White Paper Report

Report ID: 98525

Application Number: HD5088009

Project Director: Jennifer Guiliano (jenguiliano@gmail.com)

Institution: University of South Carolina Research Foundation

Reporting Period: 9/1/2009-12/31/2010

Report Due: 3/31/2011

Date Submitted: 3/31/2011

The Sapheos Project:
Transparency in Multi-image Collation, Analysis, and Representation
White Paper

Jarrell Waggoner, Dhaval Salvi, Jun Zhou, Song Wang, Jennifer Guiliano
Department of Computer Science, Center for Digital Humanities
University of South Carolina

The Sapheos Project was funded via a Level II: Digital Humanities Start Up Grant to pioneer an innovative product: a digital collation software, prototyped in MATLAB and delivered as an open-source project using C code, that provides both a back-end collation tool and a powerful front-end interface for interacting with large datasets of books. The Sapheos Project partnered with the NEH-funded Spenser Project (spenserarchive.org), with supporting faculty from Cambridge University, Washington University at St. Louis, Pennsylvania State University, and the University of Virginia. This partnership offered the opportunity to utilize a broad cross-section of materials for testing purposes as well as an established community of scholars to advise on the context of the images being utilized with the project.

"Collation is the time-consuming but necessary comparison of two witnesses (copies) of an early modern edition in order to ascertain information about the printing process. Used extensively by historians of the book, bibliographic scholars, and those interested in the material culture of print, textual collation takes on a new importance when the underlying manuscript of the text is nonexistent. For many early modern authors whose manuscripts do not survive—Shakespeare and Spenser are perhaps the best known, but there are hundreds of lesser known authors for whom the printed text is the only known authority—textual collation is important not just for reasons of material culture and print history, but also to establish the authority of the underlying text, to separate error from accident, and isolate the way the text is from the way the text ought to be…In performing collation, researchers isolate difference as a series of binary judgments, building alteration sequentially by comparing many individual witnesses to a given 'control copy' that arbitrarily fixes the text to a given state. While many of the differences between texts consist chiefly of mechanical or human error—errors in typesetting and mechanical errors common to early presses and the methods for aligning and securing type within the forms—there are more than a few instances of variance within an edition that simply can't be explained in terms of error. For example, in the 1590 *Faerie Queene* [a work of Edmund Spenser], there are variants with whole words inserted or deleted, lines abridged or added, sonnets relocated or re-ordered. Coupled with the lack of a stable underlying manuscript to fix the ground of the text, collational variance becomes part of the assemblage of the text, irreducibly part of the play of its intricate meaning." [1]

Fundamentally, Sapheos Project software links the words on a page to the images of the words, figure tags on a page to images of the figures, and transcriptions of marginalia within the XML to images of the handwriting. Our software also automatically sections and generates (x,y) coordinate pairs for page images. Written in Java and being ported to MATLAB for C compilation, this software takes images of pages with existing XML markup and inserts (x,y) coordinate pairs into hierarchical elements—lines, paragraphs, stanzas (line groups), and figures—to allow XSL transformations to closely associate textual and image data that may vary in terms of quality and size for users. Taken together these two functionalities advance the use of

digital image processing techniques to automatically analyze historical documents with minimal user intervention. This is particularly significant for medieval and early-modern scholars who grapple with closely related sets of manuscripts that are differentiated by minute characteristics including textual variants, embellishments, and handwritings. Thus, the intent of the Sapheos Project was to create a ubiquitous tool for back end collation and user interface to allow for non-computational specialist scholars to undertake digital collation on their own. As will be outlined below, the effort to create a ubiquitous tool for digital collation was met with significant challenges:

A. Image acquisition, file format and size:
Thanks to the Spenser Archive Project, we obtained an initial dataset consisting of early modern books that had been photographed by multiple repositories (Washington University in St. Louis, Yale University, the Harry Ransom Center at the University of Texas-Austin, and others). These images appear in a variety of formats (tiff, jpg, png) and masters (uncropped, single page or double pages, with color bars and rulers present, colored or gray-scale); source image size ranged from 1MB to 128MB. Above all, to avoid distressing rare materials, the images were taken by camera held perpendicular to the books at a ninety-degree angle. This results in the common but serious problem that existing approaches have failed to resolve: warping or shading and curved text lines in the spine area of the bound volume. Image warping will not only reduce the OCR accuracy, but also impair readability. For example, figure 1 shows it is an unrealistic task to collate two similar pages by overlaying the warped page images.

