Practical Halftoning with TEX

Adrian F. Clark
University of Essex
Janet: `alien@essex.ac.uk`

Abstract

The inclusion of photographic material in a book or similar publication is an expensive process and, unless care is taken, artefacts can be introduced into the reproduction. For publications in subjects like image processing, it is essential that the pictures are reproduced both cheaply and accurately. The principle of using TEX to typeset halftoned pictures is now quite familiar, but there are a number of practical problems involved. This paper discusses these problems and compares halftoning with TEX to other techniques such as direct inclusion of PostScript. Some of the advantages and disadvantages of both approaches are described, taking account of the need for low-resolution drafts from laser printers and high-resolution final output from typesetters.

Given the ability to generate halftoned output, it is but a small step to consider including colour pictures in TEX documents. Some attempts at producing colour separations using the above approaches are described.

Introduction

Even with the advent of computer typesetting, one of the most costly aspects of book production is the inclusion of continuous-tone (grey-scale or colour) material. This is because the preparation of such copy is done essentially manually, the figure usually being merged with the text only just before the production of the offset printing masters. However, with a suitably high-resolution output device such as a phototypesetter, it is possible to produce representations of continuous-tone material which are of printable quality. This paper looks at how continuous-tone copy can be inserted into a TEX document (albeit plain TEX or LATEX).

The conventional approach to the reproduction of continuous-tone material, *halftoning*, is briefly described and the three main ways of producing halftoned output from TEX are then discussed and compared. An extension to the basic techniques to prepare colour separations is then described and examples of the output which can be obtained are given. Directions for future work are then considered.

Readers who wish to learn more about conventional and digital halftoning are referred to Southworth and Ulichney-book; the latter requires some familiarity with Fourier analysis. Halftoning with TEX is not a new idea, aspects having been discussed by Simpson, Knuth and Clark. Colour and its reproduction are discussed in great detail by Hunt. The problems in printing computer-originated imagery are discussed by Stone.

Reproducing Continuous-Tone Material

Conventional Halftoning. Printing is an all-or-nothing operation: one either puts ink at a specific position on a page or one doesn't. Hence, to reproduce material with a continuous range of tones (grey levels), one must produce a region of black and white (ink or no ink) dots which the eye blends together to give the appearance of being the appropriate greys. The process by which this is done is known as *halftoning.* This is usually achieved by photographing the original through a *screen,* which consists of a pair of glass plates, on which fine opaque lines have been ruled, cemented together at right angles to each other, leaving a structure of square interstices. When photographed, the interstices act as crude pinhole "lenses," forming out-of-focus "images" of the camera lens on the plate. A high-contrast film is used in the photographic process, so that the "screened" picture consists of only light and dark dots. Light areas in the original produce larger areas in the (negative) screened image and hence smaller black dots when printed; the converse is true for dark areas of the original image.

The quality of reproduction is governed primarily by the fineness of the screen, which is normally measured in terms of the number of lines ruled per inch — newspaper photographs typically have screens of about 70 lines/inch, while magazines have screens of about 130 lines/inch. For top-quality printing, screens of up to 300 lines/inch are used.

Digital Halftoning. It will be apparent from the above that the binary nature of halftoning lends itself well to a digital approach, and there has been a fair amount of work devoted to the topic. At the simplest level, one can assign a little region of dots to generate a single pixel of the image — the more dots set to black, the darker the pixel; but this leads to a very objectionable patterned effect in regions of constant grey value. Practical techniques do not require several binary dots to represent a single pixel, and attempt to remove the patterning effects; these are known as *dithering* algorithms.

The algorithm known as *ordered dither* is closest to conventional, photographic halftoning since it passes high-contrast, high-frequency detail into the halftoned image. A mask (typically 8×8 pixels in size) of fixed thresholds is tesselated over the image, with the output at each pixel being white if the pixel holds a greater value than the corresponding threshold in the mask and black otherwise. Unfortunately, ordered dither also produces objectionable artifacts in regions of constant grey level; this is typically reduced by adding a little high-frequency noise. As an aside, if one uses a randomly-chosen threshold instead of a mask of fixed values, one obtains an image which resembles a mezzotint.

