HAL
open science

Editing ancient technical and mathematical figures: Tools and traps

Denis Roegel

To cite this version:

Denis Roegel. Editing ancient technical and mathematical figures: Tools and traps. [Research Report] LORIA - Université de Lorraine. 2015. hal-01198446

HAL Id: hal-01198446
https://inria.hal.science/hal-01198446v1

Submitted on 12 Sep 2015

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Editing ancient technical and mathematical figures: Tools and traps

Denis Roegel*

21 June 2015

Abstract

This study examines a number of problems and issues in the edition of ancient technical and mathematical figures, and review a number of solutions and possible extensions to better suit the needs of historians.

Introduction

Scientific articles, books and manuscripts may contain technical drawings. Contrary to more artistic and "free" drawings, technical drawings aim at expressing some abstract properties, or try to give a more or less accurate representation of an object, an instrument, a machine, and so on.

Although it is quite possible to imagine border cases, the features which usually define a technical drawing are its basic components, which are points, segments, curves, colored surfaces, and various labels. Here, we will focus on drawings which could be produced by hand, albeit perhaps not with the same perfection. We therefore deliberately ignore those drawings which can only be generated by a computer, for instance drawings containing a very large number of elementary features such as segments. The simulation of a tree or other natural objects are examples of drawings that will intentionally be excluded from our study. A good rule of thumb might be that most of the drawings of interest to us are those that can be

*Denis Roegel, LORIA, BP 239, 54506 Vandœuvre-lès-Nancy cedex, roegel@loria.fr

drawn in at most a few hours *on computer* and using the above mentioned basic components. Such drawings will be called *diagrams*.

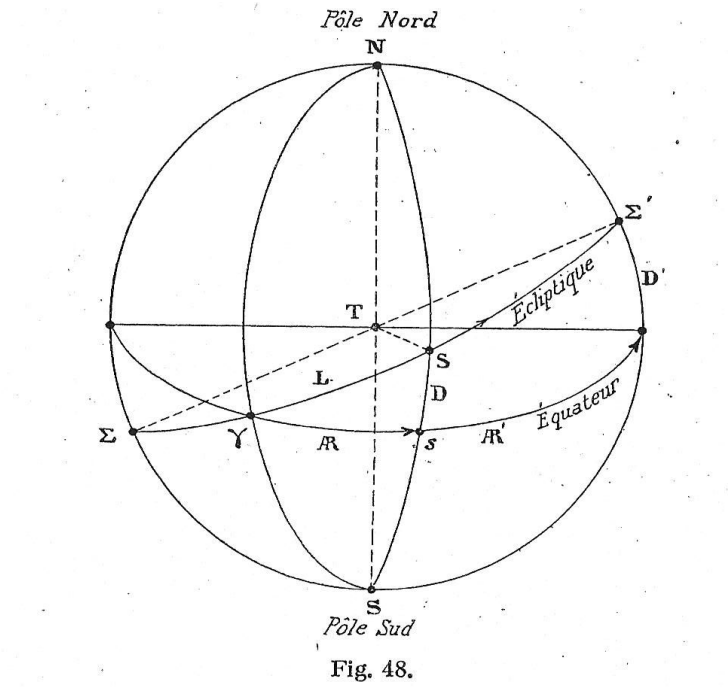


Figure 1: A figure in a book by Bouasse published in 1925 [1].

1 Issues

1.1 A first example

An example of a rather elaborate diagram that we might consider is given in figure 1 and is taken from one of the many books written by Henri Bouasse. A number of issues are raised by that figure. The first is that of its use. Since we have a good scan[1] of the original figure, we can use it in another publication if needed. The figure could then be supplemented by a description which would mention point γ, and give the meaning of other features. So far, so good.

[1] Admittedly, the scan could be better, and we have not removed the numerous stains originating either in the original paper, or in the photocopy which was used.

But we may also want to do more, or at least something else. If for instance we plan to publish a new edition of Bouasse's original book, it might be desirable to produce a faithful reproduction of that figure, but a reproduction which is perfect and without defects. This suggests that the original figure has some defects. Of course, if we look at the figure from the 1925 vantage point, the figure is nearly perfect. It has the quality which might be expected from figures in scientific books of that period. The content of the figure, or, more precisely, the way it was drawn, and the way some of its features are represented, may be objectionable, but this is another story, and we will come back to that matter in a moment.

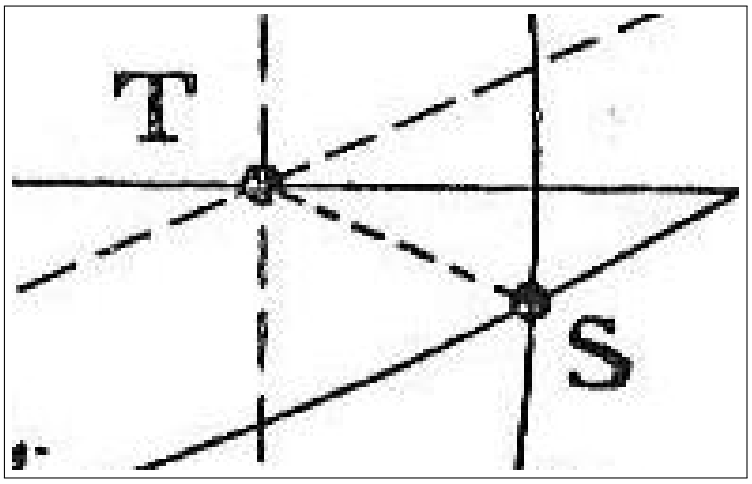


Figure 2: Zooming in on the first figure highlights the fact that it is a bitmap.

1.2 Bitmaps

The main defect of the figure, then, is to our modern eyes the fact that it is a bitmap figure. Or more precisely, the figure which is included in this document is a bitmap. If you try to zoom on it, it will look terrible (figure 2), especially when put next to vectorial fonts or figures. Other examples of bitmap drawings are given in figure 9 and are unfortunately common in today's publications. This problem should be distinguished from problems of the original figure, which may have to do with the printing process, or with some logical features of the drawing. For instance, the dashes between Σ and Σ' are at times irregular, but apart from such minor details, there are no noticeable (micro-)problems.

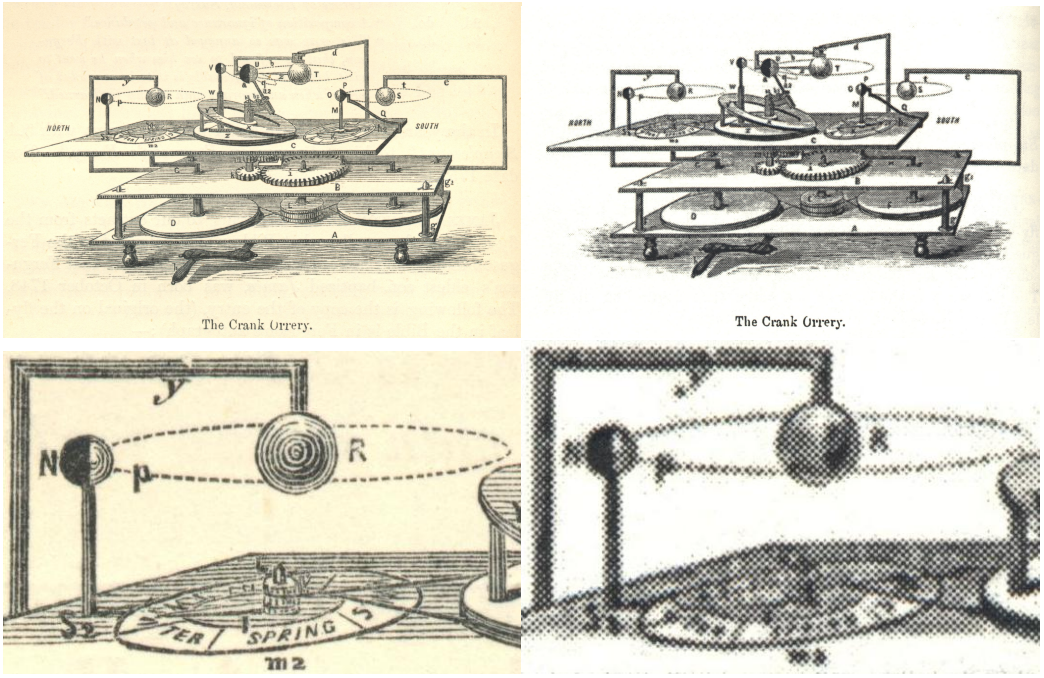


Figure 3: An original figure from 1867 (left) and its reprint in 2010 by Cambridge University Press (right). A number of details have entirely been lost in the reprint. The figures have been scanned at 600dpi from the 1867 original and from the reprint [11]. Interestingly, the *text* of the reprint (not the text of the figures) is well reproduced (see the caption "The Crank Orrery"), and the problem is not merely one of scanning resolution. It seems that the text was digitized at one resolution, and that the figures were digitized at another. It is also possible that the publisher understood that the figures required special care, but that merging figures and text caused the figures somehow to lose most of their interest. It is however surprising that this went down to the printing and was not caught earlier. On the other hand, more and more established publishers (for instance Hachette with the French National Library) are giving in to low quality "print on demand" reproductions.

The fact that the bitmap nature of the included figure is apparent suggests that the original figure was not digitized at an adequate resolution. This is indeed often the case in many of today's reprints of ancient editions, including by well-established publishing houses (figure 3). But the truth is that the original printing had defects, and that no matter which resolution is taken, either the ruggedness of the original drawing, or the insufficient scanning resolution, will show up when one has a closer look at such a drawing. This makes such a drawing awkward for modern computer usages, but it may even be a problem in printed editions featuring reproductions. Of course, most of these problems were irrelevant in the original editions, because nobody would have read a book with a microscope. Now, people do.

1.3 Digitizing a drawing

Although the main point of this article is to detail tools for recreating or improving technical or geometrical figures, we should give some guidelines about the inclusion of facsimiles.

Original drawings, if reproduced at the same scale, should be digitized at at least 300dpi, and a larger resolution will often be better.[2] The threshhold of 300dpi is related to the usual reading conditions and to the resolution of our own eyes. If a drawing is shown in closeup, a larger resolution will hide the fact that the reproduction is a reproduction. Otherwise, if the resolution is insufficient, it will show up, even though it may appear seamless when printed at the original scale. Manuscripts, and also older prints, should be digitized in color, with at least 24 bits per pixel.

1.4 Correctness

Another problem with the original drawing of figure 1 is that it is technically incorrect. The drawing shows a representation of the celestial sphere with three diameters, two meridians, and half of the equator and ecliptic. There are two important errors. First, if the vantage point does not lie in the plane of the equator, the two poles should not lie on the boundary of

[2]The necessary scanning resolution actually does not depend only on how much a drawing or photograph is magnified, but also on how much useful information it contains. The physical constraints, in particular the optical constraints, of the production of an original such as a digital photograph put a limit to that information, and hence to the useful resolution. For a deeper analysis of the problems of digitization in libraries and archives, see our recent study on sustainable digitization [25].

the sphere, but more inside.[3] Second, the great circles should be tangent to the outer circles, which they are not, especially for one of the meridians and for the ecliptic. These representation choices were common in the past, and the figures were not always meant to be accurate. We therefore have to consider whether these features should be kept or not. In some cases, the drawings might be improved.

1.5 Reproducing or improving

The problems of Bouasse's figure leads us to ponder the questions of reproduction or improvement. By "reproduction," we mean a faithful recreation of a figure, without logical improvements. Usually only the lines or curves become perfect, but the points are those of the original figure. An "improvement," instead, will introduce deeper changes.

When reproducing a figure, the first question to ask is why it is reproduced. Do we merely want to show the original drawing? Or do we want to edit it as part of a critical edition or a translation? Do we want to translate the labels and make them readable? Is the original figure so faded that it needs to be edited? If we recreate (or reconstruct) a figure, we have to ask ourselves whether we want to improve it, whether we want to show it as it was intended, or whether we only want to retain some of its features, and which ones.

A figure is of course an abstraction of some concept or object. When we reproduce a figure, when we recreate it, we create a new abstraction, we add a layer to the interpretation of the concept or object. If Bouasse's figure is corrected, we may be satisfied of having improved his figure, but at the same time we conceal the original work, and we are perhaps betraying the author's intentions. Whenever there is a doubt, it is probably best to show both the original drawing and its recreation.

Finally, an exact reconstruction sometimes helps to locate errors or omissions in an original figure. As an example, we can consider the rather complex drawing of gears by Olivier (figure 4, from [22]). Several problems were found during the reconstruction of that figure.

