A DCT image fidelity metric and its application to a text-based scheme for image display

D. Amnon Silverstein and Stanley A. Klein

School of Optometry, University of California
Berkeley, CA, USA

ABSTRACT

The discrete cosine transform (DCT) can be used to transform two images into a space where it is easy to obtain an estimate of their perceptual distance. We used this method to find the closest fit of the ASCII symbols (which includes the English alphabet, numbers, punctuation and common symbols) to rectangular segments of a gray-scale image. Each segment was converted into a DCT coefficient matrix which was compared to the coefficient matrix of each ASCII symbol. The image segment was replaced with the symbol that had the least weighted Euclidean distance. Thus, a page of text was generated that resembled the original image.

The text image format has the advantage that it can be displayed on a non-graphic terminal or printer. It can also be sent via electronic mail without requiring further processing by the receiver. The processing scheme can also be used to preview stored images when transmission bandwidth is limited or a graphic output device is unavailable.

1. INTRODUCTION

Perhaps almost as long as computers have had text output, people have been trying to use text to draw pictures. Not too long ago, this was the only way for most of us to get graphical output. Surprisingly, despite the numerous standards for storing images and the widespread availability of graphical output devices, text-based drawings are still popular. Text images have several advantages that keep them in use despite their generally low quality. Foremost, they require the fewest assumptions about the receiver. Virtually all networked computer printers and terminals can display alphanumeric information encoded in the American Standard Code for Information Interchange (ASCII), a standard which includes symbols similar to the ones found on a typewriter. Images sent in ASCII require no decoding, which makes them convenient for electronic mail, Internet news and bulletin board services. Images drawn with symbols generally have very modest storage requirements, which further facilitates their electronic dissemination.

Historically, there have been several popular ways for creating text images. Mostly, they are hand drawn, symbol by symbol, as shown by the bicyclist in figure 1. The computer can also be used to assist in the choice of symbols for image generation. For example, UNIX includes a standard program called "Banner" that uses symbols as pixels to draw large scale letters.

It is also possible to convert gray scale images, such as the images shown in figures 3 and 4, into alphanumerics[1]. In the past, a set of symbols were chosen for use as pixels of varying intensity. Symbols were chosen for their symmetry and then assigned estimated gray values. A sampled image could then be converted to an alphanumeric picture by outputing the symbol with the closest gray value to each image sample. This method generates images with very low fidelity. Still, this technique has been in use for years. The purpose of our study was to improve image matching by using what we know about vision. Specifically, given an image block, how do we find the symbol that will provide the match with the highest perceptual fidelity? To do this we need to know the specifics about the set of symbols we are using. To get started, we simply choose to use a rasterized version of Courier bold. We will discuss the problems with font choice later. We also decided to use all of the symbols available in the ASCII standard so the images we produced could be sent through electronic mail or by other common means.

```
   __e
 _ \<,_
(_)/ (_)
```

Figure 1.

A bicycle and rider drawn with symbols. Drawings like this one are often sent through electronic mail.

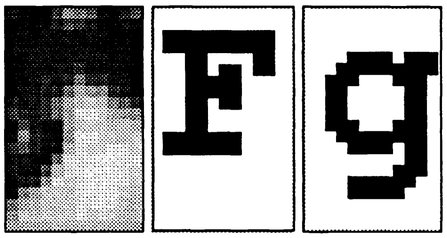


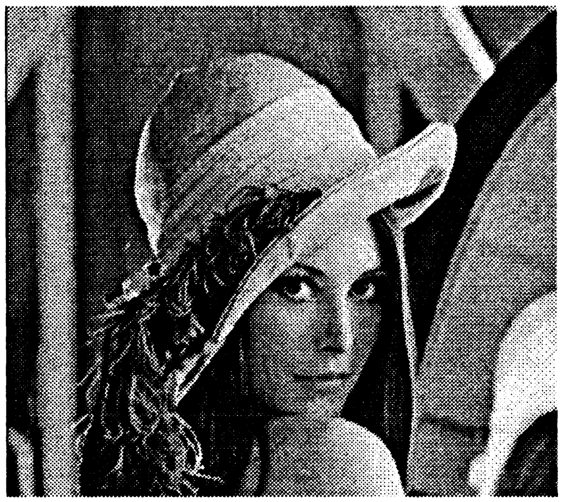
Figure 2.

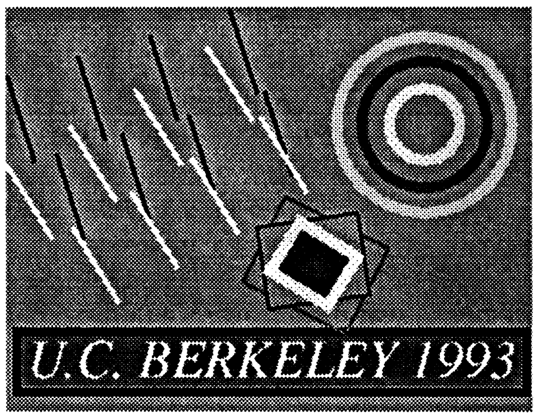
An image block is shown on the left. On the right are two possible choices for symbols to match this block. The symbol chosen should not only be a good match to the average luminance of the block, but it should also be a good spatial match.