The most popular dithering technique was developed by Floyd and Steinberg, and is known as *error diffusion*. This compares each pixel with a fixed threshold as for ordered dither. However, the threshold value is then compared with the value of the pixel and the difference (the "error") used to generate a correction factor to be added to future pixels. (In signal processing jargon, this is a "recursive filter.") The correction factor is calculated from the error according to the weighted values of neighbouring pixels, and some interest has centred around the best weights to use — see Ulichney for details.

Halftoning and TeX

There are three basic approaches to the inclusion of halftoned material in TeX documents:

1. generating a "picture" font,
2. building a halftone font, then typesetting the picture with TeX,
3. including the halftone material at the `dvi`-processing stage via `\specials`.

We shall look at each of these methods in turn.

Generating a "Picture" Font. In this approach, the entire picture is halftoned and converted to a `pk` file, which allows TeX to typeset it as a single (though large) character. The conversion may be direct from image to `pk` file, in which case a `tfm` file must also be generated, or via METAFONT. However, the run-time overhead of using METAFONT in this way is a significant one.

Having generated such a "picture" font (`picture.tfm` and `picture.pk`), one would insert the image into a floating figure in a plain TeX document with a series of commands like:

```
\font\pix=picture

\midinsert
   \centerline{{\pix\char0}}
   \smallskip
   \centerline{Picture inclusion using
               a picture font}
\endinsert
```

For LaTeX, the approach is similar but the syntax a little different:

```
\newfont{\pix}{picture}

\begin{figure}
   \centering
   {\pix\char0}
   \caption{Picture inclusion using a
               picture font}
\end{figure}
```

One problem with this approach is that some output devices — notably Digital's LN03 laser printer — have a limit to the height of a glyph which is smaller than the typical size of a complete picture. In this case, one must ensure that the picture is partitioned into a set of smaller regions.

Using a Halftone Font. An alternative approach is to devise an entire font which maps input characters onto output greys, then use TeX's boxes-and-glue approach to typesetting to construct the picture from the characters. This approach was also outlined by Knuth and popularised by this author.

```
\hbox{\vbox{\halftone\offinterlineskip % machine-generated by TEXPIC.
\def\BHT{\hbox\bgroup\ignorespaces}
\catcode`\^=12 \catcode`\_=12 \catcode`\.=\active \let.=\egroup
\catcode`\,=\active \let,=\BHT \catcode`\/=0 \catcode`\\=12
,abe___cbaadgljkigbeehe``ae_FB?=D.
,abe_^`caaddfkjic[JSbiea_]iNFA@?E.
,ace_^`baaa`dkj?DFF?60Qc_`gEH@A?F.
,bcd_^`bba`_ccAEDHOFMIBB^f[EJ@CDJ.
,acd___baa``UB>FCEMPTUQKMiMMQIOVZ.
,acd__`ba``ZUTGIQKX]^_ZLAd?CHCUUW.
,bcd_^`aa`_`YSR\deghojgTFO?CHBRPU.
,bcd_^`ba`^]_dbfjmhhmnifNH?DJNONS.
,bcd_^`ba`]]djikkmkid[aXZGAHWMQRT.
,bcd__aba`_Z[f^\[djih[[U]FDTWNVSU.
,bcd__aba`_hib\affddf]f[UFKTTR[UT.
,bdc__abba`jmkgjhillh`fafQSXXXXVU.
,bed`_abba`f[QRVccci jjgfdELUWVVXZ.
,bdd``bcbbaakiXXcjiidjigVCCWVVXVX.
,bdd``bccbbaV_LZS_c^aefaECBNWUWUX.
,bedaabcccbbJDL\LKQ\eidJDDEPWXWYV.
,cedaacccbcbP[UbUNTa[\SKFCBKTWVVX.
,cedbacccccc^aQTX]_dZTPRSHCMRXSUW.
,ceeaacddcbeeVXMdXad[RUVQLKUU[\VT.
,cgfbbddddcXe]UXZ\cceOKMQBCOW\^WZ.
,dhgddffefdiinVT\chhfOLMOBIMNNNLJ.
,ejifeefffemhlfYmkhggWRPNDRSRQOLN.
,\dihggkkkilk`ceUW_^[jfc`LSTSQPMM.
,aZ`cecgijaaaadhkhhhhifdfhkhRTVWT.
,gii]deiW]baabhdgmffeefefgeVRUXSU.
,cflg`fV`bcbcdhfgeheefhiknj_SBXQR.
,bclidb]`gdbfgfjiihhkijloggfd]R^].
,kkloo]^faffbeghkklmenmmgghhhj_aW.
,pnkop_adkgcdb]^fll^WA9PgghgijiWU.
,pmklpa_dkkjgba_dmj`QE1>?dhgnikbS.
,mllljb`bkkigijhehlORER??>kkokmjV.
,nmlmfcabkmnhiebeFJKMcY>:B:54onlc.
}}%
```