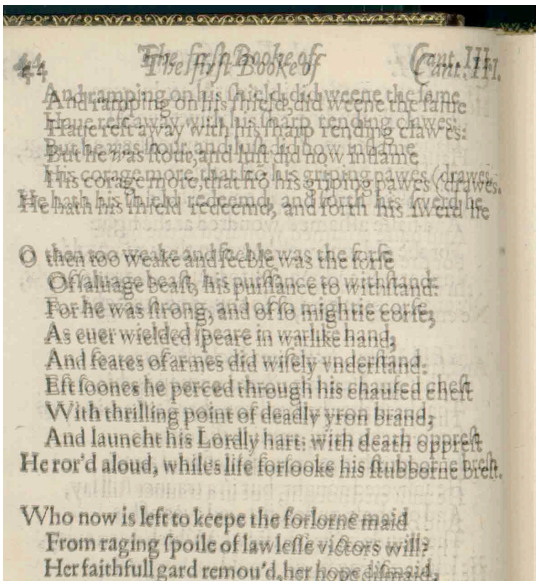


Figure 1: collation failed because of blurs caused by warping

B. Image preprocessing:
Image preprocessing included cutting the boundaries, splitting two-page images into one-page images, rough rotation and scaling, etc. We wrote several MATLAB (a high-level technical computing language) scripts to process these images in a batch order. By processing these in batch, we are able to alter multiple files at a time thereby decreasing the amount of time it takes to prepare the raw files for use.

C. Linking document images with text:
In our initial technical plan, we suggested that we would link page images to text encoded in XML. This process involved two steps: document image segmentation and injecting the segmentation information (xy-coordinates values) into XML. Here the XML file serves not only as a container for the result, but also as a reference to segmentation procedure. For example, <hi rend="face(ornamental)"> in XML represents a large ornamental letter image that may cross multiple lines. <lg> and <l> indicate the content is a poem that should be treated as a line instead of a paragraph. We had originally intended to link every word, line, paragraph, picture in the transcript to its coordinate position in the image to enable other functions like the highlighting of an image of searched-for text. To this end we can roughly segment out words, lines and stanzas. But with unusual fonts, warping and degradation present in document images, automated segmentation can never reach a 100% precision rate.

As manual correction from mis-segmentation presented in the first step, the tagging and coordinate-mapping process took longer than anticipated. We approximated a six-week process, yet even with the tool, linking text to image proved unfeasible as it took significant labor to complete every single word and line of text. For example, see figure for page image and following segmentation result in xml format.

