An informal look into the history of digital typography

David Walden
walden-family.com/texland
(Comments to me as dave@walden-family.com)

August 16, 2016

Introduction

The ditto machine at the elementary school at which my mother taught was what first sparked my interest in *printing* (and it had a swell smell). I became seriously interested in the craft of printing during four summers of college when I worked in or near the printing department of a large Fibreboard company plant that made cardboard packaging for food and drink companies (for example, cereal boxes and milk cartons). The plant had a four-color Miehle offset lithography machine that printed an array of boxes on each approximately 4.5-foot-by-6.5-foot sheet of cardboard; the plant also had big two-color lithograph machines and a couple of smaller letterpress machines. I have retained this interest in printing throughout my life.

I didn't begin to explicitly think about *typography* itself[1,2] until about 20 years ago when I adopted LaTeX for drafting and formatting books and papers I write. Then in 2012 I began to think about the *history* of printing and typography as I prepared a presentation for TUG2012 in Boston.[3] Since then I have been reading (books, papers, Internet websites) and watching YouTube videos about the history of printing and typography that in time led into the digital era.

This companion paper to my TUG2016 presentation sketches some of what I (think I) have learned in the hope that my study and thinking will be useful to someone else who is just starting to dig into this history. People who are already knowledgeable about printing and typography history can help me understand better. Certainly, by writing this paper, I am gaining more than anyone else will.

The acknowledgements and references that were left out of my TUG2016 presentation are included here.[4]

Several things became clear to me as I undertook preparing for my presentation.

First, I had not previously thought about how printing has long been a massive business throughout the world. It's also a business with broad application:[5] • newspapers, periodicals, and books; • pamphlets, reports, and legal and financial documents; • sheet music; • packaging, e.g., on can labels and cardboard boxes; • stationary, cards, etc.; • announcements, posters, etc.; • art reproductions; • money, stamps, etc.; • cloth, wall paper, etc.; and, • from the very earliest days, religious documents of all types. Even as printed materials are being replaced with images on electronic devices, printing remains a massive business.[6] Furthermore, typography seems more relevant than ever as it has to address both printed material and a variety of electronic devices and screen sizes.

Second, the dimensions of how printing and typographic activity are accomplished can vary widely:

- large scale production such as big city newspapers; medium or small sized typesetting or print shops; individuals working interactively in their homes on their desktop or laptop computers

- working with frequent tight deadlines; working with mutually agreed deadlines; working at one's own pace
- seeking great typographic beauty; putting other considerations first

One example: big newspapers such as the *Boston Globe* work with tight deadlines, and typographic beauty undoubtedly has to give way at times to more practical considerations. Another example: Donald Knuth being so concerned with typographic beauty that he delayed his work on *The Art of Computing Programming* for years while he developed a typesetting system for his personal use. And there are all the combinations in between.

Third, contrary to my naive feeling that the move to digital happened fairly quickly, it now seems to me that the evolution to digital happened over a very long time. I'll come back to this point a couple of times more.

To make some sense of this massive field, I find it useful to consider the history of digital typography in terms of four dimensions that are somewhat overlapping but nonetheless seem able to represent of the entire field. My taxonomy is:

1. moving toward digitization of newspapers (representative also of book and periodical publishing and the printing industry more generally)
2. development of digital typesetting for individuals
3. typesetting algorithms
4. digital type

The rest of this paper covers aspects of the first three of these areas in some detail and barely touches on the fourth area.[7]

1 Evolution toward and into the digitization of newspapers

Although my digging into the history of digital typography was not initially systematic, in retrospect, it has been useful to have in mind a brief sketch of the history of typesetting and printing, which of course was primarily aimed at making printed documents (e.g., books, newspapers, announcements) available to lots of people.

The original printing presses were letter presses. In the most traditional model, ink is applied to raised letters and art images clamped into a rectangular frame (a chase) and a flat sheet of paper is pressed against the inked letters and art to create a printed page. In rotary letter presses, the flat chase and its contents is slide back and forth under a cylinder carrying successive sheets of paper until the desired number of copies are available. Alternatively, the contents of a chase can be cast into a cylindrical (or partial cylinder) plate such that the inked image rolls by successive pages coming off a long roll of paper. Generally speaking, each approach is faster than the prior approach.

Initially type was set by hand using a composing stick and taking the type for different letters out of cases holding all the characters and symbols for one font. When a few lines had been prepared in the composing stick, they were transferred to a galley tray that held a column of type.[8] Then columns of type (and art work) were manually arranged within a chase and clamped into position using rectangular pieces of wood or metal (furniture) and wedges (quoins).

Until the invention of type casting machines, type had to be created and cast by hand. Stan Nelson's wonderful videos show this process.[9]

By the late 1800s linotype machines were available such that an operator at a keyboard caused molds (matrices) for characters of type to be assembled into a line; the machine then used this line of matrices to cast a solid line of type (a slug); and the slugs were stacked in galley trays to form columns or partial columns of type (the matrices went back into columns of matrices for each character in the font from which they were reused). From then on the process was as before: pages were laid out and clamped into chases either for flat bed (platen) printing or as a step in casting of cylindrical plates. (There is a wonderful video showing the operation of a linotype machine.[10]) With any lengthy print run, stereotype molds were made from which copies of the metal plates could be made.[11]

In a big newspaper there could be many linotype operators (a hundred or more in a big newspaper) creating galleys of type, and many "layout men" composing the pages in chases (perhaps in collaboration with page editors). There could be another bunch of men turning the contents of chases into cylindrical plates (via a paper mache mold), and then yet another bunch of men doing plate setup and running of the press(es).[12] (Another wonderful video shows the various steps in producing a newspaper.[13])

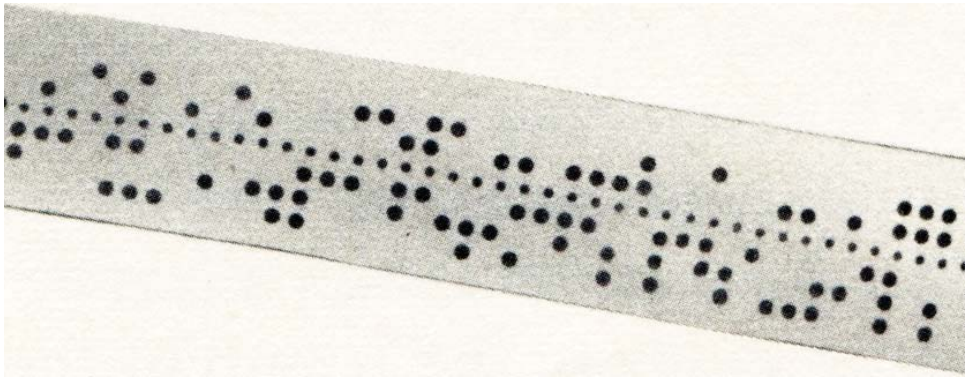

Figure 1: Teletypesetting paper tape

Later in the linotype era, operators at linotype keyboards in some institutions were replaced or augmented by punched paper tapes (Figure 1[14]) created elsewhere that drove the linotype machine. This was called "teletypewriting" or "teletypesetting." The paper tapes (ultimately in several different formats) could come from keyboard units elsewhere in the same facility or electronically from a distance, for example, from the Associate Press wire service. A big newspaper could also teletypewrite stories to casters in multiple printing plants. The videos in footnotes 10 and 13 briefly show teletypewriting input and input from a wire service. (Photos were also being distributed this way as early as the 1930s—sort of an early version of fax.) One might think of such punch paper tapes as the beginning of digital control of typesetting.

The monotype machine was being invented and perfected in the same late 1800s era as the linotype. The montype equipment consisted of two machines: the keyboard on which an operator typed lines to be cast in type; and the caster which cast the lines of type. A strip of 32-channel punched paper tape came out of the keyboard unit and was fed into the caster by the caster operator. From the paper tape, the caster cast individual letter that formed lines of type that went into a galley tray. In some ways the monotype was more flexible than the linotype, for example:[15] (a) the number of keyboards (and keyboard operators) didn't need to match the number of casters, allowing one keyboard to support multiple casters; (b) mistakes could be fixed by changing individual pieces of type rather than whole lines; (c) paper tapes could be saved indefinitely and run again for a repeat of the job while the type was remelted or distributed into cases for reuse.

The next step was phototypesetting. Phototypesetting was very popular, allowing typesetting with hot metal to be abandoned at many institutions. Phototypesetting system used keyboards on computers to prepare text and instructions for font use and type location; originally the text and instructions were punched on paper tape. The paper tapes were fed into the phototypesetters themselves where the text and instructions caused selection of individual characters on film for sizing and projection on the specified locations of a page—in the earliest days on photosensitive media from which lithograph places or plastic letterpress plates could be created. Phototypsetters could be operated in a conventional office, either by people who had previously operated, for instance, linotype machines or by writers (e.g., at newspapers) themselves.

Figure 2 shows an example of commands that were put on paper tape for a Photon phototypesetter use by Michael Barnett at MIT.[16] Figure 3 is an illustration from Seybold's

Figure 2: Barnett's reproduction of a page from chapter 3 of *Alice in Wonderland* with phototypesetter commands

book.[17] Some sort of cartridge or carrier frame containing a font was installed in the phototypesetter (shown at the top left in the right side of the figure). There was a piece of film for each character in the font. Mechanically, the piece of film for a character was placed in front of a projector light and projected on the photosensitive paper (or, later, on a screen). A lens could be adjusted to create larger or smaller sizes of the characters in the font. Finally, the mechanics were there to move across and down a page.