[3]For a possible solution to these problems, see our work on the drawing of great circles and parallels on a sphere [24]. Although our work is not totally general, and cumbersome in places, it does highlight a real problem in that kind of drawings.

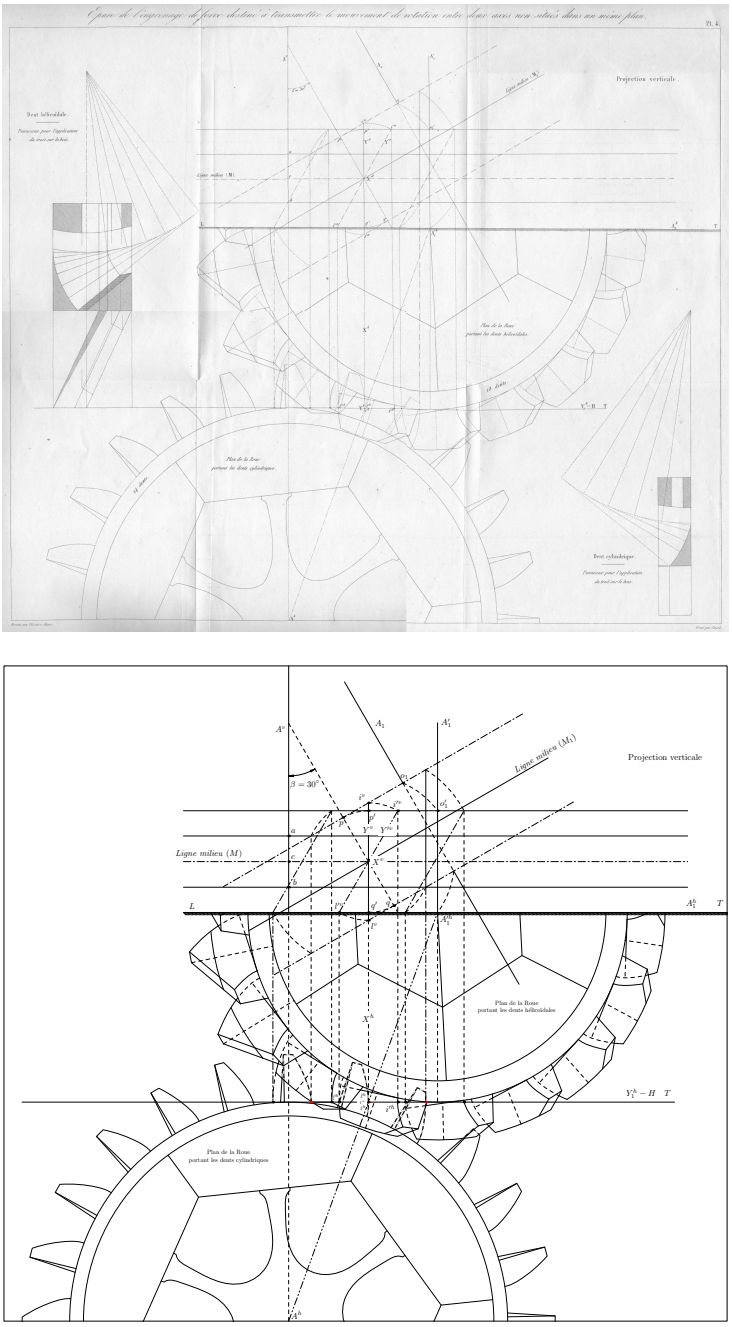


Figure 4: An example of a partial reconstruction of a complex drawing with METAPOST [22]. Top: original drawing. Bottom: reconstruction.

2 The quality of reconstructions

Depending on the purpose of a reconstruction, some tools may be more adapted than others. If the accuracy of the geometry is important, and if some points on separate lines should coincide (such as when two circles are tangent), then exact positions or intersections should be used. This is not always what is desired, For instance, in points A and G of figure 5 (traced from a manuscript), one may wonder if the lines were intended to overlap. I believe they were meant to.

This does however raise the issue that if something is meant to overlap, then the overlap should be visible without zooming. If we want to stress some feature of a drawing, we should magnify it, not conceal it, as this then leaves some doubt about the *intent* of the author. Therefore, if the lines reaching A and G had been meant to overlap, we think that they should have overlapped much more.[4,5]

However, the same author then gives a new construction, which only keeps some of the features of the original, and which is not traced from the original. One might therefore expect an exact reconstruction, or at least good junctions, and this is not the case (figure 6).

Figure 7 is another simple example, with an unfinished arc E, and inadequate junctions at B. It would surprise us if such features had been intended.

Another example is given by Friberg in figure 8. The figure is rather complex, and it is obvious that Friberg must have gone to a great length to get it as right as possible, with whatever software was used. But when you look closely at R, Q, and many other points, it will look rather messy...

Yet another subtle example is shown in figure 11, where point E was obviously placed by hand, without computing the actual intersection.

[4]We hope that our remarks are not taken as personal criticism. We are particularly aware of how difficult it is to produce some drawings, and how sometimes we become slaves of the tools, and also how sometimes some figures are messed up by third parties...

[5]One might also argue that the faithfulness to the original diagram implies that one cannot be sure that the lines will actually meet. On the other hand, given that we use smooth curves, which already depart from the original drawing, why should one want to avoid exact junctions in the very case where an approximate junction can only be noticed when zooming? This seems counterintuitive.

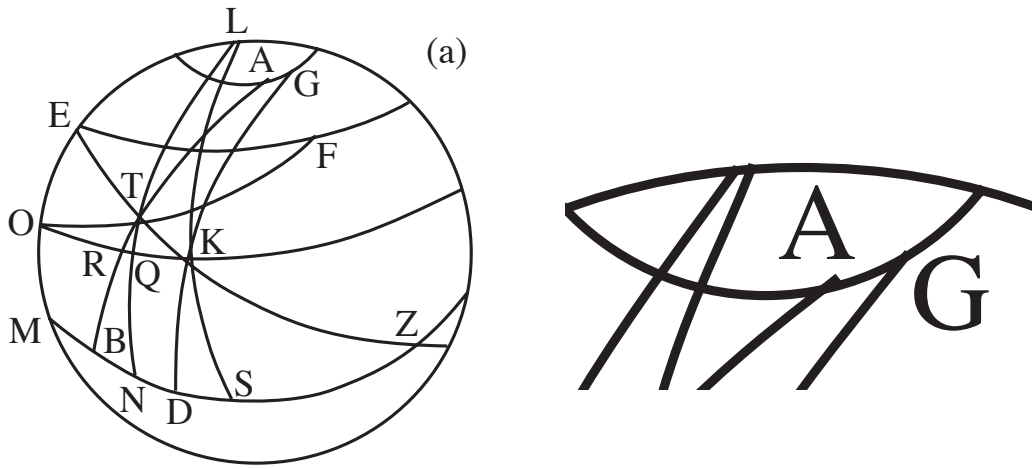


Figure 5: An excerpt of a recent reconstructed figure [32], where some junctions overlap. The overlap was probably intentional, or rather, the author intentionally did not try to obtain perfect junctions. Left: complete figure. Right: detail. The figure shown here is exactly that of the original article.

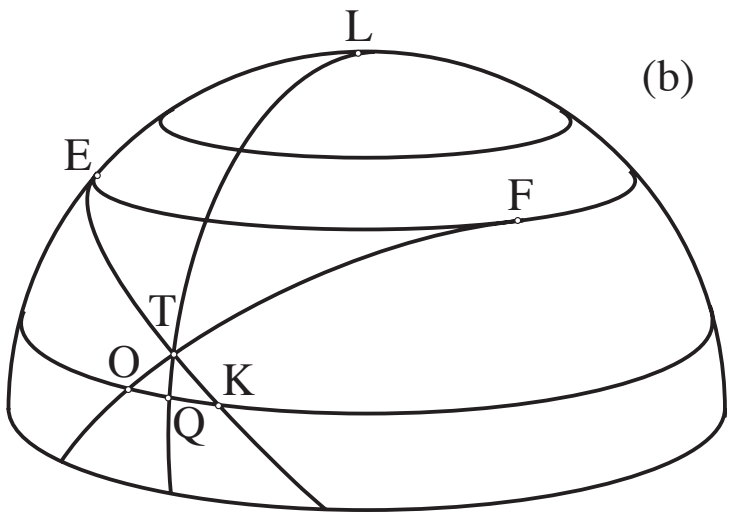


Figure 6: A new figure based on figure 5 [32]. The intent of the author is now to show a perfect construction, but that result is not achieved as every circled point is mispositioned and as the end curves are usually not tangent, see for instance in E (by zooming), and especially on the right side of the figure. The figure shown here is exactly that of the original article.

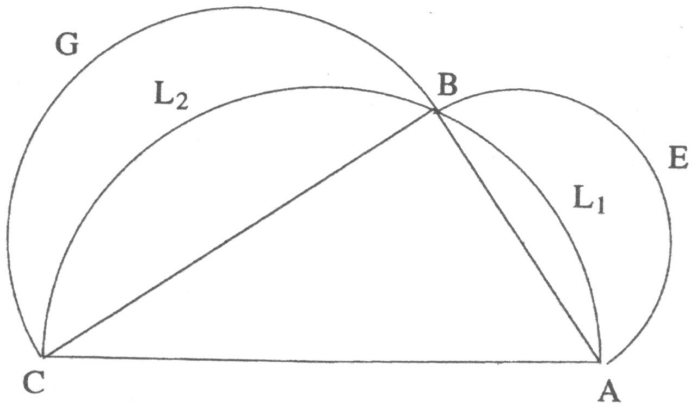


Figure 7: A drawing by Rashed [19, p. 107].

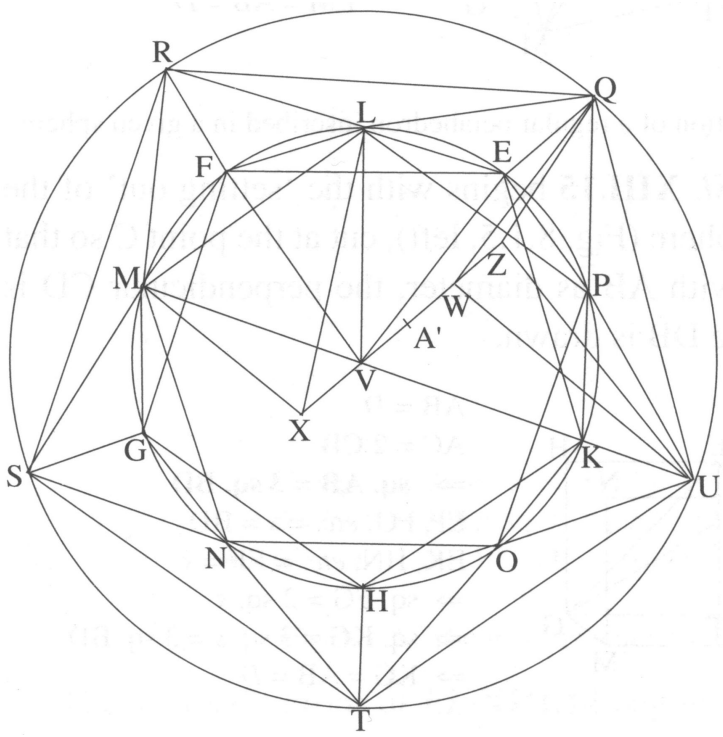


Figure 8: A drawing by Friberg [8, p. 176].

Another curious example is given in an article by Panza [17]. There, the curves of one of the figures strangely became wiggled (figure 10). This was certainly not intended, and perhaps even went unnoticed?

Some authors use software which is acceptable, but somewhere on the line the figures get mangled. Examples are for instance given by Malpangotto and Le Meur (figure 9) whose figures have all become bitmaps, although they were certainly vector graphics at the beginning [15, 13]. The loss of the vectorial nature of the drawing may have occurred with the publisher, perhaps when EPS graphics were transformed in JPEG for inclusion in the articles. A number of years ago, the same happened with one of our articles [23]. We had prepared a vector graphics and provided it as a PDF file. Alas, it was included as a bitmap. This is very noticeable in the online version, but almost invisible in print.

These examples show that the tools alone are not the solution, but what also matters is how the tools are used.

