Palo Alto Research Center

The Display of Characters Using Gray Level Sample Arrays

By John E. Warnock

XEROX

The Display of Characters Using Gray Level Sample Arrays

John E. Warnock

CSL-80-6 May 1980

Abstract

Character fonts on raster scanned display devices are usually represented by arrays of bits that are displayed as a matrix of black and white dots. This paper reviews a filtering and sampling method as applied to characters for building multiple bit per pixel arrays. These arrays can be used as alternative character representations for use on devices with gray scale capability. Discussed in this paper are both the filtering algorithms that are used to generate gray scale fonts and some consequences of using gray levels for the representation of fonts including:

1. The apparent resolution of the display is increased when using gray scale fonts allowing smaller fonts to be used with higher apparent positional accuracy and readability. This is especially important when using low resolution displays.

2. Fonts of any size and orientation can be generated automatically from suitable high precision representations. This automatic generation removes the tedious process of "bit tuning" fonts for a given display.

CR Categories: Categories: 8.2

Key words and phrases: computer graphics, fonts, gray-scale, raster displays, shading, anti-aliasing, convolution, character representation

XEROX
PALO ALTO RESEARCH CENTER
3333 Coyote Hill Road / Palo Alto / California 94304

Introduction

The motivation behind the work presented in this paper comes from research into the problems of building an interactive display system that allows users to design high quality page layouts much like those found in advertising copy. To achieve this goal, the problem of displaying a wide variety of different character fonts on an interactive display must be solved. It is important for this application that the characters on the interactive display reflect the shapes and styles that are to be in the final printed copy. The aesthetic judgments demanded to lay out a high quality page impose these requirements on such a system.

In most other text display applications where raster displays are used, each character in a font is represented as a matrix of binary values, representing white/black dots on the display. Such applications only require that the character font be legible. For this reason most text display systems provide only one character font, and this font is simple and tuned to the characteristics of the display. However, when applications arise that relate to the printing industry, then the standards regarding the printed word change dramatically. It is common, for instance, for a single advertisement to contain a dozen fonts having different sizes, weights, and styles. A technical article may contain as many as thirty different variations of fonts. In these applications the appearance of the characters, their layout, and their spacing are all important.

Systems that generate the bit matrix representations of character fonts automatically from higher level representations fall into three classes.

1. *analytic* -- Knuth's Metafont [4] system falls into this class. It allows for the inclusion of information about the strokes that make up each character so that variations on a font may be generated from a common source by applying suitable transformations to each stroke.

2. *high resolution bitmaps* -- Character masters are scanned at high precision, and these images are cut off at a threshold to obtain bit matrices at the desired resolution.

3. *parametric curved outlines* -- The outline around each character defined by a parametric function is scan converted to produce high precision bit matrices.

All of the above approaches work quite well when high precision character matrices are required. But none of the above strategies are satisfactory at low resolutions. The low resolution character matrices produced by the above schemes amount to images that are undersampled and therefore are not good representations of the information in the character. Because of the lack of an automatic scheme, the bit matrices for low resolution fonts usually are constructed, very tediously, by hand. This manual process involves an artist sitting at a console, turning individual pixels on or off in order to assess the aesthetic appearance of the resulting character on the display. If wide ranges of font sizes and rotations are needed then this latter approach is impractical.

This paper discusses a methodology for building character sets for use with low resolution raster displays that have gray-scale capabilities. There are three important goals:

1. The characters generated must have the same general appearance as the masters from which they are made even though fonts are generated for a low resolution device.

2. The characters must be free of sampling artifacts. They must not have unnatural holes or dark spots, nor should they be unnecessarily blurry.

3. It should be possible to make the characters in a line of text look properly spaced. Positioning errors due to inadequate display resolution should be avoidable.

The basic strategy described here is an old technique that is straightforward, automatic, and achieves the above goals. The technique proceeds as follows: characters represented by high precision black and white bit matrices, obtained by one of the above schemes, are filtered, using a low-pass filter. The resulting low frequency image is sampled and displayed. This filtering and resampling process is very common in the image processing field, and is used extensively to scale down images. The application of building grey encoded characters with this, and other techniques has been previously accomplished by Seitz [6] and Crow [2]. This paper reviews the general application of this method to automatic character font generation, and gives some additional results on the effects of subpixel positioning of characters that are new. The images included in this paper give an indication of how well this scheme works for various filter variations. Also a number of the side benefits of the scheme are discussed.