According to Seybold,[18] there were several generations of phototypesetting machines. The first generation was an adaptation to the prior technology, for example taking in paper tapes that previously would have gone to linotype or monotype casters. The second generation was "purpose built" for phototypesetting, while still creating images on photosensitve paper that was then photographed to make a plate. With offset lithography as a typical means of printing,[19] the step was relatively easy to go from creating plates via film characters projected on photo sensitive media to creating plates from digitally drawn images on CRTs (third generation phototypesetters). As computers and computer software became more powerful, the phototypesetting era moved toward its conclusion. (See also the encyclopedia chapter on computer-aided composition by Arthur Phillips[20] which covers the phototypesetting era???[and more]; big chunks of this are available by googling.)

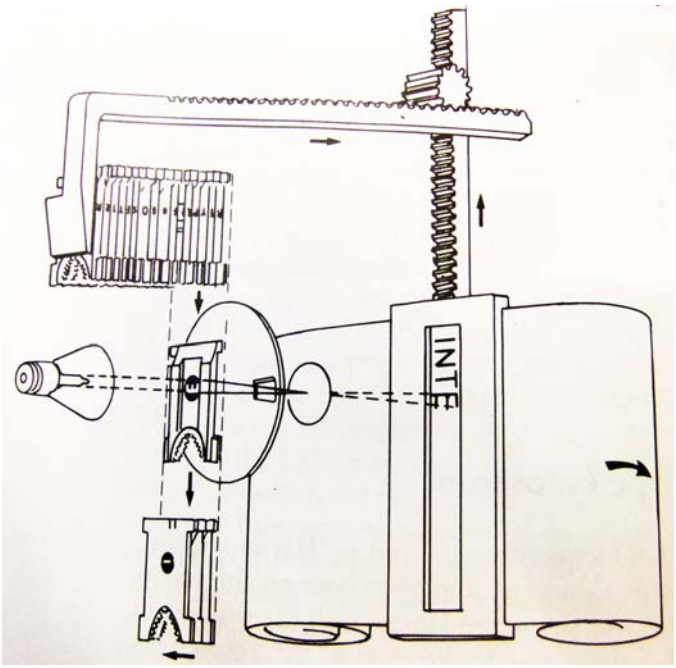

Figure 3: Phototypesetter diagram

The typical workflow in a big city newspaper (as we have read in novels, seen in movies, etc.13) was as follows: a reported covered the new (and perhaps phoned it to a rewrite person); the reporter or writer typed a draft of the story on a keyboard; a story editor reviewed and changed the draft; sometime during the day there was a meeting of what would be placed in sections and on pages of an issue of the paper; linotype/monotype/terminal operators produced columns of type; a layout editor (with help from a strong layout man in the days of heavy steel frames containing columns of lead type) would produce a page of type and images; there would be a test printing and maybe editing of the pages as necessary; depending on the era and equipment, an offset plate or a letter press plate was created (typically one or more stereoplates of each metal plate were produced.[21]

According to one of the videos listed earlier, at one point the New York Times had 150 linotype machines. In addition to in-house staffs of typesetters (e.g., at newspapers), there was a vast typesetting service industry (over 14,000 in 1988[22] supplying typesetting to publishers beyond their in-house capabilities.

Evolution to digital

Let me summarize the evolution of the technology.

- keyboard as an integral part of a linotype caster
- monotype keyboard punching paper tape to drive monotype caster
- teletypewriter tape from various sources to linotype caster
- ditto for monotype caster
- computer keyboard to produce paper tape to control phototypesetter
- phototypesetter accepts all prior forms of punched tape input
- phototypesetter projects to CRT rather than on paper
- laid out pages on paper or screen could be photographed to produce printing (lithograph mostly) plates
- editor/markup computer programs could drive phototypesetters, etc.
- on a different path, justifying or composing typewriters (e.g., “cold type” equipment such as the Varityper the the IBM Composer) could produce galleys for photographing

- editor/markup computer programs could send typeset text to their local terminals or line printers
- editor/markup computer programs could produce digital printing formats

The video in footnote 13 shows evolution at the *New York Times* with the linotype operators moving to use of display keyboards

A sketch follows of the evolution of typesetting methods at the American Mathematical Society (not a newspaper, but an organization close to our TEX world).

The AMS is where Knuth first officially presented his TEX system to the mathematics world, at the AMS 1978 Gibbs Lecture: [23,24] Knuth also arranged for the AMS to have the trademark for TEX®. Barbara Beeton provided me a detailed list (2016-08-01 email) which I have slightly paraphrased and reformatted (I still am using mostly Barbara's words); I added the footnotes.

> In the pre-computer era, most journals and books were sent out for Monotype composition. At some point some journals were brought inside for "typewritten" preparation on Varityper and IBM Composer machines (without justification, which was too complicated for math). In time, a Photon 200 was used for "direct phototypeset" of books; and the Combined Membership List, CML, was prepared from trays of edge-punched cards, fully justified (society codes flush right on the last line of each entry) with printing to a Friden Justowriter. Phototypsetting was also done from paper tapes prepared on computers. In house the paper tapes were used by a Photon 713 for the CML and some indexes (the 713 wasn't up to the job of doing books and journals). Paper tapes were also sent to remote phototypesetters (where time was rented) using software by Science Typographers Inc.[25] and Composition Technology Inc. When TEX became available, When TEX became available, it was originally used for in-house composition, mostly for proof copy but some camera ready copy; it also was used with in-house digital Alphatype and Autologic phototypesetting systems. Over time, all composition of books, journals, and the CML was shifted gradually to TEX (the CML first, along with other "administrative" publications, for which data came from databases). The in-house typesetting crew had to be trained in TEX to get decent math, and an effort was made to keep the contents of journal issues uniform. Proof copy was first produced on a Benson-Varian, system and then on various laser printers, which are still the class of proof devices in use, and final copy now goes (as PDFs) to a plate-maker.
>
> The AMS has its own print shop, which can produce folded-and-gathered signatures up to 32 pages. there is also a perfect binder, which is used for soft-bound books and journals. hard-bound books are printed in-house and jobbed out for binding. some high-volume journals are sent out as pdf files to a printing service.

As computers were used more, the ability increased to combine the traditional printing functions (line setting, justification and hyphenation, and pagination) with the editorial and greater newspaper processes and workflow (using email, central databases that could be access from remote terminals, etc.). It still can require a lot of people.[26]

There were a number of key newspapers and vendors who pioneered and spread the increasingly digital technology, such as the *Minneapolis Star Tribune* and the Atex company. Recounting that history and getting a glimmer of contemporary practice is a project for another day. *If someone already knows of such a history, please tell me.*

2 Typesettng and composition systems for individuals

Lots of history has been written about newspaper and book production and also about the commercial path to what we now know as desktop publishing, i.e., from Wang-like word processing systems through Indesign. I am going to start at a different place and

emphasize a somewhat different thread — the thread that started with interactive use of computers.

Interactive computing

Undoubtedly there were many early individual hacks that used computers to format lines of text. One particular early path is a software development activities for a series of various computers in and around Cambridge, MA, supporting interactive individual use in an era when batch processing systems were the norm. This series of computers included MIT's Whirlwind (1948–51), TX-0 (1956), TX-1 design (never built at MIT), and TX-2 (1958); it was shocking to various members of the computing world at the time that individuals were allowed to sign up for hours at a time to use these computers interactively. The TX-1 design led to the PDP-1 computer (1960–61) at Digital Electronics Corporation (DEC) and time-sharing system development at BBN (Bolt Beranek and Newman Inc., Cambridge, MA) and MIT (the first couple of PDP-1s were delivered to BBN and MIT). Also at MIT, Fernando Corbató who had used Whirlwind interactively, developed the Compatible Time Sharing System (CTSS, 1961)[27] for the IBM 709 and 7094???; this was the first really production time-sharing system. Also, in the Cambridge region, IBM (including individuals from the nearby CTSS effort) were developing the Control Program Cambridge Monitoring System (CP/CMS) time-sharing system for the IBM 360. In time, DEC developed the PDP-6, which was turned into a time-shared system (ITS) at MIT, and the PDP-10 with its DEC developed TOPS-10 time-sharing system. BBN developed the TENEX time-sharing system for the PDP-10, and that later evolved into TOPS-20 at DEC (BBN had been using the Berkeley time-sharing system, developed for the SDS-940 computer by Project Genie at UC Berkeley, after as it outgrew the PDP-1 and before it obtained its PDP-10s.) Starting before the TENEX effort at BBN (and perhaps finishing after), MIT also developed MULTICS (the design of which influenced the TENEX developers as did their experience with the SDS-940).

With the ability to sit at the console of a terminal of interactive systems such as those mentioned above, users could do interactive software development (the edit, assemble/compile, run cycle) and could apply their computers to other interactive tasks, including interactive preparation of documents to be printed. Many of these systems were early entries in a series of text editors and text formatting programs. A representative set of such systems is shown in Table 1.

