Direct and reverse synchronization with SyncTEX

Jérôme Laurens
Département de mathématiques
Université de Bourgogne
21078 Dijon Cedex
FRANCE
jerome (dot) laurens (at) u-bourgogne (dot) fr
http://itexmac.sourceforge.net/SyncTeX.html

Abstract

We present a new technology named SyncTEX used to synchronize between the TEX input and the output.

1 What is synchronization?

Creating documents with the TEX typesetting system often requires two windows, one for entering the text, the other one for viewing the resulting output. In general, documents are too long to fit in the visible frame of a window on screen, and what is really visible is only some part of either the input or the output. We say that the input view and the output view are synchronized if they are displaying the "same" portion of the document. Forwards or direct synchronization is the ability, for an output viewer, to scroll the window to a part corresponding to a given location in the input file. Backwards or reverse synchronization is the ability, for a text editor, to scroll the text view to a part corresponding to a given location in the output file.

Figure 1 is a screenshot illustrating SyncTEX supported in *iTEXMac*2, the TEX front end developed by the author on Mac OS X. The top window is a text editor where an extract of the "Not so short introduction to LATEX 2_ε" is displayed. The word "lscommand" has been selected and the viewer window at the bottom automatically scrolled to the position of this word in the output, highlighting it with a small red arrow. The grey background was added afterwards in the bottom window for the sake of visibility on printed media.

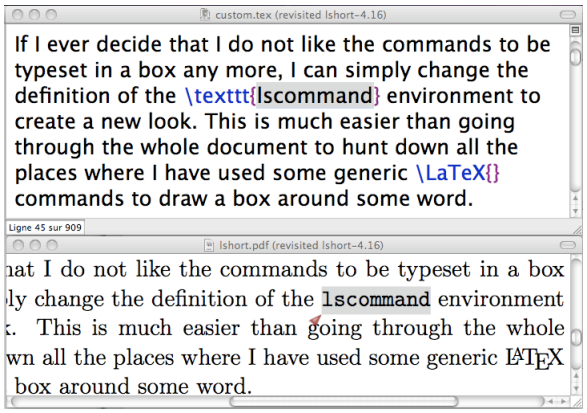

Figure 1: Synchronization in *iTEXMac*2 based on SyncTEX technology with text analysis.

2 What is SyncTEX?

This is a new technology embedded in both pdfTEX and XeTEX, available in the 2008 TEX Live and corresponding MiKTEX distributions. When activated, it gives both text editors and output viewers the necessary information to complete the synchronization process. It will be available in LuaTEX soon.

In order to activate SyncTEX, there is a new command line option:

```
pdftex -synctex=1 foo.tex
```

(or `--synctex=1`) and the same for `xetex`. With this option, a new informational line is printed at the end of the command output and reads `SyncTeX written on foo.synctex.gz`. The new (compressed) auxiliary file named `foo.synctex.gz` is used by applications for the synchronization process; this is the *SyncTEX output file*.

Setting the `synctex` option to `-1` creates an uncompressed `foo.synctex` auxiliary file, more suitable for certain operations. Setting it to 0 definitively prevents SyncTEX activation.

There is also an eponymous new TEX primitive that you can set to 1 for SyncTEX activation from the source file: `\synctex=1`. It can be used, for example, to temporarily disable SyncTEX operations for some input file by properly using `\synctex=0`. This primitive has no effect if the `-synctex=0` command line option was given, or if the first page has already been shipped out (it is then too late to activate SyncTEX).

3 Other synchronization technologies

The commercial software *Visual TEX* available on Windows has had the PDF synchronization capability embedded in its TEX engine since 1999. The commercial software *TEXtures* available on Mac OS X has had embedded synchronization since 2000, but between the text source and the DVI output. Neither implementation is freely available to the public and will not be considered in the remainder of this article.

Turning to the TEX macro level, Aleksander Simonic, the author of the *WinEdt* TEX shell on Windows, wrote before 1998 the `srcltx` macro package to enable synchronization between the text source and the DVI output. It is based on the powerful `\special` command and was later integrated into the TEX engine as "source specials". Heiko Oberdiek wrote `vpe` in 1999 where PDF technologies are used for reverse synchronization from the PDF output to the text input. In 2003, the author wrote the `pdfsync` package discussed in [5], [6] and [7], to allow synchronization between the PDF output and the text input, following ideas from Piero d'Ancona. This was based on the use of pdfTEX commands similar to `\special`, with the same limitations, namely an incompatibility with very useful packages and unwanted changes in the layout.

