Electronic Editing in LaTeX

Geoff Palmer

November 2009

1 Getting Started

As a copy editor, I've sometimes struggled to cope with LaTeX projects on hard copy. The limitations of working on paper make it difficult to carry out language correction, because the text is at final page size and so space can be very limited. A hard-copy edit offers none of the flexibility of word-processing. Furthermore, the next recipient, perhaps a trade typesetter, will have to interpret the maze of handwritten marks and key in changes – expensive, time-consuming and risky work. There has to be a better way.

The method that I'm about to set out may not be the optimal route forward, and I'm open to suggestions for improvement, but as the outcome of a process of experimentation, it's been shown to work successfully.

I was driven to investigate this course of action by the requirements of one particular task, which was to edit a maths book that needed language correction. Tracked versions of the edited files were part of the requirements. A further objective was to edit the book to follow a particular typographical style as closely as possible.

LaTeX has traditionally had a distinctive appearance in print. Certain defaults built into the system – such as page headlines in italic capitals, and specific indent schemes for displayed lists – are instantly recognisable and would appear to be somewhat limiting. However, given the many add-on packages now available, most – if not all – of those default styles can be customized.

In order to work with LaTeX documents on screen, a text editor of some kind is needed. While commercial packages such as Scientific Word provide a user-friendly "WYSIWYG" environment, they conceal rather than reveal the underlying code. Putting other considerations aside, I would prefer to see the LaTeX "for real" and learn how to make my way through it – especially given the need to customize.

I'll deal with the specific requirement to track changes below. Let's begin with some of the basics of editing LaTeX electronically.

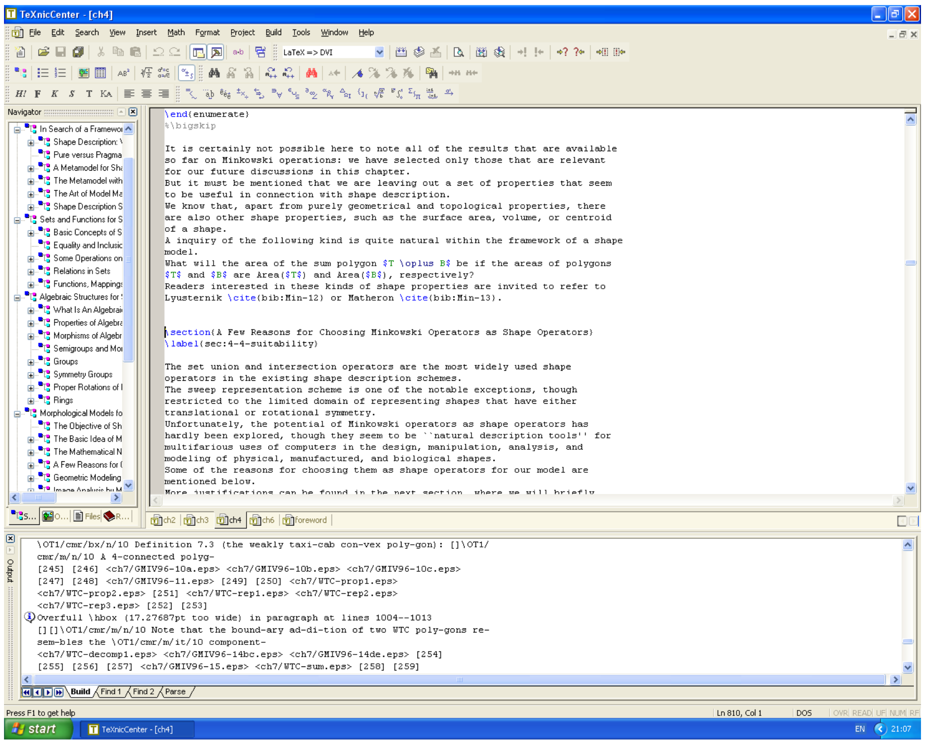
My LaTeX software set-up consists of three elements:

- The LaTeX distribution itself – I'm currently running **MikTeX 2.6**

- A suitable (freeware) editing program – in my case **TeXnicCenter**
- A choice of previewers – I use **Yap** for `.dvi` rendering and the **Adobe Acrobat Reader** for `.pdf` rendering

The MikTeX distribution is largely invisible during use. The program runs in the background and is used to "build" the page output and monitor errors and warnings. The built-in MikTeX Package Manager provides an overview of packages that are already installed in the system, and lists many more that can potentially be downloaded and installed to control specific features, such as those unwieldy page headlines, the appearance of figure and table captions, heading and subheading styles, list indents and inter-item spacing, and so on.

