ELECTRONIC PUBLISHING, VOL. 7(3), 147–161 (SEPTEMBER 1994)

Hypertext and multimedia enhancements to the TEX system

A. F. CLARK, S. L. CHEAH AND T. K. TAN

Department of Electronic Systems Engineering
University of Essex
Colchester CO4 3SQ, UK

SUMMARY

Enhancements have been made to the TEX system to support hypertext and multimedia facilities. A special previewer, `hdvi`, has been developed to give access to these features. Using TEX's `\special` mechanism, the previewer displays images, line graphics, audio, and video, as well as supporting hypertext; it also permits limited interaction with the underlying operating system. A LATEX style file has been devised to provide access to all these features. Some user feedback with the system is described and the effectiveness of the general approach is assessed.

KEY WORDS Hypertext Multimedia TEX

1 INTRODUCTION

It is now almost ubiquitous for documents to be prepared electronically: the ease with which the text may be modified—and the quality of the final product—has practically made the typewriter a thing of the past. This is particularly the case in science and engineering research, where the ability to typeset mathematics is important. Furthermore, computer networking and electronic mail have made contacting a colleague on the other side of the world as quick and easy as one in the next office. It is therefore surprising that the public dissemination of research results by electronic means is almost unheard of, despite the advantages of being able to include hypertext features, and media such as sound and video, in documents.

There are many reasons for this, both technical and sociological. The most significant of the latter appear to be related with quality and the risk of plagiarism. However, there are good technical reasons too, and the lack of common tools for preparing and viewing hypertext and multimedia documents is one of the more important. Most prospective 'electronic authors' would like to prepare their papers using familiar tools, not worrying particularly about inserting hypertext or multimedia material.

The work described in this paper was motivated by the wish to instigate an electronic journal in the area of image processing, with papers being both submitted and distributed electronically. Image processing is particularly well-suited to this medium: printing grey-scale and colour matter is significantly more expensive than text, while the quality of reproduction is almost inevitably poorer than the author's originals. For monochrome output, this is typically due to artefacts introduced while halftoning (e.g., [1]); for colour reproduction, there are additional problems due to incompatibilities between monitor and printer gamuts [2]. Conversely, a framestore with colour display is an essential tool of the

CCC 0894–3982/94/030147–15
Received 15 May 1993
Revised 30 October 1993

image processing trade, usually via colour workstations running Unix and the X window system [3].

Taking into account the wish of prospective authors to use familiar tools, an obvious approach is to attempt to 'graft' new facilities onto an existing (and popular) document preparation system. This is the approach that the authors have taken. As a cursory glance at the proceedings of any major conference in science or engineering will confirm, the most popular document preparation is probably TEX [4], especially in conjunction with the LATEX macro package [5]. What the authors have done is to develop a prototype TEX previewer that supports multimedia and hypertext features that runs under the X window system, and to provide an easy-to-use interface to these features.

There are, of course, many hypertext document preparation systems (see [6] for a recent review). However, only one hypertext system utilizing TEX has been reported: the LACE system [7] was based around the NeWS window system [8] and provided broadly similar facilities to those described herein. LACE's use of display POSTSCRIPT is more general than the approach the authors have adopted, since one may use a variety of tools to generate the document. (Under the X window system, an equivalent approach would be to use a POSTSCRIPT previewer.) However, LACE used TEX primarily for preparing fragments of a larger hyper-document system; the authors' approach, admittedly more restrictive, is to use TEX to prepare a complete, stand-alone hyper-document.

This paper discusses the desirable hypertext and multimedia features. It then examines how far the plain TEX and LATEX packages provide these features and indicates the additional features required and the mechanisms for adding them. Some important details of the actual previewer are then discussed, and a LATEX style file for accessing them is described. Finally, the paper assesses the efficacy of the authors' approach and draws some conclusions concerning the use of TEX both for an electronic journal for image processing and in a wider context.

2 FACILITIES PROVIDED BY TEX AND LATEX

The essence of hypertext is that documents may be read in a non-sequential manner. In the electronic context, this normally involves inserting connecting *links* into the document that, when activated by, say, a mouse press, provide additional information, often by means of a pop-up window. A similar interface can be provided to multimedia features: by clicking the mouse on a suitable 'hot' region of the page, stored sound or video may be replayed.

The TEX program and its default macro package were developed primarily for typesetting books and articles, and hence lack these features. The basic facilities for incorporating cross-references in documents are provided but are not well-developed. However, the LATEX macro package provides for cross-references via its `\label`, `\ref`, and `\pageref` commands. Similarly, LATEX provides mechanisms for table-of-contents production, bibliographic citations, and index generation. These provide a good starting point for providing hypertext links—and, moreover, providing them in a way which fits in well with the existing LATEX mechanisms.

