access the texture.

Compression Ratio and Visual Quality: Compression ratio and visual quality are interrelated. Because as the texture is compressed, visual artificiality will increase.

Encoding Speed: Encoding speed is not very important as it uses textures before they are stored (Beers vd., 1996).

Textures are compressed in two ways: lossy and lossless. Since lossy texture compression techniques will cause errors in the texture, the result obtained will be an approximation of the original. In this technique, processing is done according to the compression ratio, and as the amount of compression increases, artifacts increase and quality decreases. Lossless compression technique always returns a replica of the original (Strom & Wennersten, 2011) & (Eskicioglu & Fisher, 1993).

When textures are compressed at a fixed rate, they always give the same value bit per pixel (bpp). This means that a texture with a size of N*N will always be compressed to the same file size. Non-constant compression techniques have also been shown to be slower in random access since there is no structure defined to access the texture data, regardless of the content.

Texture Compression Techniques for Mobile Phones

ETC (Ericsson Texture Compression) Family

ETC format, a standard compression scheme for Android-based devices, was first developed for use on mobile devices. PACKMAN is the first version (Ström & Akenine-Möller, 2004) then ETC1(Ström & Akenine-Möller, 2005)/ETC2(Ström & Martin, 2007) versions are enhanced. Based on the fact that the human eye is more sensitive to brightness than color, ETC has made texture compression based on this idea. Therefore, only a basic color is stored in each subblock (ETC1/ETC2 consists of two subblocks) with only the brightness information tied to a single integer value. In a sub-block, there are only four different brightness offsets and, accordingly, only four different colors are available.

PACKMAN

If the data path occupies the same number of bits as its width, interruptions in the data path are avoided, which simplifies hardware implementation. Memory sizes and bus widths of mobile devices are limited, so 2x4 sized blocks were designed and each block was compressed to 32 bits and the compression rate was 4 bpp.

Only one RGB color value is stored in a block. Four bits per R, G, B represent the basic color for a total of twelve bits (also referred to as RGB444 since each has 4 bits). The remaining 20 bits represent brightness. For each pixel, the base color is additionally

modified by a constant taken from the four-input index table.

The same constant is added to all color components. This requires a pixel index of two bits per pixel to indicate which of the four values should be used. Coding can be done by exhaustively searching all parameters for least mean square error. It takes about a minute for a 64×64 texture on a 1.2GHz PC. A comprehensive search of other parameters based on average color is much faster and only takes about 30 milliseconds (Ström & Akenine-Möller, 2004).

ETC1 (iPACKMAN)

An image compressed with PACKMAN has significantly fewer luminance bands than an image where all pixels are 12 bit. To improve this, instead of a single block as in RGB444, two 2x4 sized blocks were used side by side, and 4x4 sized blocks in total were used.

The 4x4 ETC1 block is divided into two smaller blocks. Blocks can be stacked vertically or horizontally. As a result, it can choose the basic color with greater freedom. The RGB (R1 − R2, G1 − G2, B1 − B2) difference was made, and the color component that displayed the greatest deviation was noted. RGB555, or a 5-bit field from each color, was then made. This difference is represented by the black areas in the diagram below. It goes without saying that some blocks cannot be encoded in this manner by difference. In order to decide whether or not to use differential coding, a bit was added to the 4x4 block.

- Therefore, a difference bit coding is also used to indicate whether it is differential or normal,
- Flip bit indicating whether portrait (flip bit=0) or landscape (flip bit=1) orientation is used,
- 16 pieces 2-bit pixel indexes (one for each texture),
- two 3-bit table code words (one for each sub block) specifying which table to use
- two color code words used (independently or together) to encode the base color of the first sub block and the base color of the second sub block (Ström & Akenine-Möller, 2005).

ETC2

Although the diff blocks with RGB color differences increased the quality, since there was a single base color in each block, there were many errors due to compression in textures that were much more colorful. Therefore, these problems were tried to be corrected with extra blocks in ETC2.

