MPEG-4 Compression without all the Math

Peter Blomgren
Department of Mathematics
San Diego State University

November 2, 2016

Contents

1 Introduction: Rationale **1**

2 Description of the Problem **1**
2.1 1080p Movies 1
2.2 A Single Frame 2
2.2.1 Looking at a Single Horizontal/Vertical Line of the Image 2
2.2.2 Back to 2D 3

1 Introduction: Rationale

In *Math 254: Intro to Linear Algebra* I used the idea of video compression to motivate *why* we should care about orthogonal basis... The discussion was somewhat hand-wavy, and afterward I got the question

> Yesterday in the 12pm math 254 class you mentioned that, because of data constraints, for a movie to be stored on a disc, something like 95% of the movie's data would need to be discarded. That sounds really interesting to me, and I can't think of how only 5% of the data could still be seen as a smooth/complete version of the movie. How does this work?

This document is an attempt at describing the key ideas of video compression, using only concepts from calculus and half a semester of linear algebra. (Good luck to me!)

2 Description of the Problem

2.1 1080p Movies

Imagine a 2-hour 1080p24 Movie; where we are showing 24 frames/second, and each frame is 1920×1080 pixels, each pixel has a bit depth of 8-bits per color (whether that's Red-Green-Blue, or Y-Cb-Cr, is a discussion for someplace else); but the bottom line is that we have 3 bytes/pixel, so we end of with a raw datastream with

$$3\,\text{bytes/pixel} \times 24\,\text{frames/second} \times 7200\,\text{seconds} \times (1920 \times 1080)\,\text{pixels/frame} = 1,074,954,240,000\,\text{bytes}.$$

Now, keeping in mind that a standard dual-layer Blu-ray disc holds a measly 50,050,629,632 bytes of data, we need a compression ratio of `1:21.5` in order to fit the movie onto a disk. This means we can only store slightly less than 4.7% of the datastream.

2.2 A Single Frame

Let's for a moment restict our discussion to a single 1920×1080 pixel frame; and for simplicity, let's make it gray-scale.

Figure 1: Gandalf, looking particularly gray...

Next we are going to discuss how we can compress this single snapshot to use only 4.6% storage. This is going to require a little bit of mathematics...

2.2.1 Looking at a Single Horizontal/Vertical Line of the Image

First, we can consider the image to be constructed out of 1080 lines, each with 1920 pixels; which means we have a collection of 1080 vectors $\vec{r}_1, \vec{r}_2, \ldots, \vec{r}_{1080}$, each "living it up" in $\mathbb{R}^{1920}$; we can also (simultaneously) think of the as being constructed out of 1920 columns, each with 1080 (vertical) pixels; giving us vectors $\vec{c}_1, \vec{c}_2, \ldots, \vec{c}_{1920}$, each "living it up" in $\mathbb{R}^{1080}$.

What we need are some good (orthonormal) bases for $\mathbb{R}^{1080}$ and $\mathbb{R}^{1920}$. It turns out that if we given an even number, $2n$ points, we can use the $2n$ vectors generated by the functions

$$\begin{cases} \Phi_0(x) &= \dfrac{1}{2} \\ \Phi_k(x) &= \cos(kx), \quad k = 1, \ldots, n \\ \Phi_{n+k}(x) &= \sin(kx), \quad k = 1, \ldots, n-1 \end{cases}$$

evaluated in the interval $[-\pi, \pi]$, at the equally spaced points $x_j = -\pi + (j\pi/n),\ j = 0, 1, \ldots, (2n-1)$. The generated set of vectors are orthonormal!

Now, align the points x_j with the pixels, numbered from $j = 0$ to $j = (2n-1)$, horizontally or vertically. Let p_j denote the pixel value (gray-scale intensity). Now, if we let

$$a_k = \frac{1}{n} \sum_{j=0}^{2n-1} f_j \cos(kx_j) \quad b_k = \frac{1}{n} \sum_{j=0}^{2n-1} f_j \sin(kx_j),$$

be the values of the [pixel-vector]–[cos/sin-vector] dot-products. In our language the a_k and b_k coefficients are coordinates in the cos/sin-vector basis for $\mathbb{R}^{2n}$; and given the coordinates, we can fully reconstruct the pixel values:

$$p_j \equiv S(x_j) = \frac{a_0}{2} + \frac{a_n}{2}\cos(nx_j) + \sum_{k=1}^{n-1}\left[a_k\cos(kx_j) + b_k\sin(kx_j)\right].$$

We now have a set-up where we can go from "image coordinates" to [cos/sin-vector] coordinates (and back) using only dot products. What we have defined is known as the (one dimensional) *Fourier transform.*

2.2.2 Back to 2D

Even though the previos discussion gave us a nice way to build orthonormal bases in one dimension, it is far from clear *why* this is desirable. Now, consider the image we had of Gandalf; and lets perform the above procedure first in the horizontal direction (which transforms the image into 1080 lines of [cos/sin-vector] coordinates. Next, transform *that* "image" in the vertical direction. This now gives us an "image" of vertial [cos/sin-vector] coordinates of (horizonal [cos/sin-vector] of Gandalf). This is known as the two dimensional *Fourier transform.* Figure 2 shows what that looks like.

