Fundamentals of Bar Code Information Theory

Theo Pavlidis,* Jerome Swartz, and Ynjiun P. Wang
Symbol Technologies

Introduced about 20 years ago, bar codes have spread from supermarkets to department stores, the factory floor, the military, the health industry, the insurance industry, and more. The bar code industry uses the term *symbology* to denote each particular bar code scheme, while the term *symbol* refers to the bar code label itself. The last few years have seen the introduction of many new symbologies, and an ongoing debate compares them with each other and with schemes for encoding printed information on a substrate. Therefore, we need to look into information and coding theory to make meaningful comparisons.

Early codes, in particular UPC (Universal Product Code), resulted from careful studies,[1] but such studies faced the technological constraints of 20 years ago and responded to the expected applications of that time. A more recent study[2] focused only on some aspects of the encoding. The continuing drop in the price of computing hardware makes feasible in the near future the use of powerful processors that would have been deemed prohibitively expensive to use in the past. Thus, we can justify examining encoding and decoding schemes that looked impractical 20 years ago.

Recent years have also seen demands to increase the density of information packing to create a "portable data file" as opposed to the "license plate file" of conventional bar code symbols. You can see the distinction between the two forms in the following example: The bar code of a supermarket item consists of 11 digits, which represent an identifying number but not a description of the product. For example, the price look-up must be accessed in a database keyed to the number in the bar code. An alternative would be to use a much longer bar code (or other encoding) to store all the relevant information, such as price, name of the product, manufacturer, weight, inventory data, expiration date, etc. That would constitute a "portable data file," because the information could be retrieved without access to a database. While a price look-up file (PLU) is necessary and convenient in a retail environment, this may not be the case at a distribution center receiving from and shipping to remote warehouses or overseas depots.

To compare encoding and decoding schemes requires us to first look into information and coding theory. This article discusses problems and possible solutions in encoding information.

The basic problem we face is that of encoding information on some medium using printing technology. Any such encoding has the following conflicting requirements:

- We want the code to have a high density of information.
- We want to be able to read the code reliably.
- We want to minimize the cost of the printing process.
- We want to minimize the cost of the reading equipment.

It is essential to look at all the factors to reach practical and meaningful conclusions. Bar codes have been criticized often for having a low density of coded information, but some of their "inefficiencies" were introduced deliberately to facilitate their robust reading by a wide variety of means.

An important distinction exists between the method of "painting" bits on paper (*channel encoding*) and that of encoding information into bits (*source encoding*). The distinction is well understood in cod-

* Pavlidis is with the Department of Computer Science, State University of New York at Stony Brook, and serves as a scientific advisor to Symbol Technologies.

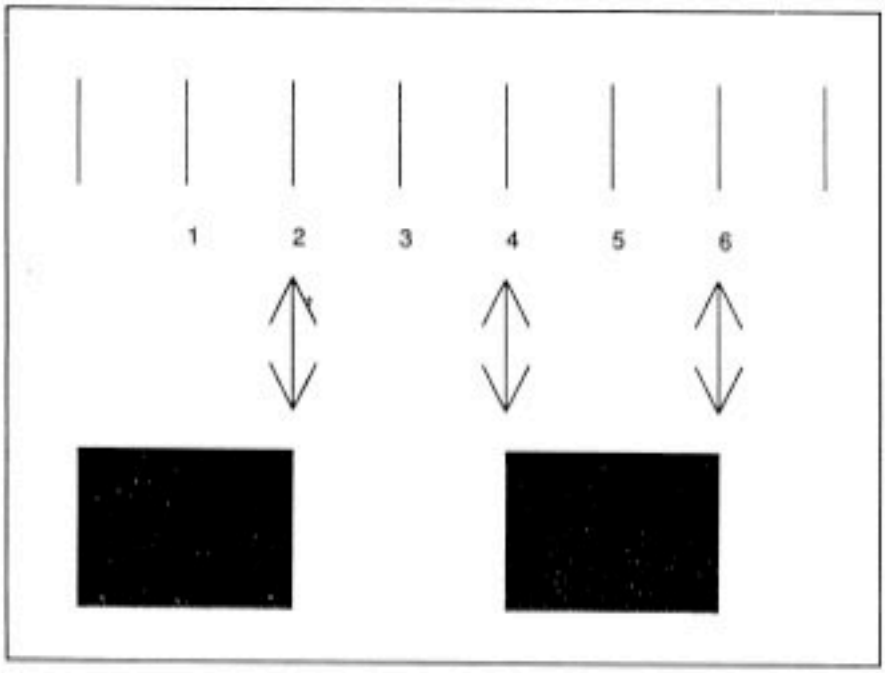


Figure 2. Derivation of the UPC code words. The seven modules (top line) have six boundaries (second line). A partition of the modules into two bars and two spaces requires three internal boundaries that can be placed in each of six locations.

because optical and magnetic recording systems are *closed systems* where the distance between the scanner and the symbol as well as the size of the symbol are fixed and known. Neither is known for bar codes, which are classified as *open systems*.

Besides self-clocking we need some means to detect the direction of the scan. Practical requirements may demand that bar codes be scanned in either direction. (Many scanning devices, including those used in retailing, have repetitive scans in alternating directions.) There are two types of solutions to this problem. One is to have a unique start/stop code word on each symbol. The other is to use only one of two code words if they are mirror images of each other. For example, three modules of a delta code with only one bar and one space yield the combinations

The last two are the mirror images of the first two and therefore cannot be distinguished if you do not know the scanning direction. This results in taking only half of all possible code words. UPC uses only code words whose mirror image is not in the code, while Code 39 uses a unique start/stop code word. We next review both of these codes in detail.

Some bar code types

Universal Product Code. UPC is the symbology used in American supermarkets for more than 15 years. What we describe here is version A, where each symbol encodes 12 digits. However, all versions use the same basic structure for the code words. Each UPC code word consists of two bars and two spaces with a total width of seven modules. The number of possible code words can be computed as follows: Two bars and two spaces require three dividers in one of six positions (between modules) as shown in Figure 2. Therefore, the total number of partitions will be the same as the number of ways to choose three objects out of six, or 20. You can then construct 40 code words, depending on whether the first element is a bar or a space. However, you cannot include code words that are mirror images of each other because of the need for bidirectional scanning. This leaves you with 20 possible code words.

While symbols must be readable by scanning in either direction, we also want to detect the direction of scanning afterwards. UPC achieves that by assigning two code words for each digit: those used on the right part of the symbol start with a bar and those used on the left part start with a space. The two versions can be distinguished because the ones starting with a bar have an even number of bar modules (even parity) and those starting with a space have an odd number of bar modules (odd parity). Table 1 shows this arrangement. (We will explain the last two columns of the table later.) You can see that each digit is assigned a single width combination. For example, 0 is 3,2,1,1. UPC specifies a symbol format as shown in Figure 3.

The symbol contains the following groups of code words:

- a left guard pattern 101
- six digits of odd parity:
  - — one digit denoting the industry type (such as 0 for grocery, 3 for pharmaceutical, etc.)
  - — five digits with the manufacturer's code
- a center guard pattern 01010 of five modules
- six digits of even parity:
  - — five digits with the item code
  - — one check digit
- a right guard pattern 101

Laser scanners in the supermarkets use essentially orthogonal patterns. The height of each half of the symbol is expected to exceed its width so that one of the two beams is guaranteed to scan a symbol half. (It is a property of a block with height greater than width that an × pattern is guaranteed to read it by at least one line of the × in any orientation of the symbol.) The symbol can be assembled afterwards because each part is identified by the parity pattern. UPC allows "scanning by halfs" because the left and the right parts of the code are inherently identifiable from the local properties of each code word.

One major source of distortions of bar codes is uniform ink spread (see the section "Noise and distortions" for more on this topic), which generally causes bars to be comparatively wider than spaces. For this reason bar code readers measure not the width of each bar or space but the edge-to-edge difference of successive bars. Figure 4 shows that the sum of the widths of a bar/space pair is not affected by ink spread.

The last column of Table 1 shows the pair of edge-to-edge distances (called the *t*-distances) for each UPC code word. The pair characterizes the character uniquely except for the pairs 1 and 7 and 2 and 8. For those, an additional calculation must be performed, typically a comparison of th

Table 9. Meaning of the minimum distance.

Minimum Distance	Meaning
1	Uniqueness
2	Single-error detection
3	Single-error correction (or double-error detection)
4	Single-error correction plus double-error detection (or triple-error detection)
5	Double-error correction

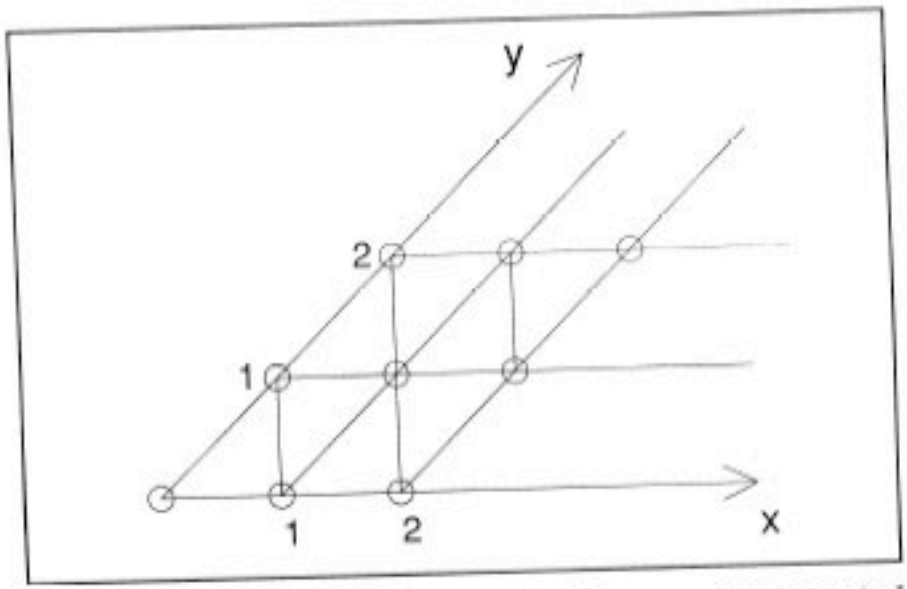


Figure 8. Topology of the diagonal distance. Each axis corresponds to the length of an interval, and points of the plane correspond to pairs of successive intervals. A single edge shift causes a transition where the sum of x and y remains constant. The plot has been distorted so the geometrical distance corresponds to the string distance. Motions along perpendicular lines represent edge shifts.

formations caused by noise and distortions during the reading of bar codes. x and y denote the widths of two successive elements in one string, and τ is a constant greater than 1.

Given two strings X and Y, we can compute their distance on the basis of Table 8. The new distance is given by the minimum sum of weights of transitions in the table, which are needed to transform X to Y or vice versa. The weights in Table 8 are based on the Laplacian channel model.

We use the term *diagonal distance* to describe this new distance because of the geometrical interpretation shown in Figure 8. There, the x and y axes represent successive elements (bars or spaces) of a string. Points along a vertical diagonal are equally close to their neighbors on the x and y axes.

We can define the minimum distance $d_{min}(C)$ of a code C as

$$d_{min}(C) = \min[d(v,w) : v,w \text{ in } C, v \neq w]$$

where $d(v,w)$ is the diagonal distance between v and w. The error-detecting and error-correcting capability of C is determined by the $d_{min}(C)$. Table 9 describes the meaning of the minimum distance for any encoding scheme.

These concepts allow us to determine the error-detecting and error-correcting capability of any code and in particular UPC and Code 39. A look at the width pattern for UPC (see that section above) suggests that the minimum pairwise diagonal distance is 2. For example, the code for the digit 7 is 1,3,1,2. A single edge shift may yield 1,2,2,2, which is not the code for any digit. One more shift might produce 2,1,2,2, which is the code for the digit 2. In other words, it requires at least two edge shifts or one shift by two modules to convert one code word into another. Therefore, UPC has single-error detection capability, meaning a single edge shifts results in an illegal code. The previous discussion (in the section on Code 39) also showed that the d_{min} for Code 39 equals 2, therefore Code 39 has single-error detection capability, commonly referred to as *self-checking*.