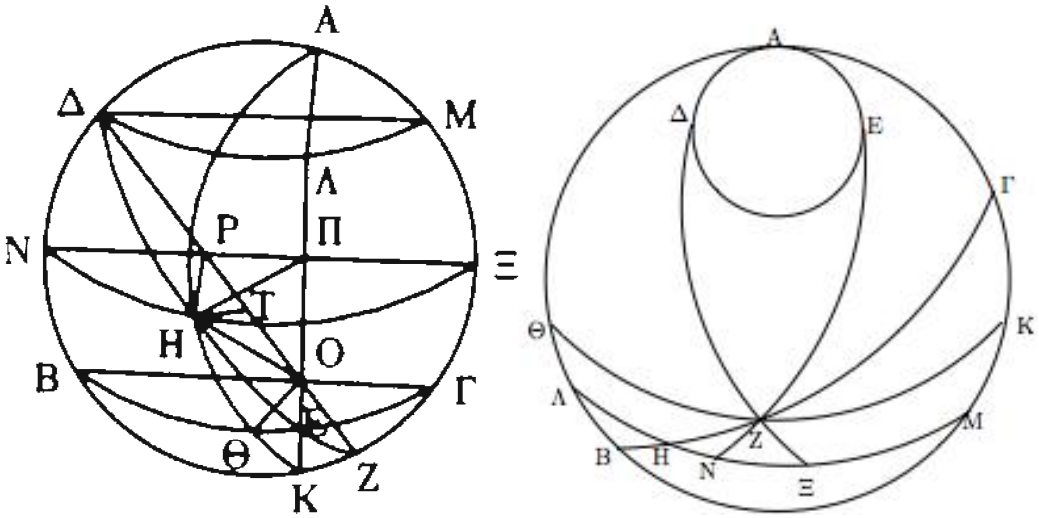


Figure 9: Recent figures by Malpangotto [15] and Le Meur [13]. These figures appear as bitmaps in the PDF and are shown here exactly as they appear in the original articles.

Now, in most cases, the imperfections are barely noticeable, especially in print. The authors often go to a great length to produce the best drawings, and it is only by scrutinizing the figures that problems become apparent. Most of the problems are about junctions: lines do not meet, or they overlap.

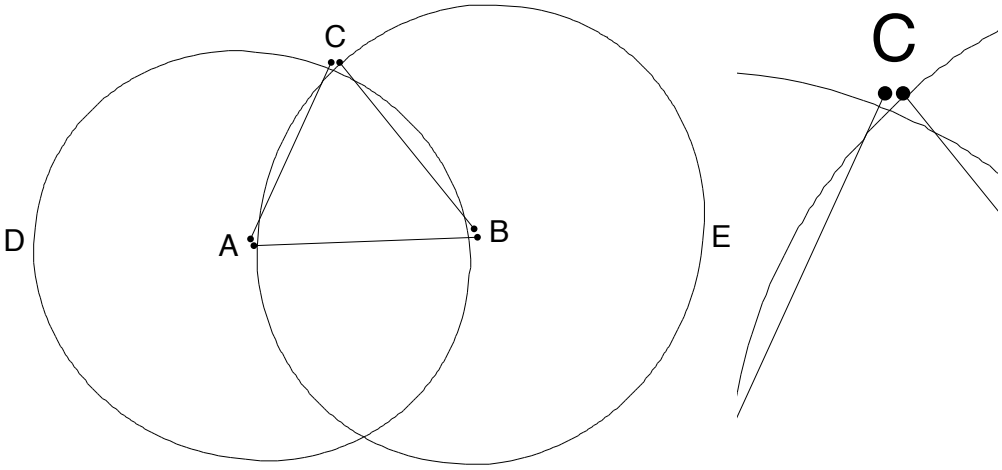


Figure 10: A recent figure by Panza [17]. Left: complete figure. Right: detail, with wiggled curves. (The figure is actually taken from the preprint, but the actual published online version is even more bizarre…) The figures shown here are exactly those of the original article.

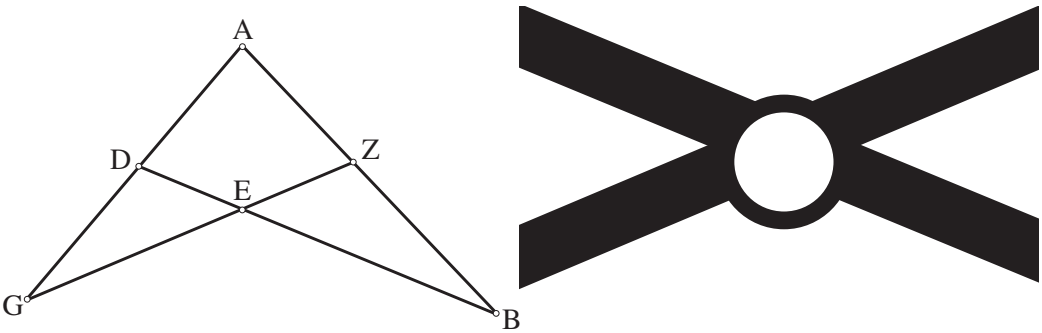


Figure 11: Left: a recent reconstructed figure, where the circled dot at point E is slightly offset from the actual intersection [33]. This is almost unnoticeable without zooming. Right: detail at point E. The figures shown here are exactly those of the original article.

3 New figures *vs.* traced figures: requirements

We can distinguish two types of figures. First, a new figure may be based on an existing one, but without the purpose of reproducing all of its features. For instance, when a manuscript shows a square which is not totally square, the new figure may show an exact square. Traced figures, on the other hand, will try to retain some of the imperfections of the original figures, such as "squares" with non parallel sides, or with lines that do not exactly meet.

In both cases, a number of techniques are useful for the edition or the creation of a new drawing. These techniques overlap to a large extent, so that it would be pointless to describe first those techniques useful for tracing a figure, and then those useful for creating a perfect drawing from scratch.

We will first review some of the basic components and operations involved in the creation of figures, and then consider what are the choices that we are offered and how we should decide on what type of reconstruction is needed.

3.1 Features

3.1.1 Setting points

In order to work on ancient diagrams, there are a number of important requirements. First of all, we need to set points, by obtaining them from another figure. There must therefore be some way to relate an original drawing to a new one. This, however, does not entail that we have to use an interface or even to include the original diagram as an overlay. It is quite possible to take measurements on a point without scanning a figure.

3.1.2 Tracing and cropping

Tracing is the operation of creating paths based on existing images. The most common paths are straight lines (segments) and curved lines, or arcs.

We should be able to draw lines of various thicknesses, colors, and styles, such as dashed lines. There should also be different styles of points, for instance the possibility to draw Neugebauer-style points (figure 12).

Some figures may be cropped using closed paths. We will not enter into the details of cropping, but some tools make it possible to crop a figure according to a user-defined path.

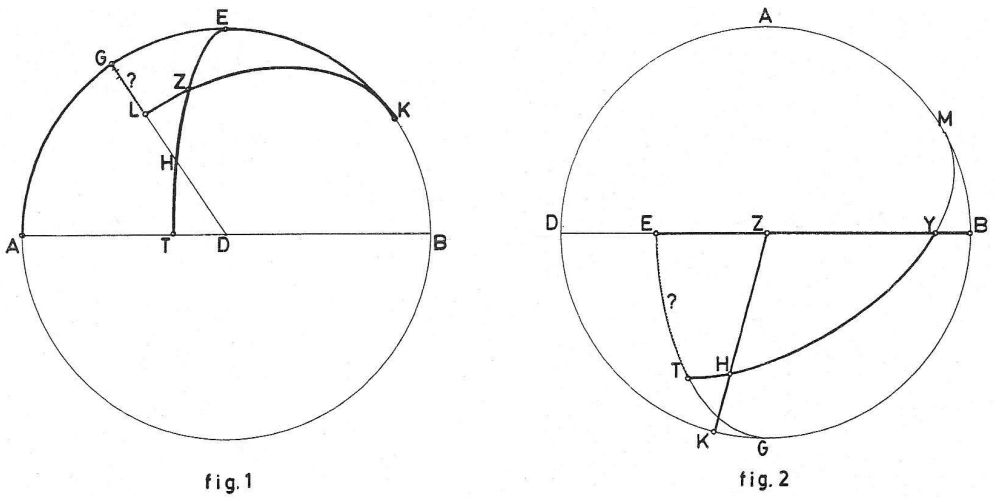


Figure 12: An example of a figure using the Neugebauer-style points [6]. This style is for instance also used in figures 6, 11, and 17.

3.1.3 Labels

We also need to be able to add complex labels, perhaps involving ancient alphabets.

3.1.4 Overlays

We may want to write over existing figures, that is, not only to base paths on existing figures, but also to retain the underlying figures (figure 13).

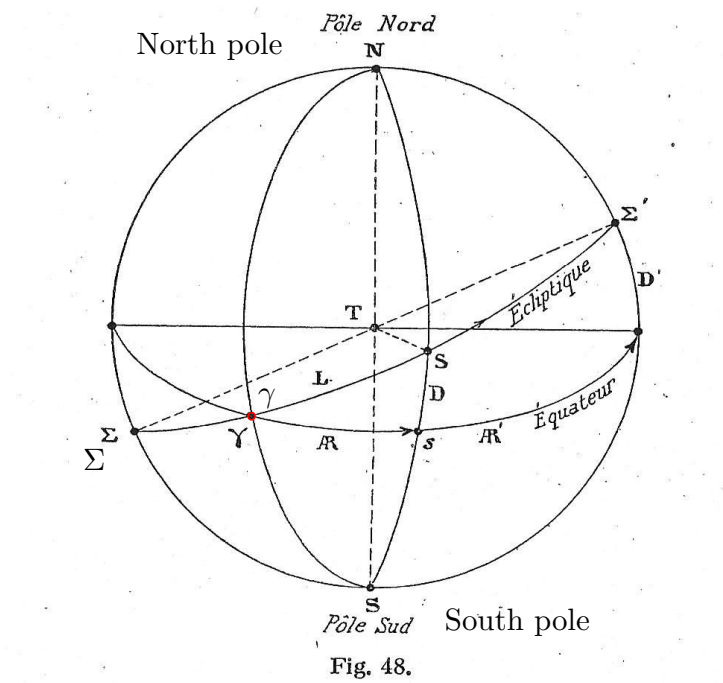


Figure 13: An example of overlay. Taking as a base the first (bitmap) image, we have added several labels, as well as a red dot. All these additions are vectorial, as zooming in will show.

3.2 Choices

3.2.1 Intentions

In order to reconstruct a figure, one has in fact to get a certain understanding of the figure, and also of the intentions of its author. For instance, drawing a diagram goes by steps: points (or starting points of lines) are chosen, *then* lines are drawn. There are intentions. A person may want to draw a straight line, or may want a line to be understood as straight, but either does not exactly manage to draw such a line, or does not care to draw the line that he/she could draw. In the case of figures 5 and 6 (and any other reconstruction), we have to distinguish the intention of the original figure (and author) and the intention of the reconstructed figure (and author). These may not be the same.

3.2.2 Tracing, or abstracting away?

The edition of ancient figures represents an interpretation of the figures. The most faithful representation is that of a facsimile. In certain cases, facsimiles can be made more readable by enhancing faded lines, but such enhancements should be made explicit, that is, they should not be concealed.

If facsimiles are not used, then we usually only want to show the relevant features of a drawing. We have to decide which features are important to us, and which ones are not. Sometimes, we want to highlight that an original drawing had lines that did not meet. But we can achieve this without having to trace the original drawing. Once this is understood, reproducing a drawing can be made much simpler.

If we want to show the differences between the figures of various manuscripts, a good idea might be to give facsimiles, but also approximate reconstructions. Tracing the original figures is perhaps not always necessary.

Of course, some features are most naturally drawn by hand, for instance the edges of a broken tablet.

Figure 19 shows an example of a figure drawn entirely in METAPOST, without any tracing, and aimed at being an approximation of the earlier figure 11. The dimensions and angles could be fine tuned, but in many cases an exact reproduction is not sought.

4 Tools

In this section, we are going to survey a number of tools and use them in order to illustrate the essential features of technical drawings. Our survey will not be comprehensive, though. We will restrict ourselves to the production of drawings for LaTeX.[6] And we will only consider representative cases. We will examine several graphical tools, and several ones based on programming.[7]

4.1 Tools with mostly mouse interaction

In this category of tools, a figure is created using the mouse and by selecting various options in an interface or in menus. Parts of a drawing can be selected and modified. Here, we will only discuss two tools, one very specialized, the other much more general.

4.1.1 DRaFT

DRaFT is a small Java program created by Ken Saito for the purpose of editing figures from ancient mathematical manuscripts [27, 29]. Its main features are the ability to trace lines or curves over an original figure given by a JPG file. DRaFT also allows for the inclusion of labels. Labels can be scaled or rotated. Lines can be colored or dashed.

When using DRaFT, the user defines points and these points are listed in a special window. Then the user joins some of these points using several types of curves.

DRaFT produces an EPS file which can then be postprocessed and included in a document. It has been used in several recent articles, in particular those by Saito, De Young, Sidoli, Malpangotto, and Le Meur [26, 31, 32, 3, 15, 5, 4, 28, 30, 13, 33].

