Schriften des Instituts für Dokumentologie und Editorik — Band 2

Kodikologie und Paläographie im digitalen Zeitalter

Codicology and Palaeography in the Digital Age

herausgegeben von | edited by

Malte Rehbein, Patrick Sahle, Torsten Schaßan

unter Mitarbeit von | in collaboration with

Bernhard Assmann, Franz Fischer, Christiane Fritze

2009

BoD, Norderstedt

Bibliografische Information der Deutschen Nationalbibliothek:
Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über **http://dnb.d-nb.de/** abrufbar.

Leicht veränderte Fassung für die digitale Publikation (siehe Vorwort).

Slightly modified version to be published digitally (see preface).

Publication réalisée avec le soutien d'Apices
Association Paléographique Internationale
Culture – Écriture – Société
Dotation J.M.M. Hermans.
http://www.palaeographia.org/apices/

Herstellung und Verlag: Books on Demand GmbH, Norderstedt
ISBN: 978-3-8370-9842-6
Einbandgestaltung: Katharina Weber
Satz: XƎTEX und Bernhard Assmann

Linking Text and Image with SVG

Hugh A. Cayless

Abstract

Annotation and linking (or referring) have been described as "scholarly primitives", basic methods used in scholarly research and publication of all kinds. The online publication of manuscript images is one basic use case where the need for linking and annotation is very clear. High resolution images are of great use to scholars and transcriptions of texts provide for search and browsing, so the ideal method for the digital publication of manuscript works is the presentation of page images plus a transcription of the text therein. This has become a standard method, but leaves open the questions of how deeply the linkages can be done and how best to handle the annotation of sections of the image. This paper presents a new method (named img2xml) for connecting text and image using an XML-based tracing of the text on the page image. The tracing method was developed as part of a series of experiments in text and image linking beginning in the summer of 2008 and will continue under a grant funded by the National Endowment for the Humanities. It employs Scalable Vector Graphics (SVG) to represent the text in an image of a manuscript page in a referenceable form and enables linking and annotation of the page image in a variety of ways. The paper goes on to discuss the scholarly requirements for tools that will be developed around the tracing method, and explores some of the issues raised by the img2xml method.

Zusammenfassung

Annotation und Referenz sind als geisteswissenschaftliche Stammfunktionen beschrieben worden, die in Forschung und Veröffentlichungen aller Arten verwendet werden. Die Online-Veröffentlichung von Handschriftenbildern ist ein grundlegender Anwendungsfall, in dem der Bedarf für Referenz und Annotation offensichtlich ist. Hochauflösende Bilder sind von großem Nutzen für die Forscher und Transkriptionen ihre Grundlage für Suchen und Stöbern. Damit erweist sich eine Darstellungsweise, in der das Bild der Seite mit seinem Text verknüpft ist, als ideale Form der Publikation von Handschriften. Diese Verknüpfung ist eine Standardmethode geworden; sie lässt jedoch die Frage offen, bis zu welcher Granularität sie erfolgen soll und wie Annotationen von Bildsegmenten erreicht werden können. Der Beitrag stellt nun eine neue Methode (img2xml) vor, mit der Text und Bild XML-basiert verknüpft werden. Diese Methode der »Spurenverfolgung« ist seit Sommer 2008 als Teil einer Reihe von Experimenten

Kodikologie und Paläographie im Digitalen Zeitalter – Codicology and Palaeography in the Digital Age. Hrsg. Malte Rehbein, Patrick Sahle und Torsten Schaßan, unter Mitarbeit von Bernhard Assmann, Franz Fischer und Christiane Fritze. Schriften des Instituts für Dokumentologie und Editorik 2. Norderstedt: Books on Demand, 2009. 145–158.

zur Text-Bild-Verknüpfung entwickelt worden und soll mit Unterstützung des National Endowment for the Humanities fortgesetzt werden. Die Methode verwendet Scalable Vector Graphics (SVG), um den Text auf dem Bild einer Handschriftenseite in einer referenzierbaren Form darzustellen. Sie ermöglicht die Verknüpfung und Annotation des Seitenbildes auf verschiedene Weise. Der Beitrag diskutiert schließlich die wissenschaftlichen Anforderungen an Werkzeuge, die auf Basis dieser Methode entwickelt werden können und untersucht einige der durch die img2xml-Methode aufgeworfenen Fragen.