Note that we do not consider here the error control capability of the check-sum digit, since we focus on the encoding of bits or bytes on the printing medium rather than the overall encoding of information.

Bar code design: Specifics

Use of the diagonal distance has enabled us to prepare a set of tables to use for specific designs. If we insist on a minimum distance d between code words, then for each code word we select we must eliminate all code words lying in a sphere of radius $(d-1)/2$.

Table 10 provides an approximate estimate of how many code words exist at a given radius away from another code word. For example, if we wish to have distance 5, then we must discard all codes at radius 2 or smaller. For a (7,2) code, Table 10 yields 1+3+5.8, or about 10. Since a (7,2) code has only 20 code words, this would allow the encoding of only two distinct symbols — not of practical interest. On the other hand, for a (15,4) code and distance 5, Table 10 yields 1+7+40.8, or about 49. Table 2 yields a total of 3,442 code words, so we can encode 3,442/49 or 70 distinct symbols. This would cover more symbols than most current codes provide and also allow double-error correction (see Table 9). UPC, a (7,2) code, has a minimum distance 2 that translates into a radius equal to 0.5. This lies outside the table, but we extrapolate to 2 (that is, we can use only half the available code words).

We proceed now to provide estimates of the density for given codes. Tables 11 to 13 list the ratio of module width X over T and

Table 3. Information content of some real codes.

Code Name	Closest Theoretical Model	Number of Symbols Used	Total Width (in modules)	H
UPC	delta(7,2)	10	7	0.474
Code 128	delta(11,3)	106	11	0.617
Code 93	delta(9,3)	48	9	0.621
Code 39	width(9,3)	44	13.5*	0.404
Codabar	width(7,2)	16	10*	0.400

* Assuming a 2.5:1 wide over narrow ratio.

locally but must be determined from the start-and-stop code word. If the first code word is decoded as P, then we conclude that we are looking at the last code word of a symbol. The major practical implication of this design is that Code 39 does not allow local detection of the scan direction and cannot be decoded, even partially, without at least one of the start-and-stop code words.

Other bar codes. While UPC and Code 39 are two of the most widely used codes, quite a few others have been implemented. One of the earliest bar codes (circa 1968) is Code 2 of 5, which is a width code using only one of the colors with the other serving only as a delimiter. It has been used for airline baggage and cargo handling, among other applications. Interleaved Code 2 of 5 is similar, using both colors with each denoting a separate character.

Another width code, Codabar, consists of four bars and three spaces. One of the bars and one of the spaces is wide, yielding a total of 12 code words. Four additional code words are obtained by using three wide bars, and four more code words using one wide bar and two wide spaces. Codabar has been used by libraries and at blood banks.

Besides UPC, (n,k) codes in wide use include the following:

- Code 93, introduced in 1982, has nine modules and three pairs of bars and spaces, plus a 47-character set and a stop code word. The basic set encodes the 10 digits, 26 uppercase characters, seven punctuation marks, and four shift code words to extend the meaning of the other characters.
- Code 128, introduced in 1981, has 11 modules and three pairs of bars and spaces. This code has 105 distinct characters plus a stop code word. It is probably the first code with a clear distinction between channel encoding and source encoding. Three of the code words are used at the start of the symbol to denote one of three types of source encoding. Two of the types involve a mixture of alphanumerics, while the third encodes the numbers between 0 and 105.

A recent development in bar coding is the introduction of stacked bar codes, also called two-dimensional codes. Such schemes use a one-dimensional code in a series of rows as the basic encoder. Code 49, introduced in 1987, is based on a new (n,k) code with 16 modules and four pairs of bars and spaces. Code 16K was introduced in 1988 based on an extension of Code 128. Identcode and PDF417 were introduced in 1989. The former is based on Code 39 and the latter on a new (n,k) code with 17 modules and four pairs of bars and spaces. Such stacked bar codes present a significant advance and additional challenges. (Because of space limitations, we will discuss these in a future article.)

For more details on bar code specifications and for a complete review of all the existing symbologies, refer to the bar-coding literature.[8-10]

Information content of bar codes

If a code has n modules and can generate $S(n)$ code words, then we define its information content, per module, as

$$H(n) = \frac{1}{n}\log_2 S(n) \tag{1}$$

If we need a width W to print the n modules on the substrate, then we define the density of the code (in bits per unit length) as

$$D(n) = \frac{1}{W}\log_2 S(n) = \frac{1}{X}H(n) \tag{2}$$

where X denotes the module width. An unrestricted delta code can encode up to 2^n code words so that its information density is

$$D_\delta(n) = \frac{1}{W}\log_2(2^n) = \frac{n}{W} = \frac{1}{X} \tag{3}$$

In a width code, with the ratio of wide to narrow equal to two, the number of effective modules will be $n-j$, if j is the number of bits set to 1. Then the number of possible code words is

$$S_w(n) = \sum_{j=0}^{n/2}\left[\begin{matrix} n-j \\ j \end{matrix}\right] \tag{4}$$

This is equal to the nth Fibonacci number, which is given by

$$f_n = \frac{1}{\sqrt{5}}\left[\left[\frac{1+\sqrt{5}}{2}\right]^{n+1} - \left[\frac{1-\sqrt{5}}{2}\right]^{n+1}\right] \tag{5}$$

This equation is a special case of a well-known result by Shannon in 1948. The first fraction in Equation 5 equals 1.618 and the second equals –0.618, so that as n increases the second term goes to zero. For values of n greater than 5, the following provides a very good approximation:

$$S_w(n) \approx 0.4472 \cdot (1.618)^{n+1} \tag{6}$$

Therefore, the density of an unrestricted width code is

$$D_w(n) = \frac{1}{W}\log_2 S_w(n) \approx \frac{1}{X}\log_2 1.618 + \frac{\log_2(1.618 \cdot 0.4472)}{W}$$

or

$$D_w(n) \approx \frac{0.694}{X} - \frac{0.467}{W} \tag{7}$$

Thus, width codes have a density of approximately 70 percent that of delta codes with the same module width.

The above densities are theoretical maxima. The need for self-clocking, redundancy for error detection, and other factors make it necessary to use lower densities. Table 3 lists the information content and some important parameters for some of the existing codes. We need to look separately at each factor that affects the information content. We start with the effects of self-clocking and proceed with the calculation of the capacity of (n,k) delta codes and the capacity of (m,w) width codes. The latter symbol denotes a code of m elements, w of them wide.

(n,k) codes. The maximum number of distinct code words of an (n,k) code is well known[1]:

Table 6. Information content of width (binary) codes with m elements, w of them wide.

(m,w)	S	H	D	Name of Related Code
5,2	10	0.415*	1	Interleaved 2 of 5
7,2	21	0.439	1	Codabar**
9,3	84	0.474	1	Code 39

* Counting both space- and bar-encoded code words.

** Actually, Codabar consists of two codes: (4,1) and (3,1) for a total of 12 code words plus four special code words obtained from a (4,3) code and another four from a (4,2) code.

efficiency is computed as

$$\frac{\log_2(S-L)}{\log_2 S}$$

the ratio of remaining bits over the bits for (n,k).

The small drop in information content suggests that codes with large n increase the information content without increasing the difficulty of decoding. Such codes may be represented by the longest element expressed in module units. Code (16,4,6) is a code with maximum element length 5. Code (11,3) has the same maximum element length, but its information content is only 0.725, which is less than that for code (16,4,6). Note that Code 49 uses a (16,4,6) code as its basis (with efficiency of 99.47 percent) and that Code 128 is really an (11,3,4) code with efficiency of 97.21 percent.

(m,w) width codes. The capacity of (m,w) width codes equals the number obtained by selecting w objects out of m or

$$S_w(m,w) = \binom{m}{w} = \frac{m(m-1)\ldots(m-w+1)}{w(w-1)\ldots 3\cdot 2} \qquad (16)$$

Their length equals $m+wr$ where r is the ratio of wide over narrow minus one. Therefore, the information content is given by

$$H_w(m,w) = \frac{1}{m+w\cdot r}\log_2 S \qquad (17)$$

Table 6 lists the capacity of some (m,w) codes assuming the width of wide elements is 2.5 times the width of narrow elements. We see that the information capacity of a (9,3) width code is 0.474, while the capacity of Code 39 (see Table 3) is 0.404. The difference, 0.070, is due to the error detection feature of the code.

Alternatives to bar codes

A natural question is whether anyone could have come up with a better way to label items than bar codes. We can abstract the problem and state it as follows:

- Given a line segment, devise a scheme of subdivision into subsegments having one of two colors so that the recorded information is maximized subject to constraints on the probability of undetectable reading errors.

The restriction to two colors is an important industrial constraint, and any departure from it will involve significant financial commitments by the industry. If we ignore the constraint of self-clocking, then we have a simple theoretical solution. Let L be the length of the segment and d the shortest length we can print. (For an ordinary laser printer, d is about 3 mils.) Then divide L into $n = L/d$ intervals and use a delta code that yields the maximum density according to Equation 3.

However, such an increase in density comes only at a significant price. We must give up the self-clocking requirement and, consequently, many of the simple techniques for reading bar codes. We can decode a bar code without self-clocking in one of three ways:

(1) By capturing a complete symbol and then using image processing techniques to analyze it. If we know the symbol length, we do not need any scaling information.

(2) By printing "timing" marks of a different color interleaved with the code markings.

(3) By printing "timing" marks of the same color but in an adjacent zone of the substrate.

The first solution requires more expensive equipment both for capturing the data and for processing them. (Current bar codes can be read by wand systems selling for about $100 per unit.) The second solution requires more expensive equipment for capturing the data but not for processing them. However, it requires a more expensive printing process. The third solution imposes a small increase in printing costs but requires more area, thus reducing the density of the encoding. It also requires an imaging type scanner, eliminating simple devices such as wands.

It is doubtful whether such expenses can be justified, because Table 4 shows that (n,k) codes with large n come close to 80 percent of the theoretical maximum. Therefore, the reduction in density because of self-clocking is not that serious.

Even if you were willing to pay the high price required for a small increase in density, it is still unclear whether you could actually achieve a net gain in information density. You can see the reasons by looking at the nature of the noise and distortions affecting the reading of information printed on a substrate. Most of the contamination of the information by noise occurs during the scanning of the printed medium, because you must deal with light reflected from a surface that might be poorly printed or dirty, experience interference with ambient light or other printing, and so forth. Noise during the transmission of the data from the scanning device to a cash register or a computer terminal is much lower. We will show in the next section that the scanning noise affects mostly the edges of segments and has minimal effects in the middle of bars or spaces.

If we do not wish to increase the total number of edges, then self-clocking eliminates very few useful combinations. For example, the unrestricted delta code for seven modules has 128 possible code words. The (7,2) code has 40 (if we ignore for the moment the bidirectionality requirement). Only 12 code words have one edge; two have none. The other 74 code words have more edges. Since the two code words with no edges could be confused with the background, we will gain only 12 code words if we give up self-clocking and do not increase the sensitivity to noise.

Electronic communications outperform printed codes for two reasons: the availability of two values (+ or −) with respect to

ing theory: the electronics of representing a zero or a one are dealt with separately from how English text, for example, is encoded into bits. The distinction is often overlooked in the discussion of bar codes because most of the early codes were defined as mappings between alphanumeric characters and "painted" paper.

In this article we review the overall principles of bar coding and discuss in detail two bar codes in wide use, UPC and Code 39, plus briefly outline some other bar codes and alternatives to bar coding. We touch on the information content of bar codes and deal with the noise and distortions that affect the reading of bar codes. Then we outline the process for bar code design using coding theory and introduce a distance between bar codes to allow the use of error detection and error correction techniques. Finally, we present design tables and a specific example showing how to increase the information density encoded in bar codes by using error detection and error correction.

