Nonlinear Intrinsic Dimensionality Computations

CHIU KUAN CHEN AND HARRY C. ANDREWS

Abstract—In pattern recognition, the raw data and dimensionality of the measurement space is usually very large. Therefore, some form of dimensionality reduction has been commonly considered as a practical preprocessing method for feature selection. Based on a method that increases the variance while maintaining local structure, a technique is developed to determine intrinsic dimensionality. A cost function is introduced to guide the maintenance of the rank order and therefore local structure. Two criteria of using the cost function to increase the variance have been introduced. Several methods of defining the local regions is suggested. A program is implemented and tested to find the intrinsic dimensionality of a variety of experimental data.

Index Terms—Feature selection, intrinsic dimensionality, minimum spanning tree, nonlinear mapping, pattern recognition, rank orders.

I. INTRODUCTION

The problem of pattern recognition is conveniently divided into two subproblems: 1) feature extraction and selection, and 2) classification of pattern classes. The first is concerned with the problem of what to measure in a given set of raw data, the second is the problem of how to make optimal decisions from the given data. Considerable achievement has been made in solving the classification problem [1]–[3]. However, the development of feature selection has seen less progress. The problem of what to measure is a difficult one. In practical applications of pattern recognition the choice of features usually depends on individual experience. The objective of feature extraction is to retain the prominent information for the process of classification, and yet there is no general rule to select a good feature measurement. However, feature extraction often results in a dimensionality reduction and can often be coupled to intrinsic dimensionality computations.

In order to take advantage of knowing the intrinsic property of a large number and dimensionality of data a variety of nonlinear approaches have been developed. Some original work on nonlinear multidimensional scaling was done by Shepard [4], [5] and further refined by Kruskal [6], [7]. Shepard and Carroll [8], Bennett [9], Trunk [10], and Fukunaga and Olsen [11] separately modified or developed different approaches for finding the intrinsic dimensionality of a point set of data.

The purpose of multidimensional scaling is to represent possibly nonmetric data, such as measures of psychological similarity or dissimilarity, in terms of metric configurations with the minimum number of dimensions. Shepard [4], [5] introduced the idea of preserving monotone relationships between the ranking distances and similarities. In metric space an analogous algorithm results. Based on the probability density function of the normalized interpoint distance in a sphere of radius a in M dimensions, Bennett [9] pointed out that for two uniformly distributed point sets the configuration with a larger interpoint distance variance will have a lower dimensionality. This function for the normalized interpoint distance

\[ d = d_{ij}/2a \]

is distributed as

\[ P_M(d) = 2^M M_d(M-1) \frac{((1/2)M+(1/2), (1/2))}{1_{C-M}} d^{C-M} \]

where \( I(M, q) \) is the incomplete beta-function [13], [14], and \( d_{ij} \) is the distance from point \( x_i \) to point \( x_j \) in the sphere. The dimensionality

Manuscript received September 20, 1972; revised May 10, 1973. This research was supported by National Science Foundation under Grant GK-20255 and by the Advanced Research Projects Agency of the Department of Defense and was monitored by the Air Force Eastern Test Range under Contract F08606-72-C-0008.

The authors are with the Department of Electrical Engineering, University of Southern California, Los Angeles, Calif. 90007.

REFERENCES

of the signal collections is reduced by an iteration technique for maintaining distance rank orders and increasing variance simultaneously.

Under the premise that a configuration of larger interpoint distance variance has a lower dimensionality, the technique of stretching and shrinking interpoint distances to increase the variance in the unfolding process has also been adopted in our method. However a cost function has been introduced to aid in smoothing the unfolding method and is formulated in terms of the rank orders.

II. COST FUNCTION AND INTRINSIC DIMENSIONALITY

In the unfolding algorithm of Bennett [9], the algorithm has to sacrifice a portion of the cluster structure in order to obtain the unfolded goal. Hence, we introduce the concept of a cost function to limit the scale of sacrifice. As an aid in illustrating the algorithm it is interesting to investigate the regions of allowable movements while preserving all rank orders. For a four point cluster \{ABCD\}, there are \(\frac{N(N-1)}{2}\) 6 edges in the graph. Their distance inequalities are

\[ AB < BD < AD < BC < DC < AC. \] (1)

The maximum number of constraints on a given point equals \(N(N - 1)/2 - 1\), the number of edges minus one. The areas where each point is allowed to move without violating the rank order are not necessarily convex regions. Such regions will be quite complex in multi-dimensional space, and the complexity of boundaries increases with the number of points in the graph and the dimension of the space. In general, for \(N\) points the constraints increase on the order of \(N^2\) and thus the movable area may be very small if it is not allowed to violate the rank order. Our problem is to find the intrinsic dimensionality of the point set. Since the rank order limits the freedom of moving a point, we would sacrifice a portion of this restriction in obtaining our goal. The use of the cost function is for preserving local structure. Since the local region plays an important factor in the unfolding process, we want it heavily weighted in the algorithm. We want to pay a high penalty for misraking the order in the local region but less penalty for nonlocal regions.