Although TEX (and hence LATEX) produces excellent typeset text and (especially) mathematics, its graphical facilities are severely limited. Plain TEX can draw only horizontal and vertical lines. The LATEX picture environment allows the user to reproduce straight diagonal lines of limited gradients and some curves and circles; however, these are achieved via custom fonts containing line- and curve-segments, an approach lacking in generality. A number of attempts have been made to circumvent the line-drawing restriction by other

means: Wichura's excellent PiCTEX [9] draws lines of arbitrary shape by means of closely spaced dots—this is flexible but extremely wasteful of memory. Alternatively, one may use the METAFONT font-generation package [4, Vol. 3] to generate an entire line-drawing, thereafter using TEX's standard font mechanisms for rendering it in the document (e.g., [10]). Finally, one may use TEX's `\special` mechanism for passing line-segments from the TEX input to the DVI output file.

In fact, the latter mechanism is by far the most commonly used, but it is not without problems, for there is no universally accepted set of `\special` commands for line-drawing. There are, however, two popular sets: the first is known as the `tpic` specials, which are derived from the line-drawing commands of the Unix `pic` processor [11]; and the second set is that supported by the DVI drivers of the popular emTEX implementation of the TEX system for MS-DOS systems.

For the inclusion of grey-scale matter in TEX documents, the simplest approach is via a halftone font [12–14]. This has the disadvantage that, for images of any reasonable size, one needs a big-memory TEX (indeed, the first big-memory TEX was developed for just this application [13]); and the size of the graphic is specific to the resolution of the chosen output device. For POSTSCRIPT [15] devices, images may be incorporated by means of `\special` commands; in this case, the image on the rendered page is independent of the device resolution. However, this requires that the image be converted to a POSTSCRIPT representation; and, since POSTSCRIPT is non-trivial to interpret, this mechanism is not suitable as a general approach for incorporating images. It is possible in principle to produce colour output using either a halftone font or `\special`s, though there are some difficulties associated with the former [16].

Ideally, one would incorporate images via a dedicated `\special` command: this would allow devices capable of rendering colour or grey scales to display the image with some fidelity, while drivers for bi-level devices would have to generate halftoned output [1]. To date, only one DVI driver, `DVIwindo` for MS-Windows, has this kind of capability.

ADDITIONAL FACILITIES REQUIRED

Hypertext facilities

As we have seen, many of the basic facilities required to make hypertext possible with TEX (and particularly LATEX) already exist. What is missing is the ability to specify hypertext links in the markup language, and to indicate the location of the link on the virtual page in the DVI file. It is, of course, possible to extend the TEX program to support such features: however, this would produce an incompatible program and might compromise the portability or robustness of the underlying program [17]. A much better approach is to use the 'escape' mechanism built into TEX, namely `\special`. This passes its argument into the DVI file as one of its 'virtual typesetter' commands and may be used to access particular features of the chosen DVI driver. All that is necessary is to define the syntax for a hypertext link, and make the DVI driver parse the `\special` and instigate the link.

The syntax chosen by the authors for their `hdvi` previewer (see below) was to have separate `\special`s indicating the beginning and end of the link. The syntax for these are:

```
\special{link begin n}
\special{link end}
```

Table 1. The `tpic` `\specials`

Argument	Meaning
`pn` *n*	set the pen size (i.e., the width of subsequent lines) to *n*
`pa` *xy*	store a 'path segment', an *x*,*y* position
`fp`	draw the current object with a full line
`ip`	draw the current object with an invisible line (this results in shading only taking place)
`da` *n*	draw the current object with a dashed line, each dash being *n* units
`dt` *n*	draw the current object with a dotted line, each dot being *n* units
`sp` [*n*]	draw the current object with a spline curve, the optional *n* indicating the length of the dot/dash
`ar` $xyx_ry_r\theta_s\theta_e$	draw a circular arc centred at *x*,*y* with x-radius x_r and y-radius y_r between angles θ_s and θ_e (specified in radians)
`ia` $xyx_ry_r\theta_s\theta_e$	as `ar` but draw an invisible arc
`sh` *s*	shade the previously rendered object (box, circle, ellipse) in grey level *s*, where $s = 0$ for white and $s = 1$ for black
`wh`	whiten (un-shade) the previously rendered object
`bk`	blackens the previously-rendered object
`tx`	sets the shading value to be used by `sh`

The origin of the coordinate system is at the upper left corner of the drawing, with the x-axis to the right and the y-axis down the page. Arguments are separated from the command and from each other by spaces. All measurements are given in units of milli-inches.

where *n* is the number of the page to which the link points. For LaTeX this may be generated using a modified version of the `\pageref` command. The entire region between the `link begin` and the `link end` `\specials` forms the 'hot' region for the link: with the `hdvi` previewer described below, one merely clicks mouse button 3 to activate the link. Separate commands for starting and ending a link were chosen for ease and flexibility: it allows the 'hot' text to span line, or even page, boundaries. (This may well be undesirable but there are enough controls over the page layout in TeX to avoid breaks.)

Line-drawing

A minimal requirement for a multimedia TeX is the ability to display more than just text! As we have seen, there are extensions that support line-drawing and grey-scale pictures—and both these features may be added via TeX's `\special` mechanism.

As mentioned above, there are two popular sets of line-drawing `\specials`, `tpic` and emTeX, which are summarized in Table 1 and Table 2 respectively. Although it is possible to produce line drawings manually with these facilities, interactive editors are frequently used: `fig` or `xfig` under Unix can be used to generate (among others) `tpic` `\specials`, while `TEXCAD` under DOS generates emTeX `\specials`. Many existing previewers support `tpic` specials, and a number of hardcopy drivers support either or both.