An example of a figure traced with DRaFT is given in figure 15, based on an original manuscript (figure 14).

4.1.2 Inkscape

Inkscape is a general open-source program for vector graphics. It is not targetted towards the edition of ancient diagrams, but it can easily be used

[6] LaTeX is free, open-source, and available on all usual operating systems.

[7] We will not consider other tools such as automatic tracers (for instance `potrace`) which trace a figure without manual intervention and produce vector graphics, as these tools still leave the burden of simplifying the original figure.

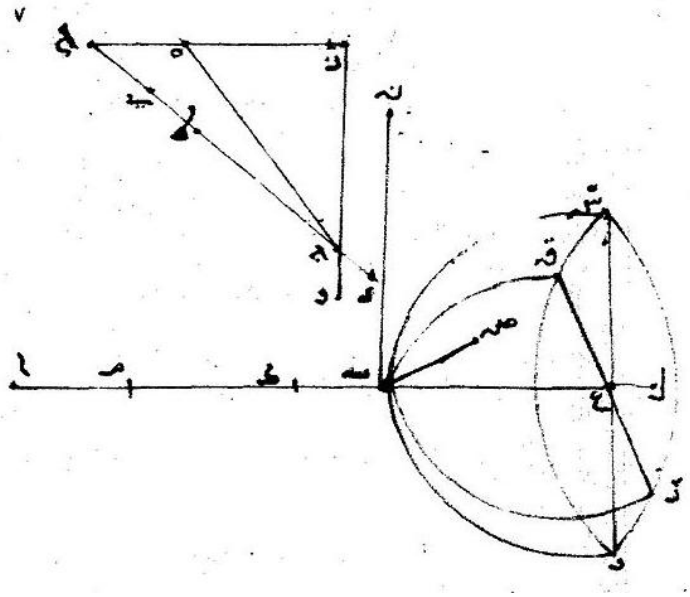


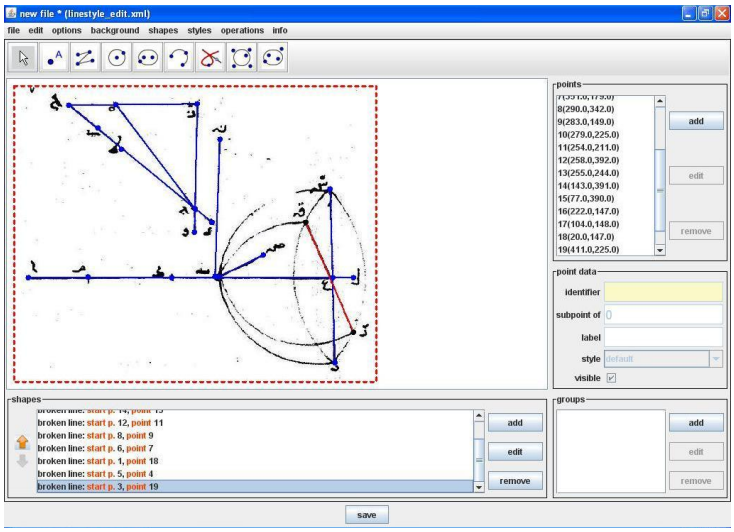
Figure 14: The original diagram used in DRaFT's tutorial [29]. (source: file `Ibn_Sahl_manuscript.jpg` on Wikimedia)

for that purpose. Inkscape outputs SVG.

As an example, we have tried to obtain more or less the same diagram as the one given in DRaFT's tutorial. The two figures, the one by DRaFT (figure 15) and the one by Inkscape (figure 16), are not identical, because we did not try to make them totally identical. Instead, in both cases, the original image was loaded in the program, the image was manually traced by the introduction of points and adequate lines, and arabic labels have been added, scaled and rotated.

Since Inkscape has many capabilities, it may prove more difficult to use, especially for beginners, because we have to wander through many buttons and menus that we do not need. In that sense, a simple program such as DRaFT may be more practical for focused beginners. But the fact is that Inkscape is quite usable for that task. In our experience, in order to trace a figure as quickly[8] as possible, one should set the straight or curved lines (with Bezier curves), placing points in several positions, in particular wherever several lines will meet or cross. For then, these points can serve as anchors for future points. (If this is not done, points can be added later,

[8]Of course, a good drawing takes time, and we are not suggesting that work should be messy, but we are looking for efficiency, and some approaches are more efficient than others.



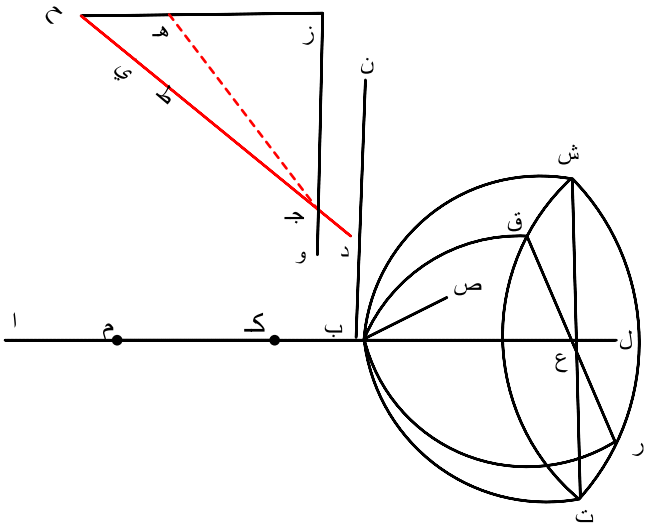
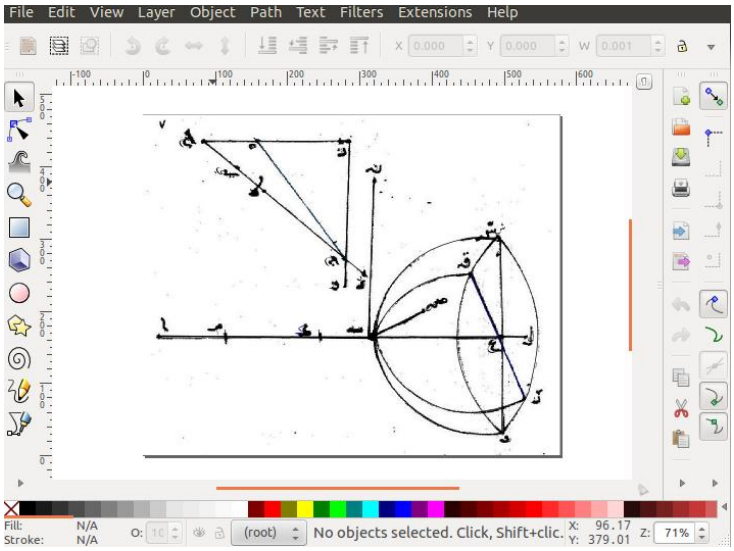


Figure 15: DRaFT's interface (above, from [29]) and the diagram produced by DRaFT, as given in the tutorial [29]. Notice (by zooming) that the lines apparently meeting at point ت actually do not meet, although they were probably meant to. There are also other anomalies.



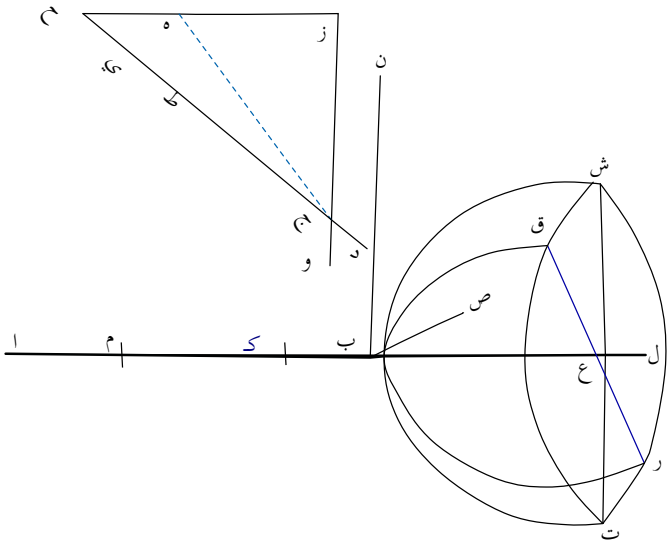


Figure 16: The reconstruction of DRaFT's diagram with Inkscape. Our junctions are not all perfect, and there are other defects, but these could probably have been improved in places. We have decided to leave the problems as they are, for instance at point ش where two lines overlap. It is important to use round caps for the end of lines, and this can be selected in Inkscape. It will improve the appearance of oblique junctions.

but it will be easier to set them right away.)

By comparison, in DRaFT, points are defined, and then curves are defined going through the points. This approach somewhat brings us closer to a programmatic one, but in fact it was above all simpler to make it that way in DRaFT. It is possible that future versions of DRaFT will allow a more interactive construction of a curve, but this has advantages and drawbacks.

When curves are defined in Inkscape, especially when tracing over an image, it is best to highlight the curves by giving them a good color and thickness.

An important problem when tracing curves is that of distinct curves meeting at some points. If no special care is taken, the curves will not meet well, like in some of the examples shown earlier. If the endpoints of two curves should coincide, then it is possible to extend one curve by the other and to merge the two points within Inkscape. However, if the points to be merged are not both endpoints, then the best way to merge the points is to select the "snapping" option, so that one point "falls" on the other, when they are sufficiently close. But even if this is not done, one always has the possibility to zoom on the drawing and to manually move the points so that they are sufficiently close. This will not be perfect, but it will be better than nothing.

Labels are easy to include and can either be written directly in one of the available fonts, or copy-pasted from another program. In our reconstruction of DRaFT's example, we have used the `Scheherazade` OpenType font which is installed on our computer, and which is freely available.

Inkscape produced a SVG file, and this file was converted to PDF using the program `rsvg-convert`. It was then included in our document.

Incidentally, we have departed from DRaFT's construction in a few ways, either with the choice of labels, with the two ticks on the horizontal line, or with a couple points that we chose to split in order to be more faithful to the original diagram.

4.2 Tools with mostly programming

Tracing a figure with an interface is not the only way to do it. In fact, we have to admit that we found it very cumbersome to trace a drawing with Inkscape, having constantly to figure out on which button to click, which mode we were in, selecting the font, moving some labels which were sometimes very small or very thin, so that we tend to prefer the programmatic approach.

With such tools, important requirements when tracing a drawing are to be able to hide the original drawing after the tracing is completed, and to have help lines for positioning additional points. Also useful is the ability to switch between various sets of labels (latin, arabic, etc.).

4.2.1 METAPOST

METAPOST [12] is a companion program to TEX and is particularly well suited for technical drawings. A METAPOST file is a text file which is processed by the METAPOST program and can produce either PostScript, EPS, or SVG. We will not enter into the details of the various formats, but formats can be converted and METAPOST outputs can be included in various documents, not only in LATEX documents (such as the present one).

Figures 17 and 18 show an example of a figure constructed with METAPOST, with one point (A) obtained as an intersection of two circles (not shown), and other points determined using rotations. This example illustrates a proof of Pythagoras' theorem, taken from Euclid's *Elements* (Book I, Proposition 47).

Figure 4 shown earlier is a much more elaborate reconstruction. In that example, the METAPOST figure depends only on a few parameters and aims at producing an exact construction in descriptive geometry. Such an exact construction may help (and in fact, did help) to identify omissions or errors in the original drawing.

We have used METAPOST to trace the figure used in DRaFT's tutorial. This was achieved using the `exteps` extension which allows for the inclusion of an EPS file within a METAPOST figure. Figure 20 shows the original figure after it was traced, together with the grid used for setting the points, as well as the traced figure alone. In order to obtain this capability, we had to make a modified version of the `exteps` extension, since the available version was meant for a permanent figure inclusion.

Figures 21 and 22 show the METAPOST source code for the first figure (with the included EPS). The core of the code is made of the definition of various points (`z0=(3u,28u)`, etc), of various curves (using `epsdraw`)

and of labels (using `epslabel`). Setting the boolean `grid` to true displays a grid, which is useful for finding the coordinates of the points, expressed as a fraction of the whole width. When tracing is done, the original figure can be hidden by setting `showeps` to false.

