An Introduction to Video Compression in C/C++

Fore June

Chapter 1

Elements of Information

2.1 What Is Information?

Basically, everyone knows that this is an information era and every day we talk about the importance of information technology and its development. Now, let us confuse you with a simple question. *What is information?* Before continuing your reading, try to answer the preceding question. Like most programmers, you may be surprised to find that after so many years of studying or working in the area of information, you really could not answer the question unless you have taken a course in information theory or have studied the subject before. To understand how data compression works, we need to first understand what information is.

Some may think that information is simply a concept and we cannot define it quantitatively. Actually, not only that we can define information, we can define it quantitatively and measure it.

Many people, including some authors of books on image compression, confuse information with data and may mistaken that information is the same as data. Actually, data is different from information. Data are used to represent information. We can have redundant data but not redundant information. We can have a large amount of data which contains very little information. For example, we can use a pseudo random number generator to generate an abundant amount of data, tens of millions of bytes. However, all these data contain very little information because if we want to transmit the information represented by these data to another person, all we need to do is to transmit the simple equation of the pseudo random number generator. The receiver can generate the huge amount of data identical to ours. As another example, when we watch news, we feel that we receive a significant amount of information if the news gives us surprises, informing us something unexpected. On the other hand, if your friend tells you that she will eat dinner tomorrow, you do not feel receiving much information as that's what you expect. For instance, consider the following two sentences,

1. India will elect her next governing party by universal suffrage.
2. China will elect her next governing party by universal suffrage.

These two sentences have exactly the same amount of data (characters). However, there is a big difference in information the two sentences would convey. Should the event happen, the first one does not give us any surprise and would not appear in any newspaper as that's what we would expect. However, the second one would give a big shock to the world and every newspaper would report the event; it gives us a significant amount of information. From these examples, we can see that information relates to unpredictability and surprises. It is a measure of the decrease of uncertainty or the gain of surprise of a receiver upon receiving a message. A perfectly predictable message conveys no information. To quantify the measure of information, scientists borrow the

concept of entropy from physics. We know that in physics, entropy is a measure of disorder or unpredictability. In information systems, we also refer to information carried by a message as entropy. However, the use of the term entropy to describe information content is artificial. There is not much relationship between the entropy of a message and the physical entropy of a physical system.

Claude Shannon, known as the father of information theory published his landmark paper in 1948 that led to the dawn of the information era. In the paper, Shannon defined the information I(E) conveyed by an event E, measured in bits as

$$I(E) = log_2 \frac{1}{p(E)} \tag{2.1}$$

where $p(E)$ is the probability of the occurrence of the event. In other words, if E is some event which occurs with probability $p(E)$ and we are informed that event E has occurred, then we say that we have received $I(E)$ bits of information given in equation (2.1). We see that when $p = 1, I = 0$. For example, if we are told that "The sun rises from the East", we do not receive any information as we are one hundred percent sure this happens. If $p(E) = 1/2$, then $I(E) = 1bit$, meaning that one bit is the amount of information we obtain when one of two possible likely outcomes is specified, like the case of examining the outcome of flipping a coin. We can also interpret I(E) given in (2.1) as the information needed to make the occurrence of E certain. Note that equation (2.1) can also be expressed as,

$$I(E) = -log_2 p(E) \tag{2.2}$$

We can also define entropy in terms of a discrete random variable X, with possible states (or outcomes) $x_1, ..., x_n$ as

$$H(X) = \sum_{i=1}^{n} p(x_i) log_2 \left(\frac{1}{p(x_i)} \right) = - \sum_{i=1}^{n} p(x_i) log_2 p(x_i), \tag{2.3}$$

where $p(x_i) = Pr(X{=}x_i)$ is the probability of the ith outcome of X. Note that a random variable is not a variable in the usual sense but rather a function that maps events to numbers. To simplify our notation, henceforth, we shall write the logarithm to the base 2 of x simply as $logx$, omitting the subscript on the "log".