1 Background

The impetus for the research outlined here comes from my experience working with the online presentation of manuscript texts, and a sense of frustration with the limitations of tools for linking the two. An example of a basic approach to the presentation of manuscript texts may be found in the "First Century of the First State University" (UNC) collection at Documenting the American South (DocSouth). Here, the primary face of a document is its HTML transcription, with links to the XML source, and to medium-quality images of the pages. For any given document, it is possible to see what the original, handwritten texts look like, but the two are not easily viewable side-by-side, and annotations are attached only to the marked-up versions of the text, not the images.

This situation flows naturally from the workflows involved in creating the transcriptions. A scholar will be responsible for transcribing the text, which will then be sent out for encoding in a TEI XML format. After that, a graduate assistant (GA), working with the scholar, will perform quality control and the insertion of notes and links to external materials (including the page images) into the XML text. Finally, the XML is transformed into the HTML version. It is certainly possible to encode links that may be actuated from the images, using image maps for example, but doing so requires the insertion of another step into the workflow and one that is not presently automatable. Given the necessary tradeoffs in time and money involved in this kind of work, such a step is not feasible for most DocSouth projects.

Because we were planning to embark on a new project that would involve the online presentation of a 19th century text, and because I had also been working with transcriptions of papyri destined for presentation alongside images, I began an investigation of what tools were available, and what might be done to automate, and therefore reduce the costs of linking text and image together.

The state of the art for linking manuscript images and texts (both transcriptions and annotations) is to treat the image as a coordinate system and for XML markup in the text to describe a rectangle on the image, containing (for example) a line of text. There are a number of existing tools that provide for user-controlled image annotation. This is typically accomplished by providing the user with drawing tools with which they may draw shape overlays on the image. These overlays can in turn be linked to text annotations entered by the user. This is the way image annotation works on Flickr, for example, and also the Image Markup Tool (IMT). The TEI P5 facsimile markup (TEI P5, chapter 11) conceives of text-image linking in this fashion also. Since these methods require a person to manually create the areas and the linkages between text and image, I began to experiment with methods for producing a representation of the text on a page image that would allow any discrete chunk of text to be referenced (Cayless 2008). This is accomplished via the use of a tool called *potrace* (Selinger) which operates by first converting the source image to a black and white version and then converts the raster image to a vector representation. This vector representation can be output in a variety of formats, including Scalable Vector Graphics (SVG), which is an XML-based format.

The resulting SVG file contains <path> elements, which describe structures in the source image, one path per structure, as cubic Bézier curves. By "structure", I mean a contiguous segment of writing on the page, which might encompass a single letter, a series of connected letters, or other artifact. The fact that each such structure is represented by an XML element means that each one can be referenced and manipulated using standard tools. If the transcription and/or notes are in an XML format also, the image may point at the text. The SVG image, since it is vector-based, is arbitrarily zoomable, but can be based on a coordinate system that will always map back to the source image. The SVG can be overlaid on the source image, parts can be made visible or invisible, transparent, of different colors, rotated, etc. An additional benefit of producing a vector representation of the page image is that it becomes quite easy to detect some aspects of page structure, like lines.

The proof-of-concept work on this method was successful, and a fully Open Source toolchain (provisionally called *img2xml*) was developed that started with a page image from a letter, with an existing transcription marked up in TEI XML, and resulting in a web view where line of text were linked to lines in the image, and vice-versa. Subsequently, the Carolina Digital Library and Archives submitted a National Endowment for the Humanities (NEH) Startup Grant to begin developing the experimental version into a functional prototype. The proposal has been funded, and development of the prototype will begin in mid-2009.

1.1 Methods

The methods and a summary of the code that has been developed thus far are outlined below. Since the project is at an early stage, and further development is anticipated under the recently-awarded NEH Startup Grant, the codebase is expected to evolve quickly. Updates will be posted on the project's website..