In producing a hypertext previewer, it was deemed important that it be as flexible as possible: hence, it supports both `tpic` and emTeX `\specials` for drawing lines. Furthermore, the popular `dvips` [18] DVI-to-PostScript conversion program supports both these types of `\specials`, and a subsidiary objective of our work was to make `hdvi` as compatible as possible with it.

Table 2. The emTEX \specials

Argument	Meaning
em:message *x*	display the message *x* after displaying the page number
em:point *n*	define coordinate *n* as the current position on the page
em:line *a,b[,w]*	draws a line between positions *a* and *b* (defined using em:point) with a line of width *w* (or, if omitted, the default width). *w* may be specified in any of TEX's units but px. Either *a* or *b* may be followed by h, v, or p to show how the line should be terminated: h specifies a horizontal cut, v vertical, and p perpendicular to the direction of the line. The default is p
em:linewidth *w*	sets the default line width to *w*
em:moveto *n*	sets the current position on the page to *n*
em:lineto *n*	draws a line from the current position on the page to point *n*

Point and line definitions are local to the page on which they are defined. Note that points need not be defined before the lines that refer to them. (There are other emTEX-specific \specials not supported by hdvi.)

Displaying images

Under the X window system, displaying colour images is not much more difficult than displaying monochrome ones, for there are well-known algorithms for converting 24-bit colour images to the 8-bit colour-mapped representation required by most workstations [19, for example]: the biggest problem concerns control of the workstation's colour map. For the reasons outlined above, the authors believe that the only truly satisfactory way of having a TEX previewer display images is via a purpose-defined \special. The form of \special adopted for hdvi is:

\special{image *format name hsize vsize*}

where *format* is a specification of the format of the image file and *name* is its name. *hsize* and *vsize* define the sizes of the region of the page in which the image is to appear (described in the usual TEX way): the image data are re-sampled if necessary to be centred within this area. The same sampling interval is used in both orthogonal directions.

This syntax was deliberately chosen to allow for additional image format types to be supported. hdvi currently supports only pbm, pgm, and ppm formats [20], but adding other popular image formats (e.g., TIFF, GIF) is straightforward.

One consequence of adopting this form of \special is that the DVI file no longer contains the entire document. While this is also the case for, say, graphics included by a DVI-to-POSTSCRIPT driver, there is an important difference: a DVI-to-POSTSCRIPT driver will generate a single output stream with all the graphics included, making distribution of the result easy; this is not so here, since a previewer produces visual output directly. Hence, if one distributes a document in DVI format for use with hdvi, one would also have to distribute any images it uses.

Using colour

The use of colour is an important aspect of multimedia documents, not just for graphics but also to convey additional contextual information to the reader. A set of \specials for generating colour POSTSCRIPT has been developed for preparing overhead transparencies

with dvips [21], though their utility is by no means limited to such an application. These have the following form:

```
\special{background c}
\special{color cmyk q}
\special{color push c}
\special{color pop}
```

The first of these is used to set the background colour for the current page and all subsequent pages to the colour *c*, where *c* is the colour *name* (see below). The color cmyk command causes all subsequent text to be rendered in colour *q*, where *q* is a quadruplet of normalized colour values representing cyan, magenta, yellow, and black (e.g., color cmyk .2 .4 .3 .1). The color push and color pop \specials permit temporary changes to be made to the rendering colour for text. Obviously, they must occur in matching pairs.

These \specials are made available to the TEX or LATEX user via a style-file, colordvi. An alternative style file, blackdvi, defines the same macros but excludes the generation of \specials, for use with non-colour devices. The style files define a number of colour names (Yellow, Plum, RedViolet, etc.) which may be used as the colour argument *c* in the \specials. These colour names and their matches are largely based on Crayola crayons; the dvips driver converts the colour name to suitable CMYK values internally. Since compatibility with existing DVI drivers was an aim in this work, these \specials were adopted for hdvi. However, it should be pointed out that the colour specification mechanism, and indeed the colour names, are not ideal for the X window system.

Line-drawings, pictures, and colour are, of course, all common in conventional documents. One could say, with some justification, that the 'multimedia' facilities described above simply correct defects in the TEX program: they do not add functionality beyond that of ink-on-paper. However, such functionality has been investigated.

Replaying audio

Many modern personal computers and workstations have capabilities for recording and/or replaying sound, so this is a natural feature to incorporate. However, doing so is not without its problems: although there are many image formats, the underlying data consist of regularly sampled pixels, so interconversion between them is not too difficult; and the X window system provides platform-independent facilities for displaying them. This is not the case with audio data: there are different, nonlinear quantization schemes, different sampling rates, and vendor-specific replay facilities. It is hoped that some standardization of facilities, representation, and—most important—programmer interface will emerge, perhaps from the Interactive Multimedia Association [22]; until then, the reader should be aware of the inherent lack of portability of the audio features of hdvi.