None of these solutions is satisfying, being either incomplete or unsafe, as we shall see.

4 Solving the synchronization problem

4.1 Stating the problem

The problem is to define a mapping between an *input record* given by an input file name and a line number in that file, and an *output record* given by a page number and a location in that page of the output file. The input record describes a certain part of the input text whereas the output record describes the corresponding location where this text appears in the output. The original TEX engine does not provide any facility for such a correspondence, except the debugging information used to report syntax errors (we call it the *current input record*). More precisely, TEX does not know at the same time both the input records and their corresponding output records. In short, TEX parses each different line of the input text file and expands the macros with its "eyes" and "mouth" (according to [2], page 38), then it sends a list of tokens into its "stomach". In turn, the stomach creates lines of text, stacks them into paragraphs, and stacks the paragraphs into pages. Once a page is complete with the objects properly positioned, it is shipped out to the output file. During this process, TEX keeps the input record information until macro expansion only (in its head), and it does not know the corresponding output record until ship out time which occurs later (in its stomach). The problem is to force TEX to remember the input record information until ship out time.

4.2 Partial solutions using macros

The first idea, developed in the `srcltx` package, is to use the `\special` macro to keep track of the input record information until ship out time. By this method, it inserts in the text flow invisible material that dedicated DVI viewers can understand. The main problem is that this invisible material is not expected to be there and can alter significantly the line breaking mechanism or cause other packages to malfunction, which is extremely troublesome.

The second idea, developed in the `pdfsync` package, is also to use macros, but in a different way because it is more difficult to manage PDF contents than DVI contents. This package automatically adds in the input source some macros that act in two steps. At macro expansion time, they write to an auxiliary file the input record information with a unique identifying tag. They also insert in the text flow invisible material to prepare TEX to write the output record information at ship out time, with exactly the same identifying tag. In this design, the problems concerning line breaking and package incompatibility remain. Moreover, the mapping between input and output records is not one to one, which renders synchronization support very hard to implement for developers.

In these two different solutions, we see the inherent limits of synchronization using macros. More generally, we can say that those macros are *active observers* of the input records. In fact, by inserting invisible material in the text flow they interact with the typesetting process. On the contrary, SyncTEX is a *neutral observer* that never interacts with the typesetting process.

4.3 How SyncTEX works

In fact, the only object that ever knows both the input and output records is the TEX engine itself, so it seems natural to embed some synchronization technology into it.

We first have to determine what kind of information is needed to achieve synchronization. For that purpose, we follow [2] at page 63: "TEX makes complicated pages by starting with simple individual characters and putting them together in larger units, and putting these together in still larger units, and so on. Conceptually, it's a big paste-up job.

The TEXnical terms used to describe such page construction are boxes and glue." The key words are "characters", "boxes" and "glue". But since an individual character requires a considerable amount of extra memory, only horizontal boxes and vertical boxes are taken into account at first. For these boxes, we ask TEX to store in memory, at creation time and during their entire lifetime, the current input record. At ship out time, we ask TEX to report for each box the stored file name, the stored line number, the horizontal and vertical positions, and the dimensions as computed during typesetting. This information will be available for synchronization: for example when the user clicks on some location of the PDF document, we can really find out in which box the click occurred, and then deduce the corresponding file name and line number. We have here the design for a neutral observer.

But if this new information is sufficient for locating, it cannot be used for synchronization due to the way TEX processes files. In fact, boxes can be created in TEX's mouth where the current input record is accurate, but in general, they are created in the stomach when breaking lines into paragraphs, for text that was parsed a long time ago and no longer corresponds to the current input record. This is particularly obvious when a paragraph spans many lines of the input text: the line breaking mechanism is engaged after the last line is parsed, and every horizontal box then created will refer to the last input line number even if the contained material comes from a previous input line. For that reason, we also ask TEX to store input records for glue items, because they are created in TEX's mouth, when the current input record is still accurate.

By combining boxes and glue management, we have accurate information about positions in the output and the correspondence with positions in the input file. In fact, things are slightly more complicated because of TEX internals: kern, glue and math nodes are more or less equivalent at the engine level, so SyncTEX must treat them similarly, but this is better for synchronization due to the supplemental information provided.

