3. Video Compression

Digital video communication today is present in many application scenarios, such as broadcasting, internet video streaming, video capture, among others. Central to all is the communication challenge of conveying source data with the highest possible fidelity within an available bit rate, or, alternatively, conveying source data using the lowest possible bit rate, while maintaining specific reproduction fidelity. In either case, there is fundamental tradeoff between bit rate and fidelity.

The fact is that digital video takes up a lot of space. Uncompressed footage from a camcorder takes up about 17MB per second of video. Because it takes up so much space, video must be compressed before it is put on the web. To better understand what is a digital video, why it is necessary to perform a data compression on it and why this process is so computational expensive, lets start from the basics of video composition.

3.1 Image Compression

A digital image or a frame of digital video typically consists of three rectangular arrays of integer-valued samples, one array for each of the three components of a tristimulus color representation for the spatial area represented in the image. Image coding often uses a color representation having three components called Y, Cb, and Cr [16]. Component Y is called luma and represents brightness, as shown in Figure 2. The two chroma components Cb and Cr represent the extent to which the color deviates from gray toward blue and red, respectively. Because the human visual system is more sensitive to luma than chroma, often a sampling structure is used in which the chroma component arrays each have only one-fourth as many samples as the corresponding luma component array (half the number of samples in both the horizontal and vertical dimensions). This is called 4:2:0 sampling [16]. The amplitude of each component is typically represented with 8 bits of precision per sample for consumer-quality video, which means that every pixel from each component could assume values from 0 to 255.

$$\begin{bmatrix} 52 & 55 & 61 & 66 & 70 & 61 & 64 & 73 \\ 63 & 59 & 55 & 90 & 109 & 85 & 69 & 72 \\ 62 & 59 & 68 & 113 & 144 & 104 & 66 & 73 \\ 63 & 58 & 71 & 122 & 154 & 106 & 70 & 69 \\ 67 & 61 & 68 & 104 & 126 & 88 & 68 & 70 \\ 79 & 65 & 60 & 70 & 77 & 68 & 58 & 75 \\ 85 & 71 & 64 & 59 & 55 & 61 & 65 & 83 \\ 87 & 79 & 69 & 68 & 65 & 76 & 78 & 94 \end{bmatrix}$$

Figure 2. The 8x8 array of Luma with 8bit of resolution [65]

Based on this structure, a raw image file with 1980x1024 spatial resolution, has more than 2 million pixels for each component, a total of 5.8MB of data. For an isolated image, it is not too much data. However, for a composition of dozens of images, the size of data becomes an issue, specially if these images will be transmitted over a network. In this scenario, we must reduce the volume of data to be sent. This process is known as compression.

The basic image compression process, used in JPEG [63, 64] standard, can be summarized by the following steps [16]. The compression process is also illustrated in Figure 3:

1. The representation of the colors in the image is converted from RGB to the YCbCr format, that consists of one luma component (Y), representing brightness, and two chroma components, (Cb and Cr), representing color.
2. The resolution of the chroma data is reduced, usually by a factor of 2. This is called subsampling and reflects the fact that the eye is less sensitive to fine color details than to fine brightness details.
3. The image is then splited into blocks of 8×8 pixels. For each block, each of the Y, Cb, and Cr data components undergoes a discrete cosine transform (DCT). A DCT is similar to a Fourier transform in the sense that it produces a spatial frequency spectrum.
4. The amplitudes of the frequency components are quantized. Human vision is much more sensitive to small variations in color or brightness over large areas than to the strength of high-frequency brightness variations. Therefore, the magnitudes of the high-frequency components are stored with a lower accuracy than the low-

frequency components. The quality setting of the encoder (e.g. 50 or 95 on a scale of 0–100 in the Independent JPEG Group's library [12]) affects the extent to what the resolution of each frequency component is reduced to. If an excessively low quality setting is used, the high-frequency components are altogether discarded.

5. The resulting data for all 8×8 blocks is further compressed using a lossless algorithm. Lossless algorithms allows for the reconstruction of the exact original data from the compressed data. Examples are the Huffman [52] algorithm or Lempel-Ziv-Welch Coding [53, 54].