The hdvi previewer was developed on a Sun workstation, so the Sun audio format ('au') was chosen for this work. As with the abovementioned facilities, audio files are incorporated into a TEX document via \specials:

```
\special{audio begin format name}
\special{audio end}
```

The audio format supported by hdvi is au.

The reason that there are two \specials rather than one is that any text in the document between the begin and the end may be used by the previewer to form the 'hot' region, so that clicking the mouse button in the region invokes the feature.

Video facilities

Video replay is a second novel medium that computer-based documents can incorporate. There has been a great deal of interest over recent years in the development of internationally agreed standards for the efficient transmission of moving imagery. At the time of writing, H.261 [23], developed primarily for video-conferencing and related applications, and its derivative MPEG 1 [24] are the most important of these. It must be emphasized that both of these are compression schemes, not file formats or interchange protocols—though the latter are under development [25, 26]. Nevertheless, there are sample codecs for both H.261 and MPEG and one may use these to illustrate the principle.

Analogously to audio, one may incorporate a moving sequence into a TEX document for hdvi via

```
\special{video begin format name}
\special{video end}
```

where *format* is either h261 or mpeg. (Either of these may be used with hdvi.) The name ipi-iif is reserved for future use.

Facilities for executing commands

A final feature that has been investigated is that of command invocation by means of a \special which causes an operating system command to be executed:

```
\special{do begin command}
\special{do end}
```

This causes *command* to be executed in a sub-shell, and hdvi to wait for it to terminate. Such a command might seem superfluous, since the TEX/previewer combination does not provide 'active' document capabilities. However, as the LATEX code in Figure 1 illustrates, one can use the \special in interactive tutorials to great effect. (The example forms part of a tutorial in image processing using the Khoros system [27]; the \docmd macro is defined in Figure 3.)

hdvi—A HYPERTEX PREVIEWER

A previewer for the X window system was developed to implement the above facilities. hdvi is based on the popular xdvi previewer. xdvi already supported the tpic \specials; the implementation of the emTEX line-drawing \specials was extracted from that in dvips. The code to support the colour \specials was also loosely based on that in dvips.

Many of the cross-references in documents are of the form 'see Figure 4', where the figure may occur on a different page. Hence, hdvi was designed to be able to display two

```
An example of the edge detection process is shown in
\figref{fig:edges}.  The following section lets you perform edge
detection interactively: follow each of the steps in turn to digitize,
edge-detect, and display an image.

\subsection{Experimental edge-detection}
\begin{enumerate}
\item Point the video camera at something interesting and
  \docmd{grab -o in.viff}{click here} to digitize an image.
\item Click \docmd{vdrf -i in.viff -o edge.viff}{here} to perform edge
  detection.
\item Click \docmd{editimage -i edge.viff}{here} to display the result.
\item Finally, click \docmd{rm in.viff edge.viff}{here} to tidy up the
  image files you have created in this example.
\end{enumerate}
```

Figure 1. Illustration of device control from a LaTeX document

pages independently in windows. When `hdvi` is executed, the first page of the document is displayed; page advance, re-sizing of the text etc. is accomplished by means of buttons and pull-down menus in the usual way (see Figure 2). Line-drawings and images are rendered at the correct location on the page. Mouse button 1 is used to magnify the region about the cursor.

Hypertext links—and the `audio`, `video`, and `do \special`s—appear in colour when used via the LaTeX style file `hyper.sty`, the colour used indicating the underlying facility (although this is usually obvious to the reader from the text of the document). When the user moves the cursor over a link, its border is drawn; this is erased when the cursor moves off the link. (This provides the user with visual feedback, and is in fact the only way a link is made visible on a bi-level display.) Clicking mouse button 3 while the cursor is located above a hypertext link causes the hypertext link to be followed: the page is brought up in a separate window on the display. Thereafter, either window may be paged forward or backward independently, and links may be followed from either window: doing so in one window causes the other window to be overwritten. Each window has a 'link back' button associated with a stack of pages displayed to permit the user to retrace his or her path through the document.

When the user clicks mouse button 3 above a region associated with the `do \special`, the appropriate command is executed via the C `system` routine. Hence, a separate process is created for each such `\special`, so one cannot assume when preparing the document that context is carried from one `do` to the next. (This is desirable too, since the result is then independent of the order in which selections are made.)

While the above `\special`s are implemented directly by `hdvi`, this is not the case for `audio` and `video`. These cause external programs to be executed, again using a `system` call via mouse button 3, with the name of the file specified in the `\special` as an argument. The program invoked depends on the file format specified, and its name is compiled into `hdvi`.