The TeXnicCenter editor provides a main window in which the `.tex` file can be edited, together with a navigator panel and an output log. A general view of TeXnicCenter is shown in Figure 1. Note that LaTeX commands are picked out in blue – more on this below.



```
\end{enumerate}
%\bigskip

It is certainly not possible here to note all of the results that are available
so far on Minkowski operations: we have selected only those that are relevant
for our future discussions in this chapter.
But it must be mentioned that we are leaving out a set of properties that seem
to be useful in connection with shape description.
We know that, apart from purely geometrical and topological properties, there
are also other shape properties, such as the surface area, volume, or centroid
of a shape.
A inquiry of the following kind is quite natural within the framework of a shape
model.
What will the area of the sum polygon $T \oplus B$ be if the areas of polygons
$T$ and $B$ are Area($T$) and Area($B$), respectively?
Readers interested in these kinds of shape properties are invited to refer to
Lyusternik \cite{bib:Min-12} or Matheron \cite{bib:Min-13}.


\section{A Few Reasons for Choosing Minkowski Operators as Shape Operators}
\label{sec:4-4-suitability}

The set union and intersection operators are the most widely used shape
operators in the existing shape description schemes.
The sweep representation scheme is one of the notable exceptions, though
restricted to the limited domain of representing shapes that have either
translational or rotational symmetry.
Unfortunately, the potential of Minkowski operators as shape operators has
hardly been explored, though they seem to be ``natural description tools'' for
multifarious uses of computers in the design, manipulation, analysis, and
modeling of physical, manufactured, and biological shapes.
Some of the reasons for choosing them as shape operators for our model are
mentioned below.
More justifications can be found in the next section, where we will briefly
```

```
\OT1/cmr/bx/n/10 Definition 7.3 (the weakly taxi-cab con-vex poly-gon): []\OT1/
cmr/m/n/10 A 4-connected polyg-
[245] [246] <ch7/GMIV96-10a.eps> <ch7/GMIV96-10b.eps> <ch7/GMIV96-10c.eps>
[247] [248] <ch7/GMIV96-11.eps> [249] [250] <ch7/WTC-prop1.eps>
<ch7/WTC-prop2.eps> [251] <ch7/WTC-rep1.eps> <ch7/WTC-rep2.eps>
<ch7/WTC-rep3.eps> [252] [253]
Overfull \hbox (17.27687pt too wide) in paragraph at lines 1004--1013
[][]\OT1/cmr/m/n/10 Note that the bound-ary ad-di-tion of two WTC poly-gons re-
sem-bles the \OT1/cmr/m/it/10 component-
<ch7/WTC-decomp1.eps> <ch7/GMIV96-14bc.eps> <ch7/GMIV96-14de.eps> [254]
[255] [256] [257] <ch7/GMIV96-15.eps> <ch7/WTC-sum.eps> [258] [259]
```

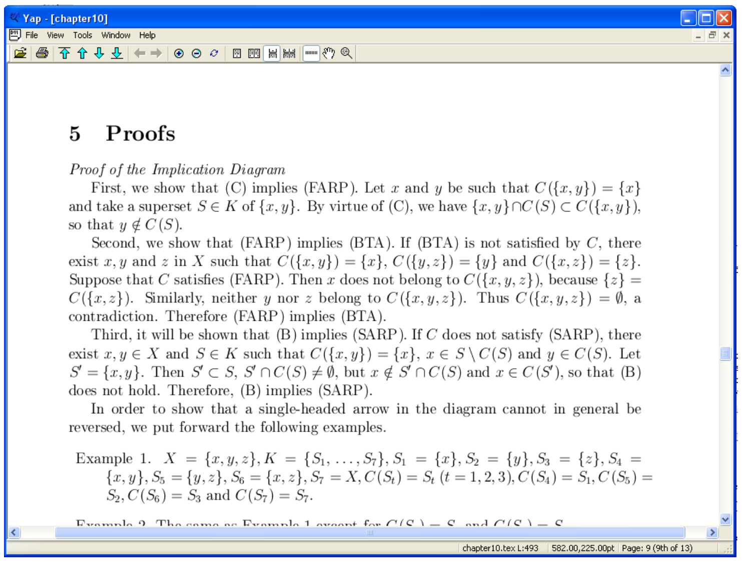
Figure 1 A general view of the TeXnicCenter editor