For a set of rank orders, the most convenient technique is to assign the integers to represent the costs. Suppose that three interpoint distances for \(x_i, x_j\) and \(x_k\) are ranked as \(d_{ij} < d_{jk} < d_{ik}\). The ranks would be assigned as

\[ R_{ik} = 1, \]
\[ R_{jk} = 2, \]
\[ R_{ij} = 3. \]

For an \(N\)-point cluster there are \(N(N - 1)/2\) interpoint distances and ranking values can be designated from 1 to \(N(N - 1)/2\). The values assigned to the ranks are a monotonic increasing integer function. The cost function could be defined as the following equations:

\[ C_{ij} = 0, \quad \text{if } R_{ij} = R_{ij}' \] (2a)
\[ C_{ij} = R_{ij} R_{ij}', \quad \text{if } R_{ij} \neq R_{ij}' \] (2b)

where \(R_{ij}'\) is a newly designated rank value after the variance-increasing shift has taken place. The previous equations are a function of rank orders \(R_{ij}\), which, in turn, are a function of distances \(d_{ij}\), and the pair itself. The cost due to the movement of point \(x_j\) is defined as

\[ C_j = \sum_{i \neq j} C_{ij}. \] (3)

Take a five-point cluster in Fig. 1 as an example. There are \(\frac{N(N - 1)}{2}\) = 10 edges. The length of each edge is shown in the figure. The ranking values are assigned as

\[ R_{AB} = 10, \quad R_{AE} = 9, \quad R_{AC} = 8, \quad R_{DE} = 7, \quad R_{AD} = 6, \]
\[ R_{EC} = 5, \quad R_{BD} = 4, \quad R_{BC} = 3, \quad R_{BE} = 2, \quad R_{DC} = 1, \]

according to the set of inequalities

\[ d_{AB} < d_{AE} < d_{AC} < d_{DE} < d_{AD} < d_{ED} < d_{DB} < d_{BC} < d_{BE} < d_{DC}. \]

Fig. 2(a) shows the cost function of pair \(AB\) for (2). In the figure, the height of each step is weighted by the same constant \(R_{AB} = 10\), hence all are equal in the diagram. The lengths of all edges are projected onto the horizontal axis. Now, ignoring the configuration but fixing the rest of the edges, when the edge \(AB\) stretches equal to or less than 3.3, the ranking value of \(R'_{AB}\) is still 10. This means there is no violation of rank order and by (2a) the cost is zero. However when edge \(AB\) stretches further the violation between distances 3.3 and 3.5 is \(R_{AB} = 90\). In a similar manner, \(C_{AB}\) can be drawn as declined step functions each occurring at the point of violation and shown in Fig. 2(a).

For edge \(DE\) the cost function is shown in Fig. 2(b). The cost \(C_{DE}\) is zero between lengths 3.5 and 4.2. This is the range that edge \(DE\) can shrink and stretch without violating the rank order.

The process of increasing the variance is similar to the one developed by Bennett [9]. Assuming a Euclidean metric, the average interpoint distance is

\[ \bar{d} = \frac{1}{N(N-1)} \sum_{i \neq j} d_{ij}. \] (4)

where \(d_{ij}\) is the Euclidean distance between \(x_i\) and \(x_j\). During the process of increasing the variance, the points at the edge of the cluster will be moved farther out. In order to avoid the calculation of large numbers, let the normalized variance be

\[ \sigma^2 = \frac{1}{N(N-1)} \sum_{i \neq j} \sum_{j=1}^N \frac{(d_{ij} - \bar{d})^2}{\bar{d}^2}. \] (5)