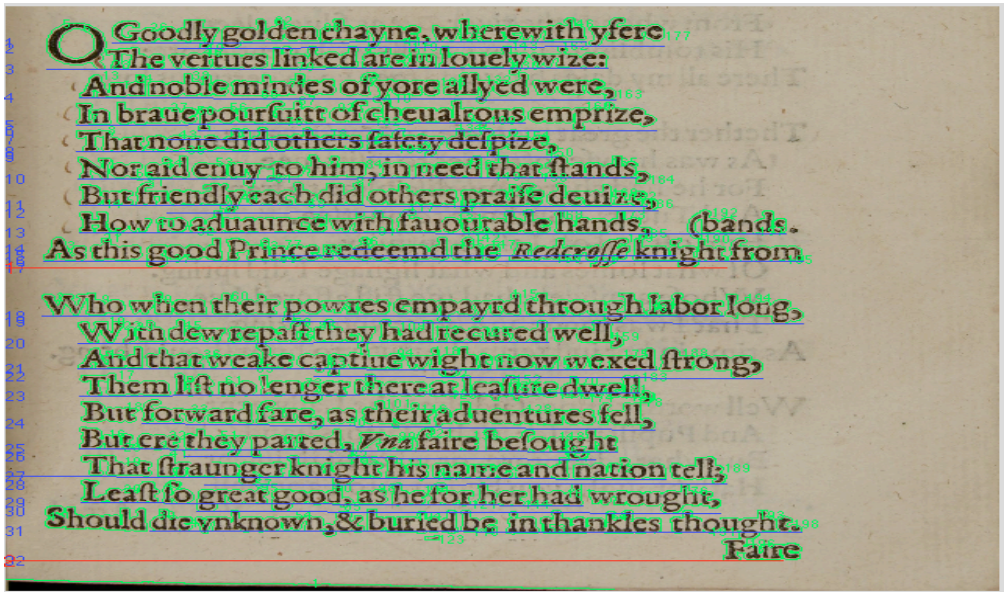


Figure 2: page with segmentation result illustrated in the image.

```
<stanzas>
:
<stanza>
    <stanzanumber>2</stanzanumber>
    <x_start>1279</x_start>
    <y_start>1518</y_start>
    <x_end>1495</x_end>
    <y_end>1518</y_end>
```

```
</stanza>
</stanzas>
<lines>
:
<line>
    <linenumber>32</linenumber>
    <x_start>1279</x_start>
    <y_start>1518</y_start>
    <x_end>1495</x_end>
    <y_end>1518</y_end>
</line>
</lines>
<words>
:
<word>
    <word>202</word>
    <x_start>1482</x_start>
    <y_start>1395</y_start>
    <x_end>1530</x_end>
    <y_end>1440</y_end>
</word>
</words>
```


This page has 2 stanzas, 19 lines and 136 words. The tool gives us 2 stanzas, 32 lines and 202 words. The manual correction for this single page needs to compare every element in the result with the image. After correction, injecting the resultant xml into the existing TEI XML transcription file requires the interjection of every TEI XML tag relating to the resulting xml element. This involves another round of manual correction if any mismatches exist. For example, if initial annual correction segments "&" as a single word, but the transcription put "&" and "buried" as one word, then that mis-identification needs to be corrected manually. As a result, we believe designing an intelligent automatic linking system is a much larger project than feasible under the auspices of this grant.

The NEH-funded TILE project (http://mith.umd.edu/tile/) at the Maryland Institute for Technology in the Humanities observed a similar problem and is currently developing a tool to improve the text-image linking process. In February 2011, the TILE project published its tool prototype. This prototype does not automatically segment image or link the text; instead it links any text block with any image block manually selected by the user and records linking in standard JSON format. In effect, it circumvents automated linking by inputting a manual identification process as central to creating the associative relations.

D. Separating printed text and hand-written text: Separating printed text and hand-written text was treated as a writer identification process [3][4]. The writer identification process is to identify writers by analyzing the structural properties of the handwriting, e.g. average height, the average width, the average slope and the average legibility of characters. In this process, the printing press was marked as a special writer. Any text content that did not display features

similar to those of the printed text are marked as hand-written text. Since computer-aided writer identification is a quite mature process in modern document image processing field, we borrowed three algorithms that worked best on our initial image dataset. They are autocorrelation (comparing shifted copies of images themselves), vertical and horizontal black run (counting number of continuous black pixels), and vertical and horizontal white run (counting number of continuous white pixels) - figure 2 shows the algorithm, and figure 3 shows the testing result. These algorithms have been implemented in the open source software – handanalyser (http://sourceforge.net/projects/handanalyser/).