The previewer is used to get an instant snapshot of the built output. There are three basic choices of output profile:

- LaTeX → DVI (Digital Visual Interface)
- LaTeX → PDF (Portable Document Format) and

- LaTeX → PS (PostScript)

The DVI previewer is quick to load and shows illustrations readily (see below), but can be slow to navigate, and – importantly – the Yap previewer (see Figure 2) doesn't seem to be searchable. The LaTeX → PDF build makes use of the familiar Adobe Acrobat Reader, which can be navigated quickly and searched comprehensively – but there is a drawback.



5 Proofs

Proof of the Implication Diagram

First, we show that (C) implies (FARP). Let x and y be such that $C(\{x,y\}) = \{x\}$ and take a superset $S \in K$ of $\{x,y\}$. By virtue of (C), we have $\{x,y\} \cap C(S) \subset C(\{x,y\})$, so that $y \notin C(S)$.

Second, we show that (FARP) implies (BTA). If (BTA) is not satisfied by C, there exist x, y and z in X such that $C(\{x,y\}) = \{x\}$, $C(\{y,z\}) = \{y\}$ and $C(\{x,z\}) = \{z\}$. Suppose that C satisfies (FARP). Then x does not belong to $C(\{x,y,z\})$, because $\{z\} = C(\{x,z\})$. Similarly, neither y nor z belong to $C(\{x,y,z\})$. Thus $C(\{x,y,z\}) = \emptyset$, a contradiction. Therefore (FARP) implies (BTA).

Third, it will be shown that (B) implies (SARP). If C does not satisfy (SARP), there exist $x, y \in X$ and $S \in K$ such that $C(\{x,y\}) = \{x\}$, $x \in S \setminus C(S)$ and $y \in C(S)$. Let $S' = \{x,y\}$. Then $S' \subset S$, $S' \cap C(S) \neq \emptyset$, but $x \notin S' \cap C(S)$ and $x \in C(S')$, so that (B) does not hold. Therefore, (B) implies (SARP).

In order to show that a single-headed arrow in the diagram cannot in general be reversed, we put forward the following examples.

Example 1. $X = \{x,y,z\}, K = \{S_1, \ldots, S_7\}, S_1 = \{x\}, S_2 = \{y\}, S_3 = \{z\}, S_4 = \{x,y\}, S_5 = \{y,z\}, S_6 = \{x,z\}, S_7 = X, C(S_t) = S_t \ (t = 1,2,3), C(S_4) = S_1, C(S_5) = S_2, C(S_6) = S_3$ and $C(S_7) = S_7$.

Example 2. The same as Example 1 except for

Figure 2 Previewing the built output using Yap

The most common current practice seems to be to include graphics in `.tex` files in one of two ways: either as `.eps` (Encapsulated PostScript) files (although other graphics formats such as `.png` should work) or via the program itself – in other words, the `.tex` file includes data from which the program builds the figures.

The `.eps` or `.png` route may be the easiest to work with, but because of a driver clash `.eps` illustrations may not show in the default `.pdf` preview (instead, blank spaces are left). There is a way round this, which is to use a package called `dvipdfm`. On many projects, the complete `.pdf` file, with illustrations in place, will be quite large, so a default version without illustrations may be more suitable for rapid and repeated previewing.