The *img2xml* process may be broken down into the following steps: pre-processing of images, generation of the SVG tracing, post-processing of the SVG, analysis of the

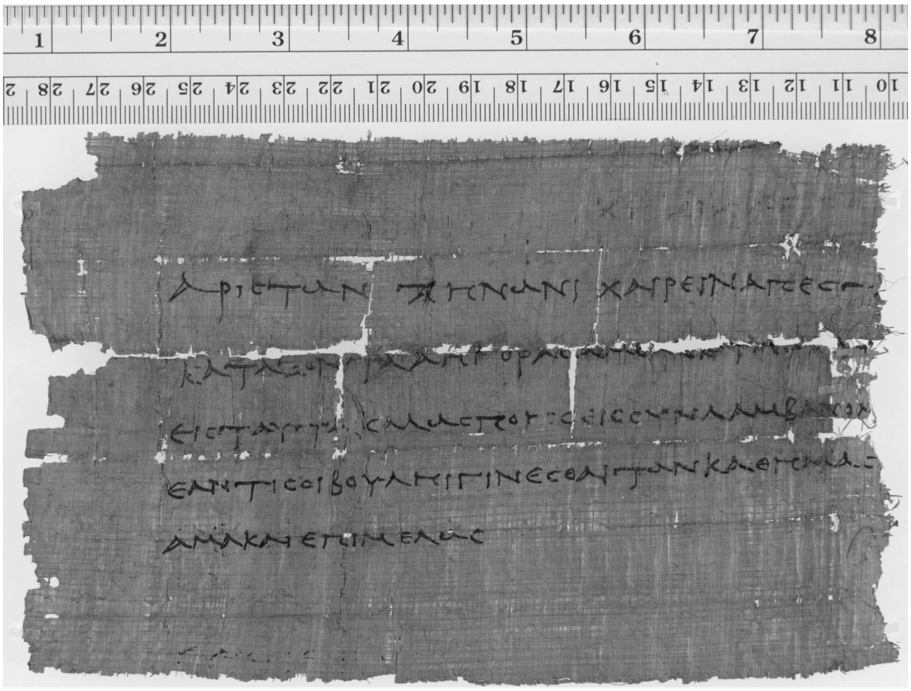


Figure 1. Fragment of a letter from Ariston to Zenon (P. Mich I 78).

SVG, and presentation. Pre-processing would involve filtering the image to whatever extent possible to improve the chances of *potrace* generating an SVG as free as possible of artifacts that are not letters. Since *potrace* relies on first flattening the image's color space to a single bit (black or white), it depends on a black or white threshold being set. During processing, a decision is made, based on the darkness of a given pixel, whether it should become black or white. With materials like papyrus, where the support itself is quite dark, setting this threshold on an unprocessed image (see Figure 1) can involve a good deal of trial and error. But since the support and the ink are different colors, a tool like *ImageMagick* can be used to delete the support from the image, leaving the ink behind (see Figure 2). This technique is also useful for images where there are blemishes (as long as they differ in color from the ink itself).

Once the tracing program and any post-tracing cleanup (including the conversion of relative paths to absolute, and the addition of id attributes to the path elements) have run, the SVG file is ready for analysis. The initial proof-of-concept system exper-