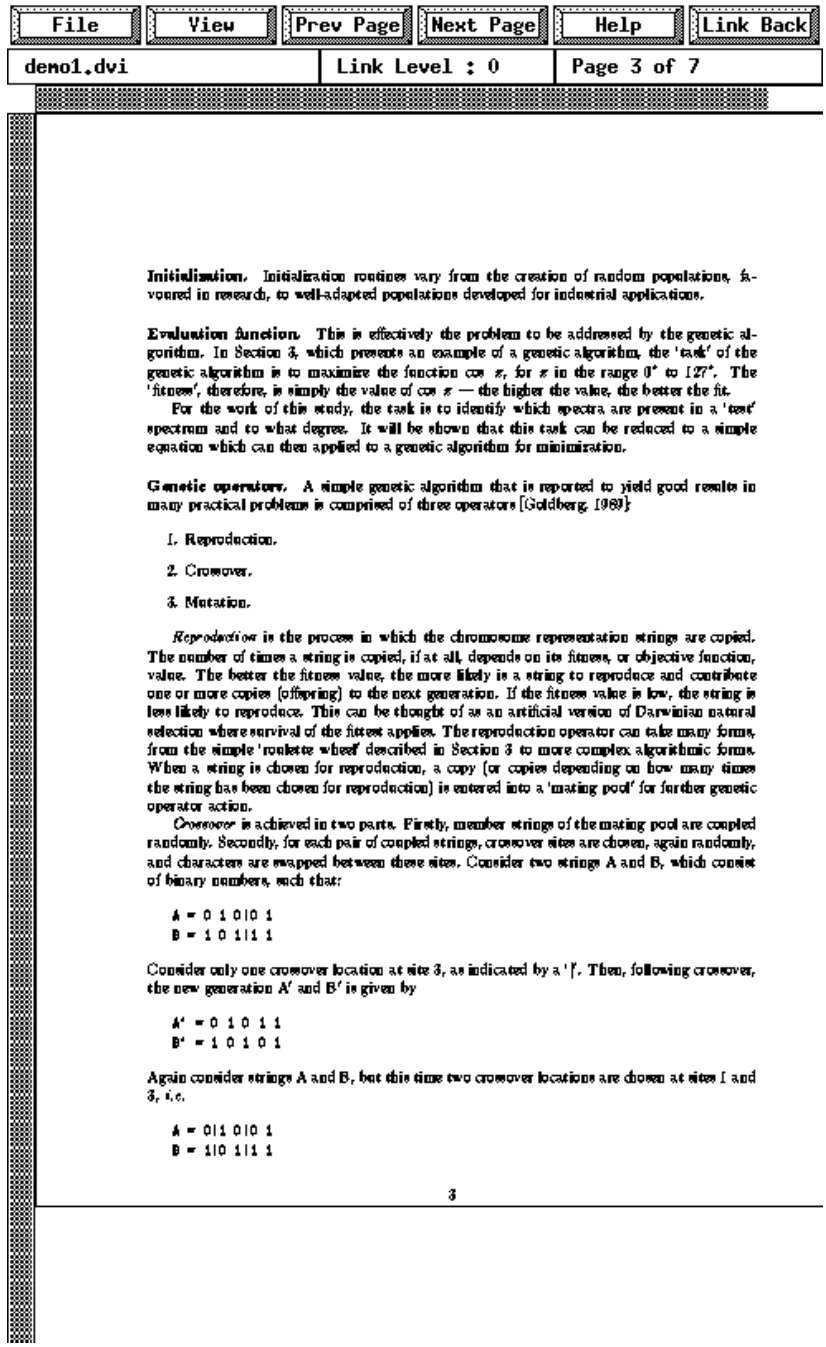

Figure 2. The `hdvi` *window layout*

The principal implementation of `hdvi` is on Sun hardware and uses Sun-supplied software for playing audio files. H.261- and MPEG-format sequences are replayed via freely distributable programs: both display the sequence in separate windows rather than on the TEX page. `hdvi` also runs on DECstation systems under Ultrix, and ports to Irix and HP-UX are under way. A Linux port is planned.

The approach of having the DVI previewer invoke external programs to perform particular functions is a powerful and extensible one, and is in keeping with the general Unix philosophy [28]; however, it also increases the effort involved in porting the hypertext system to a new platform (one might have to port a new tool for each supported file format) and increases the temptation to use manufacturer-supplied features.

It is worth pointing out that, since the PBMPLUS package has a large number of image-format conversion programs as well as library routines for accessing files in its own formats, many additional image formats may be added to those currently supported in `hdvi` by a three-step process:

1. using a `system` call, convert the image to PBMPLUS format and store in a temporary file;
2. load the PBMPLUS-format file;
3. delete the temporary file.

PROVIDING ACCESS FROM LATEX

Having developed hypertext and multimedia facilities for use with a previewer, the remaining problems concern making them available to the user. For coloured text and line drawings, user-level macros are already available; for media such as images, audio and video, the approach that has been adopted is simply to wrap macros around the `\specials`. However, there is scope for automatically inserting hypertext links. In this respect, the features already offered by LATEX and the emphasis placed on document structure by its mark-up 'language' make it somewhat easier to work with.

There are a number of LATEX facilities that may be adapted for hypertext use; these are discussed in the following paragraphs and are made available to the user in a single style file, `hyper.sty`, parts of which are shown in Figure 3.[1] In performing this adaption, the maxim was that it should be easy for the user to enable and disable the generation of hypertext `\specials` during the processing of the document by LATEX. Hence, the style file defines a TEX conditional, `\ifhyper`, which it sets to true. The user may subsequently inhibit the generation of hypertext links by `\hyperfalse` and re-enable them with `\hypertrue` anywhere in his or her document.