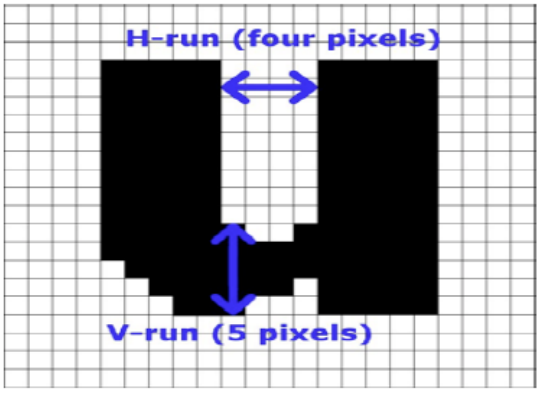
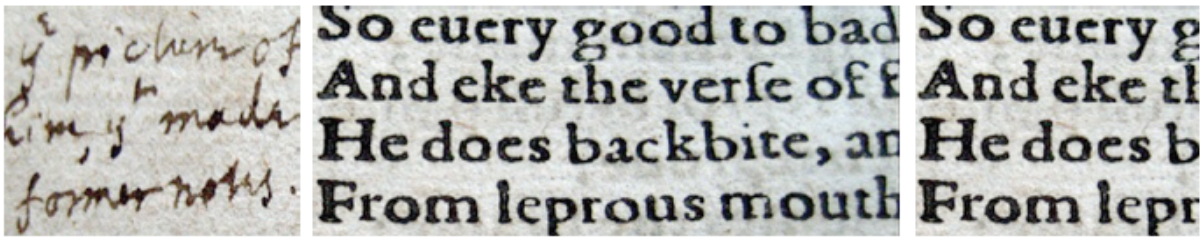


Figure 2: four continuous horizontal white pixels and five continuous vertical black pixels



A B C

A: Hand-written text B: printed content C: printed content

V-Runs	A	B	C
A			
B	39.9		
C	38.5	**0.6**	

D

Auto	A	B	C
A			
B	2.7		
C	2.5	**0.6**	

E

Figure 3: A, B and C show testing data examples; A shows a piece of hand-written text image; B and C show pieces of printed text from the same page source; D shows result running on vertical run algorithm; E shows result running on autocorrelation algorithm

E. Automatic Image Collation: Image collation is a process that identifies differences by comparing multiple images of similar documents. The differences range from semantic

difference to variations in the spacing or configuration of individual characters. The existing mechanisms in place for performing collation have traditionally been limited by requiring pristine scanned duplicates of the original witnesses to be compared. Three issues arise with these mechanisms: 1) they are usually obtained by flattening pages of a bound book, risking damage to the binding and adjacent pages; 2) without an efficient difference detection system, scholars sometimes need rigorous character-by-character manual inspections, e.g. typographic analysis; 3) or they require complex optical systems utilizing expensive hardware solutions. We have established an automatic image collation or automatic image registration that addresses these programmatic issues.

With two similar images, the target and the template, placed side by side, we first select about 70 to 100 pairs of matching points that are evenly distributed across the images [figure 4]; or example, the point under "Then" at the first line on the template page (left) should match the point exactly under "Then" at the first line on the target page (right). Based on these matching points, we warp the target image to the exact shape of template image [figure 5] – a process using the thin-plate-spline transformation algorithm in advanced digital image processing. Finally, we overlay the warped image and template image, resulting in the differences being shown as blurs on the overlaid image. [figure 6]

