Image Compression

Introduction

If we were to store all the raw data of a single image, say (1920x1080x24) bits per frame, then storing a single 2 hour movie at 30 fps would be prohibitively costly. **Compression** allows us to reduce the space an image takes on a disk. We have a reduction ratio C which can allow us to reduce the space an image takes.

There are a couple of redundancies that we can make sure we use: 1. **Coding redundancy**: We assign a code to a piece of information or events. If we use 8 bits to store the intensity, this might be more than we need, thus we waste some potential space. 2. **Spatial and Temporal Redundancy**: Since pixels are similar to neighboring pixels, and those pixels might not vary that much from frame to frame, we might be replicating the values of many pixels that might be similar at the same time. 3. **Irrelevant Information**: If our visual system ignores certain pieces of information, then it would be unnecessary to store all of them.

Coding Redundancy

We have the probability of a certain intensity value as

$$p_r(r_k) = \frac{n_k}{MN}$$

which means that out of MN pixels, there is a $p_r(r_k)$ probability of a certain intensity value. We can then find the average amount of bits used to represent each value by

$$L_{avg} = \sum_{k=0}^{L-1} l(r_k) p_r(r_k)$$

where $l(r_k)$ is the number of bits used to represent each value. This equation means that the total number of pixels required will be the product of the number of bytes and the probability that each intensity value occurs. The l value represents the code that we use to denote the intensities. Thus, if we can assign a shorter code to the intensities that appear more frequently, we can also reduce the amount of space required.

Another way of reducing space is by using **run-based encoding**. This type of encoding would involve specifying the intensity and how many pixels in a row have that intensity[1]. Thus, for a 256 pixel row, we could cut that down to just one 8-bit code plus the intensity value.

[1] the **run**

The next question becomes: what is the fewest number of bits needed to represent the information in the image? Shannon's Coding Theorem showed that we can do so in as little as 1.664 bits/pixel.

Since we are removing information from the image, we need a way of quanitfying such information. There is **objective fidelity criteria**, which deals with a mathematical function representing our loss, and **subjective fidelity criteria**. If we wanted to assign a value to the amount of loss, we could use root-mean-square error. Suppose we have and image f, and we compress and decompress a copy $\hat{f}$.

$$e(x,y) = \hat{f}(x,y) - f(x,y)$$

for pixel x, y. The total error between the two images would be

$$\sum_{x=0}^{M-1} \sum_{y=0}^{N-1} [\hat{f}(x,y) - f(x,y)]$$