It is anticipated that most users of `hyper.sty` will wish to generate coloured text, so `colordvi.sty` is `\input` if its macros have not already been defined. The `\ifhyper` conditional deliberately does not affect the generation of coloured text; however, if the user specifies `blackdvi` *before* `hyper` in the style options to the `\documentstyle` command, coloured text generation is disabled.

To permit the document to generate hypertext links in cross-references, `hyper.sty` provides several convenient macros. Having labelled (say) a figure using a command like `\label{myfig}`, one would ordinarily refer to it from elsewhere in the LATEX document

[1] The complete style file may be obtained from the first author.

```
% Define the macros that generate the actual cross-references.  These
% may be changed by the user if he or she wants a different layout
%  (e.g., 'Fig.~\ref') or is using a language other than English.
\def\hyperfigref{Figure~\ref}     % figure

% Define the macros that determine the colour of the cross-references
% and things.  Again, these are made easy for the user to change.
\def\hyperfigcol{\Red}       % figure

% Define the actual macros that the user invokes for cross-references.
\def\figref#1{\hyperLB{#1}\hyperfigcol{\hyperfigref{#1}}\hyperLE}

% The above macros make use of '\hyperLB' and '\hyperLE', for
% beginning and ending a hypertext link respectively.  These macros do
% nothing when the user has set '\hyperfalse'.
\def\hyperLB#1{\ifhyper\special{link begin \pageref{#1}}\fi}
\def\hyperLE{\ifhyper{\special{link end}}\fi}

% The following macro replaces (and uses) LaTeX's \contentsline,
% which produces a line in the table of contents (or figures, etc),
% with a hypertext cross-reference.
\let\hyper@contentsline \contentsline
\def\contentsline#1#2#3{\hyperLB{#3}%
\hypertoccol{\hyper@contentsline{#1}{#2}{#3}}\hyperLE}

% The following macros supersede those in LaTeX's citation mechanism,
% inserting a hypertext link.
\let\hyper@bibitem \@bibitem
\let\hyper@lbibitem \@lbibitem
\def\@lbibitem[#1]#2{\hyper@lbibitem[#1]{#2}\label{#2}}
\def\@bibitem#1{\hyper@bibitem{#1}\label{#1}}

% Macros that ease the generation of a 'hyper-index'.
\def\hindex#1{\ifhyper\index{#1|hyperPR}\else\index{#1}\fi}
\def\hyperPR#1{\hyperLB{#1}\hyperpcol{#1}\hyperLE}

% Access to the audio, video, and system interaction facilities.
\def\audioformat{au}
\def\videoformat{mpeg}
\def\audio#1#2{\special{audio begin \audioformat\ #1}%
\hyperaudiocol{#2}\special{audio end}}
\def\video#1#2{\special{video begin \videoformat\ #1}%
\hypervideocol{#2}\special{video end}}
\def\docmd#1#2{\special{do begin #1}\hyperdocmdcol{#2}\special{do end}}
```

Figure 3. Excerpts from `hyper.sty`

by \ref{myfig}. Instead of typing this in the document, the hypertext user would type \figref{myfig} and this generates the string "Figure~\ref{myfig}," putting a \special{link begin} before the word 'Figure' and a \special{link end} after the reference to the figure. The page number in the \special{link begin} is actually generated by a modified version of LaTeX's existing \pageref macro. As well as making the entire figure reference into a hypertext link, \figref inserts a color push/color pop pair to make it be rendered in the colour \hyperfigcol, which is defined as \Red in hyper.sty. There are two reason for inserting the word 'Figure' into the cross-reference: as well as making the hypertext link larger (and therefore easier to see), it ensures that such references are consistent throughout the document. The word is actually generated by the macro \hyperfigref; the user is free to re-define this, and \hyperfigcol, to change the appearance of the reference—for example, in a non-English document. There are analogous macros for cross-references to appendix, chapter, equation, page number, section, and table.

An obvious place in which to insert hypertext links is in the table of contents (and lists of figures and tables, of course). This is quite simple to achieve: each such entry is inserted into the document by means of a macro, \contentsline, the third argument of which is the page number. All hyper.sty does is to rename the existing \contentsline macro via TeX's primitive command \let, and then re-define \contentsline to insert a link begin \special, invoke the renamed macro, and insert a link end. Additionally, the original \contentsline macro is invoked as the argument to a \hypertoccol macro, which makes it appear in red. Re-defining \hypertoccol will change the colour of the entry.

A similar approach may be used to supersede LaTeX's citation mechanism: the definitions of the macros \@lbibitem and \@bibitem, which are used in the bibliography listing produced by the \bibliography command, must be overridden to insert \label commands, and the \@citex command modified to insert the hypertext link around a citation in the document. As with the other links, the macro \hypercitecol governs the colour of the citation.

