Video Compression

Ameya Anjarlekar

June 2020

1 Problem Statement

To implement an efficient video compression algorithm which can help reduce storage costs without much loss in video quality.
There are video compression algorithms which any computer device already has in-built in it. Thus a great deal of compression happens when we store a video. We need to implement compression on top of it. Also, we must take care about our video compression algorithm not disrupting the in-built compression algorithm.

2 Some facts

1) An image has a lot of redundancy. We can observe that the neighbouring pixels are almost same. Similarly for a video, consecutive frames are almost similar. We can use this redundancy.
2) Any image/video can be represented in form of certain basis. For example, consider a 256X256 image which is fully black. One option is to store all the pixels of the image. Or we could just store the average value and use it while reconstructing. Thus, we can observe that a lot of data will be saved.
However, we considered that the image was a plain black image. Nonetheless, it has been observed that for natural images, we can represent them in certain bases (for example, DCT, Fourier, Haar). If we represent them in these bases we get a 256X256 image(now having the coefficients in the selected basis) having most of the values nearly 0. We observe that if we approximate those values to 0, there is negligible change in the image quality (Almost invisible to the eye). This principle is used in various image/video compression algorithms such as JPEG/MPEG.

3 Compression

We consider n=5 frames. (More the frames more the compression, but the video quality might deteriorate). Consider the frames be F1, F2, F3, F4, F5.
The coded image given by C1*F1+C2*F2+C3*F3+C4*F4+C5*F5. Similarly

The barbara image (left) of size 512 x 512 and its reconstruction from just the top 100,000 (out of 262144) DCT coefficients. The difference is indistinguishable and similar results hold for wavelet transforms as well

other coded images are generated form subsequent frames. Further, they are combined together in form of a video (to use the in-built video compression algorithm of the device).
The codes C1, C2, C3, C4, C5 are binary values (0 or 1). These will be same for the subsequent set of frames.
These codes are also stored along with the video. Since they are not dependent on the number of frames in the video, their storage size is negligible for large videos.

4 Decompression

We consider each of the coded image separately. Every pixel in the coded image is a linear combination of the original pixels. Thus if y is a vector containing the pixels of the coded image. The x can be written as y= Cx where x is the vector containing the pixels of the actual frames image. Now x can be written sparsely in the form of a basis vector. Thus $x = D * \theta$. We generally break the image into various patches and then estimate x.
Now, this is a under-determined system (number of unknowns ¿ number of equations). Given that θ is sufficiently sparse and C is sufficiently random, there exists theoretical bounds which show that x can be very approximately estimated. Intuitively we can understand this by thinking that θ has very few non-zero elements. Thus, we need to just find those elements.
The loss function which we are using to estimate the actual frames is

$$L = ||y - CD\theta||_2^2 + ||\theta||_0 \tag{1}$$

Where $||.||_k$ is the kth norm. Sometimes, l1 norm is used for θ. We are using OMP algorithm to reconstruct the image. Other algorithms which are used are LASS0 (which uses L1 norm).