Figure 1: Machine-Generated TEX for Use with a Halftone Font

The halftone font in most widespread use is a variation on the "double-dot" font devised by Knuth and has some 65 grey levels. This maps the ASCII character "0" onto white, "1" onto near-white — and so on, up to "p" which represents black.

With such a font available, it is not difficult to write software which converts an image to a file which is compatible with both plain TEX and LATEX. The only point at which care is needed is in the handling of those characters which lie between "0" and "p" and have a special meaning to TEX, namely "\", "^" and "_". This is illustrated in Figure 1, which is a 32×32 representation of the "girl" image ubiquitous in image processing circles.

Including the figure in a plain TEX document is as easy as for the picture font:

```
\font\halftone=htfont

\midinsert
   \centerline{\input girl32}
   \smallskip
   \centerline{Picture inclusion using a
                   halftone font}
\endinsert
```

while, for LATEX, we would use

```
\newfont{\halftone}{htfont}
```

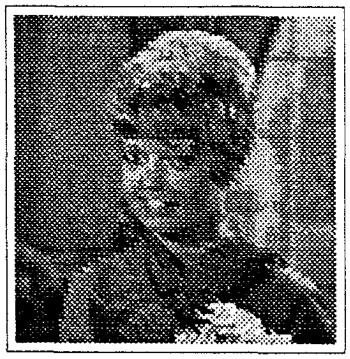


Figure 2: Picture Produced using a Halftone Font on a LaserWriter

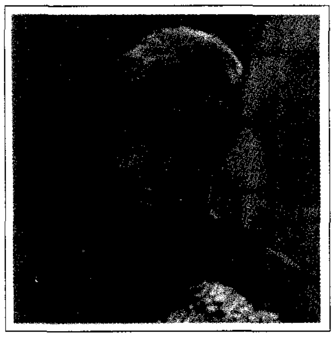


Figure 3: Picture Produced using Halftone Font on a Linotronic 300

```
\begin{figure}
   \centering
   \mbox{\input picture\relax}
   \caption{Picture inclusion using a
                halftone font}
\end{figure}
```

Note the use of **\relax** here to stop LaTeX from treating the closing brace as part of the filename. In these examples, **girl32.tex** contains the machine-generated picture, while **htfont** is the name of the **tfm** file which contains the halftone font for the chosen output device. The machine-generated TeX input assumes that the halftone font is selected by the command **\halftone**.

To obtain a reasonable number of greys in the font, the size of the printed image must depend on the resolution (number of dots/inch) of the output device. This effect is illustrated in Figure 2 and Figure 3, which are output from an Apple LaserWriter (300 dpi) and a Linotronic 300 typesetter (1270 dpi) respectively: although these are the same physical size, the LaserWriter image contains 64 × 64 pixels and the Linotronic one 256 × 256.

Some care is needed in optimising the halftone font to the output device: the author has found that existing mode_defs for printers invariably cause the halftoned image to be too dark, and one must adjust the **blackness** and **fillin** parameters while printing off a test grey scale.