2 File Organization and Basic Editing

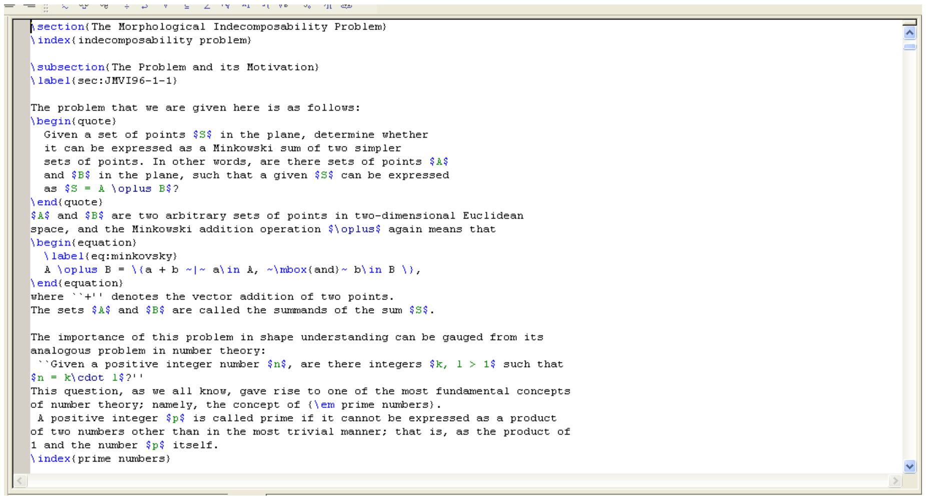
There are various classes of LaTeX document, possibly the most familiar one being the "article" class. However, a folder of `.tex` article-class files, each one bearing the title "Chapter X", isn't a properly assembled book – it's just a collection of stand-alone articles.

The LaTeX "book" class provides the facility to build contents lists and indexes automatically, and it calls for one "master file" into which the separate components can be imported in turn.

The master file begins with a preamble, in which the document settings and package options are listed, together with information about theorems and other theorem-like environments, special coding options and so on. The individual chapters are not written within a template – they are just plain `.tex` files.

To keep matters simple, the project files all need to be stored within the same folder, or within subfolders (especially useful for illustrations).

Within an individual chapter, differences begin to emerge between conventional word-processing and LaTeX editing. Most significantly, there's no need to take word wrap into consideration. In fact, you positively don't want lines of equal length and with end-of-line soft hyphenation. Hence, however untidy it may look, the example shown in Figure 3 works perfectly well.



```
\section{The Morphological Indecomposability Problem}
\index{indecomposability problem}

\subsection{The Problem and its Motivation}
\label{sec:JMVI96-1-1}

The problem that we are given here is as follows:
\begin{quote}
  Given a set of points $S$ in the plane, determine whether
  it can be expressed as a Minkowski sum of two simpler
  sets of points. In other words, are there sets of points $A$
  and $B$ in the plane, such that a given $S$ can be expressed
  as $S = A \oplus B$?
\end{quote}
$A$ and $B$ are two arbitrary sets of points in two-dimensional Euclidean
space, and the Minkowski addition operation $\oplus$ again means that
\begin{equation}
  \label{eq:minkovsky}
  A \oplus B = \{a + b ~|~ a\in A, ~\mbox{and}~ b\in B \},
\end{equation}
where ``+'' denotes the vector addition of two points.
The sets $A$ and $B$ are called the summands of the sum $S$.

The importance of this problem in shape understanding can be gauged from its
analogous problem in number theory:
 ``Given a positive integer number $n$, are there integers $k, l > 1$ such that
$n = k\cdot l$?''
This question, as we all know, gave rise to one of the most fundamental concepts
of number theory; namely, the concept of {\em prime numbers}.
 A positive integer $p$ is called prime if it cannot be expressed as a product
of two numbers other than in the most trivial manner; that is, as the product of
1 and the number $p$ itself.
\index{prime numbers}
```

Figure 3 A chapter in the main TeXnicCenter window – despite the ragged appearance, the text will still build into coherent paragraphs

When you build your file, the program will disregard short lines, space sentences automatically (although some refinement is possible) and insert paragraph breaks where it encounters blank lines.

If you break a LaTeX command across two lines, the program will read an additional space. If this space interrupts an equation, figure or table cross-reference, the program won't recognize the tag and, for example, your output might read "Figure ??."

When editing in Microsoft Word, it's normal practice to work with "Track Changes" turned on. TeXnicCenter doesn't offer a tracking facility, but there are simple ways to record old and new versions. If in doubt, rather than deleting a text passage and inserting a replacement, keep the original line in place but precede it with a "%" symbol. On any particular line, LaTeX disregards text that follows a "%" symbol (when real percentages need to be shown, the symbol has to be augmented using a "\"). Then type in your new version just below.