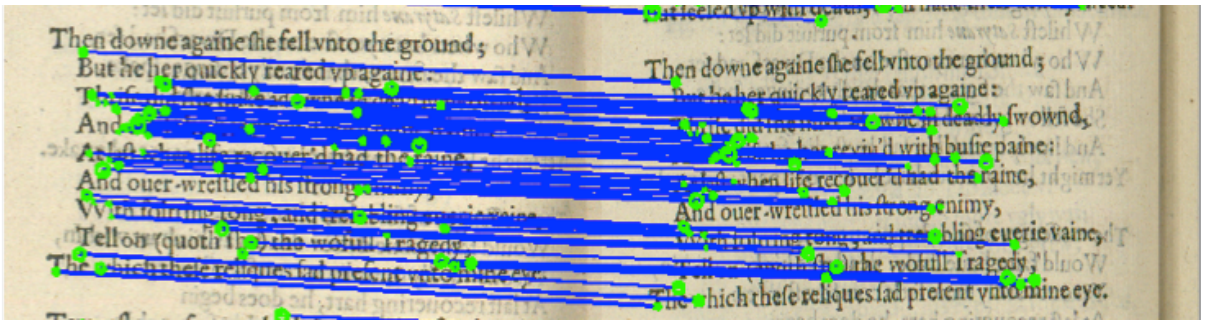


Figure 4: Point Matching

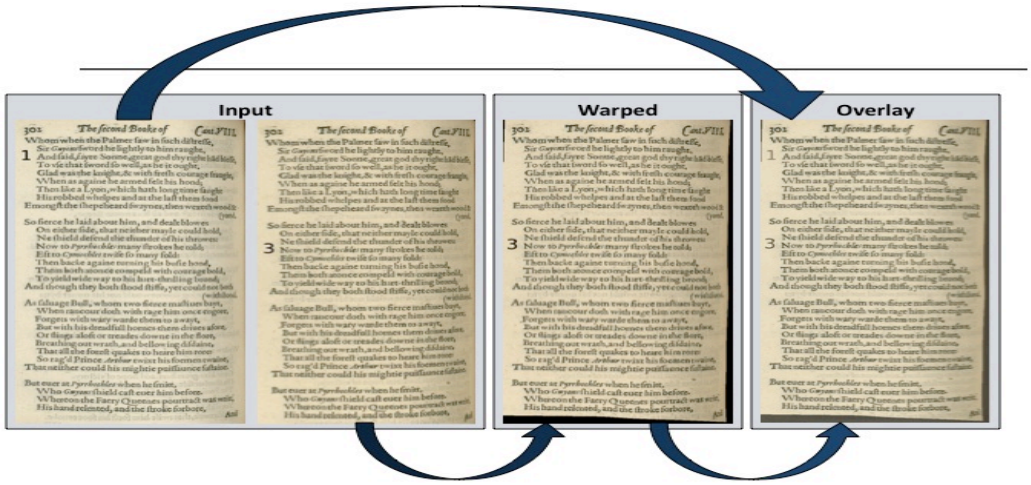


Figure 5: Warping target image to template image and overlaying them.

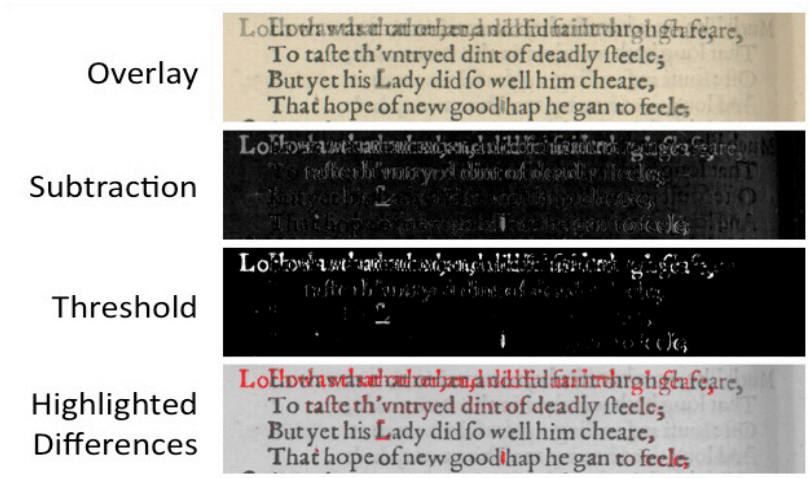


Figure 6: Difference Highlighting

To automate the collation process, precise point matching turns out to be the most critical step in the whole project. For each pair of images, we initially generate thousands of matching points using SIFT algorithm – a state-of-the-art image processing algorithm that isolates matching points. [2] We ten apply a four-tier system of filters to get rid of unnecessary points.

1. Nearest neighbor match filter: good matches should happen within designated neighboring area, e.g., the point at the bottom of template page should never match the point at the top of target page.

2. Duplicate match filter: good matches should be one-to-one matches; this is required by Thin-plate-spline transformation process.