Table 1: Interactive page layout systems (approx. start dates)

RUNOFF (1964), its predecessors, and its successors
The roffs (from 1969) — plain-text based
Pub (1971), TEX (1978-1982), Scribe (1980), and Texinfo (early 1980s), LATEX (early 1980s) — plain-text based
Wang (1971) and other stand-alone word processors
Bravo (1973), WordStar (1978), Word Perfect (1979), and Word (1983) — WYSIWYG word processors
Interleaf (1985), PageMaker (1985), FrameMaker (≈1985), QuarkXpress(1987), InDesign (1999) — DTPs

In the following, I will touch on each of the groups of interactive page layout systems listed in Table 1. With one exception (Wang, etc., initially) all these system ran or run on general purpose computers.

RUNOFF and its predecessors and successors

The initial version of RUNOFF was developed by Professor Jerome Saltzer on the CTSS system at MIT. It was implemented in the MAD language. It is arguably the first significant

text formatting program; it certainly has many important descendants. The RUNOFF description starts on page 10 its manual[28] (half of each of the documents cited in this paragraph is for the TYPSET program, a line editor for preparing files to be processed by RUNOFF.) A summary of the RUNOFF commands is shown in Table 2 (taken from page 17 of the manual). From the table, you can understand the limited but still useful nature of RUNOFF. See Figure 4 for example output from the original RUNOFF. The section for

Table 2: Summary of RUNOFF Control Words

abbreviation	control word	automatic break
.ap	.append A	no
.11	.line length n	no
.in	.indent n	
.ss	.single space	yes
.ds	.double space	yes
.bp	.begin page	yes
.ad	.adjust	yes
.fi	.fill	yes
.nf	.nofill	yes
.nj	.nojust	yes
.pa	.page (n)	yes, if n
.sp	.space (n)	yes
.he	.header xxxx	no
.br	.break	yes
.ce	.center	yes
.li	.literal	no

RUNOFF in the CTSS Programmer's Guide[29] includes a few more commands not listed in Table 2, such as .odd page, .paging mode, and .heading mode.

Saltzer acknowledges influence for RUNOFF of the following people and systems:[30] • J. McCarthy, Colossal typewrite; • S. Piner, Expensive Typewrite); • P. Samson, Justify; • Comp. Center staff Input, Edit, and File; • M.L. Lowry, Memo, Modify, and Ditto); • M. P. Barnett, Photon; • V. H. Yngve, Comit and Vedit; • R. S. ??, Madbug; • A. L. Samuels, Edits; and • F. J. Corbato, Revise. All of the projects Saltzer listed were more or less MIT related, and I know something about several of them.

Contrary to what the Wikipedia and Saltzer say, Ed Fredin (who had hired McCarthy as a consultant to BBN) says[31] that Callosal Typewriter, a basic paper tape editing program, was written by Rollo Silver for the BBN PDP-1. The PDP-1 undoubtedly had a Friden Flexwriter connected to it, which would have offered possibilities for typing out good looking documents. The program was in the DECUS library.[32]

Expensive Typewrite was originally written for the TX-0 in 1960[33] and then run on MIT's PDP-1. It was basically a tape editor that could do inputs from and outputs to either paper tape or magnetic DEC tapes. The actual editing was done in a text buffer.[34] Expensive Typewriter was the predecessor program to TECO[35] which was the predecessor of Emacs.

I'll skip over discussing Justify (or TJ-2),[36]; Input, Edit, and File;[37] and Memo, Modify, and Ditto.[38] As I understand it, Justify worked on virtual paper tapes and the others worked on virtual card decks.

My impression is that Barnett's early 1960s experiments at MIT with computer typesetting didn't have much practical influence. He did write a book[39] which is widely cited (less for content, I think, and more for propriety of noting prior workers in the field). Barnett

CC-244 MAC-M-193

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

Project MAC

TO: CTSS users.

FROM: J. H. Saltzer

SUBJECT: TYPSET and RUNOFF, Memorandum editor and type-out commands.

DATE: November 6, 1964

The command TYPSET is used to create and edit 12-bit BCD line-marked files. This command permits editing and revising by context, rather than by line number. The command RUNOFF will print out a 12-bit BCD line-marked file in manuscript format. RUNOFF contains several special features not available with the DITTO command, including type-justification.

These two commands provide an alternative to the MEMO, MODIFY, and DITTO commands, and are intended to provide experience with a different approach to editing symbolic files.

This memorandum was prepared with the aid of these two commands.

Figure 4: Saltzer used RUNOFF to produce the RUNOFF manual

was working with a Photon 560 "film setting" machine. Text and instructions (Figure 2) were typed on a Friden Flexowriter that output the typed characters on paper tape. This paper tape was converted by a program (Barnett's TYPRINT) running in MIT's IBM 709 computer into another paper tape in a format understandable by the Photon 560. Another program in the 709 (TABPRINT) could input papers tapes from non-Flexowriter sources. Barnett's book is a useful reference for what happened before his work and suggests state-of-the-art when he was working

I know nothing of the systems noted in the last three lines of Saltzer's list.

In addition to being the beginnings of text formatting programs, which are the subject of the rest of this section, a whole series of text editors developed from the work started on the early interactive computers we have been discussing. Eric Fisher made an interesting chart in November 2000.[40]

The first successor program to RUNOFF on CTSS was RUNOFF rewritten in BCPL and ported to run on Multics.[41,42] This was done by people from Bell Telephones Laboratory who were then part of the Multics project. RUNOFF (in BCPL and otherwise) was moved to other computers and also became the earliest version of roff (described below). Larry Barnes created RUNOFF for the SDS-940 project; the manual[43] says Barnes' runoff was "inspired by that of Saltzer." The 940 version of RUNOFF was ported by Bob Clements to run on the DEC TOPS-10 system. Multics RUNOFF was rewritten for TENEX by Bernie Cosell and called MRUNOFF (Figure 5). And there were plenty of other imitations and derivatives of RUNOFF on a variety of machines, for instance the Script Edit Module by Stuart Madnick for CP/CMS. Various later versions of RUNOFF has a macro capability or at least the capability to conditionally include a file of predefined text.

The roffs

The roff system[44] originally was a rewrite of RUNOFF, and then it was greatly expanded and ported to various computers, originally by Bell Laboratory people. This more powerful system existed in two forms: nroff for conventional office printers, and troff to drive the phototypesetter at Bell Labs.[45] As time went by various preprocessors were added to the nroff/troff package, and a device independent version was create—ditroff. The Free Software Foundation eventually created a version of ditroff, Groff, which remains in widespread use. (At TUG2016, Steve Izma made sure I knew about the SoftQuad derivative of troff, SQtroff, about which I asked him for more information.[46,47])

Pub, TEX, Scribe, and Texinfo

Along with the roffs, the systems described in this section are systems where one types formatting markup into a plain text file. All these were big steps past RUNOFF in power and precision. I believe they all had reasonably powerful macro capabilities. TEX and Texinfo, along with groff, are still in widespread use today.

Larry Tesler's Pub was based on the concept from Les Earnest, and Tesler calls it a scripting language that produces paginated output. It was developed for use in the Stanford AI Lab, or SAIL (where Knuth developed the original version of TEX in the SAIL programming language). Before very long, Les Earnest tried to create a business based on Pub, and Larry Tesler went to Xerox Xerox PARC where he worked closely with the group that developed the WYSIWYG Bravo (in a later subsection). Pub had a math mode, and it has a wonderful set of annotations for its 1972 manual with lots of history about Pub, RUNOFF, etc.[48] Purportedly, Pub at least partially motivated Scribe and TeX; in any case, Tesler says Don Knuth and Brian Reid built better systems

Don Knuth's (and his students, with help from a few experts on type) created TEX and METAFONT (a type design system) in the later 1970s and early l980s. It is hard to imagine a computer system that has had more written about it than these systems and their follow-on systems. Anyone unfamiliar with the TEX world, could start by

1. PROGRAM OBJECTIVE AND TECHNICAL NEED

1.1 Defense Program Addressed

The ARPA program had the following objectives: (1) To develop techniques and obtain experience on interconnecting computers in such a way that a very broad class of interactions are possible, and (2) To improve and increase computer research productivity through resource sharing. It was envisioned that by establishing a network tying IPT's research centers together, both goals are achieved. In fact, the most efficient way to develop the techniques needed for an effective network was thought to be by involving the research talent at these centers in prototype activity. Just as time-shared computer systems permitted groups of hundreds of individual users to share hardware and software resources with one another, it was thought that networks connecting dozens of such systems would permit resource sharing between thousands of users. Each system, by virtue of being time-shared, could offer any of its services to another computer system on demand. The most important criterion for the type of network interconnection desired was that any user or program on any of the networked computers be able to utilize

Figure 5: Text I formatted in MRUNOFF in 1978

reading Nelson Beebe's two retrospect papers.[49,50] The book to buy (not a user manual) is *Digital Typography*.[51] although a good bit of what is in this book is also in the *TUGboat* archive[52,53] The user groups and wider community of the TEX world represent an almost 40-year open-source project, as Knuth chose to make TEX available free to everyone (as he describes in a short video[54]) Undoubtedly thousands of people over the years have created different distributions of TEX, various TEX "engines," many "formats," and a vast number of packages all of which make TEX increasingly powerful, increasingly widely usable, increasingly customizable, and somewhat able to keep up with the rapidly changing computing and publishing worlds. Probably the most widely used "format" built upon TEX is LATEX. LATEX may not be an innovation in itself (see Scribe below), but it was a breakthrough in terms of mathematicians, economists, scientists, etc., and their assistants doing their own technical "typing."