Note that the data will also be normalized at each iteration. To increase the variance in \(d_{ij}\), let

\[ \varepsilon_{ij} = \frac{d_{ij} - \bar{d}}{\bar{d}} = d_{ij} - 1 \] (6a)
be the differential element. The last equality is due to $\overline{d} = 1$. This equation gives the stretching and shrinking process a linear scale depending on the distance $d_{ij}$. When $\delta_{ij}$ is positive, i.e., $d_{ij} > \overline{d}$, the distance $d_{ij}$ should be increased, and if it is negative, i.e., $d_{ij} < \overline{d}$, $d_{ij}$ should be decreased. When the intrinsic surface is not homogeneous in density, it may have points at the edge and thus away from the center of the cluster. To avoid such edge points moving a great distance away, a modified equation for $S_{ii}$ is defined as

$$
\delta_{ij} = \frac{d_{ij} - 1}{s d_{ij} + t}
$$

(6b)

where $s$ and $t$ are constants governing the distance $d_{ij}$ increases and decreases.

The $k$-coordinate contribution for point $x_{jk}$, at the $q$th iteration, in the new position given by point $x_{jk}$ is

$$
x_{jk}(q + 1) = x_{jk}(q) + K \alpha (x_{jk}(q) - x_{ik}(q)) \delta_{ij}
$$

(7)

where $\alpha$ is a constant that determines the rate of variance increase. $K$ is a constant to be explained in the next paragraph. The shifting of the point $x_{jk}$ should be a function of the entire collection of points. The tentative position of point $x_{jk}$ in the $k$-coordinate is then

$$
x_{jk}(q + 1) = x_{jk}(q) + \sum_{i=1}^{N} (x_{jk}(q) - x_{ik}(q)) \delta_{ij}
$$

(8)

There are two methods to move to $x_{jk}(q + 1)$. The first method is to force zero cost within a local region. The second method allows a non-zero cost for a moved point. In the first method let $N'$ be the number of points located in a hypersphere of radius $\beta \overline{d}$ centered at point $x_{jk}$. $\beta$ is a constant determining the size of the local region. It is known that a point has a neighborhood in which it can increase the variance without violating the rank order. We can use the "golden search" technique to locate the point along the direction of increasing variance. Equation (8) is one of the components of this direction. In order to speed the

1Doubling or halving the distance depending on nonviolation or violation of the zero cost criterion. (Also known as a binary search technique.)
convergence of the computation, we give each point a fixed number of moves to locate the largest zero cost position within \( \beta \).

The constant \( K \) in (8) is simply that largest value determined by the "golden search" routine that maintains the new position of \( x \) to be zero cost.

The second method is to allow a small cost in the move. Again the golden search in the direction of increasing variance is utilized and the constant \( K \) is fixed such that the cost is bounded above by \( C \). The final value of \( x_{jk}(q + 1) \) in (8) will be the one such that the cost is in less than or equal to \( C \) against \( N' \) nearest neighbors.

The iterative process mentioned in this section will not converge, since at the end of each iteration there always exists a new point with no violation of rank order from the previous point set. In other words, the always exists zero-cost regions for some points such that they can be moved in the direction of increasing variance without any violation of rank order. Hence, the change of variance is not a decreasing function in terms of iteration due to end point's continued movement. When this condition occurs the iteration should stop. Let

\[
\delta \sigma^2(q) = \frac{\sigma^2(q) - \sigma^2(q-1)}{q-1} \quad \text{for } q > 2
\]

be the ratio of the change of variance to those of all previous changes on the \( q \)th iteration. When \( \delta \sigma^2 \) is larger than some threshold, the iteration terminates. Equivalently when some large percentage of points remain unmoved, the algorithm terminates.

Up to this point we have been discussing the approach of unfolding the cluster. We have utilized nonlinear techniques to simplify the structure of the cluster so that we can determine the dimensionality by conventional mathematical methods. The final configuration is still imbedded in the original \( M \)-dimensional space. In order to reduce the number of coordinates, a rotation to principal axes is suggested. The Karhunen-Loeve expansion is used here and gives us a set of eigenvectors, with corresponding eigenvalues, which can be arranged in a decreasing order. Since our unfolding algorithm is a nonlinear transformation, the eigenvalues will not preserve the maximal amount of energy of the original data. This rules out any hope of using a mean-square-error criterion. Therefore, in this algorithm we have chosen a threshold technique. Thus a threshold value, \( \gamma \), is used as a guide in determining dimensionality. Let \( \gamma_k \) be defined by

\[
\gamma_k = \frac{\sum_{i=1}^{k} \lambda_i}{\sum_{i=1}^{M} \lambda_i}
\]

where \( \lambda_i, i = 1, 2, \ldots, M \), are eigenvalues, in a decreasing order, from the result of unfolded data and \( M \) is the number of dimensions in the original space. The values of \( \gamma_k \) increase as a function of \( k \). If we let \( k \) scan from 1 to \( M \) such that when the first value of \( \gamma_k \), say \( \gamma_R \), exceeds the threshold value of \( \gamma = 0.95 \), then \( R(R < M) \) is the dimension we seek. The value of 0.95 is empirical. Ideally, the result of an unfolding algorithm would have \( R \) eigenvalues dominating the remaining ones. In other words, those \( M - R \) remaining eigenvalues should be zero or very small. In such case, we could set the threshold value to, say, 0.998 and have no difficulty in making a decision.