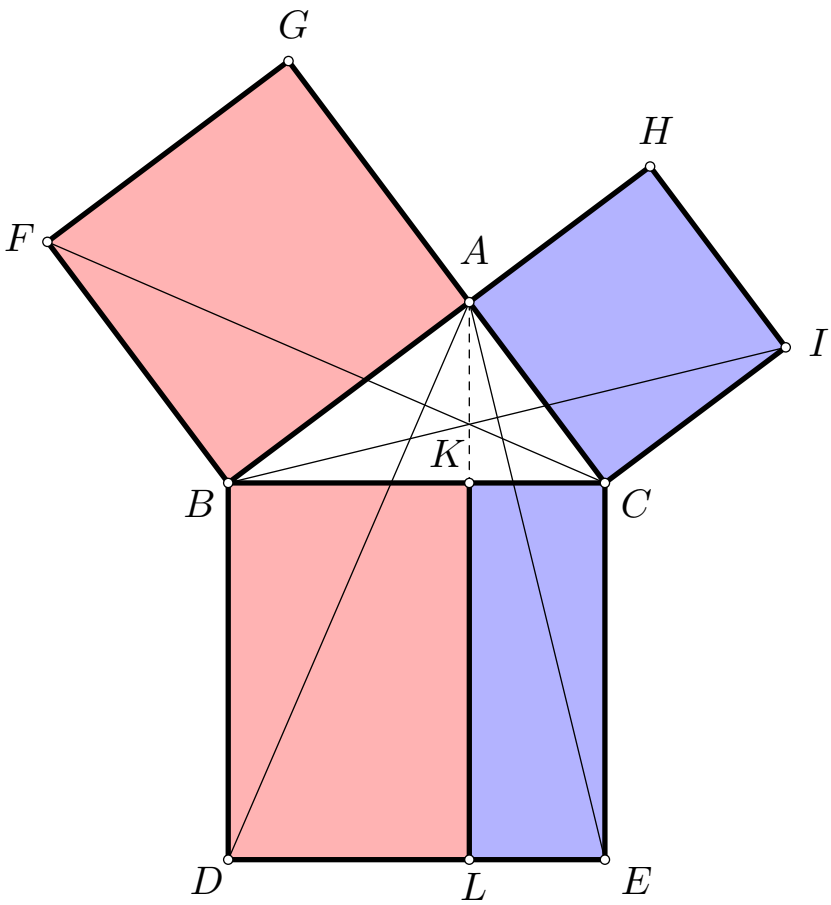


Figure 17: A METAPOST figure for a proof of Pythagoras' theorem (Euclid, *Elements*, Book I, Proposition 47). Figure 18 shows the source code. (The pink areas are equal, and so are the blue ones. In the first case, the area of the square $BAGF$ is twice that of the triangle FBC. And this triangle is equal to the triangle ABD, which is itself equal to half of the rectangle $BDLK$. The second case is similar.)

```
beginfig(1);
  numeric u;u=1cm;
  pair A,B,C,D,E,F,G,H,I,K,L;
  B=origin;C-B=(5u,0);
  A=(fullcircle scaled 6u shifted C) intersectionpoint
    (fullcircle scaled 8u shifted B);
  F-B=G-A=(A-B) rotated 90;
  H-A=I-C=(C-A) rotated 90;
  B-D=C-E=K-L=(C-B) rotated 90;
  A-K=whatever*(B-D);K=whatever[B,C];

  fill B--A--G--F--cycle withcolor (1,.7,.7);
  % ... (the three other areas)

  pickup pencircle scaled 2pt;
  draw A--B--C--cycle; % triangle
  draw A--G--F--B--D--E--C--I--H--cycle;
  draw K--L;
  pickup pencircle scaled .5;

  draw F--C;draw A--D;draw B--I;draw A--E;
  draw A--K dashed evenly;

  pickup pencircle scaled 4pt;
  for $=A,B,C,D,E,F,G,H,I,K,L:
    neupoint($); % for circled points
  endfor;
  pickup pencircle scaled .5;
  label.top(btex $G$ etex,G);
  label.lft(btex $F$ etex,F);
  ...
endfig;
```

Figure 18: The source code for figure 17 (some parts omitted).

```
numeric u;u=1cm;

% circled points
def cpoint(text c)=
  forsuffixes $=c:
    pickup pencircle scaled 2.5pt;
    draw $;
    pickup pencircle scaled 1.5pt;
    draw $ withcolor white;
  endfor;
enddef;

beginfig(1);
  pair A,Z,G,D,E,B;
  A=origin;A-G=5u*dir45;Z-G=5.4u*dir20;
  D=point .55 of (G--A);E=point .7 of (G--Z);
  B=whatever[D,E]=whatever[A,Z];
  draw A--G--Z;draw A--B--D;
  cpoint(A,Z,G,D,E,B);
  label.top(btex A etex,A);label.llft(btex G etex,G);
  label.lrt(btex B etex,B);label.urt(btex Z etex,Z);
  label.ulft(btex D etex,D);label.top(btex E etex,E);
endfig;
end
```

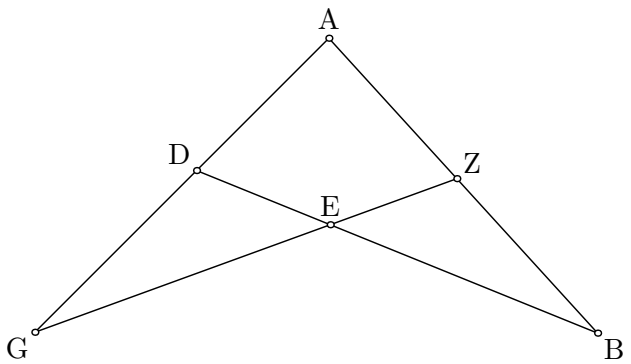


Figure 19: A better reconstruction of the figure given earlier (figure 11), but with the circled points correctly positioned. This figure was not traced from the earlier one, only constructed by visually guessing the dimensions and deducing the position of B from those of D, E, A, and Z.

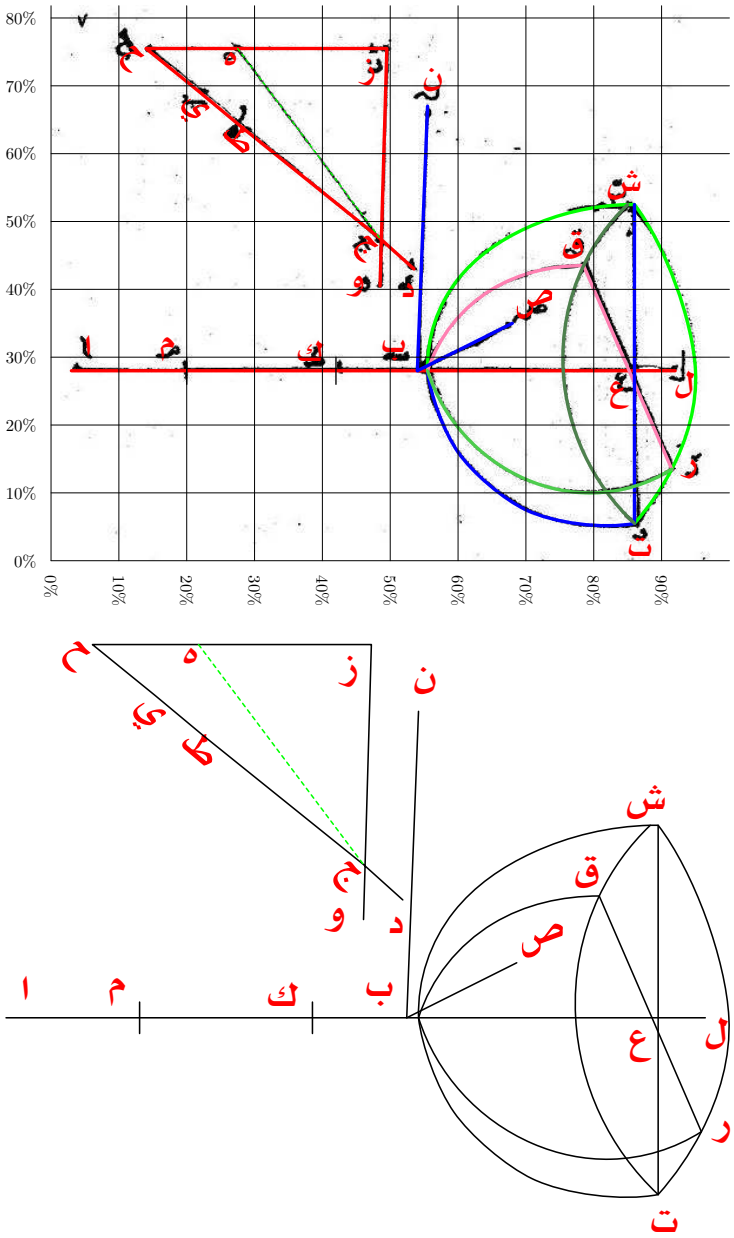


Figure 20: The reconstruction of DRaFT's example with METAPOST and the `exteps` extension.

```
verbatimtex
%&latex
\documentclass{article}
\usepackage[LAE,LFE,T1]{fontenc}
\usepackage[utf8]{inputenc}
\usepackage[arabic,english]{babel}
\begin{document}
\newcommand\ar[1]{{\selectlanguage{arabic}#1}}
etex

input exteps-roegel % modified version of exteps
showeps=true;
%showeps:=false;
prologues:=2;
beginfig(1);
begineps "saito-fig.eps";
  base := (0,0);
  scale := (0.7,0.7); % overall scaling of figure
  clipping := true;
  grid := true;
  u=pct;
  labeloffset:=1.5u; % distance of labels
  defaultscale:=2; % scaling of labels
  z0=(3u,28u);z1=(20u,28u);z2=(42u,28u);z3=(54u,28u);
  ...
  z27=(22u,69u);z28=(28u,64u);z29=(48.5u,47.5u);
  z30=(53.5u,43u);
```

Figure 21: The METAPOST code with `exteps` (1/2).

```
  drawoptions(withpen pencircle scaled .5u);
  epsdraw z0--z1--z2--z3--z4--z5--z6--z7 withcolor red;
  epsdraw z4..z9..z10..z11--z12
     ..tension1.2..z8..z18..z13 withcolor green;
  epsdraw z13..tension1.2..z14..tension1.2..z15
                ..tension1.2..z4 withcolor blue;
  epsdraw z4..z16..{right}z17--z18 withcolor (1,.5,.7);
  epsdraw z12--z13 withcolor blue;
  epsdraw z11..z17..z5..z13 withcolor (.3,.5,.3);
  epsdraw z4..z19..z20..z18 withcolor (.3,.8,.3);
  epsdraw z3--z21 withcolor blue;
  epsdraw z3--z22 withcolor blue;
  epsdraw z23--z24--z25--z26--z27--z28--z29--z30
                    withcolor red;
  drawoptions(withpen pencircle scaled .2u);
  epsdraw z25--z29 withcolor green dashed evenly;
  % ticks
  drawoptions(withpen pencircle scaled .2u);
  epsdraw (z1+(0,-2u))--(z1+(0,2u));
  epsdraw (z2+(0,-2u))--(z2+(0,2u));
  drawoptions(withcolor red);
  epslabel.top(btex \ar{...} etex,z0);
  epslabel.ulft(btex \ar{...} etex,z1);
  ...
endeps;

endfig;
end
```

Figure 22: The METAPOST code with `exteps` (2/2).

4.2.2 Ti*k*Z

Figures can also be traced with Ti*k*Z [34], a generic graphical TEX environment. The approach is somewhat similar to that of METAPOST. With Ti*k*Z, points are put in place and lines are drawn. Curved lines are a bit more tricky, because the curves provided by default in Ti*k*Z make it hard to go through a sequence of points. Bezier curves can be drawn, but control points have to be carefully selected. This is what we did, but it is rather cumbersome. A much easier way to draw curves is to use the "`hobby`" extension of Ti*k*Z (named after John Hobby, the author of METAPOST's path drawing algorithm), which makes it possible to define curves the way they are defined in METAPOST. This extension requires a rather recent version of LATEX, since it uses advanced LATEX3 features and we haven't used it here.

Another possibility in order to facilitate the positioning of points is to use the Ti*k*ZEdt editor (`http://www.tikzedt.org`).

Like we did with METAPOST, we have used Ti*k*Z to trace the figure used in DRaFT's tutorial (figures 23–25). We have in particular used this example to illustrate "label switching," that is, the possibility to go from one set of labels (for instance Arabic) to another (for instance Latin). In order to achieve this functionality, all labels are defined with the `\lab` command which takes two parameters. A boolean is then used to select which language we wish to display. This could also have been done with METAPOST.

We have also used the `onimage` package which provides the simple `tikzonimage` environment. One of the arguments of this environment is the base figure.

An option to this environment will add help lines. And once the tracing is done, the original figure can be hidden by setting an appropriate boolean to false.

Points could have been drawn using various styles, such as the one popularized by Neugebauer (see figure 11).