SyncTEX does other sorts of magic concerning file name management, the magnification and offset, but these are implementation details.

4.4 The benefits of SyncTEX

Embedding the synchronization technology deeply inside the TEX engine solves many problems and improves the feature significantly.

The most visible improvements are connected with accuracy: with SyncTEX, the synchronization process reaches in general a precision of ±1 line. With additional technologies such as implemented in *iTEXMac*2, we can even synchronize by words (see figure 1), essentially always finding an exact word correspondence between input and output.

The next improvements are a consequence of the overall design of SyncTEX. Since synchronization is deeply embedded into the TEX engines, there is no TEX macro involved in the process. As a straightforward consequence, there cannot be any incompatibility with macro packages. Moreover, no extra invisible material is added to the text flow, thus ensuring that the layout of the documents is exactly the same whether SyncTEX is activated or not. As a matter of fact, it is absolutely impossible to determine if the output was created with SyncTEX activated by examining its contents. Finally, no assumptions are made about external macros or output format, so that synchronization works the same for Plain, LATEX or ConTEXt as well as DVI, XDV or PDF.

Of course, all this needs extra memory and computational time but this is in no way significant. *In fine*, we can say that with SyncTEX, the synchronization has become safe and more precise.

5 Limits and improvements

It is indisputable that abandoning the use of macros and choosing an embedded design is a great advance for synchronization. But still it is not perfect! Some aspects of the implementation are not complete due to a lack of time, but others will prevent us from reaching the ultimate synchronization comparable to *wysiwyg* (an acronym for "What You See Is What You Get") as discussed in [7].

5.1 The DVI to PDF filters

When producing a PDF document from a DVI or XDV output, we apply a filter like `dvitopdf` or `xdv2pdfmx`. But those filters can alter the geometry of the output by changing the magnification or the offset of the top left corner of the text. In that case, the SyncTEX information, which is accurate for the DVI file, is not accurate for the PDF file. This problem is solvable by post-processing the SyncTEX output file with the new `synctex` command line tool available in the distributions, *eg*

```
xdv2pdfmx -m MAG -x XXX -y YYY foo
```

should be followed by

```
synctex update -m MAG -x XXX -y YYY foo
```

But this is not a good design. Instead, the post-processing should be embedded into the various DVI to PDF filters so that no further user interaction is required.

5.2 Using TEX's friends

Some documents are created with a complex typesetting flow where TEX is just one tool amongst others. In such circumstances, the TEX file is automatically generated from possibly many original input files by some processor. Then, synchronization should occur between the output and the original input files rather than the TEX file, which is just an intermediate step. For example, a bibliography in LATEX is generally built by BIBTEX based on a bibliography database with a `bib` file extension using an intermediate auxiliary LATEX file with a `bbl` file extension. At present, the synchronization occurs between the PDF output and the `bbl` file and not the original `bib` file, as one would prefer.

Improving SyncTEX to properly manage this situation is not extremely complicated: we first have to define a SyncTEX map file format for the mapping between the lines of the original input files and the lines of the produced TEX files, then we have to provide facilities to merge this mapping information into the SyncTEX output file. Then the processor could produce the map file, and a supplemental step in the typesetting flow would update the SyncTEX information with that map.

Sometimes it might not be appropriate to simply bypass the intermediate file. In that case, the viewer should synchronize with the auxiliary file using a text editor which in turn should synchronize with the original input file using the map file.

5.3 Accuracy and the column number

As described above, we only take into account whole lines in the input files and jump from or to lines in the text. This can suffice for textual files, but does not when mathematical formulas are involved — we would like to have a more precise position in the input. Unfortunately, when parsing the input files, the original TEX engine does not handle column positions at all. And it seems that adding support for this supplemental information might need a great amount of work, probably much greater than the eventual benefits.

5.4 For non-Latin languages

SyncTEX has been designed with a Latin language background: it relies on the fact that TEX automatically creates kern and glue nodes at parse time to manage interword spacing. For languages that do not have a comparable notion of word, the synchronization will not be sufficiently accurate and will most certainly need further investigations. This question is open and the author welcomes test files, suggestions and advice.