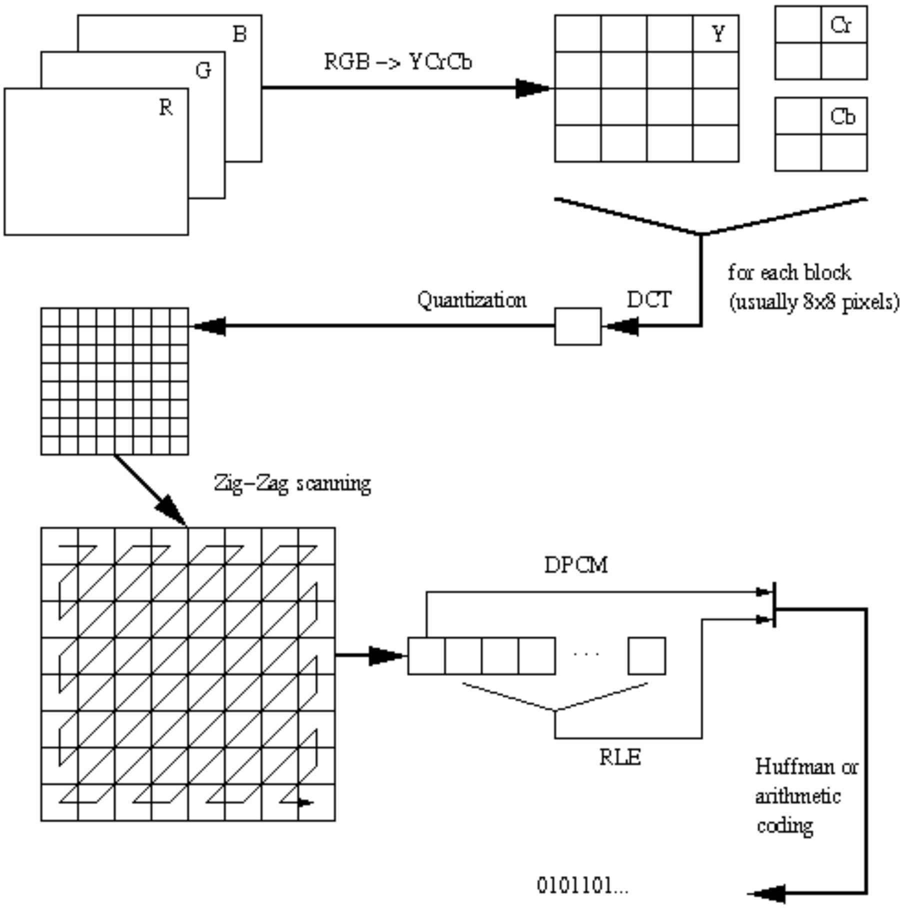

Figure 3. Image Compression Steps [66]

The compression ratio can be adjusted according to need by changing the divisors used in the quantization phase. More aggressive divisors means a greater compression ratio. Ten to one compression usually results in an image that cannot be distinguished by the naked eye from the original one. One hundred to one compression is usually possible, but will look distinctly distorted compared to the original. The appropriate level of compression depends on the use to which the image will be put to.

Figure 4, illustrates how the compression ratio affects the final quality of the decoded image. This occurs due to the quantization step, which introduces losses that can not be recovered by the decoding process. Therefore this compression process was baptized lossy compression. On the other hand, if we suppress the quantization step from the compression process, we can fully reconstruct the original image after decoding, which results in a lossless compression.