2. MATCHING IMAGE BLOCKS WITH SYMBOLS

The problem this paper addresses is shown in figure 2. Given a block of an image (in the example, a 12 x 20 pixel block from figure 3), and a choice of symbols to replace the image block (also rasterized to 12 x 20 pixels), how do we choose the best one? In the past, this has been done by assigning each image sample a symbol chosen to be a good match for the average intensity, or "DC" level, across that sample. It is possible to find a better match if the spatial distribution inside of the image block is considered. In this case, "g" may be the best DC match, but "F" is the better choice because it fits the spatial distribution better. We would like to find the highest fidelity match of symbols to gray scale images such as the ones shown in figures 3 and 4.

To find a match over a range of spatial frequencies, it is often easier to work in frequency space. Typically, images are converted to frequency space by means of a discrete cosine transform (DCT)[2]. The DCT has a number of advantages that make it popular for image processing. It is spatially local, discrete and easy to compute quickly[3]. The DCT also reduces blocking artifacts[4]. These advantages are not overwhelming and other schemes, such as the filtered mean square error or the discrete Fourier transform are probably equivalent for this necessarily crude match finding.





Figures 3 and 4.

Figure 3 shows the original "Lena" image used in later examples. Figure 4 is a test pattern with a variety of resolution tests including vernier, edge and letter patterns. Simple figures like the ones in figure 4 are often better at showing errors in processing schemes because they have little spatial masking and they tend to isolate single types of distortions.

The DCT transforms a matrix of spatial image samples into a matrix of orthogonal frequency samples. The elements of the matrix start with a value for the DC of the whole block, then progress to higher order AC components of that image block. The first few components are the "up-downness" of the block, the "left-rightness" and "criss-crossedness" of the block.

After the image block and the letters have been represented in this space, we return to the problem of finding a good fit. First we should ask, is it possible to find a good fit? figure 5a shows a scatter plot of the symbols. The horizontal axis shows the average luminance (DC value) over the entire block containing each symbol. The vertical axis shows the first vertical AC component, the "up-downness" of the symbols. The bottom of figure 5 is where a symbol that was all black on the bottom half and all white on the top half would go. The top is where a symbol that was all black on the top and all white on the bottom would go. The darkest symbol, the "M", is not very far to the left because it is made of mostly white space. The underscore is the lowest symbol, because it has all of its ink at the bottom of its area. Still, it is not all the way at the bottom of the graph because it is possible to have a symbol with more ink below the midline.

The Euclidean distance between symbols on this graph is an indicator of their perceptual similarity with the following caveats. First, the two dimensions are not equal in size. High frequency differences are less perceptible than low frequency differences[4], and so the two dimensions need to be scaled by a weighting function. But since a number of factors, such as the viewing conditions, the font and the display contrast will influence these dimensions differently, the exact weights cannot be determined without knowing these specifics. Further, edges and textures in the image may mask or facilitate the perception of errors in these two dimensions. A letter may seem to be a bad match to an image block because of a large error in one of the spatial frequency dimensions. However, it may be the case that this large error is masked by a region of texture in the image. Conversely, errors that may seem small may lead to false contours or Moiré patterns.

Despite these problems, image processing schemes such as JPEG[6] (Joint Photographic Experts Group standard for image compression) have used distance in frequency space as a measure of perceptible distance with great success. In JPEG, values of frequency components are quantized discrete levels. The number of levels is chosen so the difference between each level is close to the threshold for discrimination. To measure the perceptual distance between two image blocks in this study, we summed the weighted squares of the differences in each frequency dimension, thus obtaining the Euclidean distance as our metric of distance in perceived fidelity.