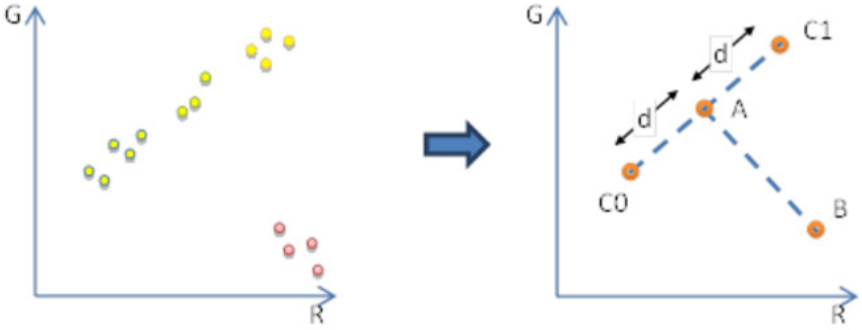
In ETC1, when the sum of the base color and the offset exceeds the valid 5-bit range [0, 31], the blocks are decoded using one of the new modes, as invalid combinations occur in the differential block. Of the 64 bits, 1 bit is spent on the diff-bit and 8 bits are spent on the R0 and dR1 values, giving 55 bits. However, it is possible to use the lowest two bits of R0 and dR1 to encode additional information. There are 3 modes for this; T, H, Planar mode. These modes are caused by the overflow of each color (R, G, B) (Ström & Martin, 2007).

In T mode, two colors A and B are compressed in RGB444 format. The remaining 3 bits encode the d value. Then the other colors are calculated as C0 = (A - (d, d, d)) and C1 = (A + (d, d, d)). This mode is applied when the colors are on a single line in Fig 7.

Figure 7

Copyright © 2024 by ISRES Publishing

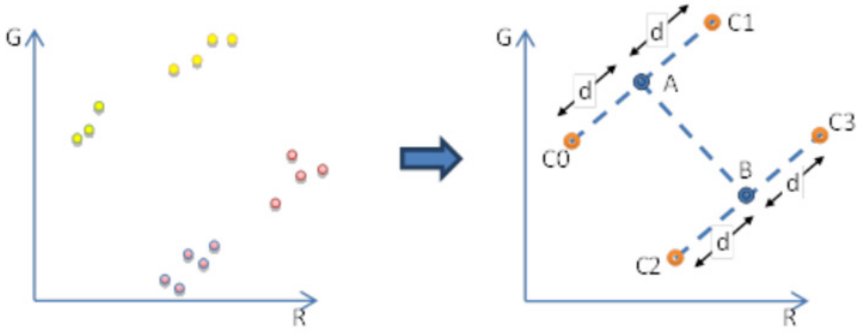
T Blocks in ETC2 (Zeng, 2016)



In H mode, unlike T3, C0, C1, C2, C3 colors are used. This mode is applied when the colors are on two lines. But the capacity of the H block is one bit less than that of the T block. Since A and B colors are symmetrical, they can be changed, and the problem is solved in Fig 8.

Figure 8

H Blocks in ETC2 (Paltashev1 & Perminov2,2014)



In planar mode, C0, CH, CV colors are available and are compressed as RGB676. And it is calculated according to the formula below (Ström & Martin, 2007);

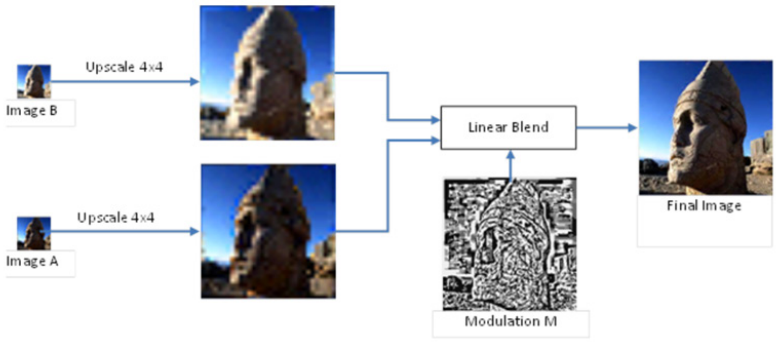
$$C(x,y) = \frac{x(C_H - C_0)}{4} + \frac{y(C_V - C_0)}{4} + C_0$$

PVRTC Family

The PVRTC (PowerVR Texture Compression) format is patented by Imagination Technologies and is designed for the PowerVR family of graphics cores (Simon, 2003). It is used on Apple mobile devices such as iPhones and iPads. There is almost no publicly available information about the PVRTC technique.

Based on PVRTC, a high-frequency and low-frequency signal are represented by two low-resolution images, A and B, scaled down by a factor of 4 in both dimensions in Fig 9. To decompress an image, images A and B are first amplified and then processed with the M signal, which indicates the processing weight for each tissue (Simon, 2003).