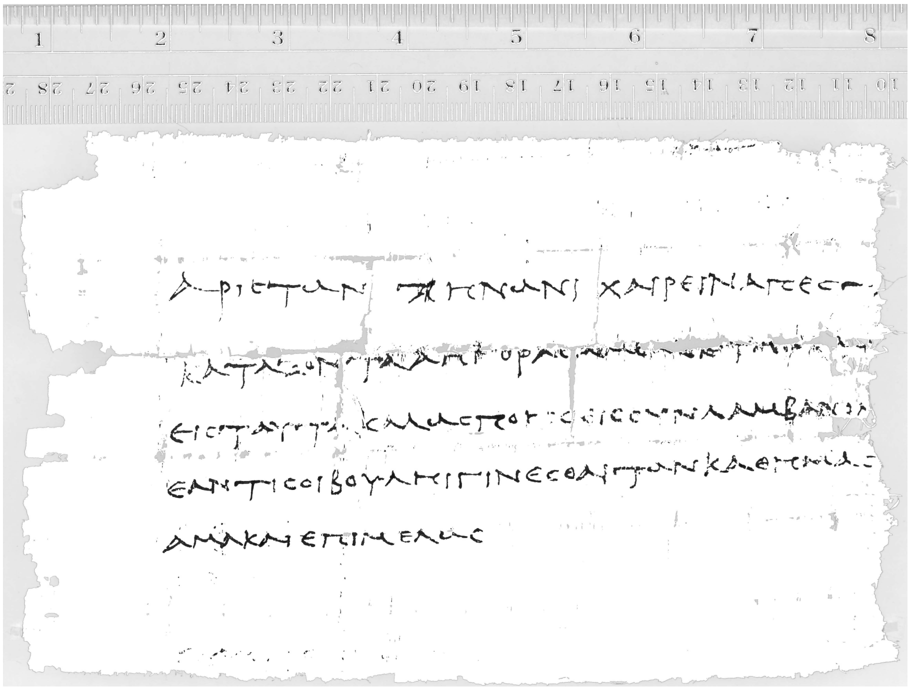


Figure 2. Fig. 1 processed to remove the color of the papyrus.

imented with very simple structure detection. Lines are detected by parsing the SVG file and loading the paths into Python data structures containing the points denoted in the SVG paths. Then, for each polygon, a bounding rectangle is determined by finding the outermost top, bottom, left, and right coordinates of the polygon. Since the original shapes are not polygons, but Bézier curves, there exists the possibility that the control points (which determine the shape of a curve, but do not lie upon that curve) will result in a rectangle that is too large or too small, but thus far in practice this does not appear to be a significant issue. As the rectangles are being extracted, each one is appended to an array, and the area of each rectangle is also appended to an array. The rectangles areas constitute a rough relative measure of size, and this measure is then used to prune objects that are unusually small or large. This helps remove artifacts such as dust spots that should be ignored as components of the document's structure. It can also be used to dispose of the outlines of pages that have been scanned against a black background.

```
def get_lines(rectangles, result, overlap):
  rect = rectangles[0]
  remainder = []
  group = Group("g%s" % rect.id)
  group.add(rect)
  for r in rectangles[1:]:
    if group.boundingrect.verticaloverlap(r) > overlap:
      group.add(r)
    else:
      remainder.append(r)
  result.append(group)
  if (len(remainder) > 0) & (len(remainder) < len(rectangles)):
    get_lines(remainder, result)
```

Figure 3. Line extraction code in Python.

Removing structures with areas smaller than two standard deviations below the mean seems to produce useful results.

After filtering, the rectangles are sorted on first the x then the y axis, which improves the efficiency of the analyses to follow. The program proceeds with line detection. During this process, shapes are added to group objects, each of which contains an array of shapes and a bounding rectangle. At the end of the analysis, these will be serialized into the output SVG as <g> elements.

The algorithm for sorting into lines is shown in Figure 3. The method starts with the rectangle at the beginning of the sorted array (which will be the topmost, leftmost shape), adds it to a group object, and then cycles through the rectangles array, looking for rectangles that overlap more than a defined percentage (50% is a good number) with the group. As each new rectangle is added, the size of the group's bounding rectangle grows. If the next rectangle in the array does not overlap, it is added to a remainder array. As long as the remainder still has shapes in it, the method is called recursively on the remainder. At the end of the process, all rectangles will have been organized into line groups.

Word detection is a process with which the project has only begun to experiment. In texts which exhibit significant space between words this should be possible using processes similar to those used in Optical Character Recognition (OCR) programs. In texts like the papyrus example from Figure 1, however, this will not work because the text is written in *scriptio continua.* If a transcription already exists, then that could be used to help align the shapes traced from the image with itself. This raises the question of how *img2xml* fits into the workflow of digitization to online production, and how it might relate to manuscript OCR.