5.5 A question of design

The two preceding limitations are consequences of a conceptual default in the actual synchronization design. With SyncTEX, the TEX engine has been modified to export some observed information useful for synchronizers. The problem is that we are able to observe only what TEX allows us to, and this is not always the best information we would like to have. It would be more comfortable and efficient if TEX already provided synchronization facilities from the very beginning. In that case, all the macros packages would have to be compatible with the synchronization code and not the opposite. That would require more work for the package maintainers but would also prevent any kind of layout and compatibility problems due to special nodes.

In a different approach, a supplemental step could be to store synchronization information for each individual character, thus increasing the memory requirements of the engine in a way similar to how XETEX handles multi-byte characters. This idea was originally proposed by Hàn Thế Thành, but it was abandoned because the SyncTEX output file was unbearably huge. With the new design, this idea can certainly be revisited with more success.

Anyway, further investigations into the arcana of the TEX program would certainly lead to a better synchronization accuracy but if we want to avoid huge changes in TEX and keep compatibility with existing macro packages, we must admit that we have almost reached some insuperable barrier.

6 Implementation in TEX Live

Without entering into great detail, we explain how the implementation of SyncTEX is carefully designed to ease code maintenance and enhancements, as far as possible.

6.1 A segmented implementation

All the SyncTEX related code is gathered in only one directory named `synctexdir`, in which the code is split into different source files. The separation is organized in order to share the maximum amount of code between the different engines, and to clearly identify the different tasks involved in the information management. All in all, we end up with 14 different change files. When building the binaries, the partial make file `synctex.mk` has the duty to manage which change file should apply to which engine and when.

6.2 An orthogonal implementation

One of the key concepts in modern code design is separation, whose purpose is to ease code management and maintenance. `WEB` Pascal does not offer facilities for code separation, nevertheless, it is possible to build all the engines with or without the SyncTEX feature, as explained in the `synctex.mk` file. It will be useful for developers whenever a problem is caused by the SyncTEX patches.

7 Which software supports SyncTEX

Up to now, we have focused on the technological aspects of synchronization, and we have described in detail the foundations. It is time to look at the concrete implementations of synchronization with different methods, because this feature is useless if it is not adopted by applications. SyncTEX not only consists of changes to the TEX engine, but gives developers tools to easily support the technology.

7.1 The SyncTEX parser library

The main tool is a library written in C, whose purpose is to help developers implement direct and reverse synchronization in their application. It consists of one file named `synctex_parser.c` and its header counterpart `synctex_parser.h`, meant to be included as-is in application sources. Both are available on the SyncTEX web site [4]. The source file takes care of all the ancillary work concerning SyncTEX information management and the header file contains all the necessary help for an optimal usage of the programming interface.

At this writing, *TEXworks* (presented by J. Kew in [1]), Sumatra PDF on the Windows platform, and Skim and *iTEXMac*2 on Mac OS X, all support SyncTEX by including this parser library. For other applications, TEX users are encouraged to send a feature request to the developers of their favorite PDF or DVI viewer.

7.2 Remark about the document viewer

It should be noticed that the tricky part of direct and reverse synchronization should be handled by the viewer only. The SyncTEX parser library is meant not for text editors but for viewers. In a normal direct synchronization flow, the user asks the text editor to synchronize the viewer with a given line in a given input file, the text editor forwards the file name and the line number to the viewer, the viewer asks the SyncTEX parser for the page number and location corresponding to the information it has received, then it scrolls its view to the expected page and highlights the location. In a normal reverse synchronization flow, the user asks the viewer to synchronize the text editor with a given location in a given page of an output file, the viewer asks the SyncTEX parser for the input file name and line number corresponding to the location; it then asks the text editor to display the named input file and highlight the numbered line.

7.3 The new `synctex` command line tool

There are cases when the inclusion of the parser library is not possible or even improbable (consider for example Adobe's Acrobat reader). For such situations, the `synctex` command line tool is the alternative designed to allow synchronization. It is just a wrapper over the SyncTEX parser library that acts as an intermediate controller between a text editor and a viewer. The description of its usage is obtained via the command line interface running `synctex help view` for direct synchronization and `synctex help edit` for reverse synchronization.

Provided that the text editor and the viewer have some scripting facilities, here is how this tool should be used. For direct synchronization, the user asks the text editor to synchronize the viewer with a given line in a given input file, the text editor forwards this file name and line number to the `synctex` tool together with some scripting command to activate the viewer, the `synctex` tool transparently asks the SyncTEX parser for the page number and location corresponding to the information it has received, then it asks the viewer to proceed with the help of the scripting command.