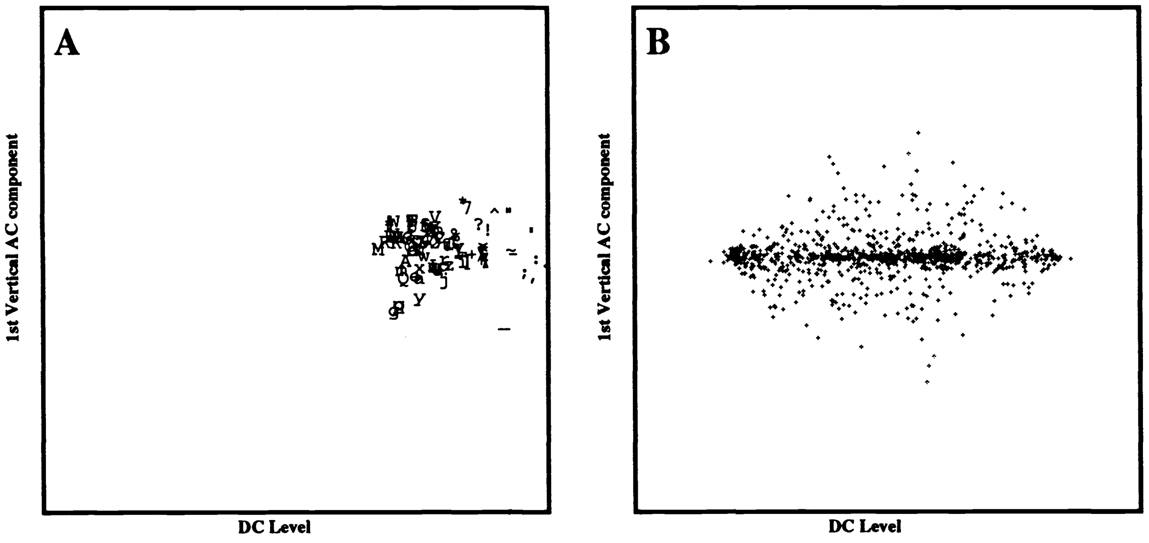
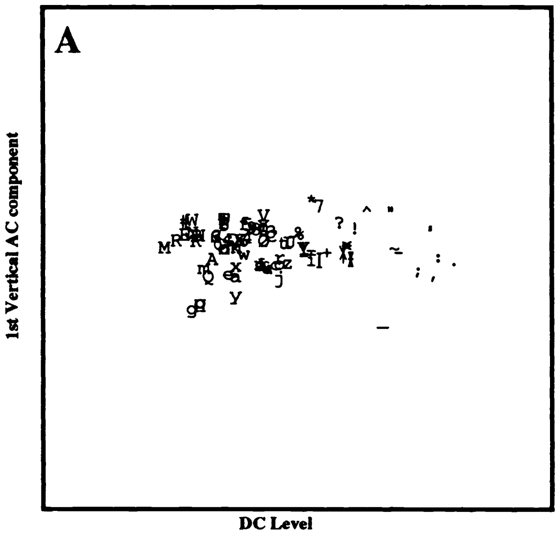


Figure 5.

Figure A shows the Courier font ASCII symbols' distribution in two of the DCT dimensions. Darker symbols are on the left and lighter symbols are on the right. Symbols with most of the ink at the top are higher on the vertical axis and symbols with most of the ink at the bottom are lower on the vertical axis. Note that the letters with decenders (such as "g" and "y") are very low and the underscore symbol is the lowest. The darkest symbol, "m," still consists of mostly white space, and so is not very far to the left. Figure B shows the image blocks from "Lena" on the same scale.

Figure 5b shows the coordinates of all of the image blocks that it takes to make up the Lena image shown in figure 3. An immediately apparent problem is the uneven coverage of the space by the symbols in figure 5a. The symbols are all clustered to the bright side of the graph because the darkest symbol, "M," is not very close to total black at all. Since all of the symbols are clumped together on the DC axis, any image made out of these symbols will have low contrast. If we make the Lena image out of these symbols, it will also be too bright since all of our symbols are on the bright end.



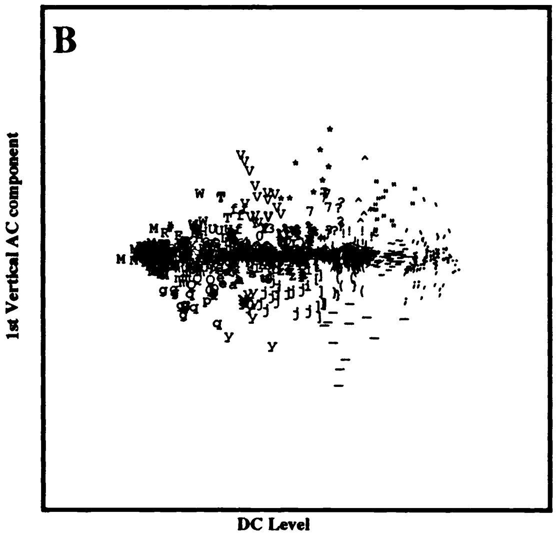


Figure 6.

Figure A shows the same symbols from figure 5a after they have been normalized by adjusting the contrast and brightness to form a least squares fit with "Lena". The symbols' positions on the graph have been stretched in the horizontally by changing the contrast and they have been translated horizontally by changing the brightness. Figure B shows the symbols that most closely match the image blocks from figure 5b.

Fortunately, changing the contrast and brightness of an image does not severely degrade its perceptual fidelity. What we would like to preserve is a good range of levels. To do this, we have allowed contrast and brightness as free parameters to finding the best fit of the symbols to the image. These two parameters were adjusted to find the least square fit of the symbols to the Lena image. Figure 6a shows the symbols after they have been appropriately scaled. The symbols are in the same relative positions as in figure 5a except they have been spread apart by adjusting the contrast and they have been translated to the left by adjusting the brightness.

After this scaling, we can choose a best match by finding the closest symbol in this multidimensional frequency space by using the Euclidean distance as an estimator. To find the perceptually closest match these dimensions need to be scaled by a set of weights, but for now we have just used equal weights. Figure 6b shows the symbol chosen for each block from the Lena image in figure 3.