For the index to a document, a somewhat different approach has been adopted. In generating index entries from a LaTeX file, one normally types "myword\index{myword}". The arguments of the \index commands are all collected in a .idx file, which may be collated into an index by a companion program of LaTeX, MakeIndex[29]. MakeIndex has a feature by which the invocation \index{myword|mac} will generate \mac{myword} in the collated index. Hence, all that is needed is a macro that wraps its argument inside link begin/link end \specials and a command to use within the document instead of \index. Since it is entirely possible that a user would wish that only some index entries form hypertext links, a new macro, \hindex, is defined instead of replacing \index. Index entries are essentially cross-references to pages, so they are rendered in the same colour as normal page cross-references.

DISCUSSION

This work was not intended to produce a production system for either hypertext or multimedia documents: it was intended primarily as an investigation into whether these facilities could be added to the existing TeX system. In this respect, it has been surprisingly successful. Although it was necessary to write a custom previewer, all the facilities have been added

without compromising the underlying programs. This can only be due to the foresight with which TEX was designed, and the flexibility underlying the LATEX system.

The hypertext features can be added into LATEX documents with very little effort. This allows papers and documents intended for conventional publication routes to be converted for interactive browsing very simply. Such documents are, of course, normally intended to be read linearly from start to finish, so `hdvi` is then used in a way that is analogous to flipping between two pages (between say text and a diagram that is being described): it is primarily a 'value-added' previewer. However, the same LATEX facilities are equally easy to use in documents that are intended solely for on-line use, as with the example in Figure 1.

It is worth noting that the Free Software Foundation's re-implementation of the Unix typesetting tools (`groff`) is capable of generating DVI files: hence, it should be possible to generate compatible hypertext and multimedia by that route.

With respect to `hdvi` itself, it has been used by both experienced and inexperienced TEX users with a view to gathering feedback to go into the development of a production hypertext previewer. Specific points include:

- The font-loading mechanism used by `hdvi` is essentially that of its ancestor `xdvi`, so the fonts normally used at Essex are sub-sampled by the program from 300 dpi fonts and hence appear a little fuzzy. This could be addressed by generating PK files matched to the screen resolution (which, unfortunately, varies from platform to platform), by anti-aliasing glyphs using grey levels (on some displays at least), or by supporting the use of X fonts.
- More control over audio and video files (e.g., volume, fast-forward, rewind) is desirable.
- The default LATEX styles are not well-suited to the use of large fonts on small 'pages.' Styles intended specifically for on-line use require development.
- When following a link to, say, a figure, bringing up a complete page wastes screen space. It would be better for `hdvi` to automatically crop the background, so that the window brought up was just large enough to contain the figure. This would not be difficult to implement, but perhaps it would be best to indicate explicitly in the DVI file that a particular page may be cropped via a `\special`:

  ```
  \special{crop-page}
  ```

- Images, audio files, and video files are included via their filename. There should be environment variables (e.g., `HDVIIMAGEPATH`) analogous to those supported by both TEX and most other utilities in the overall system that contains a list of directories in which to search for such files.
- It should be possible to specify the programs used to replay audio and video files by a similar environment variable mechanism.

Furthermore, `hdvi` itself is too slow and memory-hungry to be considered a production tool: this is partly because of buffering enforced by X and partly due to design decisions during development.

A number of enhancements would be needed to `hdvi` to make it suitable for production use. An approach that the authors favour, at least on Unix, would be to write a DVI widget for the Tk toolkit [30] and then use its underlying Tcl language to implement the hypertext and multimedia features. Such an approach could also increase the amount of interaction between document and user.

A more mature `hdvi` should utilize a client-server approach with a specified communication protocol, ideally one compatible with the URLs of the World-Wide Web (WWW) [31]. This would allow links to be made between documents so that, say, an entire conference proceedings to be viewed, or hypermedia LaTeX documents integrated into WWW. This would allow mathematical material to be made available via WWW, which is currently far from convenient, even with LaTeX-to-WWW conversion programs such as `latex2html` [32]. Moreover, the range of facilities available via `hdvi` is greater than that currently supported by WWW viewers. However, `hdvi` is much more resource-intensive than WWW viewers.

The hypertext style file was implemented for LaTeX 2.09; but LaTeX 3 is now under development and it is sufficiently different internally that the style will have to be totally re-written. In this context, it is interesting to indicate some specific difficulties that were encountered:

- The `\label`...`\pageref` mechanism is really being abused in generating the hypertext links. The value returned by `\pageref` is that in TeX's `\count0` register, which is written into the DVI file for each page of output. However, such numbers are not necessarily unique: in LaTeX, changing the rendition of the page number from say arabic to roman resets the page counter; and, in plain TeX negative values of `\count0` indicate a roman page number! To facilitate hypertext, LaTeX 3 should give each page a sequence number (1,2, . . .) that is written into the DVI file via another `\count` register.

  As an aside, if the sequence number were to be stored in `\count0` and the page number in `\count1`, many of the practical problems encountered in page selection with several DVI drivers would also be circumvented.
- The handling of headlines and footlines in LaTeX 2.09 is rather messy, especially since many contributed style files interact with them in strange ways. Ideally, one would be able to specify the colours of the headline separately from each other and from the main text of the document; this has not been done in `hyper.sty` because it would involve modifications to LaTeX's rather delicate `\output` routine. However, such capabilities (perhaps implemented in a similar way to the colours in `hyper.sty` via start- and end-of-headline 'hooks') would be invaluable in LaTeX 3.