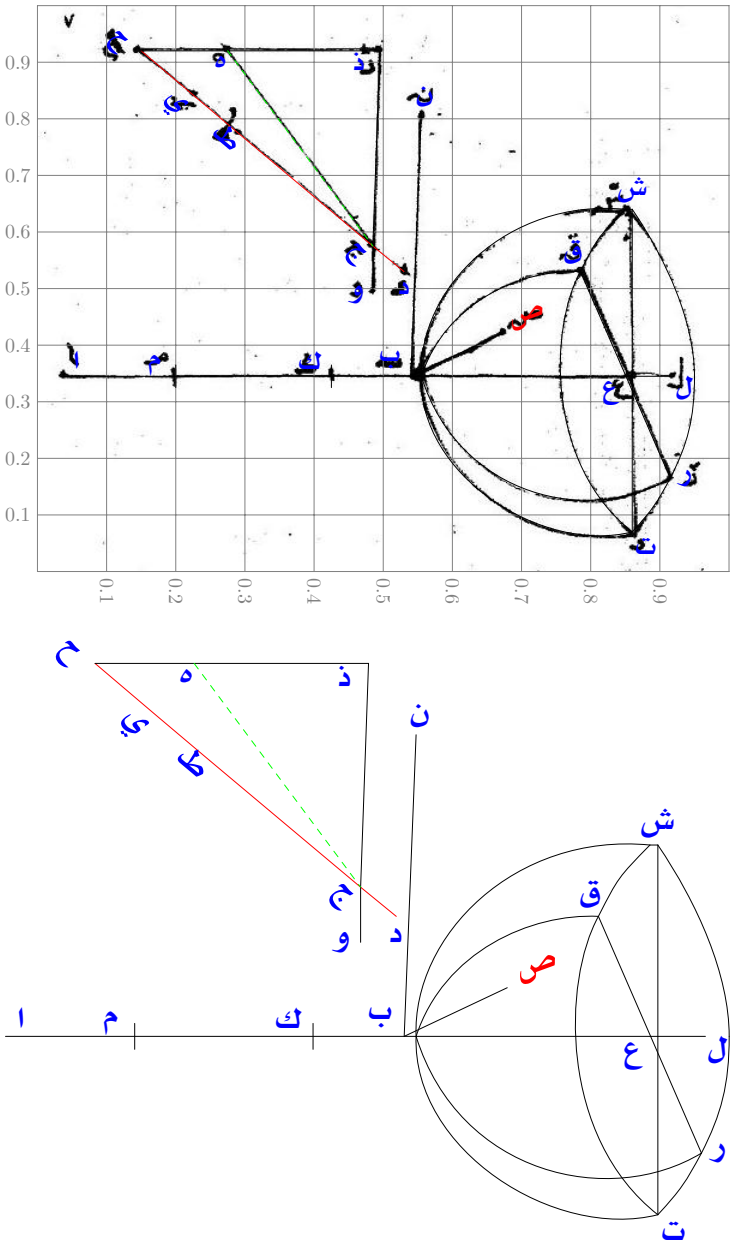


Figure 23: The reconstruction of DRaFT's example with Ti*k*Z. Care was taken to use rounded caps for the lines.

```
\documentclass{article}
\usepackage{onimage}
\usetikzlibrary{calc}
\usepackage[LFE,LAE,T1]{fontenc}
\usepackage[utf8]{inputenc}
\usepackage[arabic,english]{babel}

\newif\ifshowimage
\showimagetrue

\makeatletter
% we add a boolean to this macro defined in onimage.sty
\def\tikzonimage@opt[#1]#2{%
    \begin{tikzpicture}
        \ifshowimage
          \node[inner sep=0] (image)
                  {\includegraphics[#1]{#2}};
        \else
          \node[inner sep=0] (image)
                  {\phantom{\includegraphics[#1]{#2}}};
        \fi
        \begin{tikzonnode}{image}%
}
\makeatother

\newif\ifarabiclabels
\arabiclabelstrue
%\arabiclabelsfalse
\newcommand\lab[2]{%
  \ifarabiclabels{\selectlanguage{arabic}#1}\else#2\fi}

\pagestyle{empty}
\begin{document}

\tikzset{every path/.style={line cap=round}}
```

Figure 24: The source code of the reconstruction using Ti*k*Z (1/2).

```
\newcommand\yA{0.345}

\begin{tikzonimage}[width=\textwidth]{saito-fig.jpg}%
                   [tsx/show help lines]
\coordinate (A) at (0.037,\yA);
\coordinate (M) at  (0.2,\yA);
\coordinate (K) at  (0.425,\yA);
\coordinate (Q) at  (0.785,0.53);
...
\coordinate (Y) at  (0.227,0.84);

\draw (A) -- (L);
\draw (B) -- (C);
\draw (X) -- (T1);
\draw[red] (H)--(Y)--(T)--(G)--(D);
\draw (H)--(E)--(Z)--(G)--(W);
\draw[dashed,green] (E)--(G);
\draw (N)--(B);
\draw (X)--(X1);
\draw (X) .. controls (R2a) and (R2b) .. (R1);
...
% two ticks
\draw ($(M)+(0.00,-0.02)$) -- ($(M)+(0.00,0.02)$);
\draw ($(K)+(0.00,-0.02)$) -- ($(K)+(0.00,0.02)$);

\node[color=blue,above] at (A)  {\Large\lab{...}{A}};
\node[color=blue] at ($(M)+(-0.05,0.02)$)
                            {\Large\lab{...}{M}};
...
\end{tikzonimage}

\end{document}
```

Figure 25: The source code of the reconstruction using Ti*k*Z (2/2).

4.2.3 Inkscape + METAPOST

Another solution, which lies between purely graphical tools and programming solutions, is to use a tool of the first kind in order to produce a sketch of the tracing, and to modify it using another tool. Inkscape, for instance, generates a SVG file, and that file could be edited directly. It could also be postprocessed.

An even better solution is to convert the SVG file to METAPOST using the following commands:

```
rsvg-convert file.svg -f pdf -o file.pdf
pstoedit -q file.pdf -f mpost file.mp
```

However, the METAPOST code will be rather low level, and labels will not be converted. Nevertheless, labels and other features can be added within METAPOST.

In order to make this useable, it is best to set only points in Inkscape. But since points are actually circles, they produce complex curves in METAPOST. Even simpler is to draw small ticks in Inkscape, and this will result in small strokes (and not dots) in METAPOST. The second coordinates can then be manually (or automatically) discarded. This is what we did in figures 26–28.

With this hybrid technique, we could for instance also add irregular hand traced dashed lines and still benefit from all of METAPOST's features.

Incidentally, Inkscape also has the capability to output both a SVG and a LaTeX file, so that the figure included in a document will have fonts matching those of the document [7].

4.2.4 Other LaTeX environments

There are several other LaTeX environments or packages which could be used for our task. The simplest of these environments is certainly the standard `picture` environment. It will however be far from convenient, as tracing curves is not that easy. Bezier curves can be drawn, but the ease of drawing curves with METAPOST will be seriously missed.

Another possibility is the `pstricks` package with which tracing a figure should be very similar to the way it was done with Ti*k*Z.

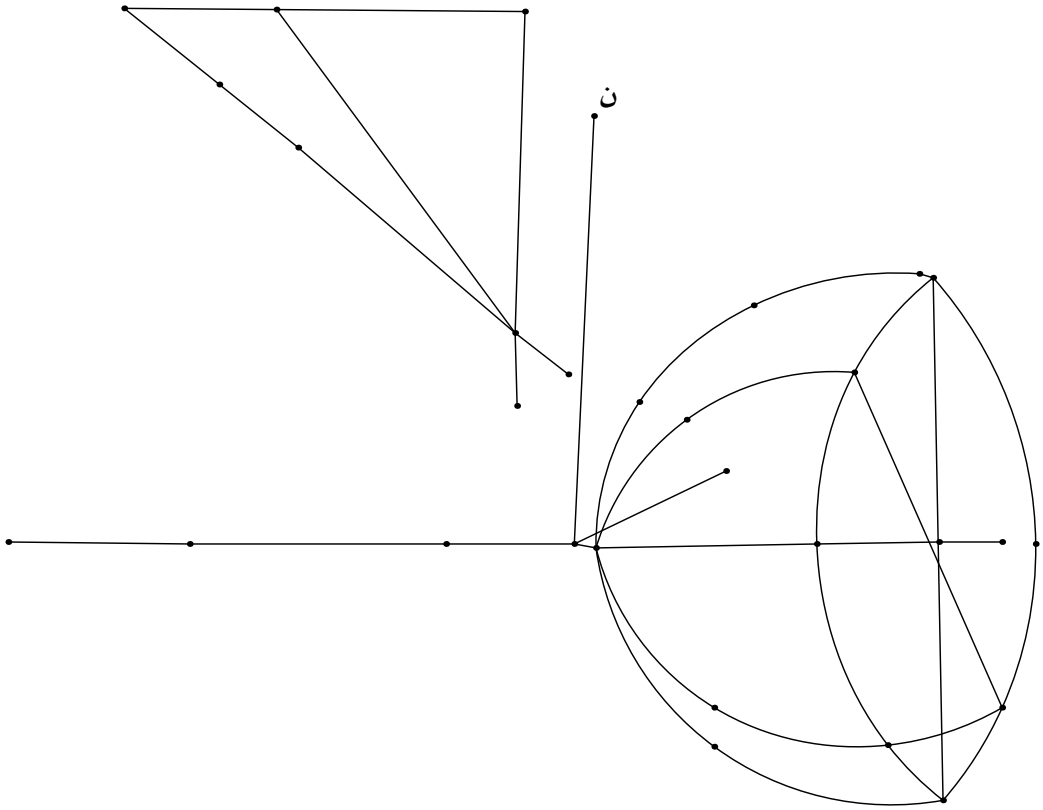


Figure 26: The reconstruction of DRaFT's example by setting a number of points in Inkscape, but by doing all further work in METAPOST. Here, we made use of only one arabic label.

```
verbatimtex
%&latex
\documentclass{article}
\usepackage[LAE,LFE,T1]{fontenc}
\usepackage[utf8]{inputenc}
\usepackage[arabic,english]{babel}
  \begin{document}
etex

numeric u;
u=1cm;

beginfig(1);
  % coordinates obtained from Inkscape
  % the mp output was slightly edited,
  % removing the second coordinates
  % and introducing a variable
  z0=(244.457001,719.875000);
  z1=(371.734009,718.867004);
  z2=(215.164001,681.492004);
  ...
  z25=(557.604980,343.089996);
  z26=(468.710999,362.281006);
  z27=(474.773010,483.500000);
  % this was forgotten, and we recompute it here:
  z28=whatever[z0,z1]=whatever[z2,z3];
  z29=z26+(0,-20);

  % scaling everything down:
  for i=0 upto 29:
    x[i]:=x[i]/1.5;y[i]:=y[i]/1.5;
  endfor;
```

Figure 27: METAPOST code for the reconstruction of DRaFT's example. Only the coordinates of the points were set with Inkscape. (begin)

```
  draw z7--z8--z9--z10--z11--z12--z13--z14;
  draw z6--z4--z1--z0--z28--z2--z3--z4--z5;
  draw z10--z27;
  draw z10--z15;
  draw z0--z4;
  draw z11..z16..z17..z18;
  draw z11..z29..z22;
  draw z11..z26..z25..z24;
  draw z20--z24;
  draw z21..z23..z24..z22;
  draw z11..z19..z20;
  draw z21..z20..z12..z25..z22;
  draw z21--z22;
  draw z18--z21;

  % labels
  pickup pencircle scaled 2pt;
  for i=0 upto 29:
    draw z[i];
    %label.top(decimal i,z[i]) withcolor red;
  endfor;
  pickup pencircle scaled .5;

  label.top(btex {\selectlanguage{arabic}..} etex,z15);

endfig;
end
```

Figure 28: METAPOST code for the reconstruction of DRaFT's example. Only the coordinates of the points were set with Inkscape. (end)

4.3 Summary

We have mainly examined two approaches for tracing an original figure, one using a mostly graphical approach, in which the mouse is used to reconstruct paths and position labels, and another one in which everything is programmed.

Both approaches have advantages and drawbacks. We found it time-consuming to use Inkscape to draw paths, especially because we always only saw the end result, and that the way to the result had vanished. There is no visible trace of which button and which menu option was used, and this can be confusing. It also means that different versions of the interface can force us to revise our habits, and of course that it is difficult to tell someone else how something was achieved.

In terms of speed, using an interface is certainly the fastest way to trace a figure, but it is not the most flexible one. With an interface, it is easy to forget to have some lines meet, especially if one line end meets another one in the middle (as in figure 5), when it is supposed to not go beyond it. This explains why some figures produced with such tools have defects. An interface does not encourage perfect drawings, and may even make it difficult.