3. Thin plate spline regression filter

4. Local consistency filter

The thin plate spline regression filter and local consistency filter are advanced image processing filters that remove matches affected by noise like document degradation, ink bleeding, etc.

The prototype was implemented with MATLAB. We evaluated this prototype by selecting about 172 large size images (1279X2444) and did ten rounds of two-page collation tests. Each test round took about 12 hours running on an eight-core processor MacPro server. All of our tests showed that all differences could be indentified correctly. However, false positives were a significant challenge and are addressed below.

F. User Interface: Constructing a user-friendly interface that can efficiently analyze historical document images was precisely our intention, and we have succeeded in completing this endeavor. Our user interface was built as a standalone JAVA application. As a cross-platform language, JAVA turns out to be the first choice for other popular image systems, e.g. ImageJ and Virtual Light Box. As promised, our application allows user to load up to four pages. Using the choice buttons at the panel bottom, the user can select and deselect any two of them to collate [figure 6]. The pages can be made translucent in order to provide a better visual result (differences shown as red marks) [figure 7]. Most important of all, while doing collating, the user does not need to align the images by tedious dragging manually. All of these tasks are performed automatically, even with warping presented in the images. The results are displayed as two pages perfectly overlaid with difference shown as blurs. Finding x-y coordinates is also included in the user interface [figure 8]. The application would be able to print out x-y coordinates of each word and line.

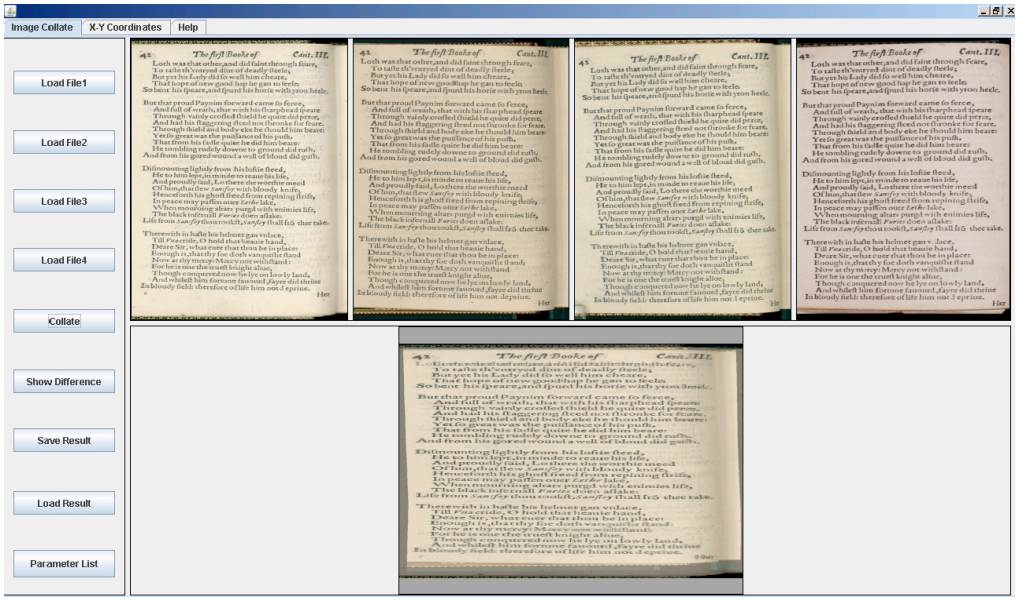


Figure 6: Collating multiple pages, result of two-pages images shown in the bottom right panel.