Brian Reid's Scribe started as his thesis research.[55,56] He gave some credit to TEX. One of its notable development directions is separation of structure and format, which is said to have later influence the development of LATEX. Reid sold Scribe to a company which charged for copies of the software. Copies were available initially for free on the condition that, to keep Pub running, it had to be paid for within 90 days (and Reid fixed the program so it would stop working if not paid for in time).

The Texinfo manual[57] says, Texinfo was "Texinfo is the official documentation format of the GNU project. It was invented by Richard Stallman and Bob Chassell." and was "loosely based on Brian Reid's Scribe and other formatting languages of the time" Purportedly Stallman (founder of the Free Software Foundation) was unhappy with Scribe being charged for. An advantage of Texinto is uses a single source file to produce output in a number of formats, both online and printed (dvi, html, info, pdf, xml, etc.) It is also used for the LATEX Reference Manual project described by Jim Hefferon at TUG2016.[58] (Karl Berry was involved with Texinfo's maintenance for many years.)

Wang and other stand-alone word processor

The Wang word processors[59] were early and very popular stand-alone systems: the model 1200 did its editing off magnetic cassette tapes[60]; the model 2200 ran on a general purpose computer but was still a single user system.[61] With the introduction of personal computers, Wang's word processor business collapsed although it made its own person-computer-based system. There were many other word processing systems similar to Wang's, and they all suffered the same fate with the advent of the personal computer.

Bravo, WordStar, Word Perfect, and Word

These are all editing and formatting systems — word processors — that ran on personal computer (or the proto PC in the case of Bravo). They were the successors to stand-alone word processors such as the Wang, and they were WYSIWYG. The latter three each dominated the market for a while.

The Bravo development (ca. 1973–79) was led by Butler Lampson and Charles Simonyi of Xerox PARC. It ran on the Alto computer, whose windows-and-mouse graphical user interface was influenced by Doug Engelbart pioneering work at the Stanford Research Institute. Lampson and Simonyi came to PARC from UC Berkeley[62] where Lampson was a key developer on the SDS-940 time-sharing system project. Alto was the prototype personal computer with a graphical user interface — the system Steve Jobs saw that influenced the design of the Mac.[63] Bravo was a WYSIWYG system where a mouse could be used for scrolling up and down; was command driven, e.g., D for delete, U for undo, I for insert; but without markup.[64] It was like using Word with mouse selection and keyboard shortcuts such as CNTL-I; CNTL-CR to end paragraph. It also had templates (style sheets). (Larry Tessler went to PARC after SAIL and was close to the Bravo project and developed a related system called Gypsy. In Gypsy he implemented ideas for copy/cut-and-paste that

he conceibed while developing Pub. Gypsy also moved away from having an editing mode; in other words, it worked like how we work in Word today.)

Rob Barnaby[65,66] designed WordStar for the commercial work and to be portable. It had cursor navigation using keyboard shortcuts other than arrow keys. WordStar dominated market in first half of 1980s, and introduced Word-like word processing to the masses. (Rob Barnaby has previously been on the development team of the TENEX time-sharing system where he used,and I believe improved, RUNOFF.)

WordPerfect was developed by Bruce Bastian and Alan Ashton's with its commercial release late in 1983. By 1986 it had supplanted Wordstar as the most popular such word processing system, having for instance automatic footnote and endnote handling. Starting in the late 1980s, Word Perfect could alternate operating in WSYIWYG mode or in "reveal code" mode where the markup could be seen.

Charles Simonyi took the Brovo ideas to Microsoft and with Richard Brodie developed Microsoft Word. It was a good system but not popular on DOS. My memory is that it became more popular on MACs, and then very popular with Microsoft Windows and bundled in Microsoft Office; and it now dominates the world, including the world of book and journal composition. My understanding is that there are over a billion users in the installed base for the Office Suite that includes Word.

There are lots of other word processors used by smaller groups of users, for example, systems particularly oriented to the organizational needs of professional writers, e.g., Scrivener and Note Bene; and users who refuse to leave an earlier system (such as WordPerfect) to move to Word.[67]

Interleaf, PageMaker, Framemaker, QuarkXpress, InDesign

The systems in this subsection are what we now call desktop publishing systems — DTPs.

Interleaf was aimed at technical publishing and distribution with integrated text and graphics. It purportedly took ideas from TEX.[68] (Some of you will remember author Tracy Kidder attending the 2014 TUG conference in Portland. Kidder has written a book called *A Truck Full of Money* (due out in September) that talks a good bit about Interleaf as a development organization and business.[69])

Paul Brainerd is the guy who was behind PageMaker.[70] Out of college he worked in operations for the *Minneapolis Star and Tribune* while they converted from hot type to computer-based typesetting. Atex was a key supplier. Next Brainerd went to Atex and stayed there until it was sold. Then he started Aldus, which created perhaps the first DTP; in any case, he is credited with coining the term DTP. PageMaker was used by professional and amateurs book designers and others. Aldus was eventually bought by Adobe.

Charles Corfield developed FrameMaker which was a competitor of Interleaf and was aimed at publishing large and very large and complex documents. Later the company tried to also compete in the home DTP market which was a loss of business focus and led to near insolvency. Adobe bought the product, refocused on the business market, and the product still has a significant following today.[71]

QuarkXpress is aimed at the professional typesetting and page layout market and out competed PageMaker in that market. For a while it was the industry standard.[72]

As noted above, Adobe acquired PageMaker and then PageMarker lost its market to Quark. InDesign was developed to be a successor to PageMaker. My impression is that InDesign cut deeply into Quark's market, although I think there is still competition between Quark and InDesign today. InDesign is used by professional book designers and typesetters (and by amateurs who want good typesetting and would never think of using LATEX).

A couple of additional notes

PostScript came on the scene in the early 1980s (continuing work Chuck Geschke and John Warnock had started at Xerox PARC). I have heard PostScript being described as a

page description language or as a language for creating vector graphics. It was originally aimed at driving printers and first became well known by its use in Apple's computers. PostScript (and EPS and PDF) have clearly changed the way the typesetting and printing worlds work.

Interactive technology has become ubiquitous in both the world of word processing and desktop publishing (this section) and in the world of newspapers (the prior section). The two worlds have substantially merged.

3 Algorithms for typesetting and composition

There are lots of areas for good algorithms that computers can apply. Some that come to mind for me are: how letters, etc., are drawn, e.g., pens/strokes, outlines, and so forth; simple line breaking and hyphenation; justification and inter-word spacing; line breaking based on paragraphs or pages (rather than simply line by line); microtype for glyph variation, kerning, and protrusion, e.g., *hz* algorithms; boxes and glue model; floats, grids, and other positioning issues; page layout models. The National Bureau of Standards produced good summary of the state of the art in 1967.[73] Seybold's 1984 book[17] gives the state of the art nearly 20 years later as does Enlund's paper.[74]

The topic I'll discuss at some length here is justification. I'll recount a bit of the history below.

Pre-printing, hand, linotype, and monotype justification

I have read that in the days before moveable type printing, scribes and calligraphers did justification through the use of various sizes of interword spaces (not necessarily same sized), abbreviations, ligatures, typographical flourishes, and so forth. In the earliest days of moveable type, printers tried to mimic justification by calligraphers by using the same sorts of techniques and even slightly differently sized letters to justify lines.

As letterpress printing became widespread, economics came to dominate aesthetics, type manufacturing and typesetting became businesses, and justification was mainly done in composing sticks with spaces (quads, slugs) of more or less standard width.

All of the above work depended on the judgement of the scrivener or typesetter and seem to have been quite tedious to accomplish.

When the linotype was invented in the later 1800s, the decision of when to break a line still resided with the human operator; but the machine could mechanically insert equal size interword spaces throughout a line. As the operator transcribed manuscript pages onto the keys of the keyboard, molds (called "matrices") for the different characters type slid down channels into a line of type molds except for interword spaces where the operator inserted a "spaceband" wedge in the line (see photo at `http://tinyurl.com/spacebands`; the molds for each character in the line are in the front face in the photo). When the operator decided that another word could not fit on the line, the spaceband wedges were pushed up uniformly (shown in the animation at `http://tinyurl.com/spaceband-animation`) as far as they could go until the right most character hit the stop defining the line width, thus creating equal size spaces between the words. Then hot lead was poured onto the line of molds for characters, and the cast line-of-type moved from into a galley tray (and the molds and spacebands traveled back to their storage areas).

The monotype came from the same 1800s time period as the linotype. The column width was set manually into the machine. As the operator typed on his keyboard, the characters to be printed were punched on paper tape. When the operator decided another word could not fit in the line, he used a nomograph to look up the proper interword spacing as a function of the number of interword spaces and the space left in the line. He typed the spacing information on his keyboard, and it was added to the paper tape (I guess in effect becoming an end-of-line indicator). After typing an appropriate number of lines, the paper tape was removed from the keyboard unit and fed backward into the type caster

unit. The caster read the interword-space-size information for each line from the paper tape, and then inserted the appropriate amount of space between words as the line was formed. Completed lines went into the galley tray effectively from the bottom up and the last character of each original keyboard line ending up at the left of the galley tray. You can perhaps imagine how this all works; I created an example (Figure 6) to get it sorted out in my mind.