Manuscript OCR is a problem without a satisfactory solution to date, though the technology is improving all the time (see, for example Gilbert and Roland Tomasi's article in this volume). The main difficulty in manuscript OCR is the lack of consistency in handwritten versus printed text. Progress can be made on relatively uniform hands with a significant amount of work in training the recognition algorithms, but the cost/benefit ratio is a problem, and this is compounded when one is faced with a situation like Greek papyri, where there are thousands of short texts, in thousands of hands. The process used by *img2xml* does share some workflow steps with OCR, which does the same sort of image downsampling prior to isolating and attempting to identify symbols and structure on the page. But *img2xml* is envisioned as complementary to OCR, and will be useful even when (or if) fully-functional manuscript OCR is a reality. *Img2xml* at its core is a tool that produces a vector representation of text, which can then be manipulated and linked. It does not necessarily attempt to aid in the production of a transcription of the text it represents, though a scholar producing such a transcription could use the SVG tracing as a reference. The tool's main payoff is likely to be on the publication and presentation side.

1.2 Further Work

The award of the proposed NEH Startup grant was announced on March 9th, 2009. This award will enable the project to proceed at an accelerated rate. It will focus on developing a system to present the diary of a 19th-century UNC Chapel Hill student, James Dusenbery. This manuscript has already been digitized and the transcription encoded according to the TEI guidelines.

A graduate assistant will be funded as part of the grant budget who will be responsible both for enhancing the previously-encoded Dusenbery Diary by adding tagging to denote the lines in the transcription and by converting the text to TEI P5, and for testing and evaluating the tools developed for viewing and working with the images and text. In addition, the grant will help fund some programmer time to further develop the tools we have begun working on. We plan to complete the following steps as part of the project[1]:

1. Potrace needs to be tested on a wider variety of manuscript images in order to determine the best practices for using it with these types of image.
2. We will develop methods for preprocessing images and detecting the ideal white/black cutoff to produce an optimal SVG tracing. Or, failing that, develop a web interface whereby users can run traces via a web interface and choose the best one.
3. The automated line detection routine developed for the proof-of-concept tool

[1] This list is adapted from the proposed workplan for the NEH grant.

works well for relatively horizontal, left-to-right writing, but the algorithm will need further development in order to handle text that runs (for example) at an angle. Various examples of nonstandard texts will be selected from our collection and the tool will be tested against these to determine its effectiveness and what additional development may be needed.
4. The automated detection of word boundaries will be evaluated as well, again by testing the analysis tool against a variety of images. Further development will be undertaken on the use of the transcription as a means for detecting word boundaries where they do not exist on the page.
5. OpenLayers was patched during the proof-of-concept work to allow the representation of complex paths and their serialization as SVG. This patch requires substantial improvement in order to make better use of OpenLayers' zooming capabilities. The work done to date will be refined and tested. Upon completion, the patch will be submitted to the OpenLayers developers for possible inclusion in the library.
6. Work will be done on the web interface to allow for simultaneous paging through page images and transcriptions/annotations.
7. A graduate assistant will work on the updates to the previously digitized Dusenbery text detailed above.
8. The project will be published as a collection within the Documenting the American South program.
9. All of the code developed for the project will be published in an open code repository along with documentation of the tools and a user manual. We will present the results (as well as ongoing work) at one or more of the major Digital Humanities and/or Digital Library conferences.

The interface that is developed will demonstrate features like links from notes to text in the image, the ability to link or pan/zoom to any part of the text in the image from the transcription, highlighting text search results on the surface of the image, and the marking of editorial emendations, such as expanded abbreviations and other types of editorial addition or deletion on the image itself.

2 Scholarly Requirements for Operating on Text and Image

In designing the presentation system for the Dusenbery Diary, I have made several assumptions about the functionality that should be present in the outcome, which I hope will serve as a prototype for other online manuscript publications. Some of these assumptions about desirable requirements for a system to link text and image which underlay the efforts to date are that:

1. the software should enable bidirectional linking (that is, both text to image and image to text).

2. it should be possible to link at the level of any structure on the page image.
3. images should be as manipulable as possible (i.e. panning, zooming, rotation).
4. the results should be presented in an online (browser-based) form.
5. any systems or methods constructed should be based on Open Source tools and released under an open license.

Assumptions like these should always be examined. That linking both from text to image and from image to text is a desideratum seems obvious. In an interface that presents both, either one should be able to become the primary focus of the reader's attention, but it should always be possible to call up the other and smoothly move between them.