But what if language correction is extensive and challenging, and full word-processing functionality is required? The answer may be to work within Microsoft Word instead.

3 Editing LaTeX within Microsoft Word

Given the facilities that a program such as TeXnicCenter can offer, it may seem like unnecessarily hard work to consider taking the `.tex` files out of the LaTeX editor altogether, and then – after detailed editing – putting them back in. But it is possible.

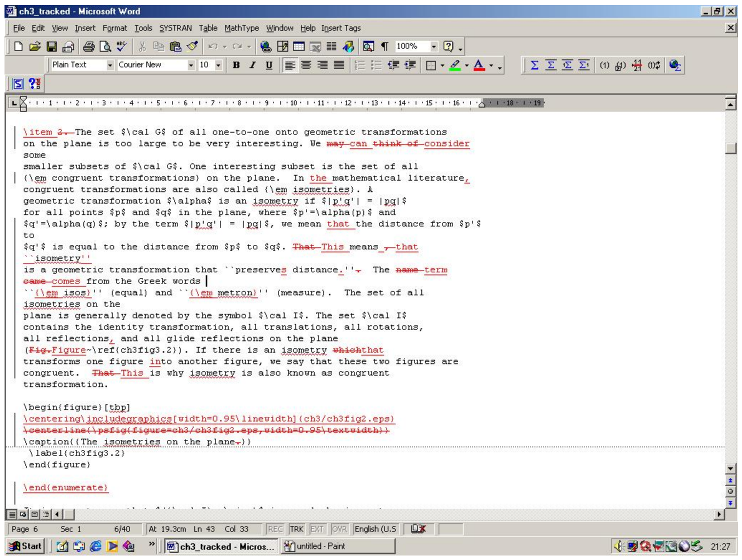
A `.tex` file is really little more than an ASCII file, one that contains a mixture of plain text and commands. There's no hidden content. If you take your chapter `.tex` file and convert it to plain ASCII `.txt` (by changing the file extension – something that can easily be done from the DOS command prompt, now better known as the console), you can then load the resulting ASCII file into Word, save it as a `.doc` and edit in that environment (see Figure 4).

I must emphasize that this process is *not* a "conversion" in the true sense (which would involve producing a Word `.doc` file as the final output, with equations rendered in a program such as MathType). Such true conversions are possible – see, for example, the GrindEQ range of maths utilities – but my approach has been simply to view the LaTeX file, complete with explicit coding, within Word. As I understand it, a fully fledged conversion to Word can cause loss of information.

However, there is an obvious loss of functionality. Unlike TeXnicCenter, which shows LaTeX commands in blue, Word doesn't distinguish between ordinary text and commands, so it becomes all too easy to delete vital coding by accident.

A further knock-on effect is that it becomes difficult to preview, because at each stage the files (preferably a state-of-the-art set of duplicates) will need to be converted back to `.tex`. To convert back, save duplicate Word `.doc`s, accept the current changes and switch "Track Changes" off, save again as ASCII, in the "Text Only with Line Breaks" format (to preserve the all-important layout), and then go back to the DOS command prompt to change the file extension back to `.tex`.