### III. Experimental Results

Several examples were used to demonstrate the technique. The first example is a 25-point uniformly deterministic point set on a hemisphere shell. The second example is also a hemisphere shell but the points are randomly generated. The third example is a 25-point helix. Example 4 is a 64-point uniformly deterministic cube from which we want the algorithm to detect nonintrinsic cluster. Finally, for comparison purpose, the modified equation for the differential element as given in (6b) is tested for the data given in Example 4.

#### Example 1

A 25-point 3-dimensional uniformly deterministic point set of data is generated on a hemisphere shell as shown in Fig. 3. The normalized eigenvalues of these data and values of \( \gamma_k \) are

- \( \lambda_1 = 1.000 \quad \gamma_1 = 0.4143 \)
- \( \lambda_2 = 1.000 \quad \gamma_2 = 0.8286 \)
- \( \lambda_3 = 0.414 \quad \gamma_3 = 1.0000 \).

1) Under the zero cost condition:

<table>
<thead>
<tr>
<th>number of iterations</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>computer running time</td>
<td>67 s</td>
</tr>
</tbody>
</table>

normalized final eigenvalues and values of \( \gamma_k \) are

- \( \lambda_1 = 1.000 \quad \gamma_1 = 0.7333 \)
- \( \lambda_2 = 0.364 \quad \gamma_2 = 0.9999 \)
- \( \lambda_3 = 0.185 \times 10^{-5} \quad \gamma_3 = 1.0000 \).

2) Under a cost less than 30 units condition:

<table>
<thead>
<tr>
<th>number of iterations</th>
<th>15</th>
</tr>
</thead>
<tbody>
<tr>
<td>computer running time</td>
<td>99 s</td>
</tr>
</tbody>
</table>

normalized final eigenvalues and values of \( \gamma_k \) are

- \( \lambda_1 = 1.000 \quad \gamma_1 = 0.5649 \)
- \( \lambda_2 = 0.770 \quad \gamma_2 = 1.0000 \)
- \( \lambda_3 = 0.323 \times 10^{-6} \quad \gamma_3 = 1.0000 \).

Fig. 4(a) and (b) show, respectively, the second and fourth iterations of the test of (1).

#### Example 2

A 30-point 3-dimensional randomly generated data set is also on a hemisphere shell. The normalized eigenvalues of these data and values of \( \gamma_k \) are

- \( \lambda_1 = 1.000 \quad \gamma_1 = 0.4692 \)
- \( \lambda_2 = 0.920 \quad \gamma_2 = 0.9010 \)
- \( \lambda_3 = 0.211 \quad \gamma_3 = 1.0000 \).
Fig. 4. (a) Second iteration of the point set in Fig. 3 under zero cost, \( \beta = 0.4 \) and \( \alpha = 0.02 \) conditions. (b) Fourth iteration of the point set in Fig. 3 under zero cost, \( \beta = 0.4 \) and \( \alpha = 0.02 \) conditions.

Fig. 5. A 25-point 3-dimensional point set as a helix is for Example 3.

Fig. 6. (a) Second iteration of the point set in Fig. 5 under a cost less than 30 units, \( \beta = 0.4 \) and \( \alpha = 0.02 \) conditions. (b) The fifth iteration of the point set in Fig. 5 under a cost less than 30 units, \( \beta = 0.4 \) and \( \alpha = 0.02 \) conditions. (Note the collapse of some points into their neighbors for graphical purposes.)