Input Character Masters

The character masters used in this work are 100x100 bit arrays. They are obtained by scan converting curved outline representations of each character. The particular process used to build these high precision master characters is not critical to the techniques presented in this paper. Figure 1 shows an example of a 100x100 bit matrix of the character "&".

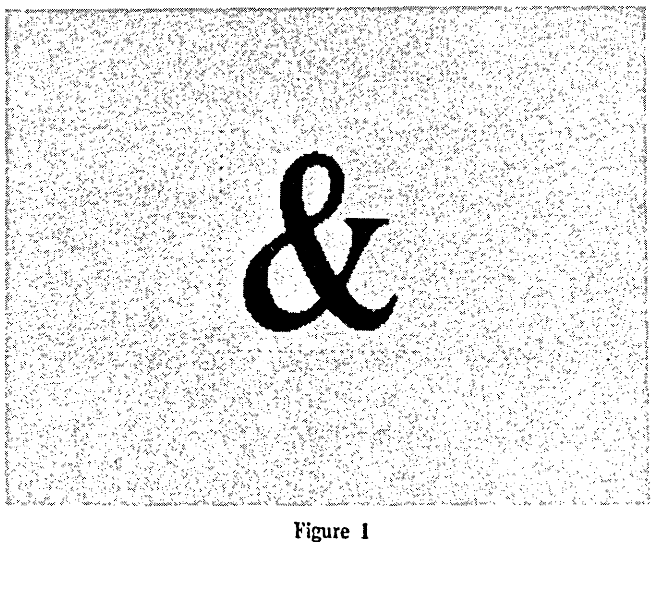


Figure 1

Gray Values

The mapping function between computed gray values and the intensities actually shown on the display must be carefully controlled in order to remove non-linearities introduced the display hardware. This problem has been addressed by Catmull [1], and the techniques suggested in his paper should be followed. For the purposes of the illustrations in this paper, gray scale wedges are provided in each image in order to calibrate the gray values used.

Filtering Process

The basic strategy in building gray encoded fonts is to take a high resolution, black and white image (a high precision bitmap in this case), to filter the image, and then to resample to produce the low resolution character.

The way in which this is achieved in practice is quite simple and intuitive. Consider figure 2 which shows the high precision character "a". The computation that is performed simulates an idealized sampling camera that is pointed at the character. As this camera scans the image, it looks at overlapping sampling areas of the image. These sampling areas correspond to the pixels that are shown on the display. The value that each pixel receives is a function of the black and white subareas within the sampling area. In particular, if only white is within the sampling area, then a white pixel is produced. If only black is within the sampling area, then a black pixel is produced. If a combination of black and white are in the sampling area, then a shade of gray is produced.

One kind of area averaging scheme that can be used for producing the gray values in this simulation consists of taking the ratio of the black subarea to the total area and using this ratio to determine the gray value. Another approach is to weight each point of the area as some function of the distance of the point from the center of the sample area. With this latter weighting function, black subareas of the master character near the center of the pixel may contribute more to the *blackness* of the pixel than do black areas near the edge of the pixel.

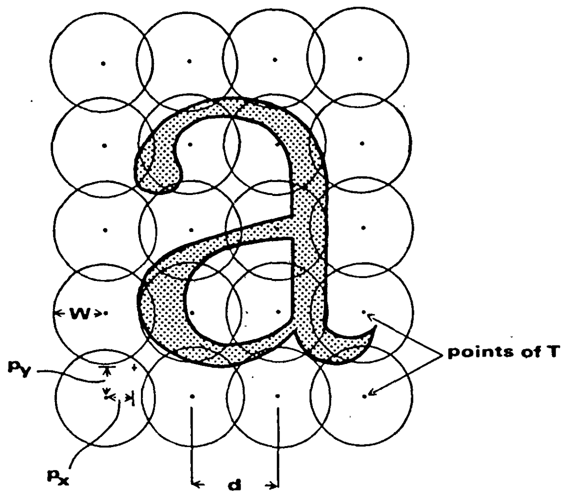


Figure 2

Three families of weighting functions (filters) are used in this work. First generalized 2-dimensional Hamming filters [7] are used. The Hamming filters used are of the form:

$$f(x,y) = S\,\mathrm{Cos}^{e}(\pi\ \mathrm{sqrt}(x^2+y^2)/W) \quad \text{for } \mathrm{sqrt}(x^2+y^2) < W/2$$
$$f(x,y) = 0 \quad \text{for } \mathrm{sqrt}(x^2+y^2)) > W/2 \qquad (1)$$