Equation (2.3) can also be expressed as

$$H(X) = \sum_{i=1}^{n} p(x_i) I(x_i), \tag{2.4}$$

Therefore, the entropy of the discrete random variable X is the average information of its states.

2.2 Memory Source

Rather than using a discrete random variable to further study information, it is more intuitive and convenient to consider a model of discrete information source as shown in Figure 2-1. In the model, the source generates a sequence of symbols from a fixed finite source alphabet $X = \{x_1, x_2, ..., x_n\}$. Successive symbols are generated according to some fixed probability law.

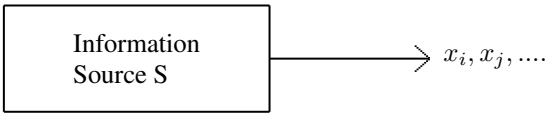

Figure 2-1. A discrete information source

For example, we can generate an English text message with a computer program; the alphabet X consists of letters {a, b, c, d, ...} and digits { 1, 2, 3, ...}. In this model we can also view each symbol $\{x_i\}$ as a state and the alphabet X as a random variable.

If the successive symbols generated from the source are statistically independent, the information source is referred to as **zero-memory source** (or order-0 Markov source), which is the simplest kind of sources one can have. Such an information source can be completely described by the source alphabet X and the probabilities with which the symbols occur:

$$p(x_1), p(x_2), p(x_3), ..., p(x_n)$$

If symbol x_i occurs, we obtain an amount of information $I(x_i)$ bits given by

$$I(x_i) = log \frac{1}{p(x_i)}$$

This means that if we receive a symbol z that is very unlikely to appear, $p(z)$ is very small; $\frac{1}{p(z)}$ and thus $I(z)$ is very large. In other words, we get a lot of surprise (information) when we receive something totally unexpected. On the other hand, if $p(z)$ is large, $I(z)$ is small. That is, we gain little information (surprise) when we get something we expect.

The probability for x_i to occur is simply $p(x_i)$. So the average amount of information per symbol one can receive from the source S is

$$\sum_X p(x_i) I(x_i) \quad bits$$

where we sum over the n symbols of the alphabet X; this average information is the entropy $H(S)$ of the zero memory source.

$$H(S) = H(X) = \sum_{i=1}^{n} p(x_i) log \frac{1}{p(x_i)} \quad bits \tag{2.5}$$

The source gives maximum entropy $H_M(S)$ when all symbols occur with the same probability $p(x_i) = \frac{1}{n}$.

$$H_M(S) = \sum_{i=1}^{n} \frac{1}{n} log\ n = log\ n \quad bits$$

Therefore, an alphabet with 256 symbols has a maximum entropy of 8 bits, which is the maximum amount of average information per symbol that the source can generate from it.

2.3 Markov Memory Source

In the zero-memory model, the occurrence of each symbol is statistically independent of each other. However, in reality symbols are statistically related to each other in most cases. For example, in an English text, the probability for letter 'u' to occur is quite small. However, if we receive a letter 'q', we know that the next letter that we shall receive is very likely to be a 'u'. If we have received the letters 'democrac', there is a large chance that the next letter is a 'y'. To better study the information content of this kind of symbol sequences, we need a model in which the occurrence of a source symbol x_i may depend upon m preceding symbols. Such a source is referred to as an mth-order Markov source and is defined by specifying the occurrence of source symbols with the set of conditional probabilities

$$p(x_i / x_{k_1}, ..., x_{k_m}) \quad for \;\; i = 1, 2, ..., n; \;\; k_j = 1, 2, ..., n \tag{2.6}$$

which is the probability of seeing x_i after we have seen m symbols.