TeX, as originally devised, had a memory limit which is too restrictive for typesetting pictures of 256 × 256 or more pixels. This was one of the factors which led to the production of expanded-memory versions of TeX. Of course, the dvi driver (and, indeed, the output device) must also be able to handle this many characters on a single page. Furthermore, the time taken for TeX to typeset a 512 × 512 picture (about quarter of a million characters!) is not insignificant. The great advantage of this method, however, is that pictures are text files (and hence can be moved around by electronic mail) and produce essentially identical output on a variety of output devices.

Incorporation at the DVI-Processing Stage. In this approach, one simply leaves a gap in the TeX output and tells the dvi driver to insert a file of printer-specific instructions by means of a \special. Until the time comes when \specials are standardised, this approach is also specific to the chosen dvi driver: for example, the syntax of the \specials accepted by one dvi-to-PostScript driver are rarely accepted by another.

The most widely-used halftoning device available to most TeX users is probably a PostScript laser printer. Since it is relevant to the following discussion of colour output, a brief summary of the facilities provided by PostScript for image display is called for (see Roth for many practical insights into using PostScript devices for halftoning).

Image data is displayed in PostScript via the **image** operator. This requires information as to the sizes of the image (in pixels) and size (in units of 1/72 inch) of the resulting display. This is followed by the pixels themselves, with each pixel normally being coded as two hexadecimal digits. The pixels are halftoned within the printer according to a notional halftone screen which is defined by three parameters:

1. the screen frequency in lines/inch,
2. the rotation of the screen in degrees from the horizontal,
3. the *spot function*.

The first of these determines the fineness of the screen (within the limitations of the device), while

```
%!PS-Adobe-2.0 EPSF-1.2
%%Creator: PSPIC 0.1
%%CreationDate: 10-AUG-90 11:15:16
%%BoundingBox:     72     72    360    360
%%EndComments
/paperxorigin   72 def        /paperyorigin    72 def
/paperwidth    288 def        /paperheight    288 def
/imagewidth     32 def        /imageheight     32 def
/picstr imagewidth string def
/plotpic { gsave
  imagewidth imageheight 8
  [imagewidth 0 0 imageheight neg 0 imageheight]
  {currentfile picstr readhexstring pop} image grestore } def
paperxorigin paperyorigin translate
paperwidth paperheight scale
plotpic
3A382D43454235383A3C2E241218131E25392B2E1F2C3E3F3D2B44A6B8C2CAAE
3B372D424741363A3A31312613181B32549872371C2B3C434D1B88A7[illegible]
3B352D43463E373A3C3C4130131AC4AFA8A9C4E8FF7[illegible]
3A342F43464036393C3F423634BCAA[illegible]886626A646A5E
3B342F444542363A3[illegible]2B1D2E97AFAFAD7E6361655E68
3B34[illegible]6B396C876F3C554F7494A6B3B5946E65696661
[illegible]2323235354A3B7B6E5F4C442F57727F77749EB58D795E756A62
342C2E3A3C33313234362D2D695F8C325E3D3055796A687A8F936B6D5350696E
32252A363A302F3130345E2A4C6D5F575034322D82928C7CB6B384634E476359
3120233131292A2C2A311C1E09667050321E1F26838E8B83B99D8D8988888E96
2D191C292C2B2728262D0D2111285D0D151F232462777E86AF7675797A839189
512F1A2025221416151C111341342D6D6245495316283340 8F7471757B808B8D
3D5840322C32231D193A3E3D3B311F1621211E221C262E2720141F776F66646F
241C1A4C312C1D644C363A3B391F31230B26292B2D292B28222A67796B60756A
352611233F26664138333734 2E1E29242B222A2D27201E1408194275B9617B77
3932101B2F374C40232F38292427171A1D1F20151B1A1205242328304C78494A
13120F04064C47283D2628372C22201316110A2D070C0B22232221211A463C63
0008130501453D301625332F384A47290E124865BCDB7E232220221A181C656A
010D1311013B443012151925383D43310D163E7BADFAC7C5322123071D143974
0A0F111219393E3612121A221E1A1E2D200E8579AD77C2C1C6131505150B1768
0A0C110D29353D39140D09221D2B372CA798928C325BC7D8B8D6EBEE06070E32
showpage
%End of Picture
```