For reverse synchronization, the user asks the viewer to synchronize the text editor with a given location in a given page of an output file, the viewer forwards this information to the `synctex` tool together with some scripting command to activate the text editor, the `synctex` tool transparently asks the SyncTEX parser for the input file name and line number corresponding to the information it has received, then it asks the text editor to proceed according to the received scripting command.

Before this tool was available, developers had no solution other than directly parsing the contents of the SyncTEX output file. This was generally made in continuation of the implementation of `pfdsync` support. Comparatively, it is more comfortable to work with a `.synctex` file than a `.pdfsync` file because the new syntax is extremely clear and straightforward, consequently reverse engineering was unexpectedly encouraged. But this practice should be abandoned for two reasons: it is certainly not compatible with forthcoming enhancements of SyncTEX, and it generally does not work when changing the magnification

and the offset in the DVI to PDF conversion, as discussed above. In order to convince developers to prefer the `synctex` tool, the specifications of the SyncTEX output file are considered private and will not be widely published.

More details concerning usage and implementation are available on the SyncTEX web site [4].

8 Applications

There are a variety of ways to use the newly available information in the SyncTEX output file. Some were considered while designing this feature, others suggested by people at the conference. No doubt this list is not exhaustive.

8.1 Better typesetting mechanisms

TEX is well known for its high quality page breaking mechanism, but the hardware constraints that were crucial 30 years ago imposed some choices and deliberate barriers. The limitation in memory usage led to a page by page design, where memory is freed each time a page is shipped out. In that situation, a page breaking algorithm cannot perform optimization in a document as a whole, but only on a small number of consecutive pages.

In order to have global optimization algorithms, one can keep everything in memory until the end of the TEX run, but that would require a big change in the engine. From another standpoint, SyncTEX has demonstrated that it is possible to trace geometrical information throughout the typesetting process. It is clear that the information actually contained in the SyncTEX output file is not suitable for typesetting purpose because it was designed for synchronization only. But with some additional adaptations, there is no doubt that SyncTEX can help in designing global optimization algorithms for even better typesetting.

8.2 Debugging facilities

During his presentation at the conference (see [3]), the author used a lightweight PDF viewer to demonstrate SyncTEX. This viewer was primarily designed as a proof of concept and as such, was meant to remain private. But one of its features might be of great interest to the TEX community, as suggested by different people at the conference, namely the ability to display over the text all the boxes, either horizontal or vertical, created during the typesetting process. As it happens, this feature was already implemented in an unknown modest PDF viewer for Mac OS X (whose name I have unfortunately lost) by parsing the result of the `\tracingall` macro in the log file.

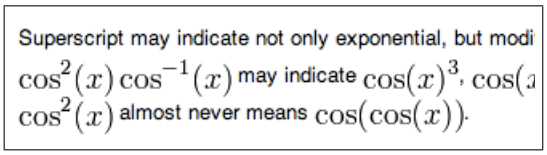

Figure 2: TEX output embedded in HTML, detail of `http://en.wikipedia.org/wiki/Special_functions` (2008/08/11)

The interest is at least twofold. It can serve debugging purposes for publishers who want to elaborate complicated page layouts, and it can also serve pedagogical purposes during TEX training sessions. For these reasons, this viewer will be available on the SyncTEX web site [4] once it has been properly factored for distribution. Unfortunately, this benefits Mac OS X users only, but adding this feature to the new cross-platform *TEXworks* will eventually be considered.

8.3 Embedding TEX output into HTML or running text

In web pages, it is rather common to include mathematical formulas as embedded images built with TEX, to compensate for the limitations of web browsers. The example given in figure 2 is particularly ugly, not only because the size of the mathematical text does not conform to the size of the running text, but also because the base lines of the formulas and the running text are not properly aligned. In fact, the included images contain no information concerning the base line, and this is where SyncTEX can come into play. The synchronization information contains the dimensions of each box containing a mathematical formula, in particular its height and depth, hence the exact position of the base line. We just have to raise the image by the proper amount to obtain a correct vertical alignment.