But we have also found it very time-consuming to position points with Ti*k*Z, and to link these points. Most awkward was the tracing of curves, because we had to position control points in the void, and to adapt them. A much more practical approach is that of METAPOST, in which we can obtain a good curve going through a number of points, and in fact there have been efforts to import this capability in Ti*k*Z.

Although this is useable, and we use it, an even better approach would be a way to position a number of points graphically, possibly in a relative way, to *name* them, to connect them, and then work out the remaining part in a purely textual way (using the names given to the points), in order to have all the benefit of automation, of consistency, and so on. None of the tools described in this survey provide *all* of these functionalities. Many, if not all, of these features could be made available through a purely graphical interface, but it would entail creeping featurism, which eventually would deter new users. On the other hand, creeping featurism in a programming language does not deter new users, for the very reason that new users do not see the new features. You only see what you use!

In any case, whatever the tools that one uses, there are two important priorities to have in mind when tracing a figure or producing an entirely new one:

- first, it is important to stick to vector graphics as much as possi-

ble, and to keep lines smooth; a figure produced by a tool such as Inkscape, DRaFT, METAPOST, or another one, should never be scanned, but directly included as vector graphics, for instance as a PDF file (but PDF files can also contain bitmaps, so that one should be careful); this also applies to figures which are part of web pages and such figures should, when possible, use a vector format such as SVG;

- second, complex drawings should be parameterized, so that a figure can easily be scaled, and the features of its lines, curves, points, or the colors, etc., easily changed.

In addition, the user should have a fine control over labels, and be able to compute accurately the positions of some points, if there is such a need.

Of course, any repetitive work should not be done with a graphical interface. For instance, if we want to change the thickness of a number of lines, or change globally the size of labels, or their colors, etc., a program will do much better.

5 Languages

Technical or mathematical drawings usually contain labels. The figures shown earlier have already illustrated the use of Latin and Arabic labels, but there may be a need to enter labels in other languages, or to add more complex annotations, such as mathematical expressions.

How this can be done, and even if this can be done, depends on the tool used to produce a drawing. Whatever tool or environment is used, the characters are typed using a certain encoding. This can be ASCII, ISO-Latin 1, UTF-8, or one of many other encodings. Such encodings are called "input encodings" and, in certain cases, they may be changed within a same text. Fonts, on the other hand, may use the same or a different encoding as the input encoding. It is for instance possible to type Arabic using the ASCII encoding, and still obtain a genuine Arabic output. However, things are much simpler when one uses an encoding capable of encoding all major scripts, and at the same time, fonts using this same encoding. This is the case with UTF-8.

5.1 Interfaces

With Inkscape, labels can be entered in UTF8, either directly, or by copy-pasting from another source, such as an editor or a web page. Various fonts are available through Inkscape. Inkscape can also include LaTeX formulæ, although we haven't used this capability.

DRaFT can also use fonts in the same way, but cannot use LaTeX mathematical expressions.

5.2 LaTeX

With LaTeX, there are a number of possibilities, depending on how text is encoded and which version of TeX and which graphical environment one is using. For instance, some fonts such as OpenType fonts are only available with XeLaTeX or LuaLaTeX. There are many available OpenType fonts, and a good way to find out which ones are installed on your computer is to check the fonts available in LibreOffice.

5.2.1 Versions of TeX

The multiple versions of TeX may be confusing for newcomers, and even for longtime users who have clung to their old habits. TeX is the fundamental program and LaTeX is a large superset of TeX. LaTeX uses TeX. pdfTeX

is an extension of TEX outputting PDF files. XƎTEX [20, 21, 9] is a more recent extension of TEX, in which OpenType fonts can be used, and which is natively based on the UTF-8 encoding. And LuaTEX [14, 18] is an even more recent version of TEX, using Lua as a scripting language.

XƎLATEX is the LATEX version of XƎTEX. LuaLATEX is the LATEX version of LuaTEX.

For end users, XƎTEX and LuaTEX offer similar functionalities. XƎTEX was the first TEX extension to provide support for OpenType fonts. However, most of its features are included in LuaTEX, so that the latter should be considered as the real successor of pdfTEX, and XƎTEX is probably bound to die. This is somewhat ironic, as only a very small fraction of LATEX users have ever used XƎTEX.

Although standard LATEX cannot use OpenType fonts directly, if XƎLATEX is used to produce a figure using OpenType fonts, that figure can be exported as a PDF, and then included in a standard LATEX file, so that in a way, OpenType fonts are not totally forbidden in LATEX.

For an introduction to multilingual writing with XƎLATEX and LuaLATEX, see the documentation of the `polyglossia` package [2].

5.2.2 Graphical environments and labels

Since Ti*k*Z is a LATEX package, it can use OpenType fonts if XƎLATEX or LuaLATEX are used.

METAPOST is more problematic, since it is currently tied to PostScript. At the moment, standalone METAPOST, that is METAPOST not called from TEX, can not include OpenType fonts, but in many cases labels can be produced with other fonts. It is possible that future extensions of standalone METAPOST will have the ability to use OpenType fonts.

However, if METAPOST is used through LuaLATEX, then OpenType fonts can be used. An example of the use of Chinese OpenType labels in METAPOST through LuaLATEX is given in figure 29, but it requires a fairly recent version of LuaLATEX.

As an illustration, we give below a number of examples of figure labels in various languages, both with standard LaTeX (and METAPOST) and with XƎLATEX (and Ti*k*Z). In LuaLATEX, it is now possible to use Ti*k*Z with or without OpenType fonts, METAPOST as an external program with standard fonts, and METAPOST included with `luamplib` package, with or without OpenType fonts.

Note that you should have a fairly recent and complete installation of TEX and its derivatives to get all this working. A basic installation of TEX, or one which is five years old, will not do.

```
\documentclass{article}
\usepackage{fontspec}
\usepackage{luamplib}

\newfontfamily\cjkfont[Script=CJK]{AR PL UKai CN}

\begin{document}

\begin{mplibcode}
numeric u;
u=1cm;
beginfig(1);
  pair A,B,C;
  A=origin;
  B-A=(5u,0);
  C=(fullcircle scaled 8u shifted A) intersectionpoint
    (fullcircle scaled 6u shifted B);
  draw A--B--C--cycle;
  label.bot(btex \cjkfont{...} $=\sqrt{3^2+4^2}=5$ etex
                    rotated angle(B-A),.5[A,B]);

endfig;
\end{mplibcode}

\end{document}
```

Figure 29: Example of inclusion of METAPOST in a LuaLaTeX file.

5.3 XƎLaTeX code examples

```
\documentclass{article}
\usepackage{tikz}
\usetikzlibrary{intersections}

\usepackage{fontspec}
\setmainfont{Linux Libertine O}
\setsansfont{Arial}
\newfontfamily\greekfont[Script=Greek]{Linux Libertine O}
\newfontfamily\greekfontsf[Script=Greek]{Linux Libertine O}

\usepackage{polyglossia}
\setdefaultlanguage{greek}

\begin{document}
\pagestyle{empty}

\def\firstcircle{(0,0) circle (4 cm)}
\def\secondcircle{(5cm,0) circle (3 cm)}

\begin{tikzpicture}
\coordinate [label=left:$A$] (A) at (0,0);
\coordinate [label=right:$B$] (B) at (5cm,0);
\path[name path=c1] \firstcircle;
\path[name path=c2] \secondcircle;
\path[name intersections={of=c1 and c2}];
\coordinate [label=above:$C$] (C) at (intersection-1);
\draw (A) -- node[below,sloped] {μήκος $c$} (B)
          -- node[above,sloped] {μήκος $a$} (C)
          -- node[above,sloped] {μήκος $b$} (A);
\end{tikzpicture}


\end{document}
```

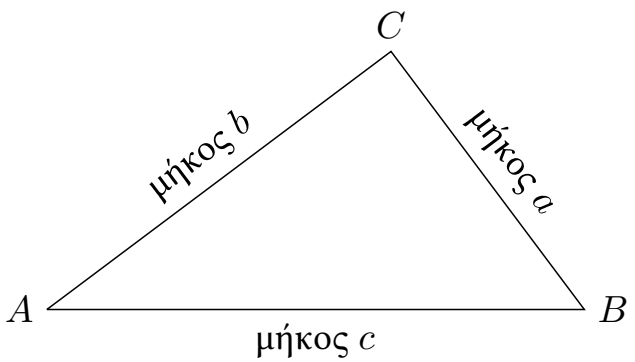


Figure 30: Greek labels in a Ti*k*Z figure with XƎLaTeX.

```
\documentclass{article}
\usepackage{tikz}
\usetikzlibrary{intersections}
\usepackage{xeCJK}
\setCJKmainfont{AR PL UKai CN}

\begin{document}
\pagestyle{empty}

\def\firstcircle{(0,0) circle (4 cm)}
\def\secondcircle{(5cm,0) circle (3 cm)}

\begin{tikzpicture}
\coordinate [label=left:$A$] (A) at (0,0);
\coordinate [label=right:$B$] (B) at (5cm,0);
\path[name path=c1] \firstcircle;
\path[name path=c2] \secondcircle;
\path[name intersections={of=c1 and c2}];
\coordinate [label=above:$C$] (C) at (intersection-1);
\draw (A) -- node[below,sloped] {长度 $c$} (B)
          -- node[above,sloped] {长度 $a$} (C)
          -- node[above,sloped] {长度 $b$} (A);
\end{tikzpicture}


\end{document}
```

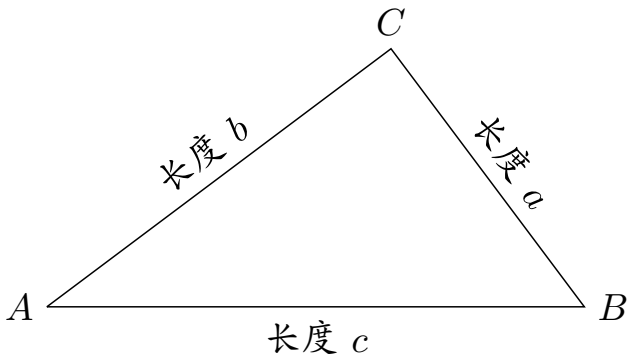


Figure 31: Chinese labels in a Ti*k*Z figure with XƎLaTeX.

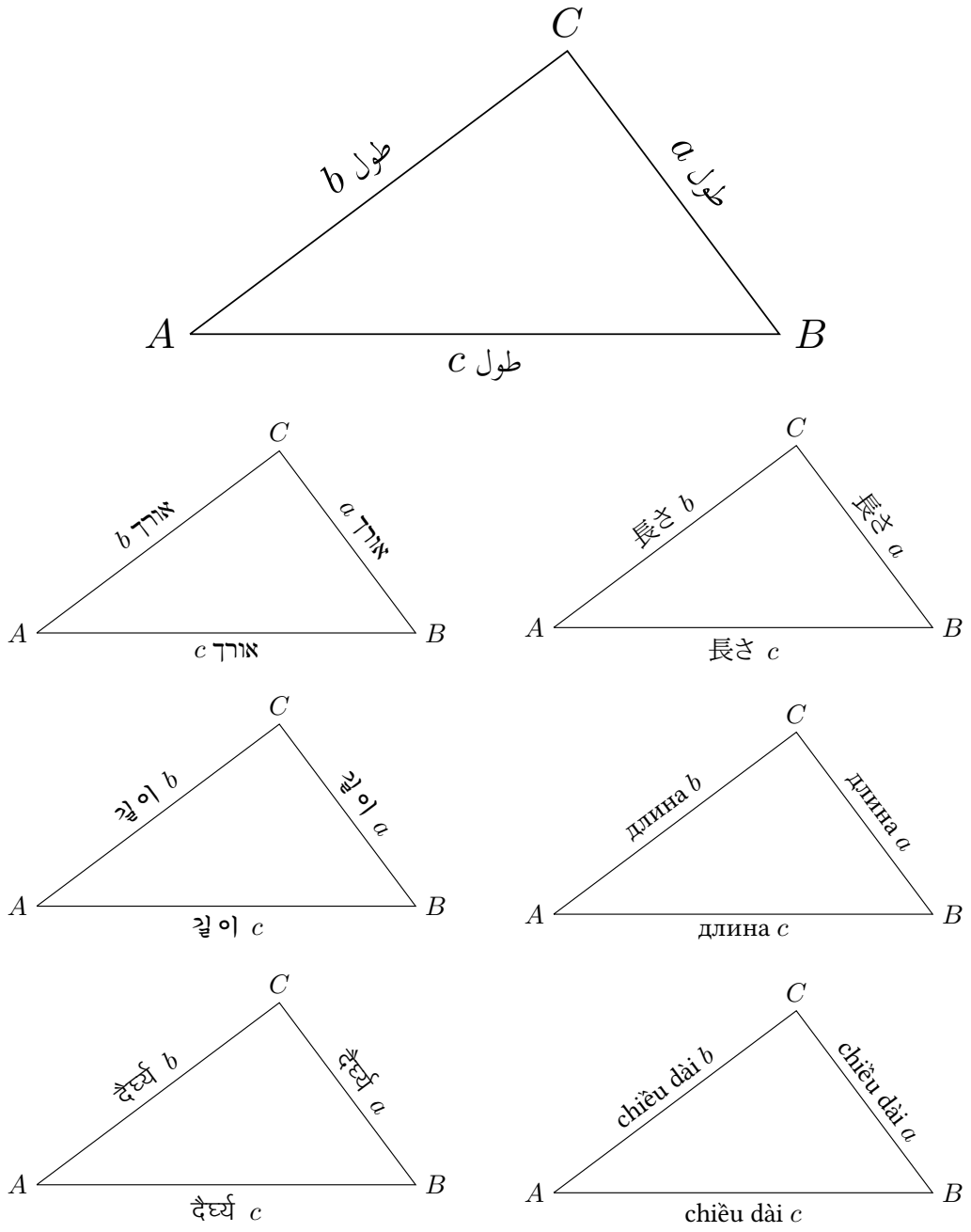


Figure 32: Ti*k*Z figures using various languages and fonts with XƎLaTeX.