Another family of filters used in this work are of the form:

$$f(x,y) = S\,(\mathrm{Cos}((\pi\ x)/W)\mathrm{Cos}((\pi\ y)/W))^{e} \text{ for } -W/2 < x,y < W/2$$
$$f(x,y) = 0 \quad \text{for } x,y > W/2 \text{ or } x,y < -W/2 \qquad (2)$$

The third family of filters used are bi-linear and are of the form:

$$f(x,y) = S\ g(x)g(y) \text{ where}$$
$$g(t) = -(4/(W^2))t + 2/W \text{ for } t >= 0 \text{ and} \qquad (3)$$
$$g(t) = (4/(W^2))t + 2/W \text{ for } t < 0$$
$$g(t) = 0 \text{ for } t < -W/2 \text{ or } t > W/2$$

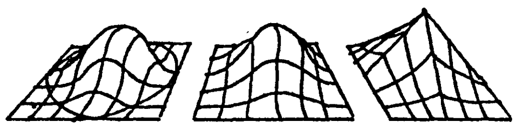
In these formulas W, e, and S are constant parameters that determine which member of the family is used.

W -- the width of the filter;

e -- controls the relative weighting of the filter near the origin. (Large values of e weight values near the origin more heavily. Note that $e = 0$ provides for the simple averaging case in the first two families.);

S -- chosen so that the sum of all the elements in an array of filter values is a desired maximum intensity I. In this work, because of display controller limitations, only 16 intensity values are used ranging from 0 to 15).

The shapes of examples of the filters in the three families are illustrated in figure 3.



1 2 3

Figure 3

All the resulting character fonts are scaled down versions of the master characters. It is assumed that the master characters' resolution is high relative to the resolution of the fonts that are produced. To make a character of a given size, the following steps are taken:

1. Given that the high precision character is represented as an nxn matrix, and given that we wish to make a character of approximately mxm pixels where $m \ll n$, we let $d = n/m$. This spacing, d, is the increment used between pixels relative to the master character.

2. Next, chose a filter width W appropriate to the sampling increment. In this case, filter widths between $.5d$ and $2.5d$ give an adequate range of results for the purposes of this work. In the camera simulation metaphor, this width corresponds to the diameter of the sampling area of a single pixel. It should be pointed out that with these families of filter functions, the wider filters are lower pass filters and tend to blur the characters. The narrower filters tend to undersample, leaving unwanted gaps in the characters.

3. The next step is to choose e (if using one of the first two filter families.) The value of e determines the shape of the filter function and therefore the relative weights assigned to the filter elements. Lower values of e make lower pass filters, and therefore tend to blur the characters. It should be noted that the values of e interact with W in that large values of e tend to cancel the effect of making W large. The values of e used in this work are between 0 and 4. This range of values brackets the usable range of results.

4. In this step the weighting array (filter), a WxW array, F, is constructed which is centered on the origin, and which has values at each array position that correspond to the value of the filter function at that position.

5. Now to compute the weighted average (gray value) for a given x,y position relative to the character matrix C, F is centered at x,y, and convolved with C to yield r_{xy}, i.e.,

$$r_{xy} = \text{sum}(F(W/2-i, W/2-j)C(x-i, y-j))$$
$$\text{for } \text{Max}(-W/2, -x) < i < \text{Min}(n-x, \ W/2)$$
$$\text{and } \text{Max}(-W/2, -y) < j < \text{Min}(n-y,$$

   Here $C(i,j) = 0$ or 1 for all i,j.

6. The last step is to select a sample grid, T, where the points of the grid are d units apart and where the starting phase of the grid is picked to be p_x, p_y $(-d < p_x, p_y < d)$ (see figure 2). The sample grid T covers C so that pixels that could have sampling areas that intersect C are included in T. For this reason T may be larger than mxm. Finally, the gray array, R, is computed on the grid points of T. The gray array, R, is the gray scale representation of the character.