Justifying typewriters

As printing moved into the photographic era, several "justifying" or "composing" typewriters were invented that helped the human operator type columns of justified lines that could be photographed for conversion into printing plates. One of these was the Varityper Office Composing Machine (photo at `site.xavier.edu/polt/typewriters/varityper.html`).

The Varityper could be loaded with "cartridges" (perhaps not the correct word) for hundreds of type styles in dozens of language (including proportional spacing) before IBM Selectrics (and the Selectric Composer) had their changeable type balls. With an extra wide carriage, the operator typed a line for a column of type and the machine mechanically kept track of the number of interword spaces typed. When the next word could not fit within the line, the operator typed a tab which mechanically recorded the amount of space left within the line width and moved the carriage far enough right so the author could retype the line of type. During this retyping, each interword space typed resulted in a the carriage moving enough right to leave a interword space such that all of the interword spaces were the same width and in total created a right justified line.

We believe a cam and the follower lever arm recorded the number of interword spaces in the line as originally typed.[75] Each space typed moves the follower level over a position so there were more positions on the cam for it to click on as spaces happened during the second typing of the line. (We don't yet understand how the machine recorded the amount of space left in the line during the first typing.)

Computerized justification

The earliest computer-based justification of which I have read was at Newcastle University.[76] In the late 1950s and early 1960s, the project used a Ferranti Pegasus computer and then an English Electric KDF9 computer to generate nicely formatted text for output via a paper tape to printing devices, in particular to a monotype machine.[77]

Michael Barnett's book[16] Barnett provides an example of justification at Newcastle Figure 8) and describes (pp. 174-176) the algorithm the Newcastle project used for justification.[78]

> To avoid splitting words in line endings whenever possible, the positions of line breaks within a paragraph are not finalized until the paragraph is completely processed. Whenever an interword space is encountered, the minimum output space is allocated to it tentatively,. When overset occurs, the word that is being process is left for inclusion in the next line. A test is made to determine if the interword spaces on the line that has just been completed can be expanded to fill the requisite measure without exceeding the limit that has been specified for the distance between words in a typeset end product. If the interword space would become excessive, a test is made to determine if the last word on the previous line can be brought down without making the spaces needed to justify that line become excessive. If the spaces would remain within the allowed limit, the word is brought down and the processing continued from that word. If the spaces would become excessive, the previous lines are tested one by one, going backward, until one is found in which the last word can be brought down to the next line and the residual material expanded without exceeding the allowed interword spacing. Processing is then restarted from that word. If the beginning of the paragraph or a line that ends with a hyphenated word is encountered before a line that ends with a word that can be brought down, the process is abandoned, and the word whose overset initiated the search is hyphenated.

[1. The sample paragraph]

Then Oppenheimer gave all the writing samples (the original, simple ones and the modified, flowery ones) to 71 students to evaluate. The result? As the grandiosity and complexity of the language increased, the judges estimation of the intelligence of the authors decreased.

[2. The words that fit on a line, showing the number of interword spaces and number of unused spaces]

```
Then Oppenheimer gave  |2+2|
all the writing samples|3+0|
(the original, simple  |2+2|
ones and the modified, |3+1|
flowery ones) to 71    |3+4|
students to evaluate.  |2+3|
The results?  As the   |3+4|
grandiosity and        |1+9|
complexity of the      |2+7|
language increased,    |1+4|
the judge's estimation |2+1|
of the intelligence of |3+1|
the students decreased.|2+0|
```

[3. What actually goes on the keyboard unit output tape]

```
Then Oppenheimer gave|2+2|
all the writing samples|3+0|
(the original, simple|2+2|
ones and the modified,|3+1|
flowery ones) to 71|3+4|
students to evaluate.|2+3|
The results? As the|3+4|
grandiosity and|1+9|
complexity of the|2+7|
language increased,|1+4|
the judge's estimation|2+1|
of the intelligence of|3+1|
the students decreased.|2+0|
```

[4. How the tape goes backwards into the caster]

```
|0+2|.desaerced stneduts eht|1+3|fo ecnegilletni eht fo
|1+2|noitamitse s'egduj eht|4+1|,desaercni egaugnal
|7+2|eht fo ytixelpmoc|9+1|dna ytisoidnarg|4+3|eht sA ?stluser ehT
|3+2|.etaulave ot stneduts|4+3|17 ot )seno yrewolf
|1+3|,deifidom eht dna seno|2+2|elpmis ,lanigiro eht(
|0+3|selpmas gnitirw eht lla|2+2|evag remiehnepp0 nehT
```

[5. What comes out of the caster and what prints]

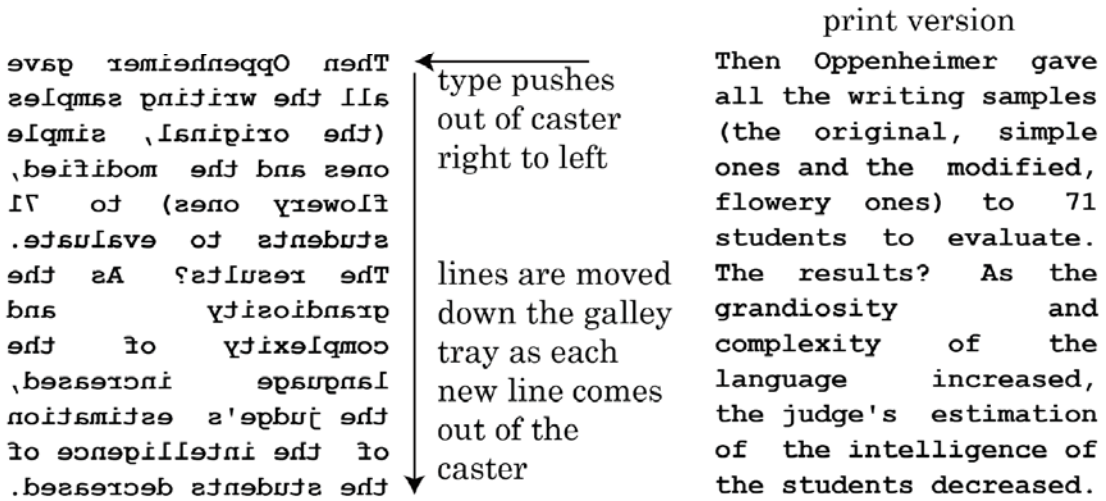

Figure 6: Monotype justification example

Figure 7: Varityper cam (from photo by Richard Polt)

Earlier in this paper, I noted that paper tapes driving linotype and monotype casters provided an opportunity for primitive-to-a-little-more-sophisticated computers to take in a paper tape with lines of text from a keyboard or communications circuit and to put out another paper tape with justified lines for the casting machine. As computers increasingly came on the scene, the keyboarding could be done into the computer and a paper tape with justified lines could be output to the typecasting device (the earliest phototypesetters in fact took in paper tapes in the formats of the lineotype and monotype machines). Over time paper tape was dropped in favor of sending bits over wires. A variety of different more or less good justification algorithms were used in different systems.

While that early Newcastle justification algorithm was working on a paragraph of lines, many text processing systems use a very simple justification method called the "greedy" or "first fit" method for breaking a paragraph into lines. Successive words from a paragraph are brought one after another to a line being formed until the next word won't fit within the specified column width. Then a line break is inserted and then next line is formed in the same way. (This is probably the approach many of us use when writing a letter on a piece of stationary paper.) This method tends to give a total amount of interword space in the paragraph that is greater (and thus probably uglier) than if a more sophisticated line-breaking algorithm was used.

Let's look at a trivial example of greedy line breaking of the paragraph "Canada and beers go well"[79] and a line width of 10 monospace characters.[80] Using the greedy algorithm, the line breaking happens as shown on the left side of the following example:

```
Canada and          Canada and
beers go            beers   go
well                well
```

If we count the spaces left at the end of each line, the first line has 0 left over spaces, the second line has 2 left over spaces, and the third line has 6 left over spaces.[81]

PROGRAMMING

A computer is very well adapted for dealing with justification since the process is largely one of counting spaces, adding together the widths of characters forming a word and subtracting these widths with an allowance for intervening spaces from a specified line length. When the residual line length becomes too small to allow another word on the line it must be distributed, equally if possible, over the existing spaces.

It should be emphasised that the set of instructions (called a programme) which causes a computer to carry out such a task, is quite independent of the actual numerical values of line length, character widths or the permitted range of widths for spaces between words. These numerical values together with a text may be regarded as the data to be processed by the programme. This means that basically the same programme suffices for any system of typesetting, for any fount or for any length of line.

Although justification is straightforward the related problem of word splitting is not. Many of the rules for splitting words are based on aesthetic considerations and are open to dispute, while even the so called 'logical rules', obtained from an analysis of letter sequences in English words, have many exceptions. The use of a stored dictionary to solve the word splitting problem tends to be uneconomic on a computer although this method can in theory cover all cases.

In view of the undesirability of splitting words quite apart from the difficulties involved in doing so it is surprising that little attention has been paid so far to the possibility of using computers to reduce the frequency of word splits. Very nearly all the work at Newcastle so far has been in exploring various aspects of this possibility.