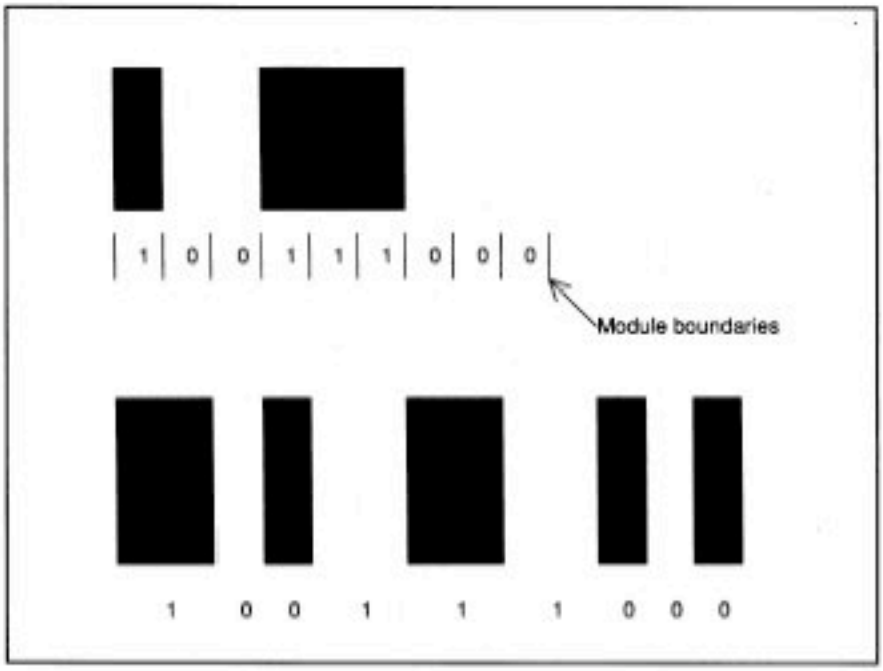


Figure 1. Encoding of the binary string 100111000 by a delta code (top) and width code (bottom).

The state of the art

Bar codes encode information along one dimension with intervals of alternating diffuse reflectivity, usually black and white color. The intervals are actually stored as rectangles whose vertical height carries no information but facilitates the scanning process. The term *bars* denotes the rectangles with the foreground color while the term *spaces* denotes the intervals with the background color between the bars. The bars have the darker color, and special care is taken to ensure that.

You might think this doesn't work on soda cans, where the bars have a metallic color that reflects more light than the background. However, the reflection from the bars is specular and directed away from the sensor. The spaces have a dull color with diffuse reflection, part of which is always captured by the sensor. Similarly, on transparent bottles the bars are not colored, so light goes through them rather than reflecting to the sensor.

There are two fundamental ways of encoding information in such a one-dimensional medium. In one we subdivide the available interval into *modules* and assign 1's and 0's to each module. Modules with 1's are painted and form the bars, while modules with 0's correspond to spaces. Obviously, a single bar or space may contain many modules. Such schemes are called *delta* codes in communication theory. A more descriptive name would have been color codes, but we prefer the standard term to limit the introduction of new terminology.

In another scheme we assign each bit to a bar or space and make that element wide if the bit is one and narrow if the bit is 0. We refer to such schemes as *width* codes, although the common name used in the bar-coding industry is *binary* codes, because only two widths are used. We prefer the term "width code" because it describes more accurately the method used and because the word binary has too broad a meaning in computer literature.

Figure 1 shows two encodings of the binary string 100111000. In both cases the minimum printed width is the same. The delta code requires nine such widths (the number of bits), while the width code requires 13 such widths if a wide element is twice as wide as a narrow element.

It was recognized from the beginning that, for bar codes to gain wide acceptance, they should be allowed to be printed in different sizes and read at a range of nonconstant distances (for example, by hand-held laser scanners) and by devices with variable scanning speed. (The speed is extremely and unpredictably variable in hand-held wands. However, even laser scanners exhibit some variation in speed.) These requirements impose the constraint that such printed codes be *self-clocking*, which means that both the number of modules and the number of bars and spaces per code word must be fixed. Thus, delta codes (which inherently have a fixed number of modules) are required to have a fixed number of bars and spaces. A code with n modules and k pairs of bars and spaces is called an (n,k) code. Width codes are required to have not only a fixed number of bars and spaces but also a fixed number of wide elements to achieve a fixed width. Among the most popular bar codes, the UPC is a (7,2) delta code (seven modules, two bars and two spaces), while Code 39 is a width code with nine elements (five bars and four spaces), three of which are wide.

The self-clocking constraint for bar codes is more stringent than that for optical and magnetic recording,[3-6] which share many of the features of bar codes. In such systems information is encoded under the run-length-limited (RLL) constraint. This constraint is specified by a pair of constants (d,k) where $0 \le d < k$, which means two consecutive 1's are separated by at least d, but no more than k, 0's. The constant d is used to control the intersymbol interference effects. The constant k is used to provide self-clocking through a phase-locked loop (PLL) with windowing.[7] This weaker constraint is possible

Table 10. Average number of code words at a given radius for various (n,k) codes.

Radius	(7,2)	(11,3)	(15,4)	(19,5)
1	3.0	5.0	7.0	9.0
2	5.80	19.3	40.8	70.4
3	4.80	35.6	122	297
4	3.40	51.2	275	931
5	1.30	48.7	430	2,086
6	0.700	42.0	562	3,777
7		25.1	573	5,473
8		15.9	522	6,831
9		5.57	382	7,191
10		2.62	266	6,792
11			141	5,520
12			76.0	4,143
13			24.2	2,643
14			9.47	1,594
15				763
16				358

Table 11. Design parameters for (7,2) codes.

d_{min}	$E_s=10^{-3}$ X/T	D	$E_s=10^{-6}$ X/T	D	$E_s=10^{-9}$ X/T	D
1	16.012	0.039	29.828	0.020	43.643	0.014
2	10.675	**0.040**	19.885	**0.021**	29.095	**0.014**
3	8.666	0.038	15.573	0.022	22.481	0.015
4	6.932	0.032	12.459	0.018	17.985	0.012
5	5.651	0.026	10.256	0.014	14.861	0.010

Table 12. Design parameters for (11,3) codes.

d_{min}	$E_s=10^{-3}$ X/T	D	$E_s=10^{-6}$ X/T	D	$E_s=10^{-9}$ X/T	D
1	17.034	0.042	30.849	0.023	44.665	0.016
2	11.356	0.049	20.566	0.027	29.776	0.018
3	9.869	**0.050**	16.777	**0.029**	23.685	**0.021**
4	7.896	0.046	13.421	0.027	18.948	0.019
5	6.986	0.043	11.591	0.025	16.196	0.019
6	5.988	0.038	9.935	0.023	13.882	0.017
7	5.422	0.034	8.875	0.020	12.330	0.015

Table 13. Design parameters for (15,4) codes.

d_{min}	$E_s=10^{-3}$ X/T	D	$E_s=10^{-6}$ X/T	D	$E_s=10^{-9}$ X/T	D
1	17.707	0.044	31.522	0.024	45.338	0.017
2	11.805	0.054	21.015	0.030	30.225	0.021
3	10.618	**0.054**	17.525	**0.033**	24.433	0.023
4	8.494	0.054	14.020	0.033	19.546	0.0236
5	7.810	0.052	12.415	0.032	17.020	**0.0240**
6	6.695	0.049	10.641	0.031	14.589	0.0227
7	6.261	0.046	9.715	0.029	13.169	0.021
8	5.566	0.041	8.635	0.026	11.706	0.020
9	5.189	0.038	7.952	0.024	10.715	0.018

the corresponding density D for a set of (n,k) codes and minimum distances d_{min}. The maximum density value is highlighted in bold. (Complete design tables appear in Wang.[16]) We illustrate their use with an example.

Example. We need 44 code words. We select $d_{min} = 3$ so that we can have single-error correction (according to Table 9). Then the radius should be 1. We start with an (11,3) code. According to Table 10, the sphere volume is 1 + 5 = 6, while the total number of code words, according to Table 4, is 252. Therefore, the number of available code words is 252/6 = 42. This is slightly less than the desired 44, but we keep it. The values of Table 10 are approximate averages, and we should be able to select 44 code words that will allow us error correction.

The density for this code is found from Table 12 to be maximum for $d_{min} = 3$. For $E_s = 10^{-6}$ we have $X/T = 16.78$ and $D = 0.029$ bits per length T. For $T = 0.6$ mil this yields 48 bits per inch and module size $X = 16.77 \times 0.6$ or 10 mils.

For the same goal we can try another combination: $d_{min} = 5$ (double-error correction) and a (15,4) code. We already saw that this yields 70 code words, so we have more than enough. From Table 13 we find that for $E_s = 10^{-6}$ the ratio $X/T = 12.4$ and the density is 0.032 bits per length T or 55 bits per inch with module size equal to 7.5 mils. This density is 14 percent higher than that obtained with a (11,3) code and has been achieved without an increase in the error rate. It is an engineering question whether this increase in density is justified by the increase in the decoding cost caused by going from 11 modules to 15.

Table 11 provides an interesting insight on UPC. The maximum density for a (7,2) code is achieved for $d_{min} = 2$. This is exactly the value used in UPC, so we might be justified in calling UPC an *optimal* density (7,2) code.

We have provided a method for applying the results of coding theory to bar codes. As new applications demand new types of bar codes, the results presented in this article make it possible to optimize the information density of the new codes under realistic constraints. The methodology is particularly useful for the design of two-dimensional or stacked bar codes. ■

Acknowledgments

We want to thank the many people who helped us with the article. In particular, Joseph Katz, Stephen Shellhammer, Boris Metlitsky, Rick Schuessler, Emanuel Marom, and Leonard Bergstein provided many useful comments on various drafts. Metlitsky also gave us the data used in Figure 5. The article was thoroughly rewritten on the basis of many constructive comments of the referees.

element widths themselves. For example, if the second element is wider than the third, we decide in favor of an even 8 rather than an even 2. Of course, for an 8 the second element is twice as long as the third, while the opposite holds true for a 2. Because of the ink spread and the preclassification provided by the *t*-distances, we use a more liberal rule.

Code 39 (three of nine). Code 39 is a width code with three wide elements out of a total of nine: five bars and four spaces between them. It has been the standard for the Department of Defense since 1980. The total number of code words that it can generate is the number of ways three items can be chosen out of nine, or 84. It has code words for the 10 digits, the 26 letters, and eight special symbols (hyphen, period, space, asterisk (*), $, /, +, and %) so that a total of only 44 code words are used. The asterisk is used only as the first and last code word of a symbol.

Table 2 shows some of the codes. A 1 means a wide element and a 0 means a narrow element. If an edge shift (the most common reading error) occurs, it will change both the bar and the space patterns. For example, 10001,0100 might become 11001,0000. The latter does not correspond to any Code 39 pattern.

The patterns for both the bars and the spaces have been chosen in such a way that changing a single bit in either of them results in an illegal code word. Spaces have only an odd number of wide elements and bars only an even number. This allows the immediate detection of single errors. The code is called *self-checking* for that reason. The number of bar patterns with two wide elements is 10 (the number of ways to choose two out of five), and there are four space patterns with one wide element. This yields 40 code words. Four additional code words are obtained by using the four patterns with three wide spaces and only narrow bars.

In contrast to UPC, Code 39 has no specifications for the arrangement of code words on the symbol except that the asterisk must be the first and last character and can appear only there. Called the special start-and-stop character, the asterisk makes it possible to determine the direction of scanning. (Of course, Code 39 and all other codes have printing specifications for the spacing of code words on a symbol.)

As Table 2 shows, Code 39 contains code words that are mirror images of each other, such as the pairs (P,*) and (K,U). Therefore, the direction cannot be decided

Table 1. Specification of Universal Product Code.