There is a risk that this process can become iterative and time-consuming. Greater familiarity with LaTeX helps, because the need to build frequently is reduced. However, if the language issues are manageable, but someone simply needs to make a "before and after" comparison, an easier option would be to perform these operations just once, on completion of editing, and then use Word's "Compare Documents" facility to reveal the changes.



```
\item ~~2.~~The set $\cal G$ of all one-to-one onto geometric transformations
on the plane is too large to be very interesting. We ~~may~~ can ~~think of~~ consider
some
smaller subsets of $\cal G$. One interesting subset is the set of all
{\em congruent transformations} on the plane.  In the mathematical literature,
congruent transformations are also called {\em isometries}. A
geometric transformation $\alpha$ is an isometry if $|p'q'| = |pq|$
for all points $p$ and $q$ in the plane, where $p'=\alpha(p)$ and
$q'=\alpha(q)$; by the term $|p'q'| = |pq|$, we mean that the distance from $p'$
to
$q'$ is equal to the distance from $p$ to $q$. ~~That~~ This means ~~,~~ that
``isometry''
is a geometric transformation that ``preserves distance.''~~.~~  The ~~name~~ term
~~came~~ comes from the Greek words
``{\em isos}'' (equal) and ``{\em metron}'' (measure).  The set of all
isometries on the
plane is generally denoted by the symbol $\cal I$. The set $\cal I$
contains the identity transformation, all translations, all rotations,
all reflections, and all glide reflections on the plane
(~~Fig.~~Figure~\ref{ch3fig3.2}). If there is an isometry ~~which~~that
transforms one figure into another figure, we say that these two figures are
congruent.  ~~That~~ This is why isometry is also known as congruent
transformation.

\begin{figure}[tbp]
\centering\includegraphics[width=0.95\linewidth]{ch3/ch3fig2.eps}
~~\centerline{\psfig{figure=ch3/ch3fig2.eps,width=0.95\textwidth}}~~
\caption{{The isometries on the plane~~.~~}}
 \label{ch3fig3.2}
\end{figure}

\end{enumerate}
```

ch3_tracked - Microsoft Word
File Edit View Insert Format Tools SYSTRAN Table MathType Window Help Insert Tags
100%
Plain Text
Courier New
10
Page 6 Sec 1 6/40 At 19.3cm Ln 43 Col 33 REC TRK EXT OVR English (U.S
Start
ch3_tracked - Micros...
untitled - Paint
21:27

Figure 4 A LaTeX file opened in Microsoft Word, showing tracked changes

Perhaps the best way to keep track of two versions of the same document – one in TeXnicCenter and the other in Word – is to run *two* PCs (see Figure 5), so that both documents are visible at the same time.

4 Customizing LaTeX

The basic LaTeX "theorem" environment produces a standard result that features a bold "**Theorem x.x**" label and italic text. The publisher may want a different appearance – perhaps using "caps and small caps" rather than bold, or a different style of punctuation or header spacing. What's more, "Definitions," "Lemmas," "Examples" and even "Remarks" are, by default, "theorem-like" environments. The `amsthm` package provides control over these environments, allowing you to customize their appearance.

Similarly, the `fancyhdr` package can be used to control page headlines, captions and so on, `emumitem` provides refinements to the displayed list environments (in fact, it's been used in this document to make bullet lists flush left, and to remove inter-item spacing), and so on. Similarly, headings in this document have been customized using `titlesec`.

The benefit of using add-on packages is that you can control appearance with a view to passing on typographical design instructions. For example, in a coded Microsoft Word edit, a copy editor might use some kind of generic coding to indicate heading levels, example environments and so on, following the general principle that the coding will leave the files in a suitable condition for any design to be applied, through suitable interpretation of the coding.

Figure 5 Two PCs, in a very compact but very useful arrangement! This set-up was forced on me by domestic upheavals, but on this particular LaTeX project it worked very much to my advantage

The opposite approach is to begin with a design in mind, and edit to replicate that design. This may be feasible in journal publishing, or where a book series design is predetermined, but it narrows down the options.

In LaTeX there's no opportunity to add explicit generic coding for design purposes. The whole idea of the program is to add the design layer and generate "final" output.

However, if the book is to be re-flowed through trade typesetting, one or other of the following criteria will still apply:

- either the book will need to be made to follow a certain standard design (perhaps by using a ready-made template);
- or it will need to be made flexible – for example, a bold subheading can be taken as being a reliable indicator of a certain heading level, and can be reinterpreted to replicate a particular design choice.

In reality, copy editing instructions often lie between these two views. For example, while a publisher may ask for coding to be generic, the intention may always be to set figure captions with a bold "**Figure x.x**," followed by a quad space and then a non-bold caption, regardless of other design considerations.