The question of what level to link at is more complicated. A straightforward answer is that linking should be as granular as possible, but it is also true that the line is a basic unit of reference (like the page) and that line-level linking is a basic requirement. Word- and letter-level linking are also important desiderata. A prerequisite for either, however, is the detection of all written symbols on the page, which can be then grouped into words, lines, and so on. Of course, word detection is much easier when letters are organized into words, with spaces in between, which is not the case in most ancient texts. This detection of document structure is the problem that OCR attempts to solve, by first matching symbols to letters and then attempting to recognize words from collections of symbols. As I noted above, the experiments to date have proceeded by doing only structure detection which does not attempt to recognize symbols. An OCR process could be combined with *img2xml* in the future, which would enable superior structure detection. At this time, *img2xml* method enables linking at the level of the symbol, and at the level of any larger structures that can be detected based on the arrangement of the symbols.

Assumptions 3 and 4 are related, since the platform dictates to a large extent the capabilities of the image viewer. Fortunately, modern browsers are highly capable in this regard.[2] Provided that the page images have been digitized at a high enough resolution, there are a number of techniques that provide for panning, zooming, and rotation. The original demonstration version of the *img2xml* viewing tool employed a Javascript mapping library called OpenLayers, which supports pan-and-zoom functionality in a variety of ways. The author has successfully linked it to aDORe djatoka (Chute), an Open Source image server that uses JPEG2000 as the service format. Experimentation to date has focused entirely on Javascript-based methods and avoided reliance on proprietary technologies such as Flash. To date, this approach has not met with any insoluble obstacles.

Finally, it was the author's conviction from the beginning that all of the tooling and all of the methods developed during this project should be released under open

[2] The main difficulty being that Internet Explorer (as of IE 7.0) does not have native SVG support.

licenses. Proprietary technologies certainly have their place, but the kind of work under development is being designed for the online publication of scholarly editions. For this kind of work, there is unlikely to be much monetary reward and the long-term survivability of the output is of primary concern. Editions produced using tools like the ones the *img2xml* project will build may not work in their current form 20 years after their release, but if the formats and code are open, it will at least be possible to migrate them to new, working forms. The use of proprietary formats and closed software makes indefinite survival a great deal less likely.

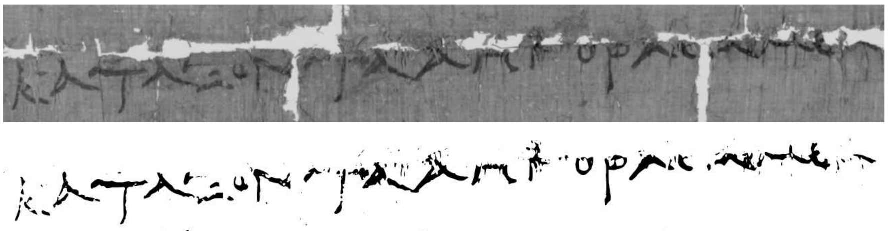
3 Problems and Solutions

In examining ways one might link between an XML transcription of the text and an XML overlay of the text, one quickly runs into problems involving overlapping hierarchies: single paths may include multiple letters or words, for example, and there may be single letters constituted from paths. As I noted above, the process of generating the SVG tracing involves the conversion of the image to a black and white (1-bit) bitmap, wherein each pixel is either 0 or 1. This makes it possible for the software to reproduce the shapes in the original source in vector format, but it also means the originally layered text has been flattened. While it might have been clear that the stroke of one letter runs over the top of a second in a color image, that layering is lost in the SVG, and the two letters are a single shape in the output. Such issues of information loss complicate the detection of structure in documents.

The figure below (derived from the papyrus in Figure 1) highlights some of these issues. Notice that the initial kappa is represented by no less than eight paths, while part of the downward stroke of the final alpha in κατἀξοντα connects to the following word, ἅ.

A variety of possible solutions to this problem presents itself, including editing the SVG so that the single path is split into two, or indicating word boundaries by drawing boxes that may intersect with paths. Even more pronounced examples of the layering problem are not hard to find: the Archimedes Palimpsest (Noel) contains a wealth of them, and highlights the value of image processing as a tool to enable scholars to read difficult texts. A palimpsest by definition contains at least two layers of text, and typically it is the obscured one that holds more scholarly interest (see Figure 5). *Img2xml* itself does not hold out any promise as a tool for separating layers—for that one needs multispectral imaging of the kind used on Archimedes—but once those layers can be teased apart, it could serve as a means to highlight the undertext on any image, or to demonstrate the interaction between layers.