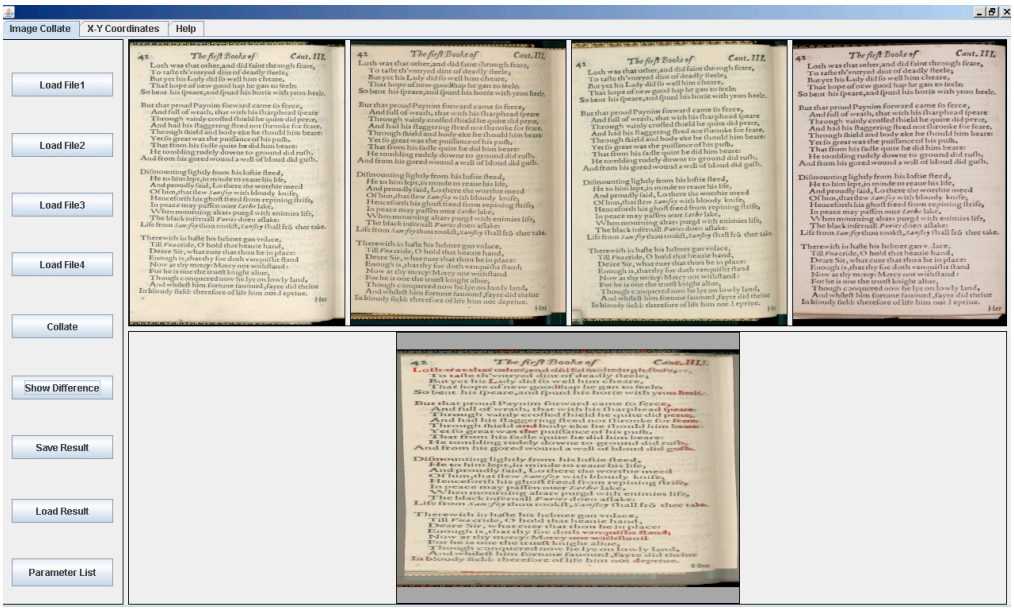


Figure 7: Differences shown as red marks

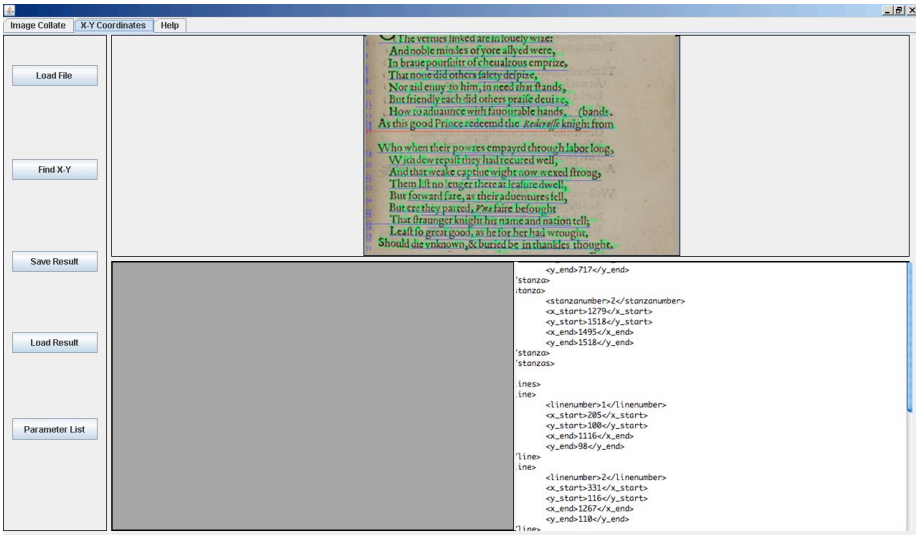


Figure 8: Finding X-Y coordinates, coordinates shown as xml format in the bottom right panel.

We had originally built an interface that would allow users to upload single images and collate two images at one time. This works perfectly for those who have small or middle scale images as it is a configured to use limited computational resources common to the average desktop computer. But what about collating a significant number of large-sized books, each with hundreds of pages, in one batch? Our experience shows collating two large-sized images, say 2200 X 2400, on a 2.26G intel core and 4GB memory PC can take up to 20 minutes per image. This is not ideal, as the need to collate hundreds or even thousands of images would dramatically affect both computational performance and the amount of time that a user would need to be stationed at the computer. We intend to continue expanding our software package, so that this software can cover not only personal needs, but also these large-scale institutional projects.