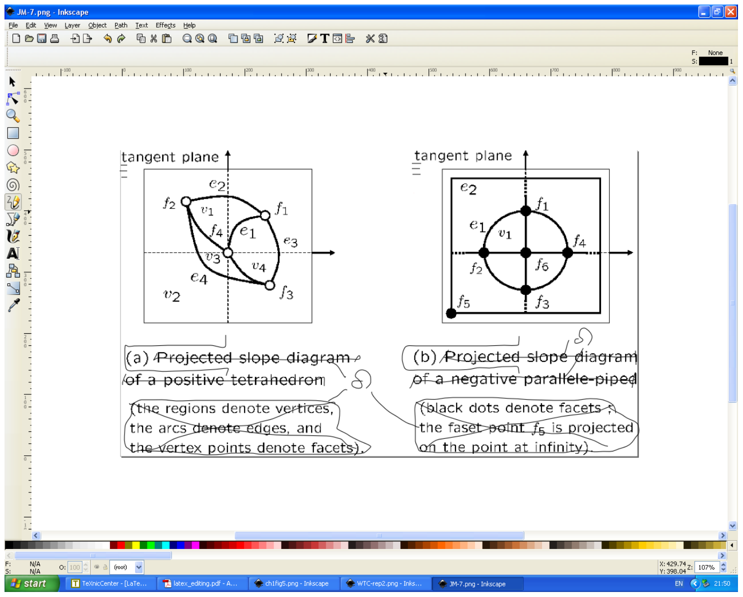
5 Including and Correcting Illustrations

As mentioned above, illustrations tend to be included in the `.tex` files in one of two forms – either as graphics files or in the form of embedded coding. Here, I will concentrate on including graphics files.

The width value in the standard code for the inclusion of a figure provides a means of sizing illustrations on the page, as a proportion of the line width. But what if the illustration also needs to be corrected?

One option is to print hard copy of the illustration, make corrections on that hard copy by hand, and then scan the marked-up copy back into the PC. However, this is a semi-mechanical process that could easily become both time-consuming and wasteful of materials.

A better option, I think, is to load the illustration into a graphics editor and add freehand annotations on screen. There are several useful freeware programs that can facilitate this process, particularly **IrfanView**, which can convert between many different file formats, and the scalar vector graphics program **Inkscape** (see Figure 6).



(a) ~~Projected slope diagram of a positive tetrahedron~~

~~(the regions denote vertices, the arcs denote edges, and the vertex points denote facets).~~

(b) ~~Projected slope diagram of a negative parallele-piped~~

~~(black dots denote facets ; the faset point f_5 is projected on the point at infinity).~~

Figure 6 A marked-up figure in the Inkscape window. The purpose of editing here is largely to remove unwieldy part labels, which can be transferred into the figure caption, and to correct some errors at the same time. The editing marks need to be kept within the figure's "bounding box"

I must emphasize that I don't have the technical know-how to actually correct the electronic figures and produce clean revised versions. My work has been limited to conveying the desired corrections to others.

One of the advantages of LaTeX over word-processing systems is that it isn't necessary to allot final numbering to chapters, sections, equations, tables, figures or any other separable entity. The program takes care of all that. However, given that the next person down the line will probably be using the final `.pdf` file output as the central visual reference, how can that person correlate the figure numbers visible on the `.pdf` file pages with the unique filename identifiers stored with the actual figures? Conversely, if the copy editor saves the amended versions using the final figure numbers, how will the next person link back to the figure filenames? One solution is to draw up a table relating the filenames and figure numbers.

6 Final Deliverables

The deliverables need to include a `.pdf` file that shows all of the graphics. My book experience so far has involved using `dvipdfm` to build that final file.

If necessary, the final `.pdf` file can be "rubber stamped" in places to show where existing illustrations will need to be replaced by amended versions. A freeware `.pdf` file reader called the **Free eXPert PDF Reader** offers a number of preset stamps, one of which very usefully reads "Information Only."

To accompany the `.pdf` file, the publishers and typesetters will need any log of illustration filenames and figure numbers (see above), and any marked-up/amended graphics images (in whatever format).

While it may be necessary to return tracked Word `.doc`s, in general, there will be no tracked `.doc` of the master project file, which remains in TeXnicCenter throughout the process.

7 Conclusion

By means of this scheme, I've successfully edited a complex LaTeX maths book electronically, fulfilling all the specific requirements of that task.

My obvious reliance on freeware packages has had a great deal to do with immediacy – in other words, the need to solve problems quickly. At the same time, the point deserves to be made very emphatically that all this technology is freely available, without the need to invest in expensive proprietary typesetting software.

This method is very much a work in progress, and I welcome any comments and/or criticisms.