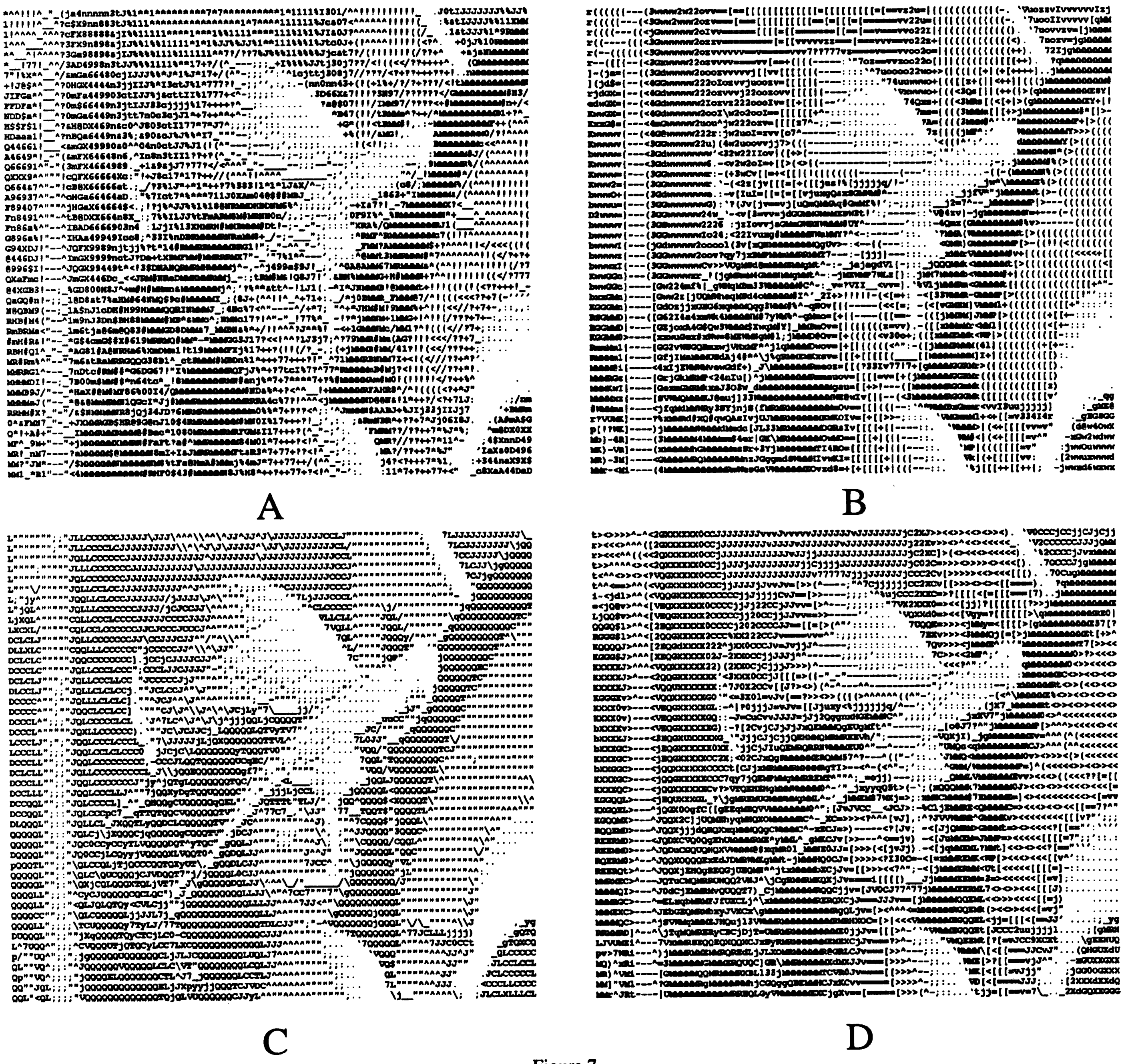


Figure 7.

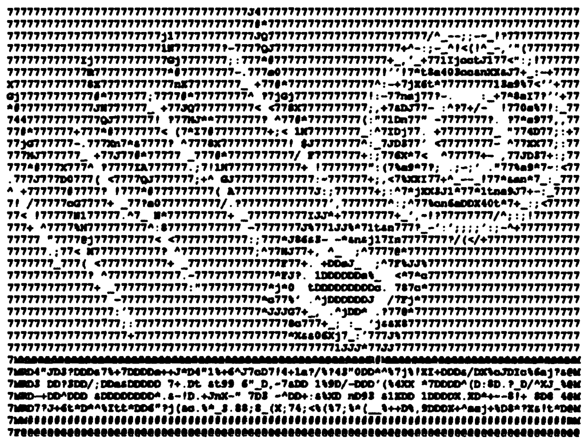
Figure A shows the image from figure 3 after it has been matched with Courier Bold symbols by using only the DC value of each block. Figure B shows the match obtained by giving the first-order AC components equal weights to the DC component. Figure C shows the match when the second-order AC components are also given an equal weight. Figure D shows the match obtained with an unequal weighting matrix. The first-order AC components were given half the weight of the DC component and the second-order AC components were given a quarter the weight of the DC component.