	Left (odd)	Right (even)	Width Pattern	*t*-Distances (odd)	(even)
0	0001101	1110010	3,2,1,1	4,3	5,3
1	0011001	1100110	2,2,2,1	3,4	4,4
2	0010011	1101100	2,1,2,2	4,3	3,3
3	0111101	1000010	1,4,1,1	2,5	5,5
4	0100011	1011100	1,1,3,2	3,4	2,4
5	0110001	1001110	1,2,3,1	4,5	3,5
6	0101111	1010000	1,1,1,4	5,2	2,2
7	0111011	1000100	1,3,1,2	3,4	4,4
8	0110111	1001000	1,2,1,3	4,3	3,3
9	0001011	1110100	3,1,1,2	3,2	4,2

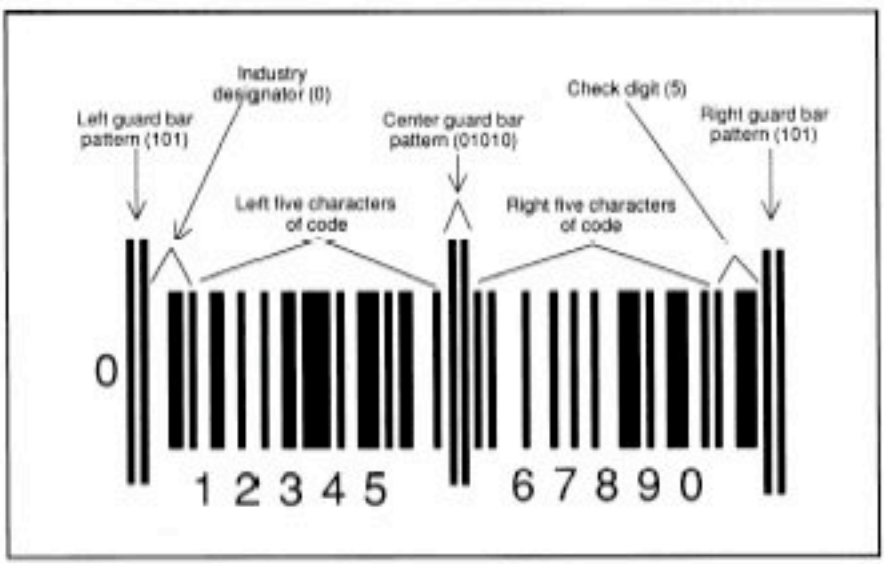


Figure 3. Illustration of the specifications of the UPC symbol. The readable characters are normally printed in OCR-B font. The diagram is not drawn to scale.

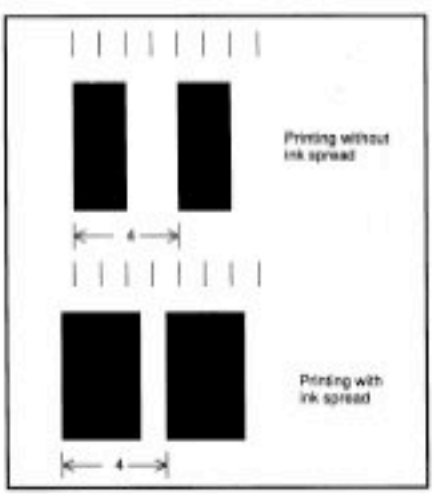


Figure 4. Illustration of the invariance of the edge-to-edge distance under ink spread.

Table 2. Part of the specification of Code 39.

	Bars	Spaces	Pattern
1	10001	0100	
2	01001	0100	
A	01001	0010	
K	10001	0001	
P	01100	0001	
U	10001	1000	
Z	01100	1000	
*	00110	1000	
$	00000	1110	

Table 7. Specific numbers of example in Figure 6.

	Bar	Space	Bar	Space	Bar	Space	Bar	Space	Bar
Ideal Widths	0.50	0.50	0.20	0.40	0.40	0.20	0.20	0.20	0.20
Measured Widths	0.50	0.50	0.30	0.31	0.40	0.29	0.20	0.20	0.20

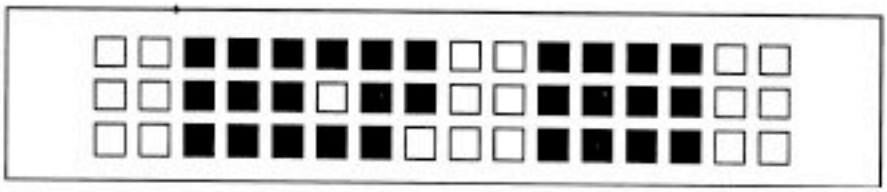


Figure 7. Possible errors in a binary string.

Notice in Figure 6 that the second, fourth, and fifth pulses have the same real widths, but the waveform of the second pulse differs from the other two. As a consequence of the convolution distortion, the second pulse appears wider than the other two. The specific numbers of the example shown in the figure appear in Table 7. Both sets of widths are expressed in the same arbitrary units. Notice that a 0.2 input width is mapped in one case to 0.3 and in the other two to 0.2. Because of the severe distortions of very narrow pulses, the minimum width of modules used in bar codes must exceed the width of the minimum detectable pulse to compensate for the distortion.

Using more sophisticated decoding techniques can help remedy the situation. For example, we could compare the width of a bar or a space not only to the widths of its immediate neighbors, but also to the widths of other code words. Such decoding might require more computing power than commonly used now, but it offers a way of tolerating higher code densities.

In addition to those already mentioned above, many other sources of noise exist. Some are multiplicative, such as the speckle noise inherent to coherent laser illumination and that due to the paper substrate. Others are additive, such as the ambient illumination (low frequency noise in the case of artificial light and shot noise in the case of sunlight, the electronics of the scanner, etc. See Barkan and Sklar[12] and Barkan and Swartz[13] for more on these topics.). While the effects of those noise sources do not necessarily concentrate at the edges, they show no preference for the interior, either. Thus, overall we have greater sensitivity at the edges.

Since most problems occur around the edges, we deal with signal-dependent distortions and noise. That makes inapplicable many of the analytical techniques used in electronic communications, where the noise is usually signal independent. Consider, for example, a message (stream of bits) of the form shown in the top row of Figure 7. A white square stands for a 0 and a black square for 1. In electronic communications a 0 can be mapped into a negative voltage pulse and a 1 into a positive voltage pulse. The nature of noise is such that it is equally likely to misread a pulse at the ends or the middle of a run of similar polarity pulses. Thus, the distortions shown in the second and third rows are equally likely.

We can encode the same sequence on paper by painting "modules" black for 1's and white for 0's. The result will be a white band of two modules, a black of six, a white of two, etc. It is rather unlikely that a module in the middle of the blank band will be read as a white, but it is much more likely that one in the edges will. This problem exists in other technologies as well, particularly optical and magnetic recording.[3-6] There, most of the noise also occurs at the edges, giving rise to intersymbol interference.

Bar code design: Preliminaries

The design of a bar code involves the following major parameters:

- the number, N, of distinct symbols that must be encoded;
- the error rate, E_r, for rejections (when a code word is flagged as unreadable);
- the error rate, E_s, for substitutions or misdecodes (when one code word is read for another);
- the density of information D measured as bits per inch, and
- the number of modules n.

The number of zones k in an (n,k) code is closely related to n. For symmetric codes we have the relation $n=4k-1$. Usually N is given and we wish to minimize the two error rates, maximize the density, and keep n low. We wish to keep n low because the complexity of the decoding algorithm and the size of decoding tables increase with n. These parameters cannot all be optimized at the same time because they all depend on the module size X. We will discuss the possible trade-offs among them next.

The trade-off between rejection and substitution rates, already well known from statistical decision theory, can be seen intuitively from the following extreme cases: If we set the rejection rate to 100 percent, then we cannot have any substitution errors. Alternatively, if we do not mind how many substitution errors occur, we can have a zero rejection rate.

The rejection and substitution rates' relative sizes are controlled by a parameter in the decision step for a given overall decoding quality. Therefore, we will focus only on the substitution rate, assuming a zero rejection rate. If we can make that small during the design of the bar code, then we can make it even smaller by allowing some rejection errors. In bar code practice we usually work with a 10^{-5} rejection rate and a 10^{-6} substitution rate.

An error typically occurs when we misread the width of an element in terms of modules. For example, instead of, say, 3,2,1,1 (the code word for 0 in the UPC), we read 1,4,1,1 (the code word for 3 in the UPC). Clearly, the smaller the module size, the more likely it will have errors when reading the code, because a given amount of edge shift will constitute a higher percentage of the total width if X is small. Moreover, the number of possible values to be discriminated should affect the error. Unfortunately, there have been no systematic studies of these relationships.

The Laplacian channel model looks like a reasonable approximation for the probability of an edge shift equal to at least one module:

$$p(X) = e^{-\frac{X}{2T}} \tag{18}$$

where T is a normalizing constant. If a code

ground; and the availability of a clock in synchronous electronic communication channels. These are not available when we print on a substrate unless we incur the significant costs of a third color (both for printing and for reading). These problems are shared with other technologies, in particular optical and magnetic recording,[3-6] where the recording polarities are binary.

Sometimes we hear the following argument: Since part of our problems are caused by the need to measure length, why not limit ourselves to looking for the presence or absence of patterns without regard to their dimensions? However, in the absence of a clock, pure detection is virtually impossible. We might not have to measure the length of bars, but we must measure the length of spaces. The only way around the lack of a clock is to allow only contact scanning (or at a fixed distance) and require that the codes all have the same size. This is the case with optical and magnetic recording systems, as mentioned above in "The state of the art."

The only realistic solution for increasing the density of encoding in a rectangular area is to use the vertical dimension. This strategy has been implemented in the stacked bar codes discussed in the section "Other bar codes."

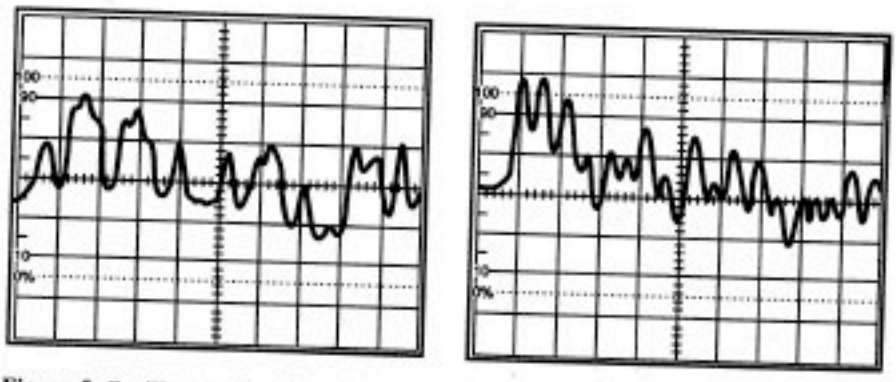


Figure 5. Oscilloscope tracings of bar codes with a laser scanner. The input for the left-hand image was produced with a low-quality dot matrix printer and the input for the right with a high-quality, high-density printer. In this case, the pulse amplitude is an indication of the pulse width (see also Figure 6).

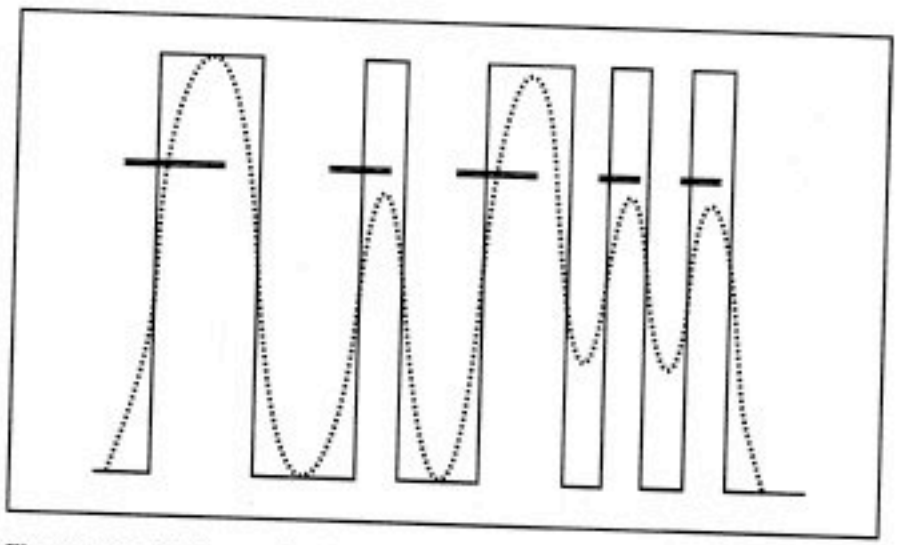


Figure 6. Results from a simulator of bar code scanning waveforms running on a Sun-3/160 workstation. The thin lines denote the ideal waveform from a bar code scan. The dotted lines represent the distorted waveform output by the sensor. The double lines denote the apparent pulse locations and widths obtained by an adaptive threshold method.