The potential capabilities of this method quickly reveal some of the shortcomings in our representational formats when one begins to explore ways of linking between



κατάξοντα ἃ ἠγοράσαμεν

ΚΑΤΑΞΟΝΤΑΑΗΓΟΡΑΣΑΜΕΝ

Figure 4. Text and transcriptions of P. Mich. I 78, line 2.

tracings, images, annotations, and transcriptions. Pure linking is simple: all that is required when the formats are XML-based is that elements have unique identifiers. The SVG is a derivative of, and provides a usable coordinate system for, the source image. If one is careful (as for example the Archimedes Palimpsest project has been) to keep multiple images in different lighting aligned, then the SVG can serve as a map for all of them. But the semantics of SVG are almost purely geometric. It encodes shapes, with additional support for links, text, embedded images, and animation. This means there is no inherent way to express the significance of a grouping or a feature in SVG, nor the relationships between them. So, for example, we may consider how one might indicate that a set of paths in the SVG document signifies a word in the transcription. Assuming this is the uncomplicated case, where the shapes on the page do not somehow join members of one word to the next, we can combine those paths into an SVG group (<g>) element or we can draw a box within whose bounds all of the paths fall. The grouping at least has the advantage of establishing a parent-child relationship between the group and its members, and the <g> element, given an id-attribute, can be referenced externally or internally. Unfortunately, this method will only work in the simple case where a structure on the page does not contain letters from two different words. Unless a workaround can be found (and some possibilities will be discussed below), the creation of a container outside the hierarchy of paths and groups will be necessary. It is easy to create a rectangle which when rendered will contain an arbitrary set of paths, but because of SVG's geometric semantics, there will be no obvious way to

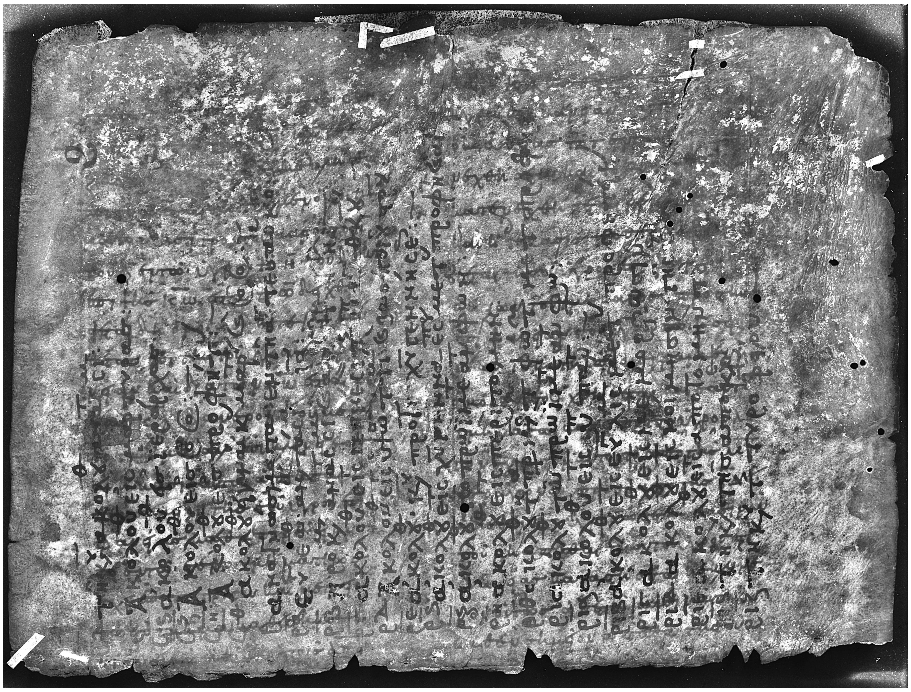


Figure 5. Archimedes *On Floating Bodies* folio 13v—under ultraviolet light.

indicate the relationship between that rectangle and the paths it "contains."[3] We are left then with a need either to work around the problem of overlapping hierarchies so that we can exploit the parent-child relationship established by <g> or to produce a method for defining relationships on the order of "isMemberOf" so that elements connected only by geometry may be semantically linked.