3. RESULTS

We can give the DC component a weight of one and all higher order frequencies a weight of zero to find the best DC match to each image block. Figure 7a shows the best DC match for Lena. All of the high frequency information in the image has been discarded by this matching. Low contrast edges, such as the hat band, are almost invisable. Further, spurious high frequency information from the symbols gives the image a grainy appearance. Figure 7b is the match obtained when the three first-order AC components of the DCT are given equal weight in choosing the match to the DC components. This image is substantially improved over figure 7a. Low contrast details are much more sharp and clear than in the DC match. The high frequency noise introduced by the DC match is also substantially diminished in this match.

There are some trade-offs to including the AC components in the match. Figure 7c shows the match when the five second-order AC components are included in the match. The DC, first-order, and second-order components are all given equal weight. Some symbols that are good matches in the AC dimensions are bad matches in the DC. This can be seen by the black and white blobs around the right eye and in other places on the image.

Figure 7d shows that the high frequency information can be included, but deemphasised, in the match by using a weighting function that gives the DC and low order components more weight in the match than the high order components. In this figure, the first-order AC components are given half of the weight as the DC component and the second-order AC components are given a quarter the weight. This image preserves the DC and matches the AC whenever possible.


A

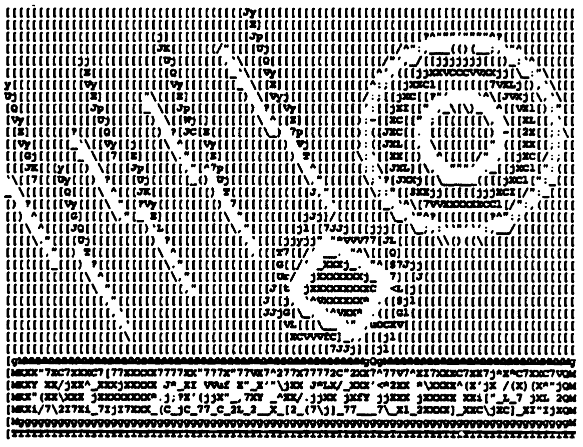

B

Figure 8.
Simple test patterns such as figure 3 are often better at showing errors introduced by processing. Figure A shows the best DC match to figure 3 and figure B shows the best AC match using a matrix that weights the first-order component match half as much as the DC component match, and weights the second-order component match one quarter as much as the DC component match.

Figure 8a and 8b show some of the advantages of using the high frequency components more clearly. Figure 4 was used as the original image. It was designed to have a variety of resolution tests. Simple figures like these are often better at showing errors in processing schemes because they have little spatial masking[6]. Figure 8a shows the best DC fit and figure 8b shows the best AC fit using a weighting matrix like the one used for figure 7d. Notice that the lines are thinner and the edges are sharper when the AC components are included. By matching only the DC component and then quantizing to a very limited number of levels, low contrast high frequency areas of the image are completely lost. This can be seen most clearly in the dark circle in the upper right corner of the two images. In the DC match, the dark circle is hardly visible but in the AC match it is sharp and clear. High contrast high frequency areas are also improved by including the AC in the match. The high contrast white circle is spread out in the DC match as are the letters. The AC match makes the circle appear rounder and sharper and it makes the letters nearly legible.