It is often convenient to use finite state diagrams to illustrate an mth-order Markov source. At a given time, we can refer to the m preceding symbols as the **state** of the mth-order Markov source at that time; as there are n symbols in the alphabet, there are n^m possible states. Figure 2-2 shows a state diagram of a **second-order** (i.e. $m = 2$) Markov source with binary alphabet X = {0, 1} and conditional probabilities

$$\begin{array}{ll} p(0/00) = 0.8 & p(1/00) = 0.2 \\ p(0/01) = 0.5 & p(1/01) = 0.5 \\ p(0/10) = 0.5 & p(1/10) = 0.5 \\ p(0/11) = 0.2 & p(1/11) = 0.8 \end{array}$$

In the diagram, states are represented by circles and state transitions are indicated by arrows labeled with the corresponding conditional probabilities. (This Figure and related examples discussed below are taken from the book *Information Theory and Coding* by **Norman Abramson**.)

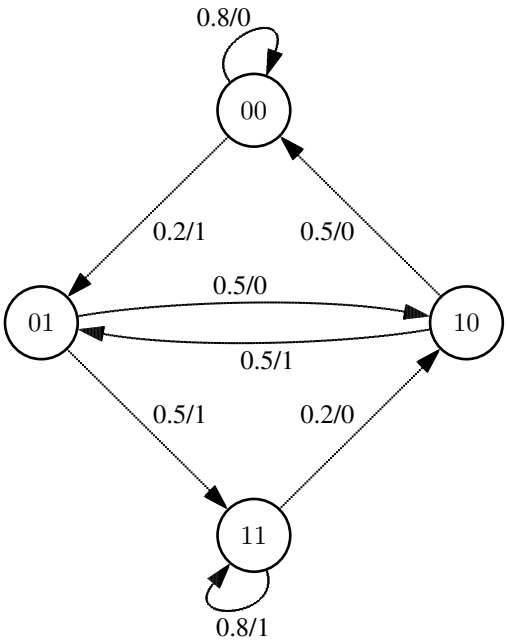

Figure 2-2. State diagram of a second-order Markov source with alphabet {0, 1}

In image processing, it is common to use a second or third order Markov model to predict a pixel value. That is, two or three previously occurred pixel values are used to estimate the value of a forthcoming pixel.

In many cases (but not all), the probability distribution over the set of states do not change with time. We refer to this kind of distribution as stationary distribution. In these cases, when we specify the conditional symbol probabilities $p(x_i/x_{k_1}, ..., x_{k_m})$ of an mth-order Markov source, we also implicitly specify the n^m state probabilities $p(x_{k_1}, x_{k_2}, ..., x_{k_m})$. The product of these two probabilities gives us the probability of the joint event "*the source is in the state* $(x_{k_1}, x_{k_2}, ..., x_{k_m})$ *and* x_i *occurs*". That is

$$p(x_{k_1}, ..., x_{k_m}, x_i) = p(x_i/x_{k_1}, ..., x_{k_m})p(x_{k_1}, ..., x_{k_m}) \tag{2.7}$$

The information we obtain if x_i occurs while the system is in the state $(x_{k_1}, x_{k_2}, ..., x_{k_m})$ is

$$I(x_i/x_{k_1}, ..., x_{k_m}) = log\frac{1}{p(x_i/x_{k_1}, ..., x_{k_m})} \tag{2.8}$$

and the average amount of information per symbol while the system is in the state $(x_{k_1}, x_{k_2}, ..., x_{k_m})$ is given by:

$$H(X/x_{k_1}, ..., x_{k_m}) = \sum_X p(x_i/x_{k_1}, ..., x_{k_m}) I(x_i/x_{k_1}, ..., x_{k_m}) \tag{2.9}$$

where the summation is over the n symbols in the alphabet X. If we average this quantity over all the n^m possible states, we obtain the average amount of information, or the entropy of the mth-order Markov source S.

$$H(S) = \sum_{X^m} p(x_{k_1}, ..., x_{k_m}) H(X/x_{k_1}, ..., x_{k_m}) \tag{2.10}$$

In other words, if we have a very long text consisting of n symbols which can be described by such a model, then $H(S)$ of (2.10) gives us the average information per symbol of the text. For example, in a long Internet message, we can take the alphabet as the generalized ASCII code, consisting of 256 symbols or characters (i.e. $n = 256$). To estimate the average information content of the message, we may take m to be 4 (i.e. using a 4-th order Markov model). There will be totally $256^4 = 2^{32}$, about 4 billion states. We have to collect the statistics over the 4 billion states to estimate the probabilities and use them to find the entropy given in (2.10), which gives us the average information per symbol. The total information of the message is then the length of the message times $H(S)$.