Noise and distortions

The design of bar codes has been influenced by the type of noise and distortions encountered during their scanning. The distinction between noise and distortions is subtle but important: If we scan a bar code twice, the effects of distortions will repeat but the effects of the noise most likely will change. We will show that the effects are most significant near the edges between the bars and the spaces.

A major source of distortions is ink spread. Bar codes are printed in two ways. On-demand printing demonstrates generally low quality and has significant ink spread. In-advance printing, although of higher quality, still is not entirely free from ink spread. A cursory inspection of boxes in the supermarket might suggest that bar codes are sharply printed. However, for decoding we need to discriminate differences in width of 0.01 inches — barely discernible by the human eye. We have already shown (in "Universal Product Code") how ink spread has heavily influenced the design of decoding algorithms (see also Savir and Laurer[1]). Clearly, this distortion only affects the edges.

Errors are also caused by the methods used for detecting bars and spaces. Because of uncertainties about the contrast and because of problems with ambient illumination, it is advisable to look for changes in the intensity of the reflected light rather than in the absolute level of the light (automatic gain controls notwithstanding). Indeed, most bar code systems use edge detection or highly adaptive thresholding techniques that look, in effect, at the slope of the waveform produced when a bar code is scanned. While the ideal signal would be a set of rectangular pulses, the real signal has a rounded form because of *convolution distortion*. This term refers to the averaging of the signal due to the finite size of the beam spot and the delays in the electronic circuits. Such rounding changes the slope and causes significant errors even in the absence of noise.

Figure 5 shows a photograph of oscilloscope traces from the scanning of actual bar codes. This is a far cry from the crisp, alternating black and white bars seen by the human eye.

The effects of distortion without noise appear in Figure 6, where we used a computer simulation. There, we assume without any loss of generality that the high levels correspond to bars and the low levels to spaces.

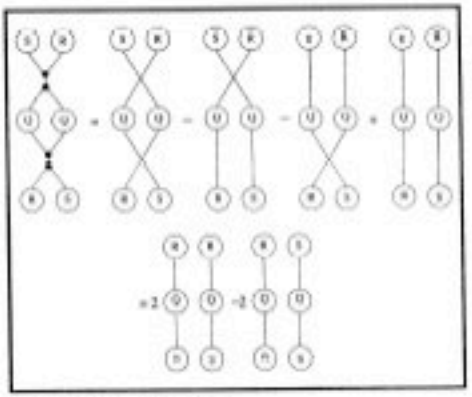


Figure 2. Line tangency proof.

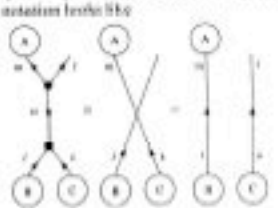
Since we are not concerned with scale factors, the only interesting feature of a determinant is whether it's zero or not. In other words, if the above diagram (which evaluates to a scalar) is zero, then the matrix is singular.

The other special symbol, delta, is just written as an arc (when necessary, I will label it with two labels):

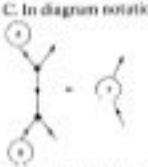
The diagram for epsilon-delta

Now we get to the interesting part: the epsilon-delta rule in diagram notation. Compare the following with Equation 2 to see how the index names work.

As an example, our proof of the vector identity of Equation 3 in diagram notation looks like



When I used this to calculate the perspective shadow, I was primarily interested in finding a transformation matrix to apply to arbitrary points C. In diagram notation, this transformation is just



Note that this diagram has one covariant and one contravariant free index, and thus, qualifies as a transformation matrix.

Now we can do the line tangency proof diagrammatically. To show the essence of the proof, I've left out the cluttering index names, a few global scale factors, and even the arrows. This diagram appears in Figure 2.

4D(3DH) epsilon-delta rule

The somewhat imposing 4D version of the epsilon-delta rule is

$$\varepsilon_{ijkl}\varepsilon^{mnop} = \delta^m_i\delta^n_j\delta^o_k + \delta^m_i\delta^n_k\delta^o_l + \delta^m_i\delta^n_l\delta^o_j - \delta^m_i\delta^n_j\delta^o_l - \delta^m_i\delta^n_k\delta^o_j - \delta^m_i\delta^n_l\delta^o_k \quad (4)$$

This is what you must use if you are solving problems in 3D homogeneous coordinates. Note that all the delta terms have the same subscripts, but they just have even or odd permutations of the (*mnop*) letters in their superscripts.

I haven't found this directly useful yet, but it lets us evaluate another useful identity. From the double summation by setting $j = l$ above. When you simplify, remember that a delta summed with itself is the trace of the identity—the constant 4. That is,

$$\delta^i_i = 4$$

When you work this out, it looks like our old friend, the 3D(2DH) rule:

$$\varepsilon_{ijk}\varepsilon^{lmno} = \delta^l_i\delta^m_j - \delta^m_i\delta^l_j$$

4D diagram notation

In diagram notation, the 4D epsilon would simply be a four-pronged node. We have to be careful, however. In 3D, a cyclic permutation of the indices of ε doesn't change its value. That is,

$$\varepsilon^{ijk} = \varepsilon^{jki} = \varepsilon^{kij}$$

In the diagram, we didn't need to explicitly identify which index was first. As long as you labeled them counterclockwise, you can start with any arc. In 4D, a cyclic permutation *does* change the sign, that is

$$\varepsilon^{ijkl} = -\varepsilon^{jkli}$$

Occupational mobility, Indianapolis, 1940: proportion of sons in each occupation according to father's occupation

Father's occupation	Son's occupation (percentage)											Total number of fathers
	Professional	Semi-professional	Proprietors, managers, officials	Clerks and salesmen	Skilled	Semi-skilled	Unskilled	Protective service	Personal service	Farming	All fathers	
Professional	28.3	6.3	7.6	27.9	15.4	9.5	2.5	0.8	1.5	0.2	100.0	(474)
Semiprofessional	15.8	19.3	3.5	17.5	23.7	12.3	2.6	1.8	3.5		100.0	(114)
Proprietors, managers, officials	7.7	3.4	17.6	30.6	14.3	19.8	2.5	1.6	2.1	0.5	100.1	(1203)
Clerks and salesmen	7.7	5.2	7.6	42.2	15.1	16.4	2.4	1.3	1.9	0.2	100.0	(1092)
Skilled	3.3	2.9	4.3	19.1	32.3	26.9	5.6	2.1	3.0	0.6	100.1	(2729)
Semiskilled	2.5	2.1	4.1	17.3	18.4	43.2	5.3	2.2	4.3	0.6	100.1	(1520)
Unskilled	2.4	1.5	2.8	13.1	15.4	30.0	28.6	2.4	3.6	0.3	100.1	(720)
Protective service	2.5	0.8	6.6	22.8	17.0	31.5	8.7	8.3	1.2	0.4	99.8	(241)
Personal service	4.9	4.3	5.5	17.1	22.6	29.9	3.7	1.8	10.4		100.2	(164)
Farming	3.8	1.6	5.9	15.2	23.1	28.7	8.9	3.6	5.1	4.2	100.1	(1635)
Sons of all fathers	5.5	3.1	6.6	22.1	21.9	27.1	6.9	2.3	3.4	1.1	100.0	—
Total number of sons	(548)	(307)	(656)	(2188)	(2163)	(2678)	(684)	(229)	(334)	(105)	(9892)	(9892)

(Rogoff, 1951, p. 410)

B7 The rate of generational mobility is currently about the same in all highly industrialized countries.

"There is relatively little difference in rates of social mobility, as measured by the shift across the manual-nonmanual line, in countries for which sample survey data exist"—the United States, Germany, France, Switzerland, Sweden, Japan, Denmark, Great Britain, and Italy (Lipset and Bendix, 1959, p. 72). From one generation to the next, about 25 per cent of the nonfarm population moves across the line between the working class and the middle class, both ways.

Here is the summary table, plus a sample of detailed figures on individual countries:

Comparative indices of upward and downward mobility (percentages)

Country	*Nonfarm populations*		
	Upward mobility (nonmanual sons of manual fathers)	*Downward mobility (manual sons of nonmanual fathers)*	*Total vertical mobility (nonfarm population mobile across the line between working and middle class)*
United States	33	26	30
Germany	29	32	31
Sweden	31	24	29
Japan	36	22	27
France	39	20	27
Switzerland	45	13	23

Country	*Populations with rural and urban occupations classified together*		
	High-prestige occupation sons of fathers in low-prestige occupations	*Low-prestige occupation sons of fathers in high-prestige occupations*	*Proportion mobile across high- and low-occupation prestige lines*
Denmark	22	44	31
Great Britain	20	49	29
Italy	8	34	16

(Lipset and Bendix, 1959, p. 25)

The testing of Negro intelligence, school children: verbal group tests

Author, date	Location	N	Subjects AGE	Subjects GRADE	Method of selection	Results			Comments of author
							I.Q. M c	I.Q. SD c	
Long, H. H. (1934)	Washington, D.C.	c-2103 c-1323 c-1258		1A 3A 5A	All 1A, 3A, and 5A c children tested in 1930.	1A 3A 5A	93.35 95.71 92.72	17.10 13.80 13.75	Decrease in I.Q. explained by increased inadequacy of community, home environment, and school activities for nurturing Negro child.
							I.Q. w	I.Q. c	
Charles, C. M. (1936)	St. Louis, Mo.	w-172 c-172	12–16 12–16		W and c boys selected in about equal number for each age-group from schools in different parts of city; schools sought in which social environment of w and c similar. Method of selection within a school's age-group not given.	*M* *SD* *Range*	98.31 12.25 60–135	88.60 11.00 55–114	Cannot determine whether superiority of w boys is due to inherent factors, to training, to environment, or to possible deficiencies in test used.
							I.Q.		
Robinson, M. L., and Mercer, M. (1947)	Washington, D.C.	c-955		3	Secured I.Q.'s of pupils in twelve public schools either in 1938–39 or in 1945–46; at least two schools from each section of city. All born in D.C., "for whom I.Q.'s could be obtained." Scores taken from office records.	*M* 1938–39 1945–46	c 97.02 99.76		Significant difference between means.
							I.Q. †		
Hess, R. D. (1955)	Chicago, Ill.	w- high–188 low–178 c- low–179		6, 7, 8, 9 6, 7, 8, 9	Sample of 545 elementary-school pupils. High-status w from professional and managerial occupations; low-status w from semiskilled and unskilled; low-status c from unskilled and unemployed. All given a standard test and a new test.*	*M* High Low	w 108.88 102.83	c 90.21	Results suggest that socioeconomic differences between high- and low-status samples in this country are exaggerated by standard intelligence tests.
National intelligence tests ‡									
							MDN SCORES		
Jordan, A. M. (1922)	Fort Smith, Ark.	w-1502 c-247	10–14 10–14	4–8 4–8	Random sampling: 52 per cent of w and 71 per cent of c school children.	10 11 12 13 14	w 77 91.4 104 106 105	c 57.5 66.7 70.6 82 79	Pronounced racial difference found; c were two years behind w at ten years, three to four years behind at thirteen to fourteen years.

c: colored. w: white. M: mean SD: standard deviation. Mdn: median.
* In a personal communication the author indicated that the standard test used was generally the *Kuhlmann-Anderson* and that the subjects were from six public schools. For results on the new test, see *Dunn-Rice Test*, table 2.
† Author gives mean mental age for each of the twelve CA groups; I.Q.'s calculated by reviewer.
‡ Scale A, Form 1 was generally employed.

