SEGMENTATION AND SCALE DETECTION ALGORITHMS FOR AUTOMATED ANALYSIS OF DIGITIZED HISTORICAL MAPS

BY

TENZING WILLIAM SHAW

THESIS

Submitted in partial fulfillment of the requirements for the degree of Master of Science in Electrical and Computer Engineering in the Graduate College of the University of Illinois at Urbana-Champaign, 2011

Urbana, Illinois

Advisers:

Senior Research Scientist Peter Bajcsy
Professor Narendra Ahuja
ABSTRACT

This thesis addresses the problems of automatic segmentation of objects in historical maps, automatic estimation of map scale and the design of a mathematical framework for understanding the uncertainties associated with map scale estimates. The problems are motivated by the lack of accuracy and consistency in the current analysis of geographical objects found in historical maps, which is conducted by unaided visual inspection.

Our approach decomposes the analysis of geographical objects into workflow steps such as object segmentation, spatial scale calibration, extraction of calibrated object descriptors and comparison of descriptors over time and multiple cartography houses. The key computer science contributions are made in the segmentation and map scale calibration workflow steps. The segmentation step is achieved by designing a template-supervised ball-based region growing method employing the Hu moments as shape descriptors. The automation of spatial calibration (map scale estimation) is accomplished by algorithms that detect and classify lines along map borders, searching for dashed neatlines intersected by latitude lines. Thus, descriptors of map objects represented by segmentation results in pixels can be converted to geographical units; for example, the area of a lake can be reported in square miles. Finally, the map scale estimation process is modeled mathematically in order to establish uncertainty of the scale results. The uncertainty framework models contributions from various sources of error in the digitized historical map images, including clutter such as text impinging on the region of interest, low contrast between light and dark dashes of the neatline, as well as other sources.

The application of our work has been to compare shape characteristics of the Great Lakes region in a dataset of approximately 40 French and British historical maps created in the seventeenth through the nineteenth centuries. The objective was to determine which colonial power possessed more accurate geographic knowledge of the region, and how this balance changed over time. We report experimental evaluations of automation accuracy based on comparison with manual segmentation results, as well as the knowledge obtained from the area comparisons. We also report the results obtained from experiments designed to allow uncertainty analysis of the scale estimation subsystem.
ACKNOWLEDGMENTS

This project has been funded by the National Center for Supercomputing Applications and the National Science Foundation. We would like to acknowledge the NSF IIS 09-10562 EAGER award for providing the funding. We would also like to acknowledge the UIUC library and Betsy Kruger for providing the images of historical maps, and Michael Simeone and Robert Markley for consultations and insights about the map content, as well as help with finding the maps of interest.
# TABLE OF CONTENTS

DEFINITIONS OF VARIABLES .......................................................................................................... vi

1 INTRODUCTION ............................................................................................................................. 1
   1.1 Background and Significance ........................................................................................................ 1
   1.2 Overview of Problems .................................................................................................................. 1

2 PREVIOUS WORK .......................................................................................................................... 4
   2.1 Previous Work Related to Segmentation Algorithm ................................................................. 4
   2.2 Previous Work Related to Map Scale Detection Algorithm .................................................... 5

3 SEGMENTATION ............................................................................................................................. 6
   3.1 Overview of Segmentation Algorithm .......................................................................................... 6
      3.1.1 Underlying Ball-Based Region Growing Segmentation Algorithm .................................. 6
      3.1.2 Shape Supervised Ball-Based Region Growing Segmentation Algorithm ................... 8
      3.1.3 Shape Features: Hu Moments .............................................................................................. 8
   3.2 Optimization of Algorithmic Parameters .................................................................................... 9
      3.2.1 Weights Associated with Hu Moments in Shape Similarity Metric .................................. 10
      3.2.2 Segmentation Seed Location ............................................................................................. 10
      3.2.3 Intensity Threshold and Ball Size Parameter Search ....................................................... 12
   3.3 Pre- and Post-Processing Operations ......................................................................................... 12
      3.3.1 Pre-Processing Operation ................................................................................................... 13
      3.3.2 Post-Processing Operation ................................................................................................ 14
   3.4 Pseudo-Code of Segmentation Algorithm ................................................................................ 14
   3.5 Segmentation Performance Evaluation ..................................................................................... 15
      3.5.1 Computational Speed ........................................................................................................... 15
      3.5.2 Segmentation Accuracy Evaluation ....................................................................................... 16