Substituting (2.9) and (2.8) into (2.10) and making simplifications, we can express the entropy (average information per symbol) in the following form:

$$H(S) = \sum_{X^{m+1}} p(x_{k_1}, ..., x_{k_m}, x_i) \times log \frac{1}{p(x_i/x_{k_1}, ..., x_{k_m})} \tag{2.11}$$

Example 2-1

Consider the Markov source of Figure 2-2, where the alphabet only consists of two symbols which are the binary digits, i.e. $X = \{1, 0\}$. Since the stationary distribution does not depend upon the initial states, we can calculate the probability for each state from the conditional symbol probabilities. One can show that the stationary distribution is:

$$\begin{array}{ll} p(00) = \frac{5}{14} & p(01) = \frac{2}{14} \\ p(10) = \frac{2}{14} & p(11) = \frac{5}{14} \end{array}$$

We summarize the relevant probabilities in Table 2-1:

Table 2-1 Probabilities for Markov Source $X = \{0, 1\}$ of Figure 2-2

x_{k_1}, x_{k_2}, x_i	$p(x_i/x_{k_1}, x_{k_2})$	$p(x_{k_1}, x_{k_2})$	$p(x_{k_1}, x_{k_2}, x_i)$
000	0.8	$\frac{5}{14}$	$\frac{4}{14}$
001	0.2	$\frac{5}{14}$	$\frac{1}{14}$
010	0.5	$\frac{2}{14}$	$\frac{1}{14}$
011	0.5	$\frac{2}{14}$	$\frac{1}{14}$
100	0.5	$\frac{2}{14}$	$\frac{1}{14}$
101	0.5	$\frac{2}{14}$	$\frac{1}{14}$
110	0.2	$\frac{5}{14}$	$\frac{1}{14}$
111	0.8	$\frac{5}{14}$	$\frac{4}{14}$

Note that $p(x_{k_1}, x_{k_2}, x_i) = p(x_i/x_{k_1}, x_{k_2}) \times p(x_{k_1}, x_{k_2})$. We can now calculate the average entropy of the system using (2.11):

$$H(S) = \sum_{X^3} p(x_{k_1}, x_{k_2}, x_i) log \frac{1}{p(x_i/x_{k_1}, x_{k_2})} \tag{2.12}$$

Substituting the probabilities in Table 2-1 into (2.12), we obtain

$$\begin{aligned} H(S) &= \tfrac{4}{14} \times log\frac{1}{0.8} + \tfrac{1}{14} \times log\frac{1}{0.2} + \tfrac{1}{14} \times log\frac{1}{0.5} + \tfrac{1}{14} \times log\frac{1}{0.5} \\ &\quad + \tfrac{1}{14} \times log\frac{1}{0.5} + \tfrac{1}{14} \times log\frac{1}{0.5} + \tfrac{1}{14} \times log\frac{1}{0.2} + \tfrac{4}{14} \times log\frac{1}{0.8} \\ &= 0.81 \text{(bit / binary digit)} \end{aligned}$$

In equation (2.10), X^m is the mth extension of the alphabet X, which has n^m symbols, $\sigma_1, \sigma_2, ..., \sigma_{n^m}$ and each σ_i corresponds to some sequence of m $x_k's$. One can define the r-th extension of an m-th order Markov source, where we group r symbols together to form one new 'super-symbol'. If we consider each of these 'super-symbols' as a symbol, our alphabet X becomes:

$$X = \{\sigma_1, \sigma_2, ..., \sigma_{n^r}\}$$

By grouping r symbols together, we have considered the correlation between r symbols, which can give us better estimate of the information content of a text. In reality, there could be long-range correlations between groups of symbols. To accurately calculate the average information, we actually need to consider an extremely long message and make the group as large as possible. Theoretically, the average information of a 'typical' stationary infinite text is given by the entropy rate $H(X)$, which is the limit of the joint entropy of n 'symbols' averaged over n:

$$H(X) = \lim_{n\to\infty} \frac{1}{n} H(X_1, X_2, ..., X_n) \tag{2.13}$$

We may also define the entropy rate, $H(S)$ using conditional probabilities:

$$H(S) = \lim_{n\to\infty} \frac{1}{n} H(X_n | X_{n-1}, X_{n-2}, ..., X_1) \tag{2.14}$$