(Shuey, 1958, pp. 92–93)

The present range of scientific opinion can perhaps be fairly illustrated in these quotations from recent reviews of the subject. From a statement by thirty-two social scientists, included as the *Appendix to Appellants' Briefs Filed in the School Segregation Cases in the Supreme Court of the United States, October Term, 1952:*

> The available scientific evidence indicates that much, perhaps all, of the observable differences among various racial and national groups may be adequately explained in terms of environmental differences. It has been found, for instance, that the differences between the average intelligence test scores of Negro and white children decrease, and the overlap of the distribution increases, proportionately to the number of years that the Negro children have lived in the North. . . . Related studies have shown that this change cannot be explained by the hypothesis of selective migration. It seems clear, therefore, that fears based on the assumption of innate racial differences in intelligence are not well founded ["The Effects of Segregation . . . ," 1955, p. 435].

The "concluding statement" of a recent review of about 170 original investigations carried out over the past forty years:

> The remarkable consistency in test results, whether they pertain to school or preschool children, to high school or college students, to drafts of World War I or World War II, to the gifted or the mentally deficient, to the delinquent or criminal; the fact that the colored-white differences are present not only in the rural South and urban South, but in the border and northern areas; the fact that relatively small average differences are found between the IQ's of northern-born and southern-born Negro children in the northern cities; the evidence that the tested differences appear to be greater for abstract than for practical or concrete problems; the evidence that the differences obtained are not due primarily to a lack of language skills, the colored averaging no better on non-verbal tests than on verbal tests; the fact that differences are reported in all studies in which the cultural environment of the whites appeared to be no more complex, rich, or stimulating than the environment of the Negroes; the fact that in many comparisons (including those in which the colored appeared to best advantage) the Negro subjects have been either more representative of their racial group or more highly selected than have the comparable white subjects; all point to the presence of some native differences between Negroes and whites as determined by intelligence tests [Shuey, 1958, p. 518].

From the final summary of another, later review of "comparative psychological studies":

> We are not convinced that genetic differences have been shown; but even if they were so shown . . . the wide overlap between white and Negro distributions of scores should be pointed out so that it is evident that within group differences are far greater than between group differences. It should also be shown that oftimes two groups of white persons differ significantly, and probably in some

Tensor multiplication (a generalization of the dot product) involves summing over a pair of covariant and contravariant indices. A typical expression looks like

$$A^{i}_{\alpha j} B^{\alpha} = C^{i}_{j}$$

Summed indices (here, α) are called bound indices and disappear from the result. Unsummed (free) indices (the i and j) survive in the result. Since bound indices are purely local to the summation, we can arbitrarily change their name; we could just as well have written the above equation as

$$A^{i}_{\beta j} B^{\beta} = C^{i}_{j}$$

Free indices, however, can only be renamed globally; you have to do it consistently for each term on each side of an equation. Index name bookkeeping is one of the biggest pains of this method.

In my last column, I also introduced the special tensor called ε. We can use ε to abbreviate the following tensor operations. The cross product

$$A \times B = C$$

becomes

$$A^{\alpha} B^{\beta} \varepsilon_{\alpha\beta i} = C_{i}$$

The adjoint of a 3×3 matrix becomes

$$(M^{*})^{ij} = \frac{1}{2} \varepsilon^{i\alpha\beta} \varepsilon^{j\gamma\delta} M_{\alpha\gamma} M_{\beta\delta} \qquad (1$$

In homogeneous land we can drop the factor of 1/2.

A programming application

This vector/covector distinction can take on practical significance in the programming arena. One game that's currently played with modern programming languages is to define compound data types and overload all the language's arithmetic operators to do arithmetic on the new data type. For example, you might define a data type for vectors (as a triple or a quadruple of numbers) and then define vector addition for the "+" operator, vector subtraction for the "−" operator, and so forth in the obvious way. The problem comes when you get to multiplication. Of the two types of vector multiplication, the dot product and the cross product, which one should the "*" symbol mean? Now that we realize that there are two types of vectors, we can remove the ambiguity. We need to define two different data types: vectors and covectors. The storage structure, addition, and subtraction operators are the same for both of these, but multiplication depends on the operand type. Suppose the variable S is declared to be a scalar, V1 and V2 are vectors, and C1 and C2 are covectors. Expressions containing the multiplication operator should be interpreted as in Table 1.

Table 1.

Expression	Product	Result
S*V1	scalar	vector
S*C1	scalar	covector
V1*C1	dot	scalar
V1*V2	cross	covector
C1*C2	cross	vector

The epsilon-delta rule

If you multiply two epsilons together along one index, the rules of tensor multiplication give

$$\varepsilon_{ijk}\varepsilon^{klm} = D^{lm}_{ij}$$

a 4-index mixed tensor. Since ε is filled with a fixed pattern of 0's, +1's, and −1's, you can work out the values of D. They turn out to be

$$\begin{aligned} D^{lm}_{ij} &= +1 \quad \text{if } l = j;\ m = k;\ l \neq m \\ D^{lm}_{ij} &= -1 \quad \text{if } m = j;\ l = k;\ l \neq m \\ D^{lm}_{ij} &= 0 \quad \text{otherwise} \end{aligned}$$

You can write this in a more compact form by inventing yet another special tensor called δ.

$$\begin{aligned} \delta^i_j &= 1 \quad \text{if } i = j \\ \delta^i_j &= 0 \quad \text{otherwise} \end{aligned}$$

In other words, δ is just an identity matrix. This might seem a bit pointless at first, but it lets us write the above as

$$\varepsilon_{ijk}\varepsilon^{klm} = \delta^l_i\delta^m_j - \delta^m_i\delta^l_j \qquad (2)$$

This fundamental identity is called the *epsilon-delta rule* and is useful as a way to simplify any expression you come across that contains the product of two epsilons.

A simple application

Take, for example, the following common 3D vector identity:

$$A \times (B \times C) = (A \cdot C)B - (A \cdot B)C \qquad (3)$$

First let's see if this makes sense in our homogeneous world of vectors and covectors. B and C can be vectors (that is, 2DH points). Their cross product is a covector (a 2DH line). We cross this with A, which means that A must be a covector too. The result is a vector. Now look at the right side of the equation. It's also a vector, a weighted sum of the two vectors B and C. The weighting factors are dot products between vector-covector pairs. Everything fits in consistently with our scheme.

Writing the left side in Einstein index notation, we get

$$A_i(B^jC^k\varepsilon_{ijk})\varepsilon^{klm}$$

In this expression the letters *i*, *j*, *k*, and *l* are all bound indices (since they each appear exactly twice) indicating that they are implicitly summed over. The letter *m* is a free index, indicating that the net result is a vector with the single superscript *m*.

We are going to want to apply the epsilon-delta rule, but its definition (Equation 2) has different letters for the indices on the epsilons. We can make our expression more similar to it by shuffling the indices of the epsilons to get the common summed index, *l*, as the first one. This is legal as long as we do an even permutation of the indices.

$$A_iB^jC^k\varepsilon_{lki}\varepsilon^{lmi}$$

Then we need to do some pattern matching with Equation 2. It's actually easiest to rewrite Equation 2, renaming the indices to match up with the above expression. (This sort of thing is a nuisance we will be able to avoid with the graphical method.)

$$\varepsilon_{lki}\varepsilon^{lmi} = \delta^m_k\delta^i_i - \delta^i_k\delta^m_i$$

Stuff in the deltas for the product of the epsilons and you get

$$A_iB^jC^k(\delta^m_k\delta^i_i - \delta^i_k\delta^m_i)$$

Multiply out and apply several identities of the form

$$A_i\delta^i_k = A_k$$

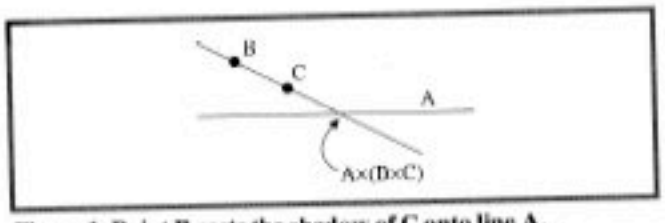


Figure 1. Point B casts the shadow of C onto line A.

You get

$$A_i B^m C^n - A_j B^j C^m$$

which, in old style vector notation, is

$$(A \cdot C)B - (A \cdot B)C$$

Ta daa! . . .

I show a geometric interpretation of this algebra in Figure 1. It's a 2D version of the perspective shadow calculation I described in my January 1988 column "Me and My (Fake) Shadow" (*CG&A*, Vol 8, No. 1, pp. 82-86). Think of B as a light source casting the shadow of point C onto the ground line A. The two sides of the equation are two ways of thinking about calculating the shadow location. You can think of it as finding the line through B and C, $(B \times C)$, and intersecting that line with line A, $(A \times (B \times C))$. Alternatively, you can parameterize the line through B and C as a linear combination of the vectors B and C.

$$\alpha B + \beta C$$

The α, β pair forms a one dimensional homogeneous coordinate for points on the line. Force this point to be on the line A by

$$A \cdot (\alpha B + \beta C) = 0$$

or

$$\alpha(A \cdot B) + \beta(A \cdot C) = 0$$

A solution to this equation is

$$\alpha = A \cdot C$$
$$\beta = -A \cdot B$$

so the intersection is

$$(A \cdot C)B - (A \cdot B)C$$

Quadric line tangency

Here's another use for the epsilon-delta rule.

In my last column, I noted that a symmetric 3×3 covariant tensor, **Q**, represents a second order curve (conic sections and the like). For points on the curve

$$\mathbf{P}\mathbf{Q}\mathbf{P}^t = 0$$

Then I baldly stated that a line L is tangent to the curve if

$$\mathbf{L}^t\mathbf{Q}^*\mathbf{L} = 0$$

where $\mathbf{Q}^*$ is the adjoint of **Q**. Why should you believe me (other than the fact that I am extremely trustworthy)? We need to express the fact that L and **Q** have exactly one point in common. To do this, we write an arbitrary point of L in the same way as mentioned above—as the weighted sum of two distinct points of L.

$$\mathbf{P} = \alpha\mathbf{R} + \beta\mathbf{S}$$

How, you might ask, do we decide what to use for R and S? Well, it doesn't matter. It's one of the funny things that often happen in mathematical proofs. We never actually have to know explicit coordinates for R or S, we just have to know that they exist and that

$$\mathbf{L} = \mathbf{R} \times \mathbf{S}$$

The points of L that intersect **Q** have

$$\mathbf{P}\mathbf{Q}\mathbf{P}^t = 0$$

or

$$(\alpha\mathbf{R} + \beta\mathbf{S})\mathbf{Q}(\alpha\mathbf{R} + \beta\mathbf{S})^t = 0$$

Multiplying this out and remembering that **Q** is symmetric gives

$$\alpha^2(\mathbf{R}\mathbf{Q}\mathbf{R}^t) + \alpha\beta 2(\mathbf{R}\mathbf{Q}\mathbf{S}^t) + \beta^2(\mathbf{S}\mathbf{Q}\mathbf{S}^t) = 0$$

This is a homogeneous quadratic equation in (α, β) of the form

$$\alpha^2 a + \alpha\beta b + \beta^2 c = 0$$

If L is tangent to **Q** this equation should have exactly one solution. Remembering our high school algebra, we find that this condition is

$$b^2 - 4ac = 0$$

or

$$4(\mathbf{R}\mathbf{Q}\mathbf{S}^t)^2 - 4(\mathbf{R}\mathbf{Q}\mathbf{R}^t)(\mathbf{S}\mathbf{Q}\mathbf{S}^t) = 0$$

We want to show that this condition is the same as

$$\mathbf{L}^t\mathbf{Q}^*\mathbf{L} = (\mathbf{R} \times \mathbf{S})^t\mathbf{Q}^*(\mathbf{R} \times \mathbf{S}) = 0$$

At this point, with conventional vector-matrix notation, we're stuck. There's no convenient way to use the commonality between cross products and matrix adjoints.