4 MAP SCALE DETECTION .................................................................................................................... 18
   4.1 Introduction ............................................................................................................................... 18
      4.1.1 Manual Calibration Methods ............................................................................................... 18
      4.1.2 Translation to Algorithmic Steps ........................................................................................ 19
   4.2 Map Boundary Extraction and Selection .................................................................................... 21
4.3 Line Detection ................................................................. 21
  4.3.1 Line Detection Algorithm ............................................ 22
  4.3.2 Boundary Image Derotation Overview ............................ 24
  4.3.3 Derotation Algorithm Description ............................... 25
4.4 Line Classification .......................................................... 26
4.5 Dash Length Calculation .................................................. 28
  4.5.1 Adaptive Thresholding for Dash Length Estimation .......... 29
4.6 Transversal Detection Subsystem ..................................... 30
4.7 Lookup Table for Converting Number of Dashes to Miles per Pixel ................................. 32
4.8 Map Scale Detection Performance Evaluation ......................... 34

5  UNCERTAINTY ANALYSIS OF MAP SCALE .................................. 35
  5.1 Variables Affecting Transversal Detection ......................... 36
  5.2 Background Clutter ....................................................... 37
  5.3 Graticule Line Contrast .................................................. 39
  5.4 Synthesis of Transversal Detection Failure Modes ................ 41
  5.5 Variables Affecting Dash Length Calculation ..................... 42
  5.6 Resolution ..................................................................... 43
  5.7 Inherent Imprecision ........................................................ 43
  5.8 Contrast ....................................................................... 44
  5.9 Sharpness ..................................................................... 45
  5.10 Dash Length Uncertainty Measurement ............................ 45
  5.11 Catastrophic Failure due to Dash Length Calculation Error ........ 48
  5.12 Catastrophic Failure and Uncertainty Synthesis ................ 49

6  RESULTS ............................................................................. 50

7  CONCLUSION ..................................................................... 54

REFERENCES ....................................................................... 56
## DEFINITIONS OF VARIABLES

<table>
<thead>
<tr>
<th>Variable</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>$I(r, c)$</td>
<td>Intensity of any of the digital images under consideration at row $r$ and column $c$</td>
</tr>
<tr>
<td>$I_A(r, c)$</td>
<td>Intensity of an automatically generated binary image produced by the segmentation algorithm</td>
</tr>
<tr>
<td>$I_M(r, c)$</td>
<td>Intensity of a manually segmented binary image</td>
</tr>
<tr>
<td>$m_{pq}$</td>
<td>The pq&lt;sup&gt;th&lt;/sup&gt; central moment of an image</td>
</tr>
<tr>
<td>$\bar{r}, \bar{c}$</td>
<td>The centroid coordinates of an image (row, column)</td>
</tr>
<tr>
<td>$\mu_{pq}$</td>
<td>The pq&lt;sup&gt;th&lt;/sup&gt; scale invariant central moments of an image</td>
</tr>
<tr>
<td>$h_i$</td>
<td>The $i^{th}$ Hu moment of an image, where $i \in {1,2,...,7}$</td>
</tr>
<tr>
<td>$\alpha$</td>
<td>Adaptive threshold sensitivity parameter for line or transversal detection</td>
</tr>
<tr>
<td>$w$</td>
<td>Window-size parameter for line or transversal detection</td>
</tr>
<tr>
<td>$s_i(r, c)$</td>
<td>The transversal detection signal evaluated at $(r, c)$</td>
</tr>
<tr>
<td>$s_i(c)$</td>
<td>The one dimensional version of $s_i(r, c)$ when $r$ is fixed</td>
</tr>
<tr>
<td>$T_{CL}$</td>
<td>Threshold for failure prediction based on clutter</td>
</tr>
<tr>
<td>$T_{CO}$</td>
<td>Threshold for failure prediction based on contrast</td>
</tr>
<tr>
<td>$S$</td>
<td>True map scale</td>
</tr>
<tr>
<td>$S'$</td>
<td>Estimated map scale</td>
</tr>
<tr>
<td>$l_i$</td>
<td>Distance between successive neatline-graticule intersections in pixels</td>
</tr>
<tr>
<td>$l_d$</td>
<td>True dash length in pixels</td>
</tr>
<tr>
<td>$l_d'$</td>
<td>Estimated dash length in pixels</td>
</tr>
<tr>
<td>$d_i$</td>
<td>Number of degrees latitude between two intersections</td>
</tr>
<tr>
<td>$N$</td>
<td>True number of dashes between two intersections</td>
</tr>
<tr>
<td>$N'$</td>
<td>Estimated number of dashes between intersections (before rounding)</td>
</tr>
<tr>
<td>$\epsilon^N$</td>
<td>Absolute error in $N:</td>
</tr>
<tr>
<td>$\sigma^2_{CL}$</td>
<td>Clutter measure for transversal detection</td>
</tr>
<tr>
<td>$\epsilon^R_{CO}$</td>
<td>Contrast measure for transversal detection</td>
</tr>
<tr>
<td>$P_{CF}^D$</td>
<td>Probability of catastrophic failure in calibration</td>
</tr>
<tr>
<td>$\epsilon^L_{CO}$</td>
<td>Error in dash length estimation due to low contrast</td>
</tr>
<tr>
<td>$\epsilon_L$</td>
<td>Error in dash length estimation due to inherent uncertainty</td>
</tr>
<tr>
<td>$\epsilon_{SH}$</td>
<td>Error in dash length estimation due to ill-defined boundaries (low sharpness)</td>
</tr>
<tr>
<td>$\epsilon_R$</td>
<td>Error in dash length estimation due to image resolution</td>
</tr>
<tr>
<td>$\sigma^2_{DL}$</td>
<td>Variance in measured dash lengths</td>
</tr>
<tr>
<td>$\epsilon_{DL}$</td>
<td>True dash length error</td>
</tr>
<tr>
<td>$\epsilon'_{DL}$</td>
<td>Estimated upper bound on dash length uncertainty</td>
</tr>
<tr>
<td>$\epsilon^C$</td>
<td>Combined uncertainty in scale estimation</td>
</tr>
</tbody>
</table>
1 INTRODUCTION