The programme used for computer typesetting strictly comprises three separate programmes, an input programme, a composing programme and an output programme. The input programme is simply concerned with reading text from punched paper tape and storing it inside the computer in a form suitable for use by the composing programme. If text

Example 8.6. *A product of the Newcastle University computer program for typesetting.*[25]

Figure 8: Newcastle example from Barnett's book

Another way to break this line is as follows:

```
Canada                Canada
and beers             and  beers
go well               go    well
```

In this second case, the first line has 4 extra spaces, the second line a 1 extra space, and the third line has 3 extra spaces. If we add up the extra spaces across all three lines, both examples have 8 extra spaces: the alternate approach to the greedy method does not produce less extra spaces. One might also think that the justified version of the first example is more attractive. However, suppose one wants a way to have lines with more extra spaces count disproportionate to lines with less extra spaces. One way to do this would be to calculate the sum of the squares of the extra spaces in each line which in these examples are 40 ($4 + 36$) and 26 ($16 + 1 + 9$)—the second example is much better using the sum-of-squares measure.

In a real paragraph with lots of words and quite a few lines, looking at *all* the places one might break lines between words could take a lot of calculation. In the line-breaking algorithm used by TEX (and many other systems since TEX) that was created by Michael Plass and Donald Knuth,[82] they use an optimization technique known as dynamic programming to reduce the size of the calculation. I won't go into the details of this excellent algorithm; you can find plenty of discussions of it by googling on "Knuth Plass line breaking,"[83] and Michael Plass's thesis on the topic is readily available.[84] TEX users reading this will not be surprised that such sum-of-the-squares calculations are part of TEX's "badness" about which we are always seeing warning messages.

Interestingly, if one googles on "dynamic programming," quite a few of the results take you to a discussion using line-breaking-into-paragraphs as the example of a use for dynamic programming.[85,80]

HJ and further sophistication

In the typesetting industry, HJ stands for hyphenation and justification although the two issues don't have to go together; one can do justification without using hyphenation, or can use hyphenation without justification. Knuth's preliminary description of TEX (`www.saildart.org/TEXDR.AFT%5B1,DEK%5D`) includes his early thoughts of justification and dynamic programming (above) and hyphenation (below). Mohamed Elyaakoubi and Azzeddine Lazrek have written a paper[86] that begins with a nice short sketch of the history of J and H.

Several basic approaches were used over the years for hyphenating end-of-line words. One approach was to try to implement the rules for hyphenation,[87] such as hyphenate between double letters (dip-ped) and breaking words at morpheme boundaries (cran-berry); this approach could have some quite complex if-then decision structures (Figure 9), required a long list of exceptions, and didn't deal with words which are hyphenated differently depending on use of the word (Frank Liang[88] and Knuth[89] give the example of the verb re-cord and the noun rec-ord). Another approach is to have a big dictionary with syllables marked in every word; however, this doesn't work for words not in the dictionary and probably also needs implementation of some rules such as not putting "ed" along on the last line of a paragraph. Liang describes in some detail an approach once used by *Time Magazine*[90] which had tables of probabilities of possible hyphenation points based on looking a successive strings of four letters in words. The NBS report[73] has a lengthy description (pp. 44–60) of the various hyphenation methods tried through the publication date of the report.

The methods Liang described in his thesis became the method TEX used (replacing TEX's initial hyphenation algorithm) and is widely used in other systems. The pattern files for this approach to hyphenation exist for lots of languages.[91] At TUG2016 Arthur Reutenauer gave a presentation (on behalf of Mojca Miklavec and himself—"Hyphenation past and future: hyph-utf8 and patgen") on reimplementation of Liang's program for

Figure 9: Example of hyphenation rules from page ??? of Seybold's book

creating hyphenation patterns.[92] A deep discussion of ways to use hyphenation patterns can be found in Sofa's thesis.[93,94]

Once hyphenation is enabled, the number of places a line can be broken goes up significantly; it's a good thing that the dynamic programming optimization technique is used.

In 1993 Herman Zapf published a paper in which he described his *hz* ideas.[95] These are apparently implemented in InDesign. In his 2000 thesis and a 2005 paper, Hán Thế Thánh describes *hz*-like micro-typographic extensions to TEX.[96,97] This includes tiny bits of expansion or contraction of the character sizes within a line to better improve justification and reduce use of hyphens plus slight protrusion of end-of-line punctuation to make the right margin look better.

With micro-typesetting, the combinations that must be looked at in the justification calculation become even greater. The use of dynamic programming continues to help a lot. And at TUG2016 Frank Mittelbach gave a presentation called "Alice goes floating—global optimized pagination including picture placements" where he described using the same dynamic programming approach to do paragraph and line breaking across page boundaries in combination with placing floats.[98]

While Frank's work for a book was complicated (his example was *Alices Adventures in Wonderland*), automation of pagination in the newspaper sense (where pages may have multiple columns, single and multi-column headlines, ads, stories, story continuations, pointers to stories on other pages, etc. (as in Figure 10 from pages 289–290 of Seybold's book[17]) is surely a much tougher job. This is a topic to come back to at the same time as the follow-on project mentioned on page 6 just before the subsection giving an evolution example.

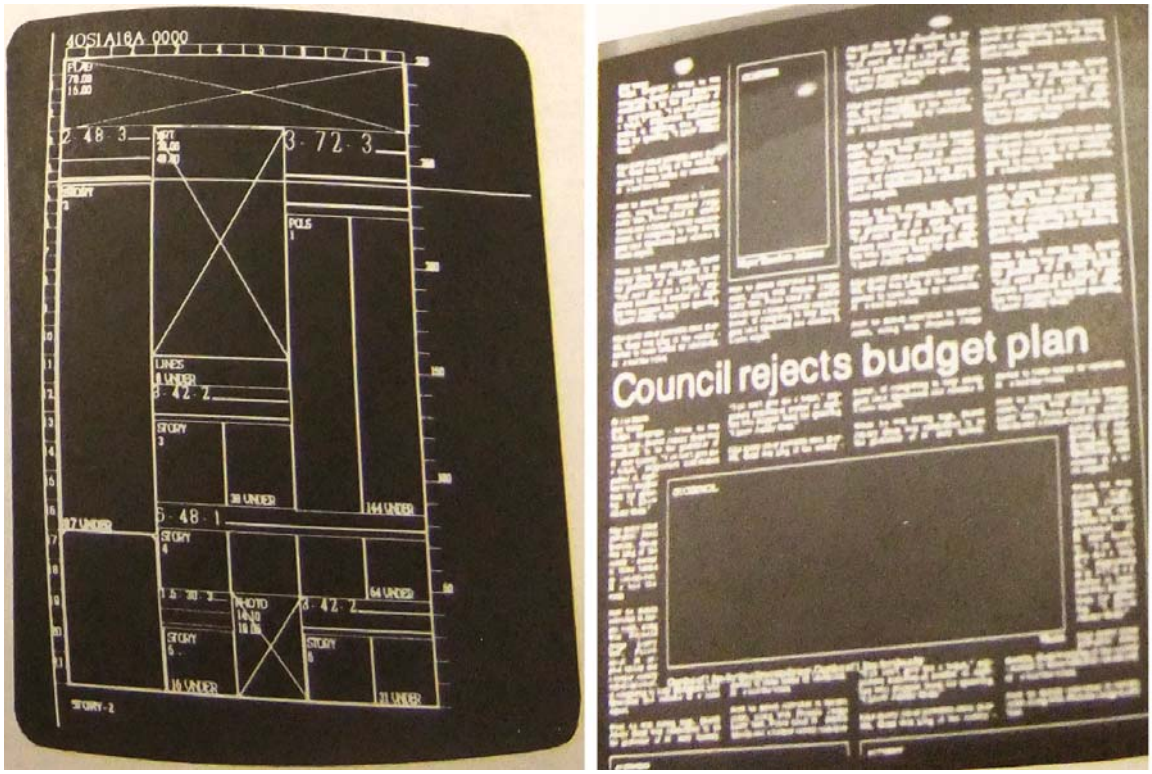

Figure 10: Screens for newspaper pagination [these images are to be replaced by non distorted versions]

4 Digital type

While this topic is important to the four-part taxonomy I used to organize my TUG2016 presentation and this paper, I didn't delve into the topic in my presentation. While I have read a bit about type design,[99] I have a lot more study to do before asserting even a basic understanding of digital type. Also, there were plenty of people at TUG2016 who were experts on digital type, the history of type, and its use, including the guest speakers. I do list here some potential subtopics of the digital type category and ask for comments, including better references:

- formats for coding characters and symbols, e.g., bitmap, type 1, type 3, truetype, opentype, etc.)[100]
- tools to help convert prior typefaces to digital or for creating new typefaces[101]
- the math for drawing glyphs (if this is separable from formats)
- issues of adapting digital type for readability,[102,103] and for various displays and printers and dealing with limits in resolution[104,105]
- how font design changed with with changes in technology[106,107,108]
- font forges and the business of selling fonts[109,110]and font protection[111] and piracy[112]

5 Reflections

As I pulled together my TUG2016 presentation (and drafted the paper version), I have thought back at what I learned from my look into the history of digital typography. Of course, I learned all the stuff I report in the paper, and a lot more stuff that didn't fit into the paper. Along the way I formed some high level observations.

- What was happening in the four dimensions of my taxonomy have become more and

more overlapping and interrelated as we have moved fully into the digital era.