Einstein to the rescue. Write the two expressions in Einstein index notation. The first is

$$R^i Q_j S^j R^k Q_k S^l - R^i Q_j R^l S^k Q_k S^j$$

and the second is

$$(R^i S^j \varepsilon_{ijm})(\frac{1}{2}\varepsilon^{mnp}\varepsilon^{rsp} Q_{nr} Q_{ls})(R^k S^l \varepsilon_{kln})$$

You can show that these two expressions are equal by renaming a lot of indices and using the epsilon-delta rule twice. It works, but you have to be careful to keep all 10 index names straight. We would like a still simpler scheme.

Diagram notation

A few years ago, the physicist Richard Feynman introduced a diagrammatic technique to express equations in high energy physics. A vastly simplified form of these Feynman diagrams can help us out here too. The following is adapted from a book called *Diagram Techniques in Group Theory*.[1]

We represent the product of a bunch of tensors in diagram form as a directed graph. The encoding is as follows:

- Each tensor in the product is a node in the graph.
- Each index is an arc. A contravariant index is directed away from the node, a covariant index is directed towards the node.
- Bound indices (those that are summed over) are arcs connecting two nodes. The directedness of the arc represents the fact that you can sum only over a contravariant-covariant pair of indices.
- Free indices are "dangling" arcs.

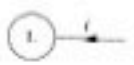
The nice thing about this is that bound indices don't need to be named (although I will sometimes do so here to more easily relate the Einstein index notation to the diagram notation). Only the topology of this diagram is important. Any rearranging that preserves topology is okay except for mirror reflections. These are not allowed because they imply transposition of some of the tensors. This is simply a sign flip for epsilon, but it might change the value of a nonsymmetric tensor.

Here are some examples. A point is

P^i

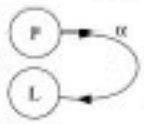
A line is

L_i



The dot product of a point and a line is

$P^\alpha L_\alpha$



which is topologically equivalent to

A transformation matrix is

T^i_j

A transformed point is

$P^\alpha T^i_\alpha$

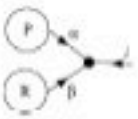
We write the special tensor epsilon (in either its covariant or contravariant forms) using a dot for the node and labeling index arcs counterclockwise around the node.

ε^{ijk}

ε_{ijk}

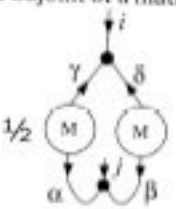
The cross product of two vectors is

$P^\alpha R^\beta \varepsilon_{\alpha\beta i}$

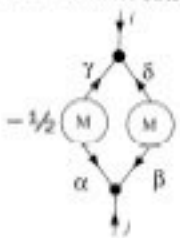


Note that mirroring this flips the sign.

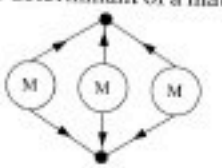
Equation 1 for the adjoint of a matrix appears in diagram form as



This is a bit clumsy. We can flip the lower epsilon over to make it a bit prettier, but we must compensate by prefixing with a minus.



If we multiply the adjoint by the original matrix, we get the identity matrix times the determinant of the original matrix. The trace (sum of the diagonal elements) of this 3×3 matrix is then three time the determinant. I'll leave it for you to verify that, graphically, the determinant of a matrix is proportional to



no errors is $(1-p)^n$ and the probability of having at least one significant edge shift is

$$P(m,X)=1-(1-p(X))^m \approx mp(X), p(X) \propto 1/m \quad (19)$$

We will use Equation 19 and the current bar code reading specifications to obtain an idea of the physical size of I. Assuming a UPC code word with four edges and an expected error rate of about 10^{-6}, we find that $P(4,X)=4\cdot10^{-6}$. Substituting into Equation 18 and taking the natural logarithm of both sides, we find

$$-\frac{X}{2I} \le -6\ln 10 - \ln 4 = -15.2 \quad \text{or} \quad X \ge 30.4I$$

For a minimum module width of 10 mils, this expression yields I equal to about 0.3 mil.

The probability of at least z significant edge shifts is

$$P_z(m,X)=\sum_{k=z}^{m}\left[\begin{matrix}m\\k\end{matrix}\right]p^k(1-p)^{m-k} \approx m^z p^z \quad (20)$$

with the approximation holding for z much smaller than m and p much smaller than $1/m$ so that their product is much less than 1.

The number of required code words N obviously imposes a lower bound on n. However, a comparison of Tables 3 and 6 shows that n is never set at that minimum. It is usually set at a higher value to allow for some error detection because of redundancy. If we keep the module size fixed (in other words, we keep the error rate fixed), then an increase in n will lower the density. If we keep the code word length fixed so the density remains fixed, then it appears that an increase in n will increase the error rate. However, the number of available code words increases exponentially with n, and we can take advantage of this increase to introduce error detection and error correction schemes. The concept has already been used empirically in all the earlier bar codes, which typically use only half of the available code words.

In the next two sections we present a methodology for a systematic analysis of the trade-offs. The methodology is based on the analysis developed by Wang.[14] Consult that work for mathematical details of the development.

Error detection is possible when we use only a fraction of the possible code words. We can set things up so that a single error will convert a code word into one not used. UPC and Code 39 have that property. If we arrange things so that it requires three errors to convert one used code word into another also used, then we can also do error correction. For each erroneous code word u we try to find another v from which u can be derived by one or two errors. Then we can assume that v is the correct version of u. We discuss such strategies next.

Table 8. Transformations of strings because of noise.

Noise Effect	Transformation	Weight	
None	$xy \to xy$	0	(1)
One edge shift to the left	$xy \to (x-1)(y+1)$	1	(2)
One edge shift to the right	$xy \to (x+1)(y-1)$	1	(3)
Warping or pair of edge shifts	$xy \to (x-1)y$	1	(4)
Warping or pair of edge shifts	$xy \to (x+1)y$	1	(5)
Warping or pair of edge shifts	$xy \to x(y-1)$	1	(6)
Warping or pair of edge shifts	$xy \to x(y+1)$	1	(7)
Combinations of edge shifts	$xy \to$ other		(8)
Merging of three into one	$xyz \to (x+y+z)$	τy	(a)
Splitting of one into three	$x \to uvy, (u+v+y=x)$	τv	(b)

Coding theory and bar codes: Diagonal distance

The cornerstone of error correction techniques is the definition of a distance between a pair of messages. The most common such measure, the *Hamming distance*, equals the number of places two strings differ. If only a subset of all possible code words has been chosen, then a corrupted message is assumed to represent the nearest correct code word.

In this article we use only elementary concepts of coding theory, which you can find in any text on the subject. The book by Hill[15] is a particularly readable source.

Bar codes are read as sequences of widths of bars and spaces. To compare such widths with each other, we must be careful about how the Hamming distance is computed. For example, the signal 324 represents a bar with width 3 followed by a space with width 2, and then a bar with width 4. If the first element is a little "fat" from ink spread during the printing process, then the signal will probably be read as 414. Using a general radix, the Hamming distance between 414 and 324 is 2, the number of differing elements. However, the correct Hamming distance is only 1, as shown by the binary representations:

```
324    111 00 1111
414    111 10 1111
```

Instead of having to go to the binary representation each time, we can introduce a different way of calculating the distance between the strings. Such a step is advisable for four more reasons:

(1) Most scanners detect the change of reflectance (edge) of the substrate rather than the existence of bars or spaces. Therefore, a single physical event will cause an edge shift. In contrast to communications systems, where the individual pulses are detected, the scanner does not see the underlying module structure.

(2) The Hamming distance between the first string in Figure 7 and either the second or third is 1, while the second string is far less likely than the third. Coding theory is based on the premise that the smaller the distance, the more likely that one string is a distortion of the other.

(3) In the case of width codes, there are no corresponding binary representations when the wide-to-narrow ratio is not an integer.

(4) Due to scanning speed variation or substrate distortions (for example, a bar code printed on a plastic bag might stretch because of the elasticity of the plastic), an element can be shortened or lengthened.

We call such effects *warping errors*, for example

```
324    111 00 1111
314    111 0 1111
```

In this case the distance of the binary representations cannot be measured by the Hamming distance because we have an insertion error.

Therefore, it is best to define a metric that will give the answer 1 when a single physical disturbance occurs.

Table 8 gives a partial list of the trans-

Table 7. Specific numbers of example in Figure 6.

	Bar	Space	Bar	Space	Bar	Space	Bar	Space	Bar
Ideal Widths	0.50	0.50	0.20	0.40	0.40	0.20	0.20	0.20	0.20
Measured Widths	0.50	0.50	0.30	0.31	0.40	0.29	0.20	0.20	0.20

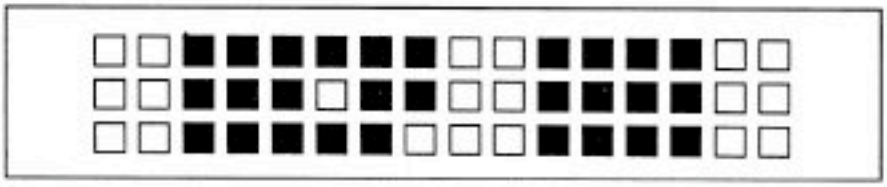


Figure 7. Possible errors in a binary string.

Notice in Figure 6 that the second, fourth, and fifth pulses have the same real widths, but the waveform of the second pulse differs from the other two. As a consequence of the convolution distortion, the second pulse appears wider than the other two. The specific numbers of the example shown in the figure appear in Table 7. Both sets of widths are expressed in the same arbitrary units. Notice that a 0.2 input width is mapped in one case to 0.3 and in the other two to 0.2. Because of the severe distortions of very narrow pulses, the minimum width of modules used in bar codes must exceed the width of the minimum detectable pulse to compensate for the distortion.

Using more sophisticated decoding techniques can help remedy the situation. For example, we could compare the width of a bar or a space not only to the widths of its immediate neighbors, but also to the widths of other code words. Such decoding might require more computing power than commonly used now, but it offers a way of tolerating higher code densities.

In addition to those already mentioned above, many other sources of noise exist. Some are multiplicative, such as the speckle noise inherent to coherent laser illumination and that due to the paper substrate. Others are additive, such as the ambient illumination (low frequency noise in the case of artificial light and shot noise in the case of sunlight, the electronics of the scanner, etc. See Barkan and Sklar[12] and Barkan and Swartz[13] for more on these topics.). While the effects of those noise sources do not necessarily concentrate at the edges, they show no preference for the interior, either. Thus, overall we have greater sensitivity at the edges.

Since most problems occur around the edges, we deal with signal-dependent distortions and noise. That makes inapplicable many of the analytical techniques used in electronic communications, where the noise is usually signal independent. Consider, for example, a message (stream of bits) of the form shown in the top row of Figure 7. A white square stands for a 0 and a black square for 1. In electronic communications a 0 can be mapped into a negative voltage pulse and a 1 into a positive voltage pulse. The nature of noise is such that it is equally likely to misread a pulse at the ends or the middle of a run of similar polarity pulses. Thus, the distortions shown in the second and third rows are equally likely.

We can encode the same sequence on paper by painting "modules" black for 1's and white for 0's. The result will be a white band of two modules, a black of six, a white of two, etc. It is rather unlikely that a module in the middle of the blank band will be read as a white, but it is much more likely that one in the edges will. This problem exists in other technologies as well, particularly optical and magnetic recording.[3-6] There, most of the noise also occurs at the edges, giving rise to intersymbol interference.

Bar code design: Preliminaries

The design of a bar code involves the following major parameters:

- the number, N, of distinct symbols that must be encoded;
- the error rate, E_r, for rejections (when a code word is flagged as unreadable);
- the error rate, E_s, for substitutions or misdecodes (when one code word is read for another);
- the density of information D measured as bits per inch; and
- the number of modules n.