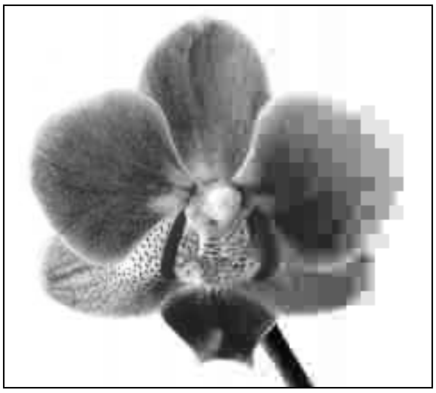

Figure 4. An image with successively more compression ratios from left to right [65]

3.2 Lossless Video Compression

Digital video comprises a series of digital images, which in the context of video are called frames. Frames are displayed in rapid succession at a constant rate. We measure the rate at which frames are displayed in frames per second (FPS).

The greater the amount of frames per second in a video, the greater will the volume of data required to represent it be. However, reducing the number of frames as a means to reduce the amount of data in a video is not a good compression solution, as the smaller is the amount of frames per second, the worse will the perception of continuity between the images displayed sequentially be. In practice, it is used rates of 20 to 30 frames per second to represent a video.

The need for video compression arises with the need to store and transmit such information over a given medium, e.g. networks. A 30 frames per second video, with the spatial resolution of 320 by 240 pixels, and a color resolution of 8 bits per pixel, results in a file of approximately 52Mbps, which makes it virtually impossible to transmit it over the current internet network infrastructure in many countries.

Because digital video is a sequence of images displayed continuously, one way of compressing it is simply compressing each picture separately. This is how much of the compression research started in the mid-1960s [13, 14]. This process basically consists in performing a lossless image compression over each single frame of the video, and it is defined by the MJPEG standard [16, 67].

In general, in the lossless compression process, each image or frame is compressed individually, without taken into consideration the redundancy between subsequent frames. Moreover, in this process the quantization step is not usually performed, i.e. each frame goes through a process of lossless compression, as described in the previous session.

This basic model is very inefficient with respect to compression ratios. If we consider high-definition videos, with bigger spatial resolution and higher frame rates, it is necessary to find an approach in which to explore further characteristics of the video, so as to have a satisfactory reduction in data volume, without deterioration in the quality of the contents.

3.3 Lossy Video Compression

We define lossy video compression, as the process by which the original video is irreparably distorted during the compression process. As the result of the lossy compression process it becomes impossible to rebuild the original file in the decoding. Lossy compression is usually required in order to achieve increased compression ratios, not feasible when using a lossless process. In this process, parts of the original data are discarded in order to drastically reduce the amount of bits needed to represent it. Ideally this is done without introducing significant distortion. In the case of videos, much of the discarded information is not perceptible to the eyes, which makes this process a good alternative to the lossless compression process. However, when compression ratios become very aggressive, it is possible to identify a very significant degradation, which can ultimately turn the contents unrecognizable.

Most modern codec's such as H.264 [17], VP8 [68], WMV, etc. are heavily based on lossy compression algorithms.

To perform a lossy video compression process, most encoders explore two of the video’s striking features: spatial and temporal redundancy. Because each video frame can be treated as an isolated image, image compression techniques can be applied directly over them, which will act to reduce spatial redundancy, taking advantage of the correlation between adjacent pixels to reduce the number of bits needed to represent each frame. As to the process of image compression, it is possible to include a quantization step in the compression process of each frame, which will reduce the amount of symbols needed to represent the information through the approximation of similar values. Once down this path, the original values can no longer be recovered, which result in the insertion of irreparable distortions in the content. As mentioned earlier, the larger the quantization step, the greater the distortions will be visible on the frames. Typically these can be identified through the formation of small blocks in images, as illustrated by Figure 4.

Furthermore, much of the depicted scene is essentially just repeated in picture after picture without significant changes, so video can be represented more efficiently by sending only the changes to the video scene, rather than coding all regions repeatedly. We refer to such techniques as inter-picture or inter coding. The ability to use temporal redundancy to improve coding efficiency is what fundamentally distinguishes video compression from the intra frame compression process exemplified above.

The basic method of inter compression is to calculate the prediction error between corresponding blocks in the current and previous frames. The error values are then sent to the compression process. Compressed frames generated using prediction are usually called P-frames. When we use both previous and future frames as the reference, the frame is called B-frame (bidirectional frame), as shown in Figure 5.