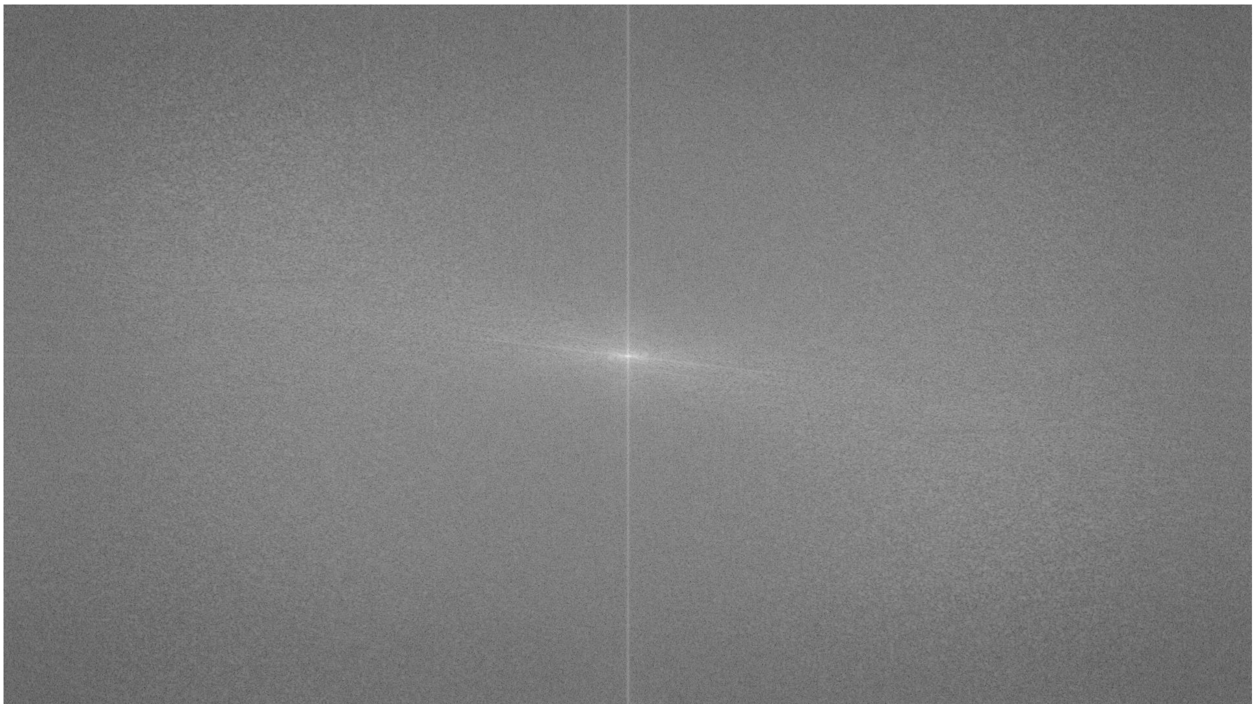

Figure 2: The two dimensional Fourier transform of Gandalf.

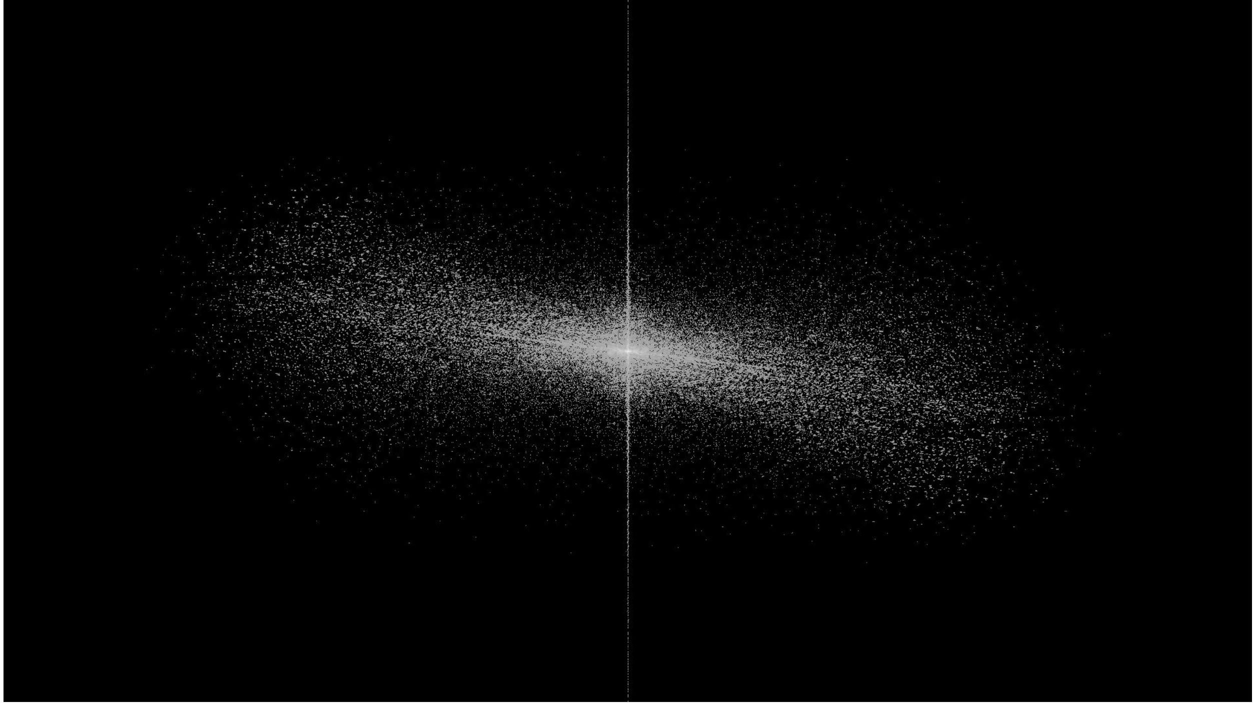

Figure 3: Gandalf, Fourier transformed and filtered: only the largest 4.6% of coefficients have been kept.

Figure 4: Gandalf, Fourier transformed and filtered: only the largest 4.6% of coefficients have been kept.