The number of zones k in an (n,k) code is closely related to n. For symmetric codes we have the relation $n=4k-1$. Usually N is given and we wish to minimize the two error rates, maximize the density, and keep n low. We wish to keep n low because the complexity of the decoding algorithm and the size of decoding tables increase with n. These parameters cannot all be optimized at the same time because they all depend on the module size X. We will discuss the possible trade-offs among them next.

The trade-off between rejection and substitution rates, already well known from statistical decision theory, can be seen intuitively from the following extreme cases: If we set the rejection rate to 100 percent, then we cannot have any substitution errors. Alternatively, if we do not mind how many substitution errors occur, we can have a zero rejection rate.

The rejection and substitution rates' relative sizes are controlled by a parameter in the decision step for a given overall decoding quality. Therefore, we will focus only on the substitution rate, assuming a zero rejection rate. If we can make that small during the design of the bar code, then we can make it even smaller by allowing some rejection errors. In bar code practice we usually work with a 10^{-7} rejection rate and a 10^{-6} substitution rate.

An error typically occurs when we misread the width of an element in terms of modules. For example, instead of, say, 3,2,1,1 (the code word for 0 in the UPC), we read 1,4,1,1 (the code word for 3 in the UPC). Clearly, the smaller the module size, the more likely it will have errors when reading the code, because a given amount of edge shift will constitute a higher percentage of the total width if X is small. Moreover, the number of possible values to be discriminated should affect the error. Unfortunately, there have been no systematic studies of these relationships.

The Laplacian channel model looks like a reasonable approximation for the probability of an edge shift equal to at least one module:

$$p(X) = e^{-\frac{X}{T}} \tag{18}$$

where T is a normalizing constant. If a code word has m edges, then the probability of

$$S(n,k) = \begin{bmatrix} n-1 \\ 2k-1 \end{bmatrix} = \frac{(n-1)(n-2)\ldots(n-2k+1)}{(2k-1)(2k-2)\ldots 3\cdot 2} \quad (8)$$

This is the number of combinations of $2k-1$ out of $n-1$ objects. Derive the result by observing that there are $n-1$ module boundaries and $2k-1$ bar/space boundaries (also see the section "Universal Product Code"). These combinations represent only the number of distinct zones, and we should expect twice as many† code words because each zone pattern can be given one of two color arrangements. On the other hand, we cannot use half of the zone patterns, because of the need for bidirectional scanning. For each width pattern we must reject the one obtained by reversing the width sequence. These two factors cancel each other, and we are left with the number given by Equation 8.

UPC is a (7,2) code, therefore it has 20 possible code words (as we have already seen in that section above). Since the number of modules is n, the information content is

$$H(n,k) = \frac{1}{n}\log_2\begin{bmatrix} n-1 \\ 2k-1 \end{bmatrix} = \frac{1}{n}\sum_{i=1}^{2k-1}[\log_2(n-i) - \log_2(i)] = \frac{1}{n}\sum_{i=1}^{2k-1}\log_2\left(\frac{n}{i}-1\right) \quad (9)$$

For a given n, S is maximum when $2k-1$ is exactly half of $n-1$, which implies $n-1 = 2(2k-1)$ or

$$n = 4k-1 \quad (10)$$

Codes where n and k satisfy the above equation are called *symmetric* codes. Table 4 lists the values of S and H for various (n,k) codes.

We observe from Table 4 that a (7,2) code has H equal to 0.617, while from Table 3 we see that UPC has H equal to 0.474. The difference, 0.143, is the loss because of the need for error detection.

It is possible to derive an approximate but concise expression for $S(n,k)$ using Stirling's formula,[11] which approximates $n!$. For symmetric (n,k) codes (that is, a $(4k-1,k)$ code) Equation 8 yields

$$S_s(n) = \begin{bmatrix} n-1 \\ \frac{n-1}{2} \end{bmatrix} \approx \frac{2^n}{\sqrt{2\pi(n-1)}} \quad (11)$$

Equation 11 shows the "loss" factor of (n,k) codes compared to the theoretical maximum of 2^n for delta codes. The loss factor is

$$\frac{1}{\sqrt{2\pi(n-1)}}$$

Table 4. Characteristics of some (n,k) codes.

k	(n,k)	S	H	Name of Related Code**
2*	7,2	20	0.617	UPC
3	9,3	56	0.645	Code 93
3*	11,3	252	0.725	Code 128
4	14,4	1,716	0.767	
4*	15,4	3,432	0.783	
4	16,4	6,435	0.791	Code 49
4	17,4	11,440	0.793	PDF417
5*	19,5	48,620	0.819	

* Symmetric codes.
** Recall that the actual codes do not use all possible code words.

Table 5. Statistics for some (n,k,m) codes.

n,k	m	L	$S-L$	Efficiency
11,3	5	6	246	99.56%
	4	6+30=36	216	97.21%
	3	36+90=126	126	87.46%
16,4	8	8	6,427	99.98%
	7	8+56=64	6,371	99.88%
	6	64+224=288	6,147	99.47%
	5	288+672=960	5,475	98.15%

Taking the binary logarithm of S and dividing by n we find that

$$H_s(n) \approx 1 - \frac{\log_2[2\pi(n-1)]}{2n} \quad (12)$$

This yields 0.626 for the (7,2) code and 0.7285 for the (11,3) code, which are quite close to the correct values 0.617 and 0.7252. Most important, it shows the effect of increasing n. As $n \to \infty$, $H(n) \to 1$, the maximum theoretical information content. The equation for the density is

$$D_s(n) = \frac{1}{X}H(n) = \frac{1}{X} - \frac{\log_2[2\pi(n-1)]}{2W} \quad (13)$$

where W is the width of a code word.

(n,k,m) codes. (n,k) codes with large n contain some bars or spaces of width equal to $n-2k+1$, which ranges from 4 for a (7,2) code to 11 for a (19,5) code. Very wide intervals are detrimental for the reading process because they can be confused with margins or other demarcations. Therefore, some symbologies omit the combinations containing very wide intervals. We will show that this eliminates relatively few codes.

Let u be a particular (integer) width. If u is greater than $n/2-k+1$, then the code can have at most one instance of u. For example, a (16,4) code can have at most one width equal to 6, but two widths equal to 5. We can easily calculate the number of code words containing an element of such width. The particular width can occur in any of $2k$ places. This leaves $n-u$ modules to be distributed into $2k-1$ places. There are

$$\begin{bmatrix} n-u-1 \\ 2k-2 \end{bmatrix} \quad (14)$$

such possibilities. Therefore, if m is greater than or equal to $n/2-k+1$, the number of "lost" code words is

$$L = 2k\sum_{u=m+1}^{n-2k+1}\begin{bmatrix} n-u-1 \\ 2k-2 \end{bmatrix} \quad (15)$$

We use the notation (n,k,m) to denote a code that has all the code words of an (n,k) code except those with width higher than m. Table 5 lists L for a set of codes. The

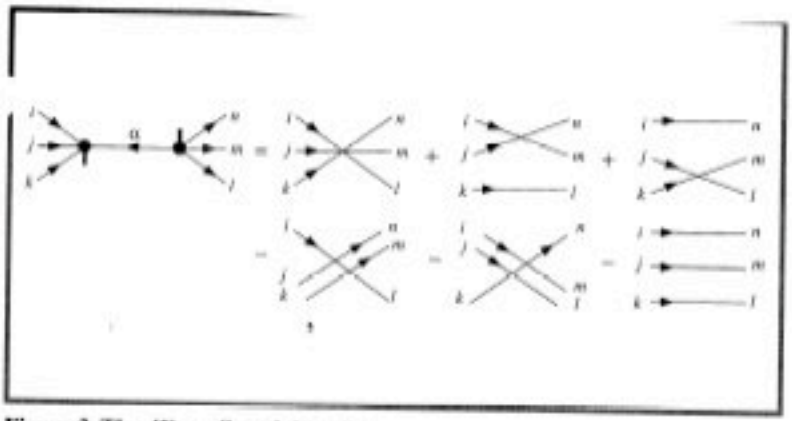


Figure 3. The 4D epsilon-delta rule.

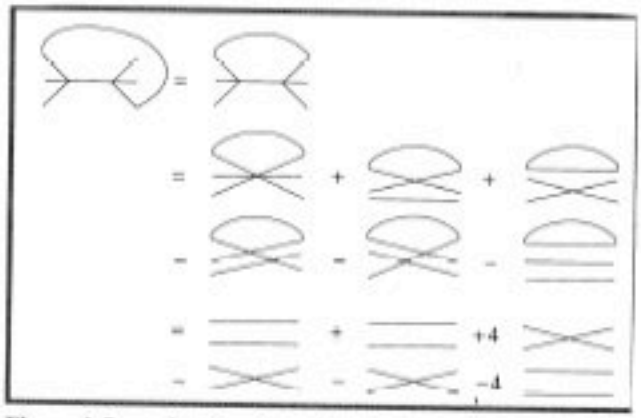


Figure 4. Summing the 4D epsilon-delta rule over an additional index.

Whenever signs matter, we need to put a little tick mark between the intended first and last arc. The diagram is

ε^{ijkl}

Exchange of any neighboring arcs (counting the tick mark as an honorary arc for now) flips the sign. In fact, in 4D, mirroring just the epsilon portion of a diagram does not change the sign.

The 3DH homogeneous line between two points is

$P^{\alpha}R^{\beta}\varepsilon_{\alpha\beta ij} = L_{ij}$

Note that this tensor has two covariant indices. The contravariant form is

$L_{\alpha\beta}\varepsilon^{\alpha\beta ij} = L^{ij}$

The 4D epsilon-delta rule appears in Figure 3. You can remember it by noting that the positive terms have an odd number of crossings (one or three) and the negative terms have an even number (zero or two). The double summation can be found graphically by connecting the *i* and *l* branch together. This gives us the graphic we see in Figure 4. Note that a closed loop is the graphical representation of δ^i_i and is equal to 4. This all simplifies to the graphic we see in Figure 5.

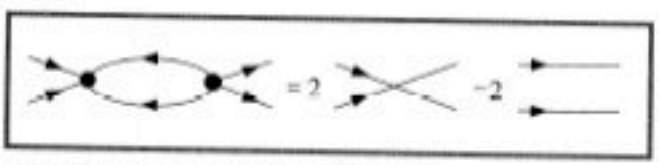


Figure 5. The doubly summed 4D epsilon.

An application

A sample application of this technique appears in Figure 6. This figure shows the 4D(3DH) perspective shadow calculation.

Whew

The epsilon-delta rule and the nifty diagrammatic technique make a lot of pretty complicated derivations easy. I expect they will be useful in a lot of other problems too. The observant reader will note that I've played a bit fast and loose with some global scale factors and sign changes. And that's . . . okay. 'Cause, gosh darn it, we're homogeneous. ❑

Reference

1. G.E. Stedman, *Diagram Techniques in Group Theory*, Cambridge University Press, United Kingdoms, 1990.

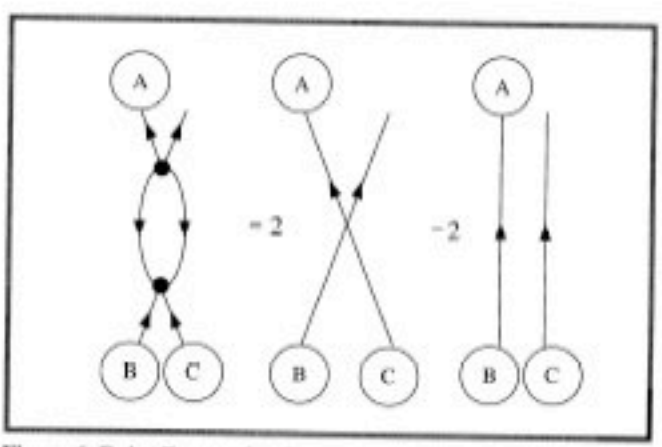


Figure 6. Point B casts the shadow of point C onto plane A.