Figure 4: A Machine-Generated PostScript Image

the second is normally set to 45°for monochrome pictures. The spot function determines the shape of the halftone cell; for a modern LaserWriter, this defaults to,

```
{ abs exch 2 copy add 1 gt
   { 1 sub dup mul exch  1 sub dup mul
      add 1 sub }
   { dup mul exch dup mul add 1 exch sub }
ifelse }
```

which, for grey levels less than 128, builds a circular white cell in a black background and, for grey levels greater than 128, black cells in a white background. We can see that this is essentially a direct digital implementation of the photographic technique.

Generating a file of PostScript (or, more precisely, *encapsulated* PostScript) is as easy as generating the TEX input for use with a halftone font (Figure 4). As we can see, the major difference between TEX and PostScript is in the representation of the image data. Each pixel is represented as a character in TEX, while in PostScript it is encoded in hexadecimal. This reflects PostScript's ability (in principle, if not in practice) to display 256

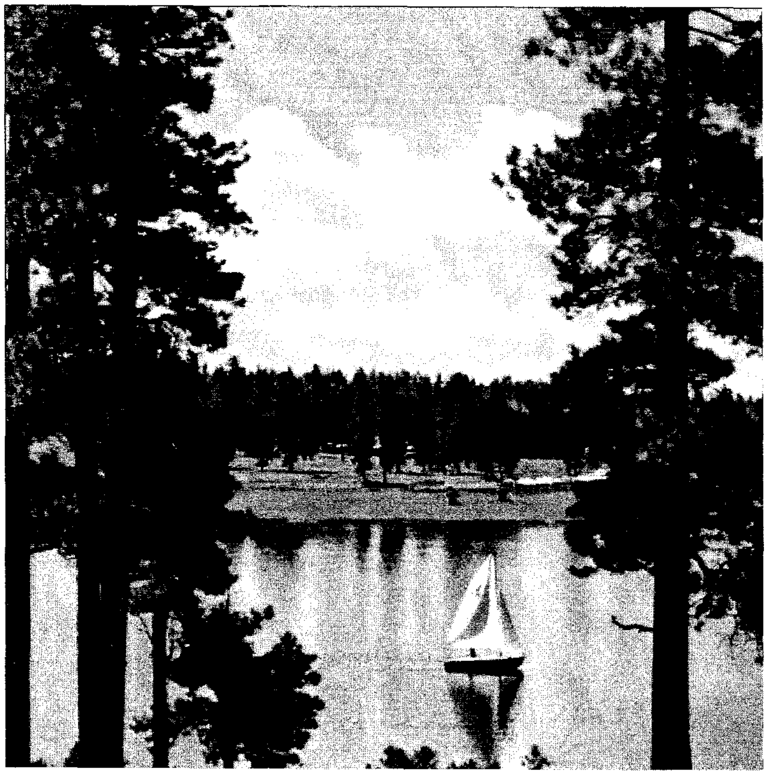


Figure 5: 512 × 512 Image Output via PostScript

different grey values. The output from a typesetter of such an image is certainly good enough to be used for camera-ready copy (Figure 5). The means by which the PostScript picture is inserted into a TEX document depends on the `\special`s offered by the `dvi` driver; for James Clark's `DVItoPS`, the appropriate plain TEX syntax is,

```
\midinsert
  \def\picht{4in}
  \special{dvitops: import boat.ps
      \the\hsize \picht}
  \smallskip
  \centerline{$512 \times 512$ Image
      Output via PostScript}
\endinsert
```

where `\picht` gives the height of the desired figure and `\the\hsize` its width. For LATEX, the syntax is:

```
\begin{figure}
   \def\picht{4in}
   \vspace{\picht}
   \special{dvitops: import boat.ps
           \the\textwidth \picht}
   \caption{$512 \times 512$
         Image Output via PostScript}
\end{figure}
```

where `\the\textwidth` is the width of the figure. In both cases, `DVItoPS` ensures that the aspect ratio of the image is retained, centring the image in the text as necessary. For other `dvi`-to-PostScript drivers, the `psfig` macros can be used to achieve equivalent results.