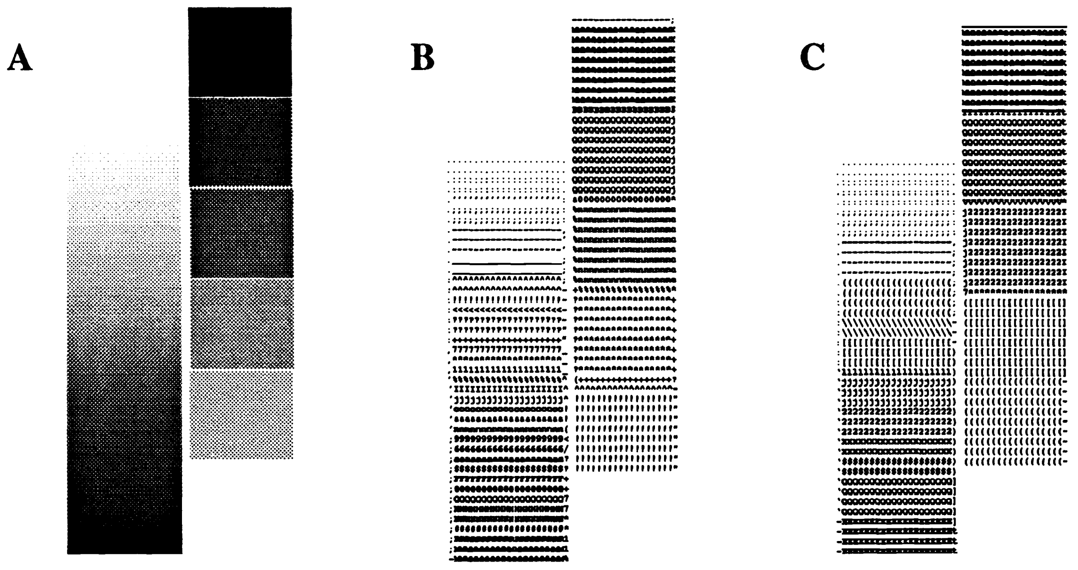


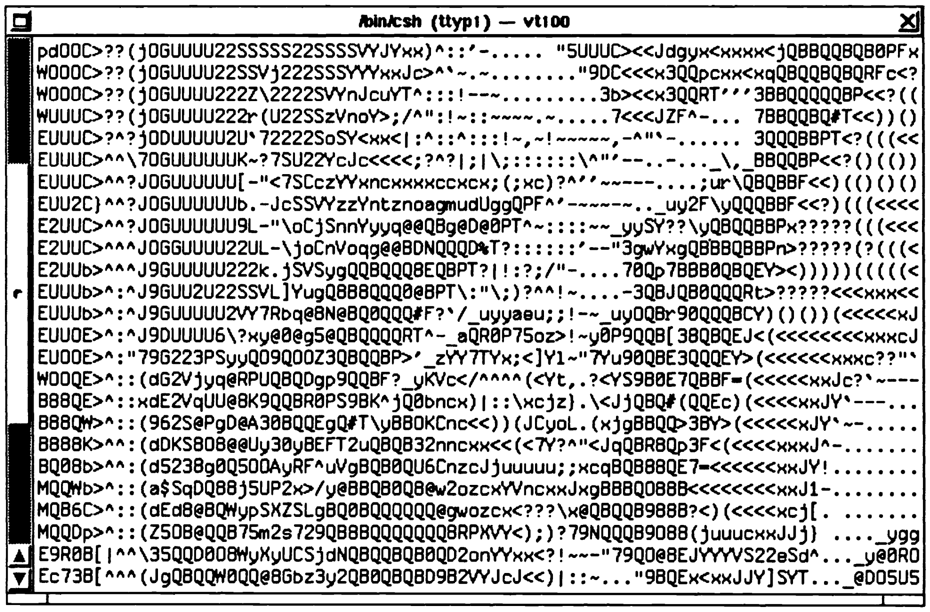
Figure 9.

The gray-scale "wedge" and squares in figure A are shown after being processed into symbols. Figure B shows the symbols chosen by just the DC fit. Figure C shows the fit when the higher order frequencies are included in the decision. The inclusion of the higher order frequencies removes some of the spurious high frequency noise that can be seen in figure B.

Figure 9 shows that in addition to preserving high frequency details, the AC fit provides a superior match to the very low frequencies as well. The gray wedge in figure 9a is matched using the best DC fit in figure 9b. Spurious high frequency patterns make the wedge appear less smooth than the wedge in figure 8c, which includes the higher frequency components.

4. APPLICATIONS

By including the high frequencies in the fit, we can find a superior match to the best DC fit. Further, this match is far superior to the popular method of using just a few symbols to form a gray-scale. But is this match sufficiently good for any purpose? Given enough symbols, the symbol match image can have any degree of fidelity to the original with the exception of the problems of brightness and contrast already discussed. But more typically, the number of symbols is more restricted. The smallest symbol array generally available on a computer display is 24 rows by 80 columns. Figure 10 shows a screen from a terminal in this configuration. Although the quality is very low, the image is still recognizable and it has the advantage that it can be displayed on virtually any terminal or printer in existence. This format would be sufficient for some applications. For example, before downloading an image with a modem, a process which may take an hour, the image could be previewed by converting it into symbols.



```
pdOOC>??(jOGUUUU22SSSSS22SSSSVYJYxx)^::'-.....  "5UUUC><<Jdgyx<xxxx<jQBBQQBQB0PFx
WOOOC>??(jOGUUUU22SSVj222SSSYYYxxJc>^`~.~........"9DC<<<x3QQpcxx<xqQBQQBQBQRFc<?
WOOOC>??(jOGUUUU222Z\2222SVYnJcuYT^:::!--~.........3b><<x3QQRT'''3BBQQQQQBP<<?((
WUUUC>??(jOGUUUU222r(U22SSzVnoY>;/^":!~::~~~~.~.....7<<<JZF^-...  7BBQQBQ#T<<))()
EUUUC>?^?jODUUUUU2U`72222SoSY<xx<|:^::^:::!~,~!~~~~~,-^"`-......  3QQQBBPT<?(((<<
EUUUC>^^\7OGUUUUUUK~?7SU22YcJc<<<<;?^?|;|\;::::::\^"'--..-..._\,_BBQQBP<<?()(())
EUUUC>^^?JOGUUUUUUU[-"<7SCczYYxncxxxxccxcx;(;xc)?^''~~---....;ur\QBQBBF<<)(()()()
EUU2C}^^?JOGUUUUUUb.-JcSSVYzzYntznoagmudUggQPF^'-~~~~~.._uy2F\yQQQBBF<<?)(((<<<<
E2UUC>^^?JOGUUUUUU9L-"\oCjSnnYyyq@@QBg@D@0PT^~::::~~_yySY??\yQBQQBBPx?????(((<<<
E2UUC>^^^JOGGUUUU22UL-\joCnVoqg@@BDNQQQD%T?::::::'--"3gwYxgQBBBQBBPn>?????(?(((<
E2UUb>^^^J9GUUUUU222k.jSVSygQQBQQQ8EQBPT?|!:?;/"-....70Qp7BBB0QBQEY><)))))(((((<
EUUUb>^:^J9GUU2U22SSVL]YugQ8B8QQQ0@BPT\:"\;)?^^!~....-3QBJQB0QQQRt>?????<<<xxx<<
EUUUb>^:^J9GUUUUU2VY7Rbq@8N@BQ0QQQ#F?`/_uyyaeu;;!-~_uyOQBr90QQQBCY)()())(<<<<<xJ
EUUOE>^:^J9DUUUU6\?xy@0@g5@QBQQQQRT^-_aQR0P75oz>!~y0P9QQB[38QBQEJ<(<<<<<<<<xxxcJ
EUOOE>^:"79G223PSyyQO9QOOZ3QBQQBP>'_zYY7TYx;<]Y1~"7Yu90QBE3QQQEY>(<<<<<<xxxc??"`
WOOQE>^::(dG2Vjyq@RPUQBQDgp9QQBF?_yKVc</^^^^(<Yt,.?<YS9B0E7QBBF=(<<<<<xxJc?`~---
B8BQE>^::xdE2VqUU@8K9QQBR0PS9BK^jQ0bncx)|::\xcjz}.\<JjQBQ#(QQEc)(<<<<xxJY`---...
B8BQW>^::(962S@PgD@A30BQQEgQ#T\yBBOKCnc<<))(JCyoL.(xjgBBQQ>3BY>(<<<<<xJY`~-.....
B888K>^^:(dDKS808@@Uy30yBEFT2uQBQB32nncxx<<(<7Y?^"<JqQBR8Qp3F<(<<<<xxxJ^-.......
BQ08b>^^:(d5238g0Q500AyRF^uVgBQB0QU6CnzcJjuuuuu;;xcqBQB88QE7=<<<<<<xxJY!........
MQQWb>^::(a$SqDQ88j5UP2x>/y@BBQB0Q8@w2ozcxYVncxxJxgBBBQ088B<<<<<<<<xxJ1-........
MQB6C>^::(dEd8@BQWypSXZSLgBQ0BQQQQQQ@gwozcx<???\x@QBQQB988B?<)(<<<<xcj[. .......
MQQDp>^::(Z50B@QQB75m2s729QB8BQQQQQQQ8RPXVY<);)?79NQQQB9088(juuucxxJJj)  ...._ygg
E9R0B[|^^\35QQD008WyXyUCSjdNQBQQBQB0QD2onYYxx<?!~~-"79Q0@8EJYYYVS22eSd^..._y@0RO
Ec73B[^^^(JgQBQQW0QQ@8Gbz3y2QB0QBQBD9B2VYJcJ<<)|::~..."9BQEx<xxJJY]SYT..._@DO5U5
```