One can show that the two quantities defined in (2.13) and (2.14) are basically equal (i.e. $H(S) = H(X)$).

What (2.14) tells us is that when reading a text, if we use a sufficiently large number of symbols to predict the forthcoming symbol, the accuracy of prediction reflects the average information of the text. If we can predict the next symbol very well, there is a lot of redundancy and the average information which is equal to the entropy rate is very small. Since n tends to infinity in (2.13) and (2.14), we have exhausted the search of any long-range correlations in the text.

2.4 Information of Text and Kolmogorov Complexity

In the previous section, we have discussed that the average information a text contains is given by its entropy rate. The problem of this approach is that entropy rate is defined using an infinite text of symbols. We have also discussed a model of information source and some of its properties. We are interested to know how closely the model and theories relate to the physical process of information generation in real life. In practice, every text is of finite length and we can only estimate its information content using predictions based on a finite number of preceding symbols. In some situations, such an estimate can be totally off. For instance, consider a simple experiment that uses the following program, **randtest.cpp** to generate 10 million bytes of data using a pseudo random number generator:

```
//randtest.cpp: Generates 10 million bytes of data
//Compile: g++ -o randtest randtest.cpp
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

int main()
{
  int x;
  char *pc = (char *)&x;          //points to low byte of x
  FILE *fp;

  fp = fopen ( "randnums", "wb" );//open binary file "randnums"
                                  // for write, no error checking
  for ( int i = 0; i < 10000000; ++i ){ //output 10 million bytes
    x = rand();                         //output only lowest byte
    putc ( (int) *pc, fp );             //output one byte
  }
  return 0;
}
```

The 10 millions bytes of data are saved in the file "randnums". Now let us use the common compression utility **gzip** to compress the data, saving the output data in the file "randnums.gz" and check the file sizes:

```
$ ./randtest
$ gzip -c randnums > randnums.gz
$ ls -l randnums*
  -rw-r--r-- 1 user user 10000000 2009-11-21 11:10 randnums
  -rw-r--r-- 1 user user 10001557 2009-11-21 11:10 randnums.gz
```

We see that the compressed file size is about the same as the original size. This implies that we were not able to compress the data and the average information of the file is 8 bits per symbol. If we calculate the entropy using Equation (2.12), we shall obtain a similar value. In other words, the file "randnums" contains about 80 million bits of information! If we need to transmit these data to a friend we need to transmit 80 million bits! We know that this could not be true because the data are generated from a simple program using a simple pseudo random number generator (**rand**()). We can simply send our friend the program along with the pseudo random number generator, which together may contain less than 1 Kbytes of data. Our friend can then use the program to reproduce all the 80 million bits of the file "randnums". Therefore, the information that the file "randnums" contains is actually much less than 80 million bits. Does this mean that the definition of entropy rate given in (2.13) or (2.14) is inconsistent with our experience of information of data? The main reason for the inconsistency is that we have only considered a finite number of symbols in the compression process. The utility program **gzip** is based on the Ziv-Lempel algorithm which uses a finite look-ahead buffer for searching a string that matches the string under consideration. Because of the limited buffer size, all characters appear random to the encoder and the text cannot be compressed. In practice, any pseudo random number generator (PRNG) has a a finite period, which means that the sequence repeats itself after a certain number. Within a period, the generated numbers appear random. In general, the period of a PRNG is very long for it to have practical use. If we have generated a sequence that is much longer than the period and our compression program has used a look-ahead buffer larger than the period, the huge file will be compressed to a very small one, which is consistent with our intuition that the sequence actually contains very little information. Therefore, in some situations entropy rate may not be a good estimate of the **average** information of data as it requires an infinite number of bits of data in the measurement to give the correct result.