There are a variety of possible solutions to the issues presented by the hierarchical approach. For example, the paths forming κατάξοντα ἅ from Figure 3 could be split apart using a process that could find the intersection points of a word-dividing line (e.g. from a rectangle containing the word κατάξοντα) and the path representing the two letters alpha, and then split that path into two derivative paths, each of which would be associated with a different word and could become children of a <g> connected to a word in the transcription. This method would avoid damage to the actual tracing while

[3] I say "contains" in quotes, because the containment is only apparent when the document is rendered visually.

allowing the types of reference that are likely to be useful. The derivative paths could be placed in the same document and only activated as needed. But such a method would immediately raise the need for semantics again. We would want to be able to discover the relationship between the original, connected "…α ᾰ́" and its divided members since there is no inherent way to distinguish between one <path> element and another.

Regardless of the method used to associate paths and partial paths with words, letters, or notes, then, it will be a requirement that semantics be added to the SVG document somehow. Some possibilities for adding semantics to SVG documents include embedding relationship metadata using SVG's <metadata> element or developing a microformat (Microformats), perhaps depending on the class attribute, which is available on all displayable elements (Schepers). A further possibility would be to create an external document using the Resource Description Framework (RDF) that defined the relationships between elements within the SVG. This would have the further advantage that it could be used as the means to link the transcription and notes to the SVG as well, using a language with richer semantics than a plain URI.

4 Conclusions

The range of possibilities presented by the experimentation done to date on this method demonstrate its potential as a tool for the presentation of scholarly research on digitized manuscripts. Details of the implementation, particularly as regards the methods used to link between SVG tracing, transcriptions, and notes, remain to be worked out. There are also a range of issues, such as the layering problem and the need for semantic linking that must be explored. But there are a range of potential solutions to these problems which will be explored as part of the NEH Grant. The progress of the project may be observed at the project repository, where code and documentation will be released as they are produced.

Bibliography

Cayless, Hugh. "Linking Page Images to Transcriptions with SVG." *Proceedings of Balisage: The Markup Conference (2008).* <http://www.balisage.net/Proceedings/html/2008/Cayless01/Balisage2008-Cayless01.html>.

Cayless, Hugh. "*Img2XML* project repository." <http://github.com/hcayless/img2xml/tree/master>.

Chute, Ryan. *aDORe djatoka.* <http://african.lanl.gov/aDORe/projects/djatoka/index.html>.

Documenting the American South (DocSouth). <http://docsouth.unc.edu>.

The First Century of the First State University (UNC). <http://docsouth.unc.edu/unc>.

Flickr. <http://flickr.com>.
Holmes, Martin. *The Image Markup Tool.* <http://tapor.uvic.ca/~mholmes/image_markup/>.
ImageMagick. <http://www.imagemagick.org/index.php>.
Microformats. <http://microformats.org>.
Noel, William, et al. *The Archimedes Palimpsest.* <http://archimedespalimpsest.org/index.html>.
OpenLayers, v. 2.7. <http://openlayers.org>. (Release notes for v. 2.7 at http://trac.openlayers.org/wiki/Release/2.7/Notes).
Schepers, Doug. *Reinventing Fire* 'Blog Archive' SVG Text, Semantics, and Accessibility. November 7th, 2006 at 5:24 am. <http://schepers.cc/?p=11>.
Selinger, Peter. *Potrace, v. 1.8.* <http://potrace.sourceforge.net>.
Selinger, Peter. *Potrace: a polygon-based tracing algorithm.* <http://potrace.sourceforge.net/potrace.pdf>.
TEI Consortium. *TEI P5: Guidelines for Electronic Text Encoding and Interchange.* Ed. Lou Burnard and Syd Bauman, 2008. <http://www.tei-c.org/release/doc/tei-p5-doc/en/html/index.html>.
World Wide Web Consortium. *Resource Description Framework (RDF).* <http://www.w3.org/RDF/>.
World Wide Web Consortium. *Scalable Vector Graphics (SVG).* <http://www.w3.org/Graphics/SVG/>.