Figure 10.

A screen image from a VT100 style terminal. Despite the lack of pixels, the image is still recognizable as figure 3. This image quality is sufficient for some uses, such as previewing images over a very slow network connection or telephone line. The weights are the same as in figure 7d, but the match was optimized for this font.

Before this method can be used in the real world, we need to address the problem of the choice of font mentioned earlier. Images that have been optimized for Courier will not be optimal when displayed in other fonts. Figure 11 shows scatter plots of two different fonts. Figure 11a shows Courier plotted in the same way as it was in figure 5 with the exception that it has been normalized on the DC axis by the darkest and lightest symbols.

Figure 11b shows the scatter of the screen font used in figure 10. Since these symbols were designed for use on the screen, they are simple and constructed of fewer pixels. Since they have fewer pixels, the symbols have fewer possible DC levels and hence fall into discrete DC bins as can be seen in the figure. By comparing a symbol's position on these two different charts, we can see how much error there will be when an image that has been matched using the data from 11a is displayed using the symbols in 11b. As can be seen from the two figures, most of the symbols fall in the same general area on either chart.

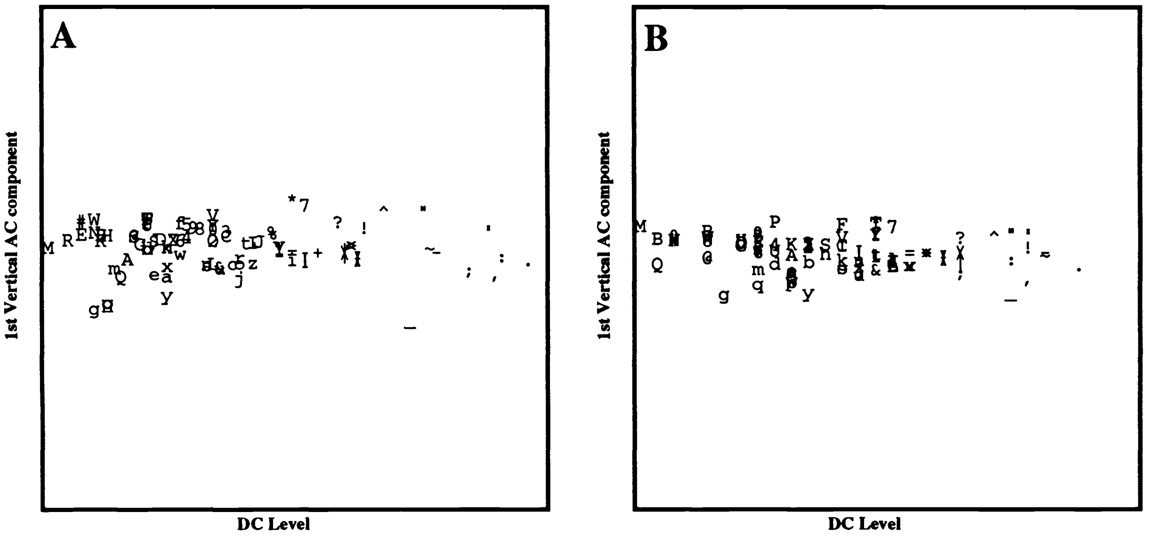


Figure 11.