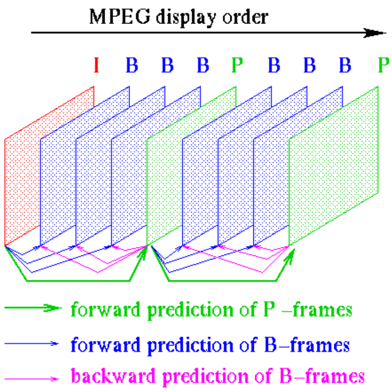
MPEG display order
I B B B P B B B P
forward prediction of P –frames
forward prediction of B–frames
backward prediction of B–frames

Figure 5. The i-frames, p-frames and b-frames

In motion compensation, depicted in Figure 6, is the process by which each image frame is divided into a fixed number of, usually, square blocks. For each block in the frame, a search is made in the reference frame over the area of the image that allows for the maximum translation that the coder can support. The search is for the best matching block, to give the least prediction error, usually minimizing either the mean square difference, or the mean absolute difference, easier to compute. Typical block sizes are in the order of 16x16 pixels, and the maximum displacement might be around 64 pixels from a block's original position. Several search strategies are possible, most use some kind of sampling mechanism. The most straightforward approach is exhaustive search.

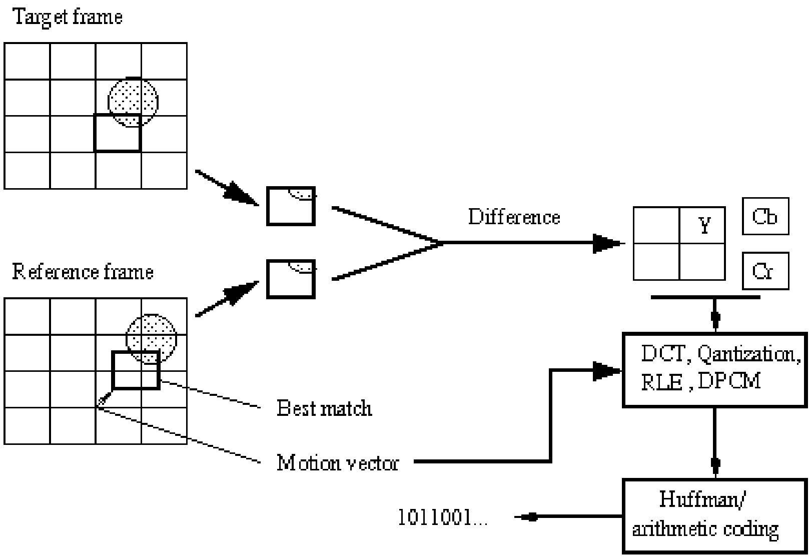

Figure 6. Motion Compensation [66]

An example of the block structure generated in order to perform a motion compensation is exemplified by Figure 7, that borrows from the MPEG-4 test sequence known as "Foreman"[1] [55]. Note that the stationary background is represented by a large numbers of blocks with very similar motion vectors (represented by the short lines starting from each block centre).