9 Concluding remarks

9.1 Synchronizing by word

In *iTEXMac*2, synchronization is enhanced to attain the precision of a word or even a character, by combining SyncTEX with some text analysis. This was rather easy to accomplish because *iTEXMac*2 manages both the text input and the PDF output, and also because the PDF library on Mac OS X has text facilities. But this does not mean that only an integrated TEX environment is able to reach such a level of accuracy. It is in fact more a matter of communication between different applications.

In fact, the text editors and the viewers allow some inter-process communication through the command line, for example `mate -l LINE foo.tex` asks the *TextMate* text editor to edit `foo.tex` at line `LINE`. Only the line number is used and this is perfectly suitable for programmers because in general, compilers and debuggers only need line numbers. But TEX users are not programmers. In *iTEXMac*2 this kind of practice has been reconsidered for reverse synchronization and the information passed to the text editor contains not only the file name and the line number, but also an extract of text surrounding the character under the mouse. This supplemental information is the hint used by the text editor to have a better focus on the synchronization target. For direct synchronization, the same idea applies and the PDF viewer is asked to highlight a location in a given page, with the help of a similar textual hint.

In order to achieve synchronization by word or character, text editors and viewers should use a textual hint, both as senders and receivers. Of course this requires some coding effort because input text and output text are not exactly the same (due to line breaking for example), but the expense is affordable as soon as efforts are combined. Once again, users are encouraged to submit feature requests to the developers of their favorite tools.

By the way, the new `synctex` command line tool anticipates the use of a textual hint by editors or viewers through its `-h` command line option.

9.2 An historical standpoint

Synchronization with SyncTEX appears for the 30th anniversary of TEX; we can legitimately wonder why and whether such a long period of gestation was necessary. In order to explain this delay, let us review the ingredients that made SyncTEX possible.

As in many situations of software design, a favorable context comes concurrently from available technologies and available workers. Regarding technological aspects, we can say in a reduction not very far from reality that SyncTEX is nothing but a clever usage of the Web2C implementation of TEX. Of course, developing on Mac OS X was rather easy and very efficient, but any other environment would certainly provide the same result at the price of more programming work.

Concerning people, the author has claimed since the beginning of `pdfsync` that some synchronization should definitely be embedded in the TEX engine, in the hope that someday, someone *else* would do it. Hàn Thế Thành was aware of the problem three years ago (not one year ago as claimed by the author during his presentation [3]), but he could only take some time for coding this in summer 2007, probably under the friendly pressure of some users dissatisfied with the limits of `pdfsync`. Although his first attempt was hardly usable and finally abandoned, it introduced the author to the minutiae of the Web2c implementation of TEX. Initially, SyncTEX was targeted at pdfTEX but Jonathan Kew helped in adapting it to XETEX and also with the integration into TEX Live.

This short review seems to indicate that technologies like SyncTEX could easily have become available many years ago. One can attribute the delay to a lack of effort devoted to the human interface of TEX, which is highly regrettable. With SyncTEX and tools like *TEXworks*, first steps are made in the right direction, because TEX really deserves a good human interface, not just a user interface.

10 Acknowledgements

The author gratefully thanks Hàn Thế Thành without whom this work would never have started and Jonathan Kew without whom this work would not have reached the present stage. He received important remarks and valuable help from members of the pdfTEX, XETEX, *iTEXMac*2 and TEX Live development teams; thanks to all of them.

References

[1] Jonathan Kew. *TEXworks*: Lowering the barrier to entry. In this volume, pages 362–364.

[2] D. Knuth. *The TEXbook*. Addison Wesley, 1983.

[3] Jérôme Laurens. *SyncTEX presentation at TUG 2008*. `http://www.river-valley.tv/conferences/tug2008/#0302-Jerome_Laurens`.

[4] Jérôme Laurens. *SyncTEX web site*. `http://itexmac.sourceforge.net/SyncTeX.html`.

[5] Jérôme Laurens. *iTEXMac*, an integrated TEX environment for Mac OS X. In *TEX, XML, and Digital Typography*, volume 3130/2004 of *Lecture Notes in Computer Science*, pages 192–202. Springer Berlin / Heidelberg, 2004.

[6] Jérôme Laurens. The TEX wrapper structure: A basic TEX document model implemented in *iTEXMac*. In *EuroTEX 2005, 15th Annual Meeting of European TEX Users*, 2005.

[7] Jérôme Laurens. Will TEX ever be *wysiwyg* or the PDF synchronization story. *The PracTEX Journal* 2007(3), 2007.