5.4 METAPOST code examples

```
verbatimtex
%&latex
\documentclass{article}
\usepackage{CJKutf8}
\begin{document}
\newcommand\korean[1]{\begin{CJK*}{UTF8}{mj}#1\end{CJK*}}
etex

numeric u;
u=1cm;
beginfig(1);
  pair A,B,C;
  A=origin;
  B-A=(5u,0);
  C=(fullcircle scaled 8u shifted A) intersectionpoint
    (fullcircle scaled 6u shifted B);
  draw A--B--C--cycle;
  label.llft(btex \korean{ㄱ} etex,A);
  label.lrt(btex \korean{ㄲ} etex,B);
  label.top(btex \korean{ㄴ} etex,C);
  label.bot(btex \korean{길이} $=\sqrt{3^2+4^2}=5$ etex
                    rotated angle(B-A),.5[A,B]);
  label(btex \korean{길이} $=3$ etex rotated angle(B-C),
          .5[B,C]+2.5labeloffset*unitvector((B-C) rotated 90));
  label(btex \korean{길이} $=4$ etex rotated angle(C-A),
        .5[A,C]+2.5labeloffset*unitvector((C-A) rotated 90));


endfig;

end
```

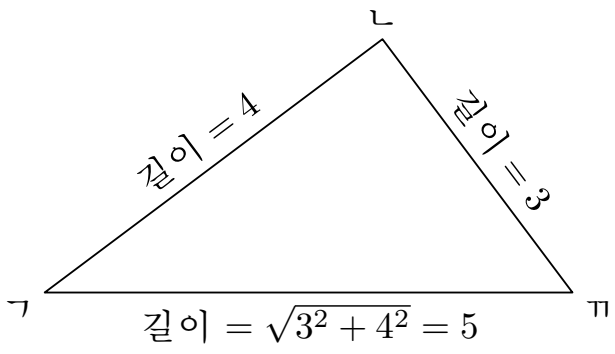


Figure 33: Korean labels in a METAPOST figure with pdfLaTeX.

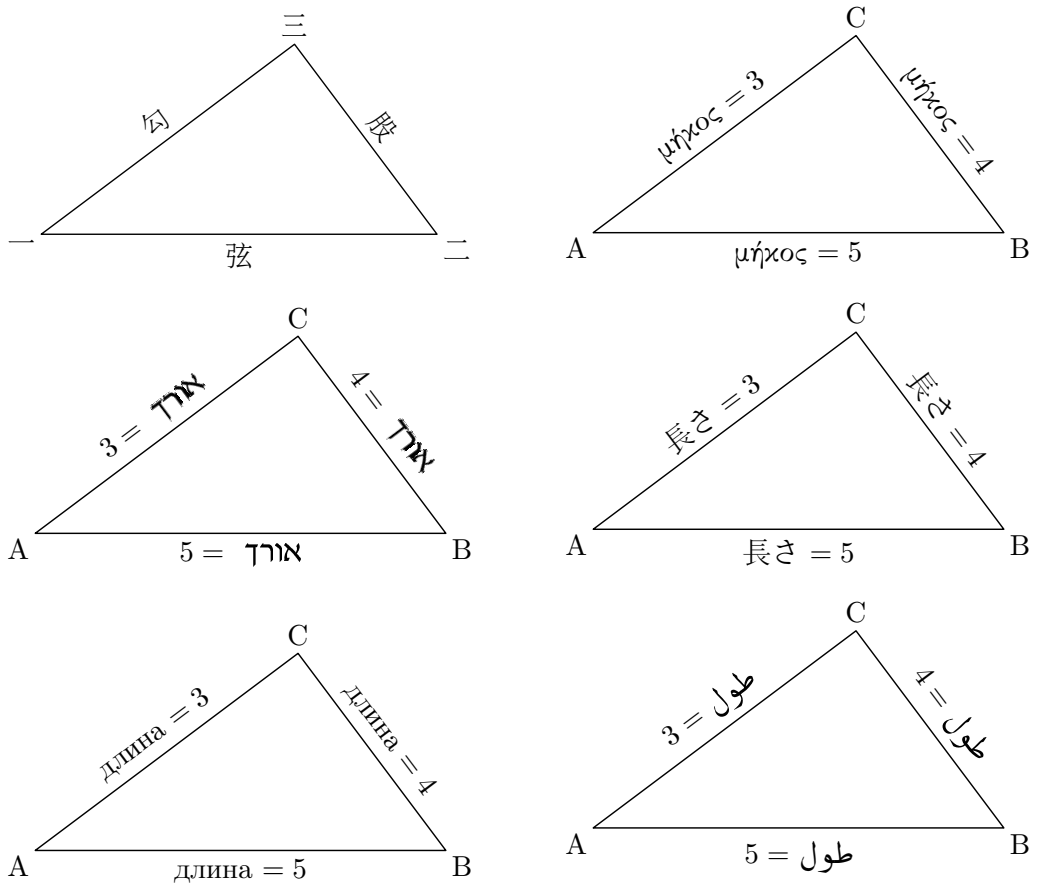


Figure 34: METAPOST figures using various languages and fonts with pdfLaTeX.

Conclusion

The examination of a number of figures drawn in the current literature has shown the existence of a number of problems and issues. Modern figures, produced using graphical interfaces, often contain subtle problems which were not intended by their authors, but crept in because the interfaces make it difficult to get the drawings right. Certain drawings, produced by tracing figures, such as ancient mathematical diagrams, may intentionally contain imperfect junctions, but these drawings nevertheless also have a more "perfect" side, for instance smooth curves, or consistency of strokes, and so on. There therefore appears to exist a real need to achieve both a certain perfection in tracing, usually by some hand tracing, and at the same time to be able to control other features at a higher level, and this can often best be obtained by programming. Several solutions have been examined, and we have argued for the extension of some tools to even better suit the needs of the user. We hope that our study will help historians to produce even better critical figures, especially with the advent of UTF8/OpenType compliant versions of TEX.

References

[1] Henri Bouasse. *Propagation de la lumière. Théorie de la réflexion vitreuse et métallique*. Paris: Delagrave, 1925.

[2] François Charette and Arthur Reutenauer. Polyglossia: An alternative to Babel for XƎLaTeX and LuaLaTeX, 2015. CTAN: macros/latex/contrib/polyglossia.

[3] Gregg De Young. Diagrams in ancient Egyptian geometry — Survey and assessment. *Historia Mathematica*, 36:321–373, 2009.

[4] Gregg De Young. Mathematical diagrams from manuscript to print: examples from the Arabic Euclidean transmission. *Synthese*, 186:21–54, 2012.

[5] Gregg De Young, Takanori Suzuki, Ken Saito, and Ken'ichi Takahashi. Diagrams in Greek mathematical texts, 2011. `http://www.greekmath.org`.

[6] Marie-Thérèse Debarnot. *Kitāb Maqālīd 'ilm al-Hay'a : la trigonométrie sphérique chez les Arabes de l'Est à la fin du Xe siècle*. Damas: Institut français de Damas, 1985.

[7] Johan B. C. Engelen. How to include an SVG image in LaTeX, 2013. CTAN:`info/svg-inkscape`.

[8] Jöran Friberg. *Amazing traces of a Babylonian origin in Greek mathematics*. Singapore: World Scientific Publishing Co., 2007.

[9] Michel Goossens. The XƎTeX Companion: TeX meets OpenType and Unicode, 2010.

[10] Michel Goossens, Frank Mittelbach, Sebastian Rahtz, Denis Roegel, and Herbert Voß. *The LaTeX Graphics Companion*. Boston: Addison-Wesley, 2007.

[11] Ebenezer Henderson. *Life of James Ferguson, F.R.S., etc.* Edinburgh: A. Fullarton & Co., 1867. ["facsimile" reprint by Cambridge University Press in 2010].

[12] John D. Hobby. METAPOST, a user's manual, 2013. `www.tug.org/docs/metapost/mpman.pdf`.

[13] Guy Le Meur. Le rôle des diagrammes dans quelques traités de la « Petite astronomie ». *Revue d'Histoire des Mathématiques*, 18(2):157–221, 2012.

[14] LuaTEX Reference Manual, 2015. `www.luatex.org`.

[15] Michela Malpangotto. Graphical choices and geometrical thought in the transmission of Theodosius' *Spherics* from Antiquity to the Renaissance. *Archive for the History of Exact Sciences*, 64:75–112, 2010.

[16] Paolo Mascellani, Pier Daniele Napolitani, and Veronica Gavagna. M2.5.2 — Minutes of the workshop on "The problem of diagrams and drawings criticism in mathematical texts", 2005.

[17] Marco Panza. The twofold role of diagrams in Euclid's plane geometry. *Synthese*, 186(1):55–102, 2012.

[18] Manuel Pégourié-Gonnard. A guide to LuaLATEX, 2013.

[19] Roshdi Rashed, editor. *Histoire des sciences arabes*, volume 2. Paris: Éditions du seuil, 1997.

[20] Will Robertson and Khaled Hosny. The XƎTEX reference guide. May 21, 2013.

[21] Will Robertson and Khaled Hosny. The `fontspec` package: Font selection for XƎLATEX and LuaLATEX, 2015. CTAN: macros/latex/contrib/fontspec.

[22] Denis Roegel. A complex drawing in descriptive geometry. *TUGboat*, 28(2):218–228, 2007.

[23] Denis Roegel. An early (1844) key-driven adding machine. *IEEE Annals of the History of Computing*, 30(1):59–65, 2008.

[24] Denis Roegel. Spheres, great circles and parallels. *TUGboat*, 30(1):80–87, 2009.

[25] Denis Roegel. Sustainable digitization, 2012. `http://locomat.loria.fr`.

[26] Ken Saito. A preliminary study in the critical assessment of diagrams in Greek mathematical works. *SCIAMVS*, 7:81–144, 2006.

[27] Ken Saito. DRaFT reference manual (for DRaFT release 1.1b25), 2012. `http://www.greekmath.org`.

[28] Ken Saito. Traditions of the diagram, tradition of the text: A case study. *Synthese*, 186:7–20, 2012.

[29] Ken Saito and Gregg De Young. DRaFT tutorial, 2013. `http://www.greekmath.org`.

[30] Ken Saito and Nathan Sidoli. Diagrams and arguments in ancient Greek mathematics: lessons drawn from comparisons of the manuscript diagrams with those in modern critical editions. In Karine Chemla, editor, *The history of mathematical proof in ancient traditions*, pages 135–162. Cambridge: Cambridge University Press, 2012.

[31] Nathan Sidoli. What we can learn from a diagram: The case of Aristarchus's *On the sizes and distances of the Sun and Moon*. *Annals of Science*, 64(4):525–545, 2007.

[32] Nathan Sidoli and Takanori Kusuba. Naṣīr al-Dīn al-Ṭūsī's revision of Theodosius's *Spherics*. *Suhayl*, 8:9–46, 2008.

[33] Nathan Sidoli and Takanori Kusuba. Al-Harawī's version of Menelaus' *Spherics*. *Suhayl*, 13:149–212, 2014.

[34] Till Tantau. The Ti*k*Z and PGF Packages — Manual for version 3.0.0-cvs, 2014. `http://pgf.sourceforge.net`.