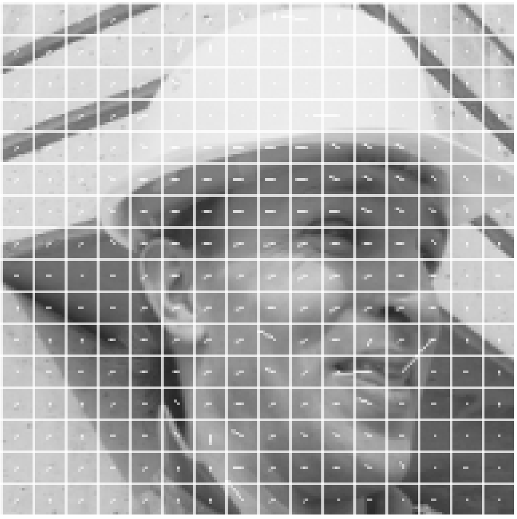

Figure 7. Motion Vectors in a Video Frame

3.4 Video Compression for Internet Distribution

Internet Video distribution is not a new process, however, it has become more and more popular in the past few years, mainly as consequence of improvements in connectivity and bandwidth. Although there's been a significant increase in available bandwidth, the Internet today has a limited capacity to efficiently transfer content in situations where higher data rates are required, e.g. uncompressed video. To allow for the exchange of video content using a common Internet connection, the information must be compressed to low data rates, preferably in the order of a few kilobits per second. The first popular solution for this process was developed by Real Networks, in 1997, with RealVideo, based on H.263 [57] video codec, which delivered a video with no more than a 28.8 Kbps dial-up connection to the Internet.

As time went by, new and better compression algorithms became available, allowing better quality at a lower bitrates. In the past few years, Internet video

[1] – The Foreman sequence, available publicly, shows closeup of a talking man followed by a camera panning and view of a building being constructed, and is commonly used by ITU-T Video Coding Experts Group for video processing tests

compression technology has been converging around the H.264/AVC compression standard [17], which is one of the existing *codecs* with highest compression ratio efficiency.

The intent of the H.264/AVC project was to create a standard capable of providing good video quality, at substantially lower bit rates, than previous standards, e.g. MPEG-2 [56], H.263 [57], or MPEG-4 Part 2 [58], without increasing the complexity of design. A design excessively complex would render this solution impractical and too expensive to implement. The standardization of the first version of H.264/AVC was completed in May 2003, has become ever more popular since then.

The evolution of video compression for Internet distribution is a constantly changing scenario, where the dominating technology is always being replaced. It started with RealVideo, which was the most popular technology from 1997 to 2004. In this year Microsoft's Windows Media Services took its place with the new WMV9 codec, that introduced a new process by which to obtain significantly better video compression quality. This technology rapidly became the most popular technology for internet video, forcing those using Real's solution to migrate to Microsoft's platform. However, with the prioritization of Windows Vista project, Microsoft stopped the evolution of their media platform, which allowed Adobe to develop a similar platform and compete for this market. Adobe, however, did not have a good video compression standard (Sorenson Spark [59]). To make their product more attractive, they focused on user experience and integration of the player inside web pages [60].

With this approach, Adobe's Flash Media platform called the attention of a startup called YouTube, in 2005, which decided to adopt Adobe's technology as their video platform. With the popularization of Internet video, due in a great part to the YouTube phenomena, the Flash Media Platform started to gain market against Microsoft's technology, and became the dominant Internet video technology.

To improve video quality, Adobe added H.264 support to their Flash Player in 2007, and, since then, it became the de facto standard for video on the web.

Analyzing the evolution of Internet video technology, we highlight the constant change in video compression standards. This short history is relevant to the work presented here because for every format/standard change, all legacy content must be re-encoded so that it complies to new standard. This fact has a great impact to large broadcasters, who are forced to transcode very large amounts of video at a large cost to convert their legacy collections. In addition, this process is usually quite complex, time consuming, resource demanding, and very slow.

One last observation is that Internet video consumption is no longer restricted to the computers or web browsers. Today there is a proliferation of devices with playback video capabilities, and different video format support. This means that it is usually the case that a single video needs be transcoded several times, to ensure compatibility with different devices. Worst, of all, legacy content must also be transcoded every time a new device, with a new supported format, is put in to the market.

Content producers, from an economical point of view, have little interest to invest in a large transcoding infrastructure. Firstly, because they do not know when a new format will become popular, and, secondly, because they only need to perform the transcoding process once a new format comes in the market. If they invest in a large hardware infrastructure, capable of handling transcoding jobs, chances are the resources will remain idle a great part of the time.

On demand resource allocation made possible by the Cloud, on the other hand, fits perfectly in this scenario, as it allows for the provision of computer processing and storage capabilities according to current needs. When no longer needed, such resources are released.

Based on this, the ability to process video efficiently in a Cloud environment becomes strategic for media production companies. This is the focus of this research. In the following chapter we detail the proposed architecture, showing how it can take advantage when deployed in a Cloud platform.