Figure 9
PVRTC technique (Paltashev1 & Perminov2,2014)



The main idea has something in common with wavelet compression, where the entire image is split into high-frequency and low-frequency signals. A low-frequency signal is represented by two low-resolution images A and B, scaled down by a factor of 4 in both dimensions. A high frequency signal is a modulation signal M with full resolution but low precision. To decompress an entire image, images A and B must first be amplified and then blended using the modulation signal M, which specifies the blending weights per texture (Simon, 2003).

ASTC Family

ASTC (Adaptive Scalable Texture Compression) was jointly developed by ARM and AMD and introduced in 2012 and is an open royalty-free format. ASTC format has a fixed size 128-bit block. However, it has a 4x4 to 12x12 size structure for 2D textures and a 3x3x3 to 6x6x6 size structure for 3D textures. ASTC is one of the most flexible formats because it also supports LDR, HDR, 2D and 3D textures. ASTC is one of the most flexible formats because it also supports LDR, HDR, 2D and 3D textures (Jorn, 2012).

Conclusion

Texture Compression has an important place both in computer graphics and in the use of mobile devices. The mobile platform still has much weaker hardware than desktop computers. Therefore, existing resources must be used in the most efficient way. Computational power, memory size and memory bandwidth are some of the limited resources. Texture compression is an optimization technique used to reduce the demand on memory size.

Most textures on the GPU use “block compression” formats to save sampling bandwidth, memory usage, and for faster texture loads. The common theme between them is that they are all lossy and have fixed compression ratio and memory bandwidth.

References

Aila, T., Miettinen, V., Nordlund, P., & Oy, B. (2003). Delay Streams for Graphics Hardware.

Akenine-Moller, T., Strom, J., & Research, E. (2003). Graphics for the Masses: A Hardware Rasterization Architecture for Mobile Phones.

Anuradha, D., & Bhuvaneswari, S. (2016). A detailed review on the prominent compression methods used for reducing the data volume of big data. Annals of Data Science, 3, 47-62.

Beers, A. C., Agrawala, M., & Chaddha, N. (1996). Rendering from compressed textures. Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques, 373-378.

Copyright © 2024 by ISRES Publishing

Campbell, G., DeFanti, T. A., Frederiksen, J., Joyce, S. A., Leske, A., Lindberg, J. A., & Sandin, J. (1986). Two Bit/Pixel Full Color Encoding.

Das, D., Kumar, R., & Chakrabarti, P. P. (2005, January). Dictionary based code compression for variable length instruction encodings. In 18th International Conference on VLSI Design held jointly with 4th International Conference on Embedded Systems Design (pp. 545-550).

Delp, E., & Mitchell, O. (1979). Image Compression Using Block Truncation Coding. IEEE Transactions on Communications, 27(9), 1335-1342.

Eskicioglu, A., & Fisher, P. (1993, Ekim 19). A survey of quality measures for gray scale image compression. 9th Computing in Aerospace Conference. 9th Computing in Aerospace Conference, San Diego, CA, U.S.A. https://doi.org/10.2514/6.1993-4514

Fenney, S. (2003). Texture Compression using Low-Frequency Signal Modulation.

Hakura, Z. S., & Gupta, A. (1997). The Design and Analysis of a Cache Architecture for Texture Mapping.

Huffman Coding. (2024). Lavivienpost.com.

lavivienpost https://www.lavivienpost.com/wp-content/uploads/2021/12/huffman-steps1.jpg.webp

Iourcha, K., Nayak, K., & Hong, Z. (2003). System and Method for Fixed-Rate Block-based Image Compression with Inferred Pixels Value.

Ivanov, D., & Kuzmin, Ye. (2000). Color Distribution - a New Approach to Texture Compression.pdf. Computer Graphics Forum.

Jayasankar, U., Thirumal, V., & Ponnurangam, D. (2021). A survey on data compression techniques: From the perspective of data quality, coding schemes, data type and applications. Journal of King Saud University-Computer and Information Sciences, 33(2), 119-140.

Jorn, N. (2012). Adaptive Scalable Texture Compression. Eurographics Association. High Performance Graphics, pp. 105-114.

Kocer, S., Jama, B.S.A ., Er Y. , Dundar, B. (2022). Analysing Trends and Applications in Data Compression Techniques, In S. Kocer, O. Dundar (Eds.), Current Studies in Basic Sciences Engineering and Technology (pp. 114–128). ISRES Publishing.