In conclusion, one should consider whether the original objectives of this work have been met, namely to provide facilities for an electronic journal without requiring prospective authors to immerse themselves in the detail of hypertext and multimedia facilities. We believe that these aims have been achieved: documents prepared with `hyper.sty` may be printed as before, with or without colour, or viewed on-line with `hdvi`.

REFERENCES

1. Robert Ulichney, *Digital halftoning*, MIT Press, Cambridge, Massachusetts, USA, 1987.
2. Maureen C. Stone, Willian 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).
3. R. W. Scheifler and J. Gettys, 'The X window system', *ACM Trans. Graphics*, **5**(2), 79–109, (April 1986).
4. D. E. Knuth, *Computers and Typesetting* series, Addison-Wesley, Reading, MA, 1986.
5. Leslie Lamport, *LaTeX User's Guide and Reference Manual*, Addison-Wesley, Reading, MA, 1986.

6. C. Boyle and K. Ratliff, 'A survey and classification of hypertext documentation systems', *IEEE Transactions on Professional Communication*, **35**(2), 98–111, (June 1992).
7. L. Carr, S. P. Q. Rahtz and W. Hall, 'Experiments in TEX and hyperactivity', *TUGboat*, **12**(1), 13–20, (March 1991).
8. J. Gosling, D. Rosenthal and M. Arden, *The NeWS Book*, Springer-Verlag, New York, 1989.
9. M. J. Wichura, *The PiCTEX Manual*, The TEX Users Group, P. O. Box 9506, Providence, Rhode Island 02940, USA, 1986.
10. P. Wilcox, 'METAPLOT: machine-independent line graphics for TEX', *TUGboat*, **10**(2), 179–187, (July 1989).
11. J. L. Bentley, 'Little languages', *Communications of the ACM*, **29**(8), 711–721, (August 1986).
12. Donald E. Knuth, 'Fonts for digital halftones', *TUGboat*, **8**(2), 135–160, (July 1987).
13. Adrian F. Clark, 'Halftone output from TEX', *TUGboat*, **8**(3), 270–275, (November 1987).
14. F. Sowa, 'Integration of graphics into TEX', *TUGboat*, **12**(1), 58–63, (1991).
15. Adobe Systems Incorporated, POSTSCRIPT *Language Reference Manual*, Addison-Wesley, Reading, MA, 1985.
16. A. F. Clark, 'Practical halftoning', *TUGboat*, **12**(1), 157–165, (1991).
17. D. E. Knuth, 'Errors of TEX', *Software — Practice and Experience*, **19**(7), 607–685, (July 1989).
18. T. Rokicki, *DVIPS: A TEX Driver*, Available by anonymous FTP from TEX archives.
19. M. Gervautz and W. Purgathofer, *A simple method for color quantization: Octree quantization*, in "Graphics Gems", A.S. Glassner, ed, 287–293, Academic Press, New York, 1990.
20. J. Poskanzer, *PBMPLUS*, Available by anonymous FTP from X window system archives.
21. J. L. Hafner, 'FoilTEX, a LATEX-like system for typesetting foils', *TUGboat*, **13**(1), 347–356, (October 1992).
22. W. K. Pratt, Personal communication.
23. CCITT, 'Draft revisions of recommendation H.261: Video codec for audiovisual services at $p \times 64$ kbits/s', *Signal Processing: Image Communication*, **2**(2), 221–239, (August 1990).
24. International Standards Organization, *IS-11172. Coding of Moving Pictures and Associated Audio for Digital Storage Media up to About 1.5 Mbit/s*, International Standards Organization, 1992.
25. A. F. Clark, 'Image processing and interchange—the imaging model', in *Proceedings of the SPIE Conference on Image Processing and Interchange*, pp. 106–116, San Jose, California, USA, February, 1992.
26. C. Blum and G. R. Hofmann, 'ISO/IEC's image interchange facility (IIF)', in *Proceedings of the SPIE Conference on Image Processing and Interchange*, pp. 130–141, San Jose, California, USA, February, 1992.
27. J. Rasure and M. Young, 'Open environment for image processing and software development', in *Proceedings of the SPIE Conference on Image Processing and Interchange*, pp. 300–310, San Jose, California, USA, February, 1992.
28. B. W. Kernighan and R. Pike, *The Unix Programming Environment*, Prentice Hall, Englewood Cliffs, NJ, 1984.
29. L. Lamport, `MakeIndex`*: An Index Processor for LATEX*, Available by anonymous FTP from TEX archives.
30. John K. Ousterhout, *An Introduction to Tcl and Tk*, Addison-Wesley, Reading, MA. In press.
31. T. J. Berners-Lee, R. Cailliau, J.-F. Groff and B. Pollermann, 'World-wide web: the information universe', *Electronic Networking: Research, Applications and Policy*, **2**(1), 52–58, (1992).
32. N. Drakos. `latex2html`. Available from the author. Computer Based Learning Unit, University of Leeds.