Comparing the Approaches. The main features of the methods considered here are compared in Table 1. In terms of the quality of the rendition, there is little to choose between the methods, at least on high-resolution devices. The generation of a picture font definitely requires the largest programming effort, particularly since the fine details of the format of `pk` files differ slightly between implementations (*e.g.*, between VMS and UNIX). The compactness and portability of the halftone font perhaps wins, but the flexibility in being able to print a PostScript image at any size is an important feature. When it is important to avoid patterning in regions of constant grey tone, the generation of a picture font, since it is under program control, probably provides the simplest implementation—although it is possible to devise PostScript spot functions which dither.

picture font	halftone font	PostScript
picture description is specific to output device	picture description is portable	picture description is PostScript-specific
picture font is device-specific	halftone font is device-specific	image description is device-independent
one (TEX) character for the entire picture	one byte/pixel, 64 levels	two bytes/pixel, "256" levels
cannot easily perform geometric effects (*e.g.*, rotation of image)	cannot perform geometric effects	easy to perform geometric effects
easy to use dithering techniques	difficult to use dithering techniques	possible to use dithering techniques
very difficult to optimise rendition of picture	grey-level font must be optimised by tweaking `mode_def`	grey-level rendition may have been optimised by manufacturer
vertical spacing in document automatically taken into account	vertical spacing in document automatically taken into account	vertical spacing in document must be explicitly left
big-memory TEX not required	big-memory TEX required for pictures of reasonable size	big-memory TEX not required

Table 1: Comparison of Halftoning Approaches

Reproducing Colour

We have seen that there are several approaches to including monochrome pictures in TEX documents. This provides the basic means for incorporating *colour* pictures, since colour images are conventionally printed from a set of *colour separations* or layers using different coloured inks. Four coloured inks are conventionally used: the subtractive primaries (cyan, magenta, yellow) and black. There are two reasons for including a black separation: since printing inks are never pure colours, mixing the three primaries produces a dirty brown rather than a true black, so dark regions require "added blackness"; and black ink is substantially cheaper than coloured inks.

Given an image in terms of red, green and blue values for each pixel, which is the most common representation of a colour image, a simple-minded way to derive the separations would be as follows. First, we convert the red, green and blue values for each pixel to the corresponding values in terms of the subtractive primaries:

$$\begin{aligned} \text{cyan} &= 1 - \text{red} \\ \text{magenta} &= 1 - \text{green} \\ \text{yellow} &= 1 - \text{blue} \end{aligned}$$

where the value 1 represents a fully-saturated colour. The black value can then be determined by finding the minimum of the three colours:

$$\text{black} = \min\{\text{cyan}, \text{magenta}, \text{yellow}\}.$$

We can then reduce the coloured values by the amount of common blackness:

$$\begin{aligned} \text{cyan} &= \text{cyan} - \text{black} \\ \text{magenta} &= \text{magenta} - \text{black} \\ \text{yellow} &= \text{yellow} - \text{black} \end{aligned}$$

Finally, when the pixel is known to be close to black (*i.e.*, the red, green and blue values are all small), the amount of blackness is increased. (In practice, it is rare for all the black to be removed in this way, since it can cause problems at colour boundaries; a value of about 50% black removal seems to be the most common.)

Having generated the colour separations in this way, it is simple to print them using one of the above techniques. To produce convincing results, however, one must take account of the *orientation* of the halftone screen. To avoid moiré effects when the screens are superimposed, the conventional approach to halftoning employs screens at angles of 0°, 15°, 45°and 75°for yellow, cyan, black and magenta respectively. This is most conveniently implemented via the PostScript screen (see above).