- It was a continuing revelation to me throughout my study how the evolution to digital has been happening for so long; there has been so much intermixing over so many decades of mechanical, photographic, electronically digital technology..
- There is disintermediation, consolidation and despecializaton all over the place. Typesetting and design used to be separate specialties, and now every typesetter is a designer or the reverse. For my mother-in-law's oral history that my wife produced in 1982, my wife typed and pasted up a photo-ready manuscript; she went to a photo and offset vendor to have the photos sized right and turned into half-tones and to have her 8 1/2 x 11 inch manuscript pages photo reduced to 6 x 9 and for printing of a few dozen copies; and then she went to a separate place for binding. Today I produce a book doing all the photo work myself in Photoshop (and the cover in Illustrator), typeset the book myself using LaTeX, produce a ready-to-print PDF, give it to a big printing company (e.g., Lightning Source) or little print shop (e.g., Copyman in southwest Portland, Oregon), and it comes back printed and bound. At the professional level, some people claim that designers have "replaced" printers as well as typographers.[113]
- More generally I feel that the disintermediation, consolidation, and despecialization has led to a lowering of standards. The word processing and desktop publishing systems (and systems like groff and TeX et al.) put powerful typesetting tools in the hands of every amateur and full-time designer, many of whom are not truly professionals. With a little work, anyone can typeset a book or journal article. This lowering of standards is exacerbated by the myriad formats and display devices that must be supported today, for example, hardcopy, ebook, and HTML formats and digital screens of all sizes.
- I suspect that such disintermediation, consolidation, and despecialization is a done deal, and there will be no turning back in general. However, some people beyond the true professionals will still care about publishing aesthetics even if they like being able to do lots of the steps themselves. I have no illusion that the TeX world will again be important to the publishing world at large. I do look forward to seeing automatic aesthetics (such as the pagination work which Frank Middelbach described in his TUG2016 presentation) becoming more available to the TeX world — to the world in which I work; and hopefully a few ideas from the TeX world will continue to migrate into the mainstream systems as they have from time to time in the past.

One more thought on the digital world. There has never been a better time for the independent researcher. In addition to traditional libraries (and library networks with inter-library borrowing privileges), we now have vast content available via YouTube, Google Books, and professional society and journal digital archives (some open access), and we have web search engines to help us find things. Our own TUG web server makes a significant contribution in the area of digital typography.

Acknowledgments

Many people answered questions about the topic of this paper and the mechanics of producing the paper. I will just list them without details about who helped more. Those of you who helped a lot know who you are, and my appreciation is great. I also appreciate the help of everyone else.

Thank you to: Steven An, Pavneet Arora, Charles Barret, Barbara Beeton, Karl Berry, Charles Bigelow, Dan Briklin, Paris Burstyn, Paul Ceruzzi, Yosem Companys, James Cortada, Bernie Cosell, Peter Davis, Jim Dempsey, Joseph Feinstein, Christine Finn, Dan Halbert, Paul Hampson, Jack Haverty, David Hemmendinger, Steve Izma (and others who gave me comments at TUG2016), Pete Kaiser, Jeff Kenton, Valerie Lester, Dave Mankins, Alan Marshall, Ralph Muha, Petri Paju, Steve Peter, Ken Pogran, Richard Polt, Norbert Preining, Brian Randell, Arthur Reutenauer, Frank Romano, Michael Silton, Joshua Smith, Rick Smith, Bärd Sørbye, Tor Olav Stein, Heather Tamarkin. Tom Van Vleck, Dave Waitzman, Gail Walker, Wally Weiner, Keith Weinstein, Ben Woznick, and anyone who I am forgetting.

Notes and references

Introduction

[1]Robert Bringhurst, *The Elements of Typographical Style*, 2nd edition, Hartley & Marks Publishers, 2002.

[2]James Felici, *The Complete Manual of Typography: A Guide to Setting Perfect Type*, Peachpit Press, 2002.

[3]David Walden, My Boston: Some printing and publishing history, *TUGboat*, vol. 33 no. 2, 2012, pp. 146–155, tug.org/TUGboat/tb33-2/tb104walden.pdf; a revised and somewhat extended version of the 2012 paper, Printing & Publishing in Boston: An Historical Sketch, is at walden-family.com/bbf/bbf-printing.pdf

[4]My TUG2016 presentation (with the images I used from google images and elsewhere on the web without bothering to think about licensing) is at walden-family.com/dave/personal/digitype.pdf; the userid is the letter a and the password is the letter b.

[5]I derived this list from a source that I forgot to note and cannot find again.

[6]$84B in the USA in 2014 and $734B wordwide, according to an Internet web page

[7]A quite nice brief sketch of much of this history is at tinyurl.com/garamond-printing on the page "From the manuscript to the print workshop" under the following headings: the early period; innovations; mecanisation; the dawn of computerisation; into the digital era.

Newspapers

[8]Setting type by hand, Letterpress Commons, 2015, letterpresscommons.com/setting-type-by-hand/

[9]Stan Nelson, five videos from OutofSortsFilm, as updated December 2011, tinyurl.com/nelson-typecasting

[10]Newspaper Typesetting, 1884 to 1970s: Linotype circa 1960 Salesian Vocational & Technical Schools, video posted to web on August 11, 2012, bit.ly/linotype-film

[11]Stereotype printing: wikipedia.org/wiki/Stereotype_(printing)

[12]Into the mid-1900s men surely did (almost?) all of the noisy and heavy work of typesetting galleys and composing pages.

[13]David Loeb Weiss (director) and Carl Schlesinger (narrator), Farewell etaoin shrdlu, 1978, https://vimeo.com/127605643

[14]I have lost the Internet source of this image — google.images?

[15]Fred Williams, The Monotype Story, spring 1984, http://tinyurl.com/williams-monotype

[16]Page viii, Michael T. Barnett, *Computer Typesetting: Experiments and Prospects*, MIT Press, 1965.

[17]John W. Seybold, *The World of Digital Typesetting*, Seybold Publications, Inc., Media, PA, 1984., page 74

[18]Ibid.

[19]In Tim Inkster's presentation at TUG2016, "The beginning of my career," he sketched how lithography works. I also have a description at walden-family.com/bbf/bbf-printing.pdf, page 9.

[20]Arthur Phillips, Computer-Aided Composition, *Encyclopedia of Computer Science and Technology*, volume 5, Jack Belzer, Albert G. Holzman, and Allen Kent editors, Marcel Dekker, Inc., 1976, pp. 267–374.

[21]Coincidentally, the company at which I worked for 27 years doing computer and communications research and development, Bolt Beranek and Newman, for several years owned a large maker of paper mache mats for making stereoplates: walden-family.com/bbn/bbn-print2.pdf, book page 99.

[22]Frank Romano, draft from April 2016 of *History of Desktop Publishing*; Frank provided me with drafts of four chapters: (a) It gets personal — software; (b) Imagesetting and the pre-press revolution; (c) DTP destroys the typesetting industry; (d) Typesetting.

[23]Knuth Gibbs lecture:tinyurl.com/ams-gibbs

[24]Gibbs Lecture: chapter 2 in Donald E. Knuth, *Digital Typography*, CSLI Publications, Stanford, CA.

[25]Observations on TEX from a Divergent Viewpoint: http://tug.org/TUGboat/tb04-2/tb08letters.pdf

[26]For instance, according to the staff list on its website, the *Boston Globe* has over 250 people in its departments that collect the news and prepare it for printing (i.e., excluding the advertising, circulation, physical print, etc., parts of the company).

Interactive

[27]*Compatible Time-Sharing System (1961–1973) Fiftieth Anniversary Commemorative Overview*, David Walden and Tom Van Vleck editors, IEEE Computer Society, 2013, history.computer.org/pubs/2011-06-ctss.pdf

[28]J. H. Saltzer, TYPSET and RUNOFF, Memorandum editor and type-out commands, MIT Project MAC, MAC-M-193, November 6, 1964, web.mit.edu/Saltzer/www/publications/CC-244.html

[29]J. Saltzer, Manuscript typing and editing: TYPSET, RUNOFF, *CTSS Programmer's Guide*, Section AH.9.01, January 15, 1966, web.mit.edu/Saltzer/www/publications/ctss/AH.9.01.html

[30]Ibid.

[31]Edward Fredkin, phone and email conversations, September and October 2002.

[32]DECUS PDP-1 library: tinyurl.com/decus-library

[33]Matthew G. Kirschenbaum, *Track Changes: A Literary History of Word Processing*, Harvard Belknap, 2016, pp 19–20.

[34]Expensive Typewriter, PDP-1 document PDP-22, MIT Electrical Engineering Department, August 1, 1972, tinyurl.com/expensivetypewritter

[35]Dan Murphy, The Beginnings of TECO, *IEEE Annals of the History of Computing*, October-December 2009, pp. 110–115, tenex.opost.com/anhc-31-4-anec.pdf; in this paper Murphy describes the early interactive use of the PDP-1 and TECO as his effort to greatly improve on Expensive Typewriter.