Storage and Selection of the Gray Scale Fonts

Once a set of characters has been generated, their gray matrices are cached in memory, and used out of this cache as needed for storage into a frame buffer or for output directly to a display. If subpixel placement of characters is desired then multiple gray matrices are made for each character. These matrices differ from one another in that the phase (p_x, p_y) of T relative to C is different. For example, if 1/4 pixel subpositioning is desired in x, then 4 versions of each character are made. In this case p_x would be 0, $d/4$, $d/2$, and $3d/4$ respectively. These multiple representations are then selected with the 2 low order bits of the x position. The display of these different versions gives the illusion of subpixel shifting of the character.

Storage of a character into a frame buffer must take into consideration the values already in the buffer. This is especially true when two characters are spatially close together. Because small characters may not have a full white pixel between them, gray scale character arrays will overlap one another. In this case the overlapping gray values must be combined in order to produce an acceptable visual appearance where they overlap. In this work, the gray values are added so that two gray values will combine to produce a darker gray. If the resulting summed value is out of the prescribed range of gray values, then the maximum value is used. In this work, 0 is interpreted as white and 15 is interpreted as black, and overflow results in 15 being used. This strategy roughly approximates the results that are obtained if the image of two adjacent characters is filtered directly. The approximation strategy can probably be improved upon with more sophisticated combination algorithms, but the good results obtained with simple addition seem to justify the simplicity of the approach.

The Subjective Assessment of the Results

Because of the aesthetic judgments that are made when viewing text, and because of the variations in display and film parameters, very few absolute statements can be made about the results. Instead, somewhat conservative observations will be made, and the reader can form a partial judgment of the results from the photographs provided.

1. The legibility of various sizes of gray scale Timesroman fonts have been assessed. The size of the fonts is measured as the number of raster lines (pixels) required for the height of an upper case "X". At font sizes of 5 pixels and less, most individual characters cease to be recognizable, but most words can still be read (mostly by context and shape). At a font size of 6, individual characters are all recognizable (with some difficulty). At a font size of 7 text is easily readable, and the font style is easily recognizable as Timesroman. At a font size of 10, the character quality is good -- the variable thicknesses of the strokes start showing, and the serifs are all visible. At font sizes of 12 and larger, the character quality is very good in that the roundness of the circular strokes is faithfully represented. Figure 4

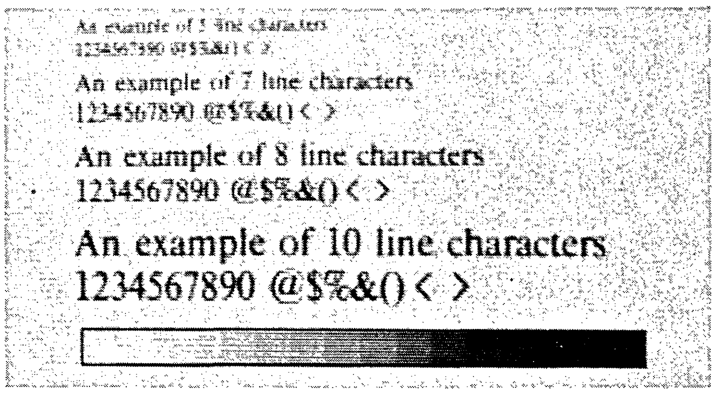

An example of 5 line characters
1234567890 @$%&() < >
An example of 7 line characters
1234567890 @$%&() < >
An example of 8 line characters
1234567890 @$%&() < >
An example of 10 line characters
1234567890 @$%&() < >

Figure 4

gives an indication of these results but the photographic and printing process used in making the figure has masked what can be seen on the display.

2. A comparison has been made between a black and white font that is directly sampled from the masters, and a gray scale font that is sampled from the filtered masters. For sizes from 4 pixel high fonts, to the size of the masters (100 pixels), the gray scale fonts are significantly more readable, more consistently textured, and more faithful to the masters, than are the directly sampled fonts. Figure 5 shows this comparison for a Timesroman font that is 8 pixels high.