A more fundamental approach to estimate the information of data or text is to consider the Kolmogorov complexity, also known as algorithmic entropy, which is a measure of the computational resources required to generate the text. For example, consider the following string of length 96:

go!go!go!go!go!go!go!go!go!go!go!go!go!go!go!go!
go!go!go!go!go!go!go!go!go!go!go!go!go!go!go!go!

The string can be described by a short English Language description like, "go! 32 times" which consists of only 12 characters.

More formally, the complexity of a string is the length of the string's shortest description in some fixed universal description language. One can show that the Kolmogorov complexity of a string cannot be too much larger than the length of the string itself. Suppose **P** is a program that outputs a string **x**, then **P** is a description of **x**. The length of the program, $\mathbf{l}(\mathbf{P})$ is essentially the complexity of the string **x**. We can now make a formal definition of Kolmogorov complexity.

Kolmogorov (algorithmic) complexity $K_u(x)$ of a string x with respect to a universal computer u is defined as

$$K_u(x) = \min_{P:u(P)=x} l(P) \tag{2.15}$$

the minimum length over all programs P that print x and halt.

One can show that if K_1 and K_2 are the complexity functions relative to description languages L_1 and L_2, then there exists a constant c, which depends only on languages L_1 and L_2, such that

$$|K_1(x) - K_2(x)| \leq c, \quad for\ all\ strings\ x \tag{2.16}$$

This implies that the effect of choosing a description language on K is bounded.

One can prove that Kolmogorov complexity is the minimum number of bits into which a string can be compressed without losing information. In other words, it is the information the string contains. Therefore, a string is incompressible if its length is equal to its Kolmogorov complexity. One can also show that when the string is sufficiently long, the entropy of the string converges to the Kolmogorov complexity. Therefore, the information contained in a fractal image or a set of pseudo random numbers is very small as the data can be generated by a simple program using an algorithm.

2.5 Data Reversibility

By now we know that data are different from information. We can use different amount of data to represent the same piece of information. In practice, a set of data may have a lot of redundancy and data compression is achieved by getting rid of the redundancy. There are two kinds of data compression, lossless and lossy. In lossless compression, no information is lost and the exact original data set can be recovered. In lossy compression, information is lost and the original data set cannot be recovered. In other words, in lossy compression, we throw away some information in order to achieve a higher compression ratio. *What kind of information do we want to throw away?* Naturally, we want to throw away the irrelevant information and retain the important information. Given a set of data, *how do we decide on which portion of data is more important than others*? In fact, separating the relevant and irrelevant information is the state of the art of lossy data compression. The following discussion gives a brief idea how this can be done.

Suppose we want to know about the age of the people in a country of a million. We would not want to remember the age of every individual. If we just want to remember one value concerning age, we would most likely remember the average age of the population; we don't care about the age of the president or the age of any 'great leader' of the country. The average value gives us a

brief idea about the population of the country. Indeed, the average value of a data set is usually the most crucial value. On the other hand, we cannot reconstruct the whole set from its average value. In other words, when we average the values of a set of data, we lose information; the process is irreversible. This is also true in the physical world. For instance, consider the case that we put a drop of red ink in a glass of water. We will see that the ink spreads over and eventually the whole glass of water becomes red. However, no matter how long we continue to observe the glass of red water, we will never see the process reverses itself and the ink pigment forms a drop again. Microscopically, the process is reversible; when an ink pigment molecule interacts with a water molecule, there is nothing that forbids them to go in one direction or the other. From the point of view of information theory, the process is reversible because we have recorded the information of every single molecule. However, macroscopically, the entropy law of physics forbids the process to be reversible. This is because when we observe the glass of water macroscopically, we observe an unaccountable number of molecules simultaneously; we are observing the average behavior of the molecules. Because of the averaging effect, information is lost in the process and thus it is irreversible.