[36]TJ-2 Text Justifying Program, PDP-1 document PDP-9-1, Massachusetts Institute of Technology, May 9, 1963, www.dpbsmith.com/tj2.html

[37]Fernando Corbató et al., *The Compatible Time-Sharing System: A Programmer's Guide*, The MIT Press, 1963, bitsavers.informatik.uni-stuttgart.de/pdf/mit/ctss/CTSS_ProgrammersGuide.pdf, pp. 71ff.
[38]Ibid., pp. 82ff.
[39]Michael T. Barnett, *Computer Typesetting: Experiments and Prospects*, MIT Press, 1965.
[40]Chart of text editors: web.mit.edu/kolya/misc/txt/editors
[41]See 1.7.7.BCPL at multicians.org/features.html; multicians.org is maintained by Tom Van Vleck.
[42]Multics Programming Manual: http://tinyurl.com/multics-prog, pp. 3-619ff.
[43]SDS-940 RUNOFF: dtic.mil/dtic/tr/fulltext/u2/707402.pdf
[44]troff: en.wikipedia.org/wiki/Troff
[45]Experience with the Mergenthaler Linotron 202 Phototypesetter: cs.princeton.edu/~bwk/202/
[46]Email of 2016-07-29.
[47]SQtroff: tinyurl.com/SoftQuadsqtroff
[48]Pub manual: nomodes.com/pub_manual.html
[49]25 years of TEXMETAFONT: tug.org/TUGboat/tb25-1/beebe-2003keynote.pdf
[50]TEX/METAFONT retrospective: tug.org/TUGboat/tb26-1/beebe.pdf
[51]See the footnote for Knuth-Plass.
[52]http://tug.org/tugboat/contents.html
[53]Knuth publications in *TUGboat*: tug.org/TUGboat/Contents/listauthor.html\#Knuth,Donald
[54]Knuth makes TEX available: tinyurl.com/knuth-freeTeX
[55]Scribe thesis: tinyurl.com/scribethesis
[56]Scribe manual: tinyurl.com/scribemanual
[57]Texinfo history: tinyurl.com/Texinfohistory
[58]http://tug.org/tug2016/abstracts/hefferon.txt
[59]en.wikipedia.org/wiki/Wang_Laboratories
[60]Wang 1200 manual: tinyurl.com/wang1200manual
[61]Wang 2200 manual: http://tinyurl.com/wang2200manual
[62]By way of the Berkeley Computer Corporation.
[63]Alto and Bravo manuals: tinyurl.com/alto-bravo-manual
[64]Ibid.
[65]Barnaby on WordStar: digibarn.com/stories/wordstar-rob-barnaby/
[66]Rubenstein on Barnaby: tinyurl.com/rubenstein-barnaby
[67]Kirschenbaum's book[33] talks about lots of the systems used over the years by fiction writers. There were also lots of systems that were popular with one group or another at one time but have not gone out of business; ChiWriter was one that Norbert Preining mentioned at TUG2016.
[68]Karl Berry was with Interleaf for a while.
[69]Karl Berry was with Interleaf for a while.
[70]Paul Brainerd oral history: tinyurl.com/brainerd-oralhistory
[71]Charles Cornfield interview: http://tinyurl.com/cornfield-interview
[72]My experience with Quark is developing one of my books in LATEX and being required by the publisher to convert it to Word for streaming into Quark where the publisher's "art department" did the final composing of the book.

Algorithms

[73]Mary Elizabeth Stevens and John L. Little, *Automatic Typographic-Quality Typesetting Techniques: A State-of-the-Art Review*, National Bureau of Standards Monograph 99, Issued April 7, 1967. digicoll.manoa.hawaii.edu/techreports/PDF/NBS99.pdf
[74]Nils Enlund and Hans E. Anderson, The early days of computer aided newspaper production systems, *History of Nordic Computing 2*, IFIP Advances in Information and Communication Technology volume 303, Springer Berlin Heidelberg, 2009, tinyurl.com/enlund-newspapers
[75]This description is derived from messages from Ken Pogran in April and May in 2016 assisted by photos from Richard Polt who maintains the The Classic Typewriter Page, http://site.xavier.edu/polt/typewriters/
[76]Newcastle project: tinyurl.com/newcastletypesetting
[77]More Newcastle: tinyurl.com/morenewcastle
[78]I'd love to know to find a report from Newcastle that describes their method for justification rather than using Barnett's secondary source.
[79]In honor of TUG2016 being in Toronto, Canada.
[80]This line-breaking example is almost a copy of the examples at tinyurl.com/TusharRoy-dynamic-prog.
[81]In real life we probably would not count extra spaces on the last line of a paragraph.
[82]Donald E. Knuth and Michael F. Plass, Breaking Paragraphs into Lines, reprinted in Knuth's *Digital Typography*, CSLI Publications, Stanford, CA, pp. 67–155, originally published in 1981 in *Software—Practice and Experience*.
[83]At defoe.sourceforge.net/folio/knuth-plass.html, the Knuth-Plass algorithm is illustrated all the way through box-and-glue.
[84]Plass thesis: Optimal Pagination Techniques for Automatic Typesetting Systems, Report No. STAN-CS-81-879, Stanford University, June 1981, tug.org/docs/plass/plass-thesis.pdf
[85]MIT algorithms course lecture: tinyurl.com/demaine-dynamic-programming
[86]Mohamed Elyaakoubi and Azzeddiine Lazrek, Justify Just and Just Justify, *Journal of Electronic Publishing*, vol. 13 no. 1, 2010, tinyurl.com/justjustify
[87]Ibid.
[88]Liang thesis: tug.org/docs/liang/

[89]Ibid.
[90]Ibid., pp. 4–5.
[91]Hyphenation patterns: tug.org/tex-hyphen/
[92]http://tug.org/tug2016/abstracts/reutenauer-hyph.txt
[93]Petr Sojka, Competing Patterns in Language Engineering and Computer Typesetting, PhD thesis, Masaryk University in in Brno, 2005, tinyurl.com/sojkathesis
[94]There also is continuing research in other approaches to hyphenation, for example, Nikolaos Trogkanis and Charles Elkan, Conditional random fields for word hyphenation, *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, 2010, pp. 366-374, aclweb.org/anthology/P10-1038
[95]Hermann Zapf, About micro-typography and the *hz*-program, *Electronic Publishing*, vol. 6 no. 3, September 1993, pp. 283-288, in which Zapf says, "Digital typography will set the future trends of aesthetics in typesetting. With all the programs available today there is no excuse any more for mediocre typography in books or magazines.", bit.ly/zapf93
[96]Hán Thế Thánh, Micro-typographic extensions to the TEX typesetting system, dissertation, Masaryk University Brno Faculty of Informatics, October 2000, reprinted in *TUGboat*, issue 21:4, December 2000, http://tug.org/TUGboat/tb21-4/tb69thanh.pdf
[97]Thế Thánh thesis: tug.org/TUGboat/tb25-1/thanh.pdf
[98]http://tug.org/tug2016/abstracts/mittelbach.pdf

Type

[99]Robert Bringhurst, *The Elements of Typographical Style*, 2nd edition, Hartley & Marks Publishers, 2002.
[100]Luc Devroye, Formatting Font Formats, *TUGboat*, Volume 24 (2003), No. 3, Proceedings of EuroTEX 2003, pp. 588–596, tug.org/TUGboat/tb24-3/devroye.pdf
[101]Lynn Ruggles, Letterform Design Systems, Department of Computer Science, Stanford University, Report No. STAN-CS-83-97, April 1983, bit.ly/ruggles83
[102]Kevin Larson, The Science of Word Recognition or how I learned to stop worrying and love the bouma, Advanced Reading Technology, Microsoft Corporation, July 2004, www.microsoft.com/typography/ctfonts/WordRecognition.aspx
[103]Kevin Larson, TUG2016 presentation, Reading between the lines: Improving comprehension for students, \Conference{larson.txt}
[104]An interview with Charles Bigelow, Yue Wang interviewer, *TUGboat*, vol. 34 no. 2, 2013, pp. 136–167, tug.org/TUGboat/tb34-2/tb107bigelow-wang.pdf
[105]Charles Bigelow, TUG2016 presentation, Looking for legibility, \Conference{bigelow-legibility.txt}
[106]Charles Bigelow, TUG2016, Probably approximately not quite correct: Revise, repeat, \Conference{bigelow-legibility.txt}
[107]Robert Bringhurst, *Palatino: The Natural History of a Typeface* tradebook edition, David R. Godine Publisher, 2016; his TUG2016 presentation: \Conference{bringhurst.txt}
[108]Hermann Zapf, About micro-typography and the *hz*-program, *Electronic Publishing*, vol. 6 no. 3, September 1993, pp. 283-288, in which Zapf says, "Digital typography will set the future trends of aesthetics in typesetting. With all the programs available today there is no excuse any more for mediocre typography in books or magazines.", bit.ly/zapf93
[109]D. C. Dennison, Boston companies push type design into future, *The Boston Globe*, Boston, MA, August 26, 2012, bit.ly/dennison12
[110]Phyllis R. Hoffman, *Matthew Carter: Reflects on Type Design*, Master of Science thesis project, School of Printing Management and Science in the College of Imaging Arts and Sciences of the Rochester Institute of Technology May, 1999, http://scholarworks.rit.edu/theses/3850/
[111]Charles Bigelow, Notes on typeface protection, *TUGboat*, vol. 7 no. 3, pp. 146–151, http://tug.org/TUGboat/tb07-3/tb16bigelow.pdf
[112]Hermann Zapf, Call for Foundation of a 'Sir Francis Drake Society,' *Electronic Publishing*, vol. 7 no. 4, December 1994, pp. 261-263, bit.ly/zapf94

Reflections

[113]*Computers and Typography 2*, compiled by Rosemary Sassoon, Intellect Books, Portland, OR, 2002 — Ian McKenzie-Kerr, Book design: before and after, pp. 69–74; David Jury, Changes in the relationship between printer and designer: craft before, during, and after graphic design.