Khalid A “Introduction to data compression”, Third edition-2006

Knittel, G., Schilling, A., Kugler, A., & Straßer, W. (1996). Hardware for superior texture performance. Computers & Graphics, 20(4), 475-481.

Lavivienpost, https://www.lavivienpost.com/huffman-coding-and-decoding/,2024

Mccabe, D., & Brothers, J. (1998). DirectX 6 Texture Map Compression. Game Developer Magazine 5, 42-46.

Moffat, A. (2019). Huffman coding. ACM Computing Surveys (CSUR), 52(4), 1-35.

Paltashev1, T., & Perminov2, I. (2014). Texture Compression Techniques. sv-journal.org/2014-1/06/en/index.php?lang=en

Pavlo, A., (2024). Lecture# 03: Data Formats & Encoding II. https://15721.courses.cs.cmu.edu/spring2023/slides/05-compression.pdf

Pereberin, A. V. (1999). Hierarchical Approach for Texture Compression.

Pigeon, S. (2003). Huffman coding. Lossless Compression Handbook, 79-100.

Said, A., & Sayood, K. (2003). Arithmetic coding (pp. 101-152). Academic Press, San

Diego, CA.

Salomon, D. (2007). A concise introduction to data compression. Springer Science & Business Media.

Sayood, K. (2017). Introduction to data compression. Morgan Kaufmann.

Severance, D. G. (1983). A practitioner's guide to database compression tutorial. Information Systems, 8(1), 51-62.

Simon, F. (2003). Texture compression using low-frequency signal modulation. Eurographics Association. Graphics Hardware. pp. 84-91.

Strom, J., & Wennersten, P. (2011). Lossless compression of already compressed textures. Proceedings of the ACM SIGGRAPH Symposium on High Performance Graphics, 177-182. https://doi.org/10.1145/2018323.2018351

Ström, J., & Akenine-Möller, T. (2004). PACKMAN: Texture compression for mobile phones. ACM SIGGRAPH 2004 Sketches on - SIGGRAPH '04, 66.

Ström, J., & Akenine-Möller, T.(2005). iPACKMAN: High-quality, Low-complexity Texture Compression for Mobile Phones. New York, NY, USA: ACM. Proceedings of the ACM SIGGRAPH/EUROGRAPHICS Conference on Graphics Hardware. pp. 63-70.

Ström, J. & Martin, P. (2007). ETC2: Texture Compression Using Invalid Combinations. Aire-la-Ville, Switzerland: Eurographics Association. Proceedings of the 22Nd ACM SIGGRAPH/EUROGRAPHICS Symposium on Graphics Hardware. pp. 49-54.

Subathra, S., Sethuraman, M., & Babu, J. V. (2005, December). Performance analysis of dictionary-based data compression algorithms for high speed networks. In 2005 Annual IEEE India Conference-Indicon (pp. 361-365). IEEE

Witten, I. H., Neal, R. M., & Cleary, J. G. (1987). Arithmetic coding for data compression. *Communications of the ACM*, *30*(6), 520-540.

Zeng, J., Fang, L., Pang, J., Li, H., & Wu, F. (2016). Subpixel image quality assessment syncretizing local subpixel and global pixel features. *IEEE Transactions on Image Processing*, *25*(12), 5841-5856.

About the Authors

Sabri KOÇER, PhD, He graduated from the Electrical Engineering Department of Selcuk University. He completed his graduate and his doctorate in Gazi University. Currently, Necmettin Erbakan University, Faculty of Engineering, Computer Engineering is working. Electronics, Computer, Telecommunication, Signal Processing and Biomedical studies in the area.
E-mail: skocer@erbakan.edu.tr , **ORCID:** 0000-0002-4849-747X

Özgür DÜNDAR, PhD, works at Necmettin Erbakan University, Department of Astronautical Engineering. He graduated from the Electrical and Electronics Engineering Department of Selcuk University. He worked as an Automation Engineer for a while. His master's and doctorate degrees are from Selçuk University, Institute of Science and Technology, Department of Electrical and Electronics Engineering. Special fields of study are Automation, Robotic, Communication, Electromagnetic and Micro Strip Patch Antenna designs.
E-mail: ozdundar@erbakan.edu.tr , **ORCID:** 0000-0002-4142-4446

Similarity Index
The similarity index obtained from the plagiarism software for this book chapter is 17%..

Copyright © 2024 by ISRES Publishing