1) Under the zero cost condition:

- number of iterations = 3
- computer running time = 35 s
- normalized final eigenvalues and values of \( y \) are:
  \[ \begin{align*}
  \lambda_1 &= 1.000 & \gamma_1 &= 0.8836 \\
  \lambda_2 &= 0.123 & \gamma_2 &= 0.9918 \\
  \lambda_3 &= 0.00928 & \gamma_3 &= 1.0000.
  \end{align*} \]

2) Under a cost less than 30 units condition:

- number of iterations = 3
- computer running time = 39 s
- normalized final eigenvalues and values of \( y \) are:
  \[ \begin{align*}
  \lambda_1 &= 1.000 & \gamma_1 &= 0.6584 \\
  \lambda_2 &= 0.500 & \gamma_2 &= 0.9873 \\
  \lambda_3 &= 0.0192 & \gamma_3 &= 1.0000.
  \end{align*} \]
Example 3
A 25-point 3-dimensional helix data set is generated as shown in Fig. 5. The normalized eigenvalues of these data and values of $\gamma_k$ are

$$\begin{align*}
\lambda_1 & = 1.000 & \gamma_1 & = 0.6909 \\
\lambda_2 & = 0.365 & \gamma_2 & = 0.8315 \\
\lambda_3 & = 0.277 & \gamma_3 & = 1.0000.
\end{align*}$$

1) Under the zero cost condition:

- number of iterations = 2
- computer running time = 28 s
- normalized final eigenvalues and values of $\gamma_k$ are
  $$\begin{align*}
  \lambda_1 & = 1.000 & \gamma_1 & = 0.9167 \\
  \lambda_2 & = 0.0906 & \gamma_2 & = 0.9998 \\
  \lambda_3 & = 0.722 \times 10^{-3} & \gamma_3 & = 1.0000.
  \end{align*}$$

2) Under a cost less than 30 units condition:

- number of iterations = 5
- computer running time = 43 s
- normalized final eigenvalues and values of $\gamma_k$ are
  $$\begin{align*}
  \lambda_1 & = 1.000 & \gamma_1 & = 0.9804 \\
  \lambda_2 & = 0.0193 & \gamma_2 & = 0.9993 \\
  \lambda_3 & = 0.743 \times 10^{-3} & \gamma_3 & = 1.0000.
  \end{align*}$$

Fig. 6(a) and (b) show, respectively, the second and fifth iterations of test (2).

Example 4
A 64-point 3-dimensional uniformly deterministic cube is used in this example.

The normalized eigenvalues of these data and values of $\gamma_k$ are

$$\begin{align*}
\lambda_1 & = 1.000 & \gamma_1 & = 0.3333 \\
\lambda_2 & = 1.000 & \gamma_2 & = 0.6667 \\
\lambda_3 & = 1.000 & \gamma_3 & = 1.0000.
\end{align*}$$

1) Under the zero cost condition:

- number of iterations = 3
- computer running time = 98 s
- normalized final eigenvalues and values of $\gamma_k$ are
  $$\begin{align*}
  \lambda_1 & = 1.000 & \gamma_1 & = 0.7795 \\
  \lambda_2 & = 0.144 & \gamma_2 & = 0.8918 \\
  \lambda_3 & = 0.139 & \gamma_3 & = 1.0000.
  \end{align*}$$

2) Under a cost less than 30 units condition:

- number of iterations = 1
- computer running time = 46 s
- normalized final eigenvalues and values of $\gamma_k$ are
  $$\begin{align*}
  \lambda_1 & = 1.000 & \gamma_1 & = 0.3510 \\
  \lambda_2 & = 0.925 & \gamma_2 & = 0.6755 \\
  \lambda_3 & = 0.924 & \gamma_3 & = 1.0000.
  \end{align*}$$

Fig. 7(a) indicates the first stage of the iteration.

Example 5
The same set of data as in Example 4 for the solid cube is tested under the following conditions: $c = 30$ units and using the modified equation for differential element as given in (6b) with $r = 1.5$ and $t = 1.0$. The normalized eigenvalues and values of $\gamma_k$ at the third iteration are

$$\begin{align*}
\lambda_1 & = 1.000 & \gamma_1 & = 0.3779 \\
\lambda_2 & = 0.992 & \gamma_2 & = 0.7529 \\
\lambda_3 & = 0.654 & \gamma_3 & = 1.0000.
\end{align*}$$

The total computing time is 107 s. The corner 8 points have moved much less drastically than before. Fig. 7(b) shows the first and second iteration of the test.

IV. Conclusions
This note presents a modification to the Bennett [9] intrinsic dimensionality computation algorithm through the introduction of a cost function. While increasing variances and maintaining rank orders is the underlying theme, the addition of a cost function fidelity criterion provides stability to the algorithm. The computational considerations are on the order of $N^2 M$ where we have $N$ points imbedded in $M$-dimensional space. Instead of restoring rank order, the cost function guides the movement of the points. Both zero cost and a small cost criteria have been implemented in a program and the results demonstrate that the latter criterion is better than the former in finding intrinsic dimensionality. Three different structural examples are provided for illustration purposes.