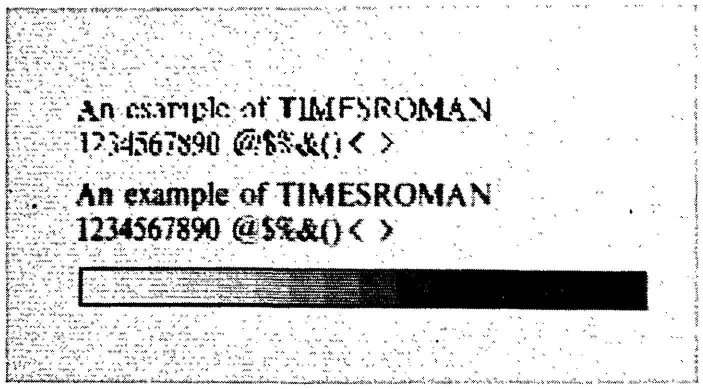

An csample of TIMFSROMAN
1?34567890 @$%&()< >
An example of TIMESROMAN
1234567890 @$%&()< >

Figure 5

3. Various weighting (filter) functions have been tested. In particular, various values of *W* and *e* (in the filter formula (1) given above) have been tested on a 8 pixel Timesroman font. Figure 6 shows the results of this test as applied to the character "a". The columns represent variations in *W* whereas the rows represent variations in *e*. The columns represent

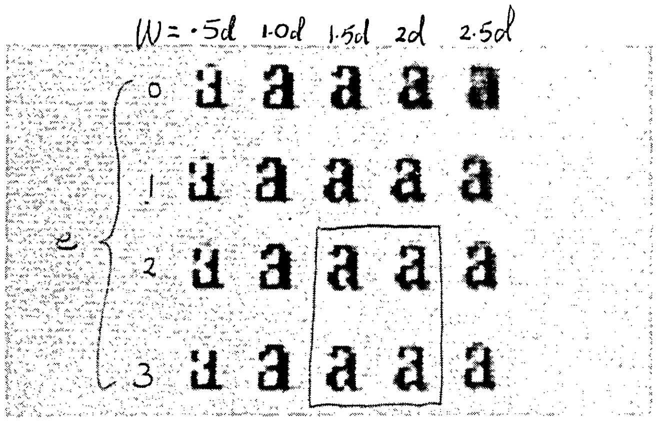
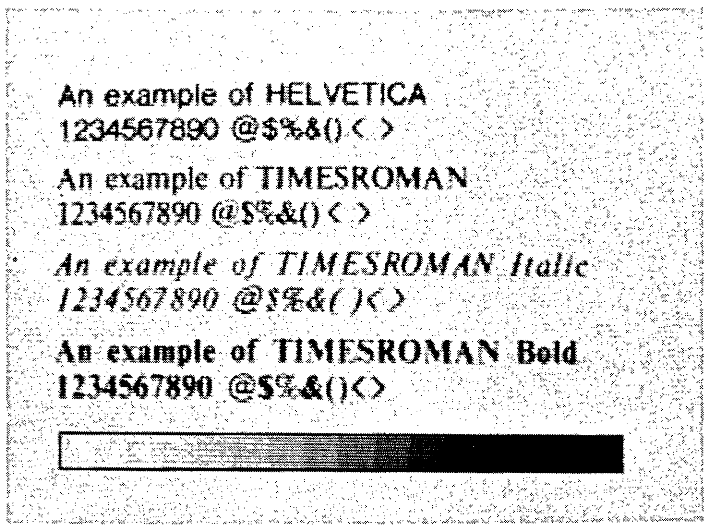


Figure 6

values $W = .5d$, d, $1.5d$, $2d$, and $2.5d$ respectively. The rows represent values $e = 0, 1, 2,$ and 3 respectively. It has been found that values of W in the range $1.5d$ to $2d$, and values of e in the range 2 to 3 give good results for all font sizes tested.

The other two families of filters (2) and (3) have been used for generating characters. From the results it has been found that the differences between corresponding members of different families is very subtle indeed. No preference judgment of one filter family over another could be made from this set of experiments.

4. Different master fonts and styles have been converted to gray scale fonts. In particular Helvetica, Timesroman, Timesroman Italic, and Timesroman Bold have been converted to gray scale fonts. Figure 7 shows the results of this conversion.



An example of HELVETICA
1234567890 @$%&()< >
An example of TIMESROMAN
1234567890 @$%&()< >
An example of TIMESROMAN Italic
1234567890 @$%&( )< >
An example of TIMESROMAN Bold
1234567890 @$%&()<>

Figure 7

2 bits
00 white
01 g_1
10 g_2 $\}$ $g_1 \vee g_2 = g_1 + g_2 =$ black
11 black

?