Further funding will be sought to continue the project. As any application developer working today is well aware, turning a successful prototype into fully functional system needs multiple software engineering phases. We intended to develop this innovative software system into robust, open-source software that will benefit all humanists. This section outlines activities that should be involved in our continuing work:

G. Collaboration: Collaboration can bring us more source images to refine our algorithms, targeted users to evaluate our system, and opportunities to integrate our system to other humanities projects. During our grant period, we have begun to explore possibilities for collaboration with projects such as the Shakespeare Quartos Archive group. We intended to hold workshops with computational scientists working in the digital humanities to assess the system and provide feedback to a multi-disciplinary community of scholars engaged in digital collation – a constituency including digital collections librarians, museum staff and digital humanists. In addition, we hope to offer an initial period of high-level technical support and system adjustment for participants. Through these activities, we encourage the adaptation of this software system to preserve and analyze digital assets while acquiring more image data to refine our software system.

H. Software refinement: As we discussed above, turning a prototype into a robust, fully function system requires a lot of refinements. Those refinements include expanding our system to work with document images that contain colors, drawings, partial matching, watermarks, burn marks, stains, ink bleeding and other degradation factors. Heavy degradation is the major source of numerous false positives. Although we will never be able to totally overcome this difficulty, we want to integrate some intelligent judging system so that with a little user interaction the precision rate can reach 100%. A further refinement of our software would deal with increasing system speed and efficiency to handle high-resolution, large-scale data sets of historical document images. Currently, the resolution and file size slows system performance speed.

I. Grant products.

In conclusion, our aim was both a) to demonstrate how historical document can be automatically collated and analyzed and b) to create prototypes that implement our algorithms.

a. A couple of papers are posted online at www.cdh.sc.edu/paragone/resource.html, including this technical white paper and "Image Registration for Digital Collation" by Jarrell Waggoner and Jun Zhou submitted to ICDAR (International Conference on Document Analysis and Recognition) 2011.

b. The prototype described here is available for download at www.cdh.sc.edu/paragon.

J. Glossary:

Matlab
: a numerical computing environment and fourth-generation programming language.

C
: a general-purpose computer programming language.

Java

a general-purpose, concurrent, class-based, object-oriented, cross-platform programming language.

XML

a set of rules for encoding documents in machine-readable form.

Feature

In computer vision and image processing, the feature is a piece of information that is relevant for solving the computational task related to a certain application.

Thin-plate-spline

the non-rigid transformation model in image alignment and shape matching.

References:

[1] Randall Cream, "The Sapheos Project: Transparency in Multi-image Collation, Analysis, and Representation," NEH Grant Application, 2009.
[2] Lowe, David G. (1999). "Object recognition from local scale-invariant features". Proceedings of the International Conference on Computer Vision. 2. Pp. 1150-1157
[3] Peter A. Stokes, "Palaeography and Image-Processing: Some Solutions and Problems", Digital Medievalist 3 (2007-2008)
[4] Bulacu, Marius, Lambert Schomaker, and Louis Vuurpijl. 2003. Writer-identification using edge-based directional features. ICDAR, 2:937-941

Participants:

Dr. Jennifer Guiliano, PI, Associate Director of the Center for Digital Humanities at the University of South Carolina
Dr. Song Wang, Co-PI, Associate Professor in the Department of Computer Science at the University of South Carolina
Jarrell Waggoner, Ph.D Candidate in the Department of Computer Science at the University of South Carolina
Jun Zhou, Lead Programmer at the Center for Digital Humanities at the University of South Carolina
Dhaval Salvi, Ph.D Candidate in the Department of Computer Science at the University of South Carolina
Dr. Randall Cream, Post Doctoral Scholar in the Department of English at the University of South Carolina (2008- May 2010)
Maliek Mcknight, Programmer at the Center for Digital Humanities at the University of South Carolina
Shawn Maybay, Digital Humanity Specialist at the Center for Digital Humanities at the University of South Carolina
Bernardo Quibiana, Programmer at the Center for Digital Humanities at the University of South Carolina