References

[8] R. Shepard and J. Carroll, "Parametric representation of non-
A Black-White Representation of a Gray-Scale Picture

T. H. MORRIN

Abstract—A hardware-implemented algorithm combining dynamic thresholding and edge detection allows black-white representation of gray-scale images for computer storage. The output resembles a highly detailed sketch, with resolution and contrast superior to a halftone representation. The same algorithm can be applied to text and line drawings, so that all three types of images can be stored using the same compression algorithm.

Index Terms—Dynamic thresholding, edge detection, image compression, image storage.

INTRODUCTION

For computer-based image systems, conventional methods of representing gray-scale images in a black-white medium present some problems. The dot patterns of halftone reproductions [4], [6], for example, have relatively poor spatial resolution and do not lend themselves to compression by the techniques used to reduce storage requirements for black-white text or graphics.

The proposed approach, combining dynamic thresholding and edge detection, tends to concentrate the pictorial information in regions which the human eye would find interesting (because informative) in the original gray-scale picture. The output looks something like a very highly detailed sketch. This approach has several advantages.

1) The output is superior to halftone representations in spatial resolution (for a given density of elementary points) and contrast. The contrast of the input image may actually be enhanced, although there is a loss of information about background tones.

2) Since the information entropy of the output is relatively low, the gray-scale images can be compressed by the same techniques as are used for storing typed and graphic information. Halftone images cannot readily be compressed with these techniques.

3) The dynamic thresholding algorithm can be implemented as a hardware attachment to a scanning device at relatively low cost. A raster scan at a bit rate of several million bauds can be achieved without parallelizing arithmetic modules.

4) The same algorithm can also be applied directly to text and graphical information with an attendant increase in the output quality of such images. Thus, documents containing text, graphics, and gray-scale images can be stored by the same algorithm with very little loss in performance or in acceptability of output.

THE ALGORITHM

The algorithm to be described is a combination of dynamic thresholding and edge emphasis. The edge emphasis process, described below, creates an effect similar to that of high frequency contrast enhancement processes seen in the literature (e.g., [3]). Thresholding of the enhanced image is accomplished by comparison with an ambient gray level, which is established by computing an average of sample points in the neighborhood of the current sample.

Fig. 1. Black-white decision algorithm. (a) Definition of composite difference. (b) Black-white decision.

The ambient generator which produced the results in Fig. 2 is simply a first-order, digital, low-pass filter which operates on the average (sum) of the picture element being scanned and the one directly above it in the previous line. The previous line is used to improve noise rejection and to allow the detection of edges parallel to the direction of scan. The time constant of the filter is eight pels. The points upon which the filters are operating are ahead of the point upon which the black-white decision is made. This causes transitions toward white to result in a black output “pulse” in the case where the output has been white at immediately preceding points.

Differences in adjacent gray values are taken for edge detection in both the horizontal and vertical directions. Following the notation given in Fig. 1, the intensity of the current sample A, the previous sample B, and the adjacent sample in the previous line C are combined into a composite difference 

\[
D_c = (A - B)/2 + (A - C)/2. 
\]

The domain of this quantity is represented by the X axis of the plot in Fig. 1. The Y axis is the range of the current gray scale A. Each time a value of \( D_c \) is calculated, a point \((X_0, Y_0)\) is defined, where \( X_0 = D_c \) and \( Y_0 = A \).

The linear function shown on the graph \( Y = mX + b \) is characterized by the values of \( m \) and \( b \). The value of \( m \) is fixed at \(-1\), which was determined by computer simulation to be roughly optimum. \( m \) can be considered to be an index of sensitivity to edges. As \( m \) approaches zero, the black-white decision is made based only upon a comparison of the ambient with the current gray value. As the magnitude of \( m \) becomes large, the decision is based mainly on the value of the composite difference obtained at each point in the image. For large \( m \), the process becomes very noise sensitive and the output begins to assume an appearance somewhat similar to a halftone image of poor quality.

The value of \( b \) is determined by the ambient gray level described above. The line on the graph “floats” up and down as the ambient changes. There is a slight offset in \( b \) to bias the decision toward white, since more of the image is expected to be white. This offset is tapered so that it is larger at the white end of the range and goes to zero at the black end. There are two reasons for doing this. 1) Areas of white in an image tend to be noisy due to paper fibers, film granularity, etc.