5. Different ranges of gray values have been tried. In particular fonts have been built with 2, 3 and 4 bits per pixel. Although no photographs are presented, it has been found that 2 bits per pixel does not provide sufficient gray scale continuity to provide the desired illusion required for making fonts automatically. Three bits per pixel were found to be adequate with four bits being slightly better.

- depends on how fussy you want to be.
- standard TV tends to smear the gray levels, so must be careful

6. The gray scale character "&" has been blown up in order to investigate the gray values. The reader can see the results in figure 8. The line of "&"s below the blown up version indicates how this character appears on the display.

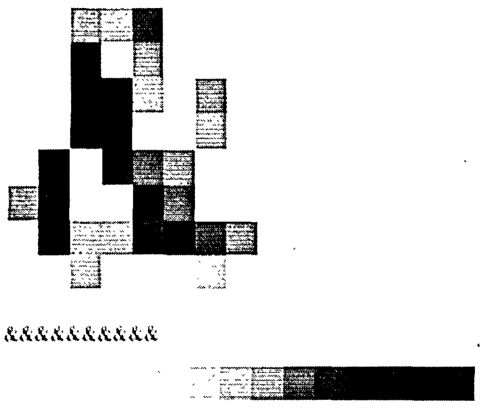


Figure 8

7. Rotated characters have been created and displayed. Figure 9 shows various fonts rotated through 20 degrees. Note that because of the subpixel positioning of the characters, no raster line positioning artifacts in the base line are present.

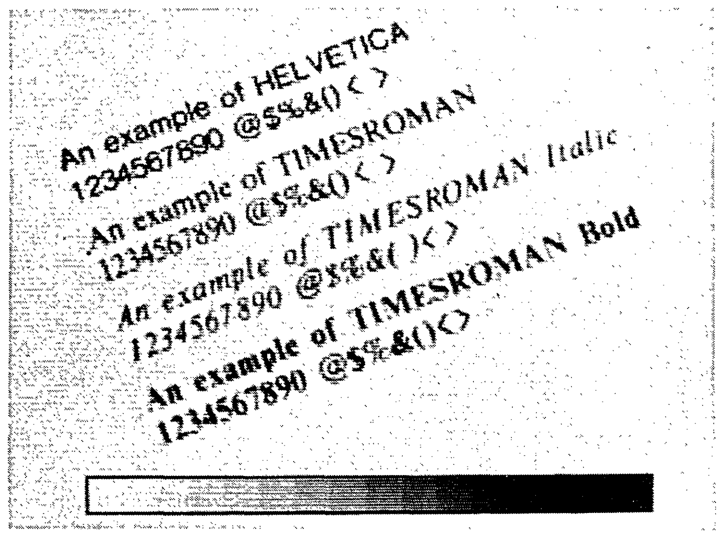


Figure 9

Summary

This work reviews a traditional methodology as it is applied to character generation. This methodology is effective for generating a wide variety of fonts of different sizes automatically from high precision masters. These fonts can then be used with low resolution gray scale raster displays. Advantages of this technique include:

1. Apparent subpixel resolution can be attained by building multiple versions of each character. These versions differ from each other only by the phase of the sampling grid relative to the master character. This technique allows characters to be positioned and spaced with more apparent accuracy.

2. Very small fonts (less than 6 pixels high) can be generated and used. Although these characters are not directly legible, they are useful for thumbnail layouts and for assessing the *feel* of a page. These small fonts do reflect the texture and densities of the various fonts they represent.

3. Rotated fonts can be generated automatically. These rotated fonts can be used when text is not to be constrained to horizontal lines.

Bibliography

1. Catmull, Edwin. A Tutorial on Compensation Tables, *Quarterly Report of SIGGRAPH-ACM,* Vol. 13, 2, August 1979. pp. 1-7.

2. Crow, Frank C. The Aliasing Problem in Computer Generated Shaded Images, *Communications of the ACM,* Vol. 20, 11, November 1977 pp. 799-805.

3. Gonzalez, Rafael C. and Wintz, Paul. Digital Image Processing, Addison-Wesley Publishing Company, Inc., London, 1977.

4. Knuth, Donald E. METAFONT, A System for Character Shaping, Stanford Artificial Intelligence Laboratory, *Report No. STAN-CS-79-000,* 1979.

5. Pearson, D. E. Transmission and Display of Pictorial Information, John Wiley & Sons, New York, 1975.

XEROX

XEROX

The Display of Characters Using Gray Level Sample Arrays

by John E. Warnock

CSL-80-6