The final problem in terms of colourimetry concerns the purity of the colours of the inks. Magenta ink, for example, is invariably "contaminated" with yellow (*i.e.*, it removes some blue as well as green), and cyan ink is contaminated with magenta. To obtain reasonable colour separations, one must take account of the impurity of colour, a process known as *under colour removal.* We can do this as follows:

$$\begin{aligned} \text{magenta} &= \text{magenta} - f_{\text{y}}\,\text{yellow} \\ \text{cyan} &= \text{cyan} - f_{\text{m}}\,\text{magenta} \end{aligned}$$

where f_{y} is the amount of yellow to remove under magenta and f_{m} the amount of magenta to remove under cyan. Typical values for these are 0.5 and 0.3 respectively. This step should, of course, be performed before determining the black separation as above.

Having generated the separations in this way, one can produce print them using one of the techniques outlined above. A set of separations are shown in Figure 6: these are PostScript output, plotted at 300 dots/inch, so that the directions of the halftone screens can clearly be distinguished.

Discussion

The techniques for preparing monochrome pictures are, as the figures show, impressive in result. With access to an output device of suitable resolution (say, 1200 dpi or better), it is possible to achieve halftoned output of publishable quality. However, there is significant variation between output devices, and it is necessary to tune the font or adjust the digital image accordingly. For example, with an identical input file, the Linotronic 300 typesetter at Aston University produces much darker results than an identical machine at the University of London Computer Centre, reflecting a difference in the adjustment of the machines rather than a fundamental disparity.

The colour separations presented above demonstrate the success of the general approach. However, the colour separation algorithm adopted here is too crude: for example, it should really ensure that there is some overlap of the separations in the region of sharp colour boundaries, to avoid unsightly white gaps should there be minor mis-registration when printing. (Making good colour separations digitally is one of the reasons why electronic prepress equipment is rather expensive!) One must also perform a significant degree of tuning of the colour conversion process to the target output device, much more so than for monochrome output.

With regard to the incorporation of halftoned output into TEX documents, we have already seen that there is a set of solutions to the problem for monochrome output. However, it does not seem possible to provide as neat a solution to the plotting of colour separations. One would like to provide a simple command, let us call it `\colourpic`, which defines the four files required to plot the separations. The obvious way in which `\colourpic` would work would be to plot the black separation in the main document and to generate auxiliary files which gave positioning information for the other three colours. Unfortunately, such information is only available within an `output` routine. (On a PostScript output device, one could alternately make the printer output this information.) This prohibits a solution of the simplicity and generality of that for monochrome pictures. However, easy-to-use macros for incorporating colour separations are under investigation.

Acknowledgements

The high-resolution figures for this paper were plotted on a Linotronic 300 made available to the author by Peter Abbott of the Aston University Computing Service. The author would like to thank Peter Wade of the Essex University Printing Service and Richard Kirk of Crosfield Electronics for helpful discussions.

References

[1] Adrian F. Clark. Halftone output from TEX. *TUGboat*, 8(3):270–275, November 1987.

[2] Malcolm Clark, editor. *TEX Applications, Uses, Methods*, chapter 28: Nontraditional Uses of METAFONT (by Richard O. Simpson). Ellis Horwood, Chichester, Sussex, UK, 1990 (*Proceedings of the TEXexter88 Conference*).

[3] R. W. Floyd and L. Steinberg. An adaptive algorithm for spatial greyscale. *Proc. SID*, 17(2):75–77.

[4] R. W. G. Hunt. *The Reproduction of Colour.* Fountain Press, London, UK, 3rd edition, 1975.