Scatter plots of two different fonts are shown for comparison. As in figure 5, the horizontal axis is the DC value of the symbol. The values have been normalized so the left edge is the value of the darkest symbol and the right edge is the lightest symbol. The vertical axis is the same as in figure 5. Figure A shows Courier font and figure B shows screen font used in figure 10 and figure 12. Most of the symbols fall in the same general area in both graphs.

The same image using identical symbols is shown in the two different fonts in figure 12. The image was optimized for the font in figure 11a. Since the symbols are similar in different fonts, the image is not severely degraded by the font change. The most noticeable distortion is the change in aspect ratio. An image could be made more resistant to errors introduced by changes in font if a large number of fonts were sampled. Symbols that had widely different representations in different fonts would not be used. The other symbols could be averaged to find the best fit across the different fonts.

The techniques discussed produce images that can be sent through electronic mail, posted on electronic bulletin boards and displayed on virtually any output device. Informally, colleagues who were sent images like these reported that they were recognizable. This was despite the choice of Courier as the font, which is doubtlessly not optimal.

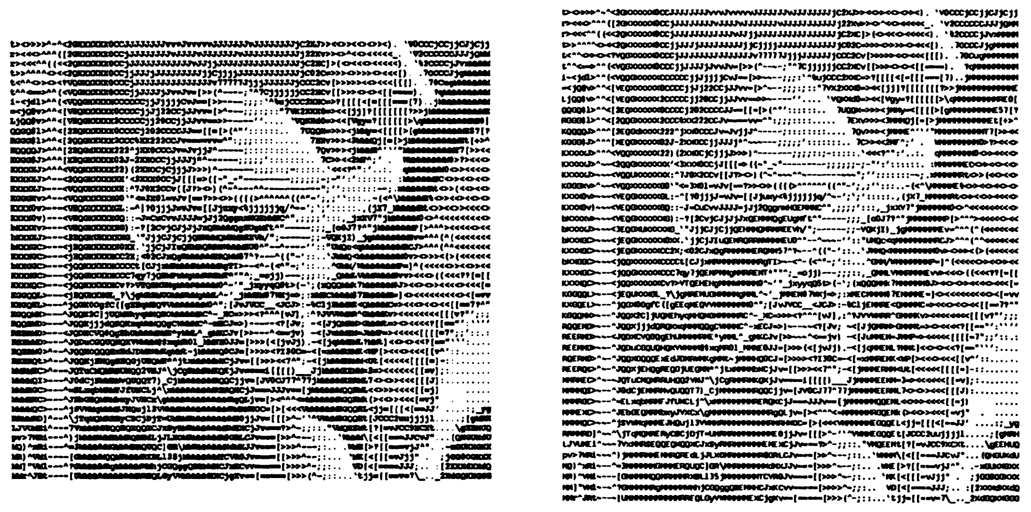


Figure 12.
Figure A is the same as figure 7d. In figure B, the identical symbols are shown in the screen font graphed in figure 9. Although the image was optimized for Courier, the image quality is not severely degraded by the change of font. The most noticeable distortion is the change in aspect ratio.

4. ACKNOWLEDGEMENTS

We thank Greg Tharp for his assistance with the software.
This work was supported by AFOSR 89-0238

5. REFERENCES

1. R. Gonzalez and P. Wintz, *Digital Image Processing*, Addison-Wesley Publishing, Massachusetts, 1977.

2. M. Rabbani and P. Jones, *Digital Image Compression Techniques*, Vol TT7, SPIE Optical Engineering Press, Washington, 1991.

3. E. Feig, "A fast scaled-DCT algorithm," SPIE Proceedings on Image Processing Algorithms and Techniques, Vol. 1244, San Jose, February 1990.

4. S. A. Klein, D. A. Silverstein, and T. Carney "Relevance of human vision to JPEG-DCT compression," SPIE Proceedings on Human Vision, Visual Processing, an Digital Display III, Vol. 1666-18, San Jose, February 1992.

5. J. G. Robson, "Spatial and temporal contrast sensitivity functions of the visual system," J. Opt. Soc. Am., 56, pp. 1141-1142 1966.

6. G. K. Wallace, "Overview of the JPEG (ISO/CCITT) still image compression standard," SPIE proceedings on Image Processing Algorithms and Techniques, Vol. 1244, San Jose, February 1990.

7. S. A. Klein and T. Carney "Perfect displays and perfect image compression in space and time," SPIE Proceedings on Human Vision, Visual Processing, an Digital Display II, Vol. 1453, 1991.