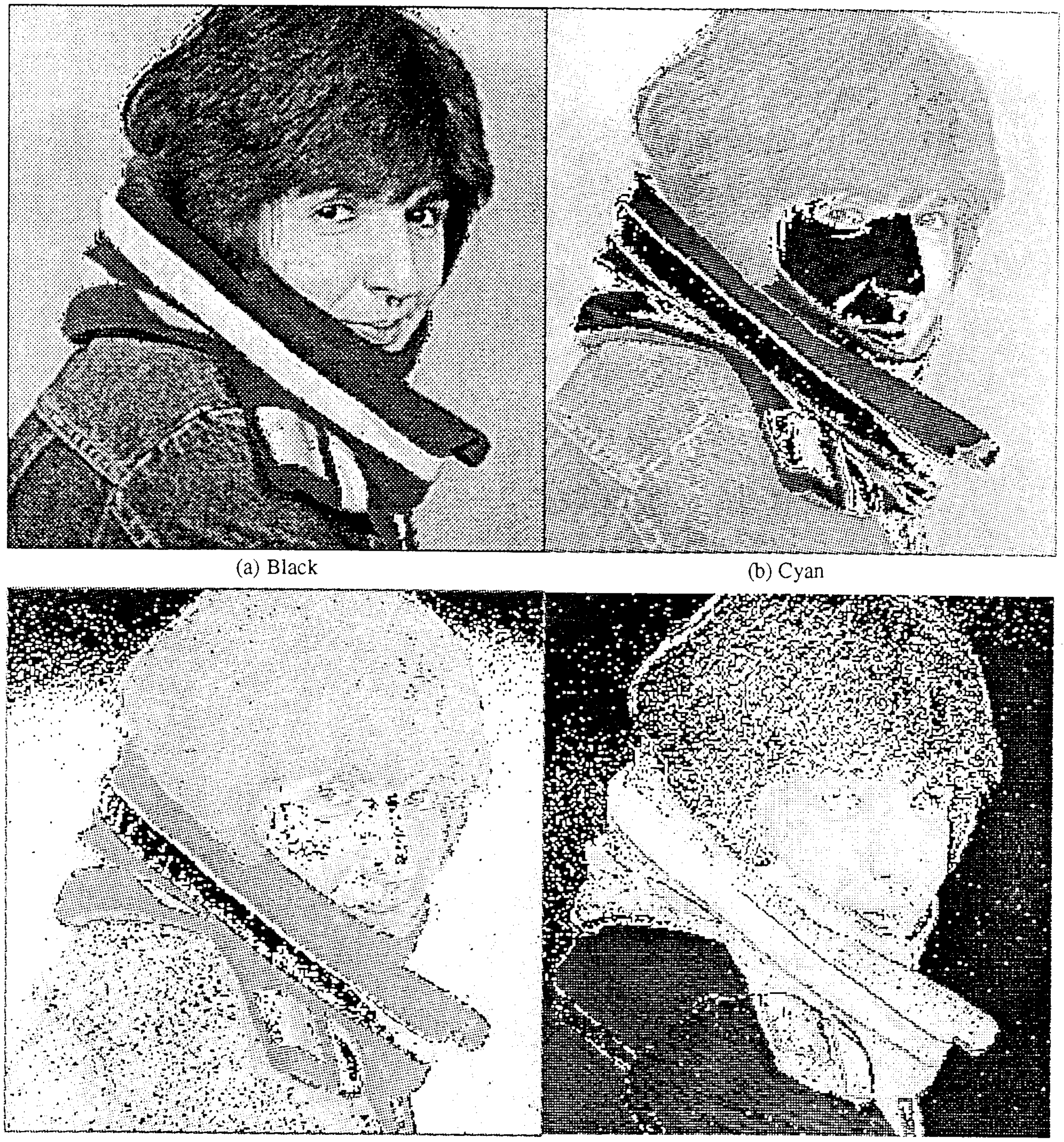
[5] Donald E. Knuth. Fonts for digital halftones. *TUGboat*, 8(2):135–160, July 1987.

[6] Stephen F. Roth, editor. *Real World PostScript.* Addison-Wesley, Reading, Massachusetts, USA, 1988.

[7] M. Southworth. *Colour Separation Techniques.* Graphic Arts Press, Livonia, Michigan, USA, 2nd edition, 1985.

[8] Maureen C. Stone, William B. Cowan, and John C. Beatty. Color gamut mapping and the printing of digital color images. *ACM Transactions on Graphics*, 7(44):249–292, October 1988.

[9] Robert Ulichney. *Digital Halftoning.* MIT Press, Cambridge, Massachusetts, USA, 1987.



(a) Black

(b) Cyan

(c) Magenta

(d) Yellow

Figure 6: Colour Separations Plotted via PostScript