1.1 Background and Significance

An important question facing historians is how knowledge of different geographic regions varied between nations and over time. A specific example of this question is how to characterize differences in geographic knowledge of the Great Lakes region possessed by the French and British from the seventeenth to the nineteenth century. This can be accomplished by examining French and British historical maps from different points in this time period, and judging the accuracy of these maps relative to modern geographic knowledge. Previously, this would have been done qualitatively, through visual inspection.

The objective of this effort is to translate visual inspections of historical map analyses into an algorithmic sequence and to automate the execution of the algorithmic sequence. In order to design a successful algorithmic sequence, the accuracy and uncertainty of the computations must be satisfactory, as well as the computational speed. The significance of this work is in addressing the translation and automation objectives, and supporting discoveries in cartography enabled by contributions from computer science.

1.2 Overview of Problems

This work addresses four problems related to analyzing historical maps:

- translation of visual inspections of historical map analyses into an algorithmic sequence
- automated segmentation
- automated map scale estimation
- uncertainty modeling of map scale estimates

The first problem is to translate visual inspections of historical map analyses into an algorithmic sequence. We have addressed this problem by decomposing the visual inspections into steps
captured in Figure 1. The algorithmic workflow consists of template shape-based segmentation of geographic objects of interest followed by spatial calibration and shape analysis of the segmented objects. The workflow is illustrated with digitized historical maps of the Great Lakes region (38 maps consisting of 21 British and 17 French maps). In the case of each Great Lake, the algorithm segments the lake from cropped images, and computes its surface area in square miles. The differences between these calculated areas and the modern figures can then be taken as a measure of the accuracy of regional geographic knowledge at the time when each historical map was made. We report experimental results for the area measure applied to all five Great Lakes over the map dataset described above in Chapter 6.

The second problem is to automate segmentation of objects such as lakes in historical maps. Assuming that the lakes have not changed in shape significantly since the time period being considered, we approached the problem by designing a segmentation algorithm for extracting regions whose shape is most similar to that of a given example. The algorithm uses ball-based region-growing segmentation combined with the seven Hu moments [1] to evaluate shape similarity. The ball-based segmentation places a circular region into a seed location and grows the region subject to color homogeneity and spatial contiguity constraints. Each resulting region is described by the Hu moments and compared to the Hu moments of a given example. The algorithm searches over a space of parameters including the region growing criteria and seed placement. We report results pertaining to the performance of the segmentation algorithm in

Figure 1. The algorithmic workflow for extracting cartographic information from historical maps.
Section 3.5, including statistics of automatic segmentation accuracy compared to that achieved by manual segmentation.

The third problem is to automate map scale estimation in order to enable comparisons of map objects. For example, for a comparison of geographic quantities, such as lake area or shore length, it is necessary to determine the scale of the map under consideration, so that a conversion can be obtained between pixels and a physical unit such as degrees or miles. The common calibration technique is to use two points of known distance apart and manually estimate the scale by dividing the physical distance by the number of pixels corresponding to that distance on the map. However, this technique is difficult to automate because it assumes the presence of two known points and a high degree of consistency in their map presentation, which would be needed by a detection algorithm. Instead, we have employed another calibration technique based on examining the dashed neatline which is present around the border of most of the maps under consideration, and the relationship of the neatline dashes with respect to latitude or longitude lines. This map calibration technique seems to be more suitable for automation. We present experimental results comparing the manual and automatic calibration techniques in Section 4.7.

The fourth problem is to design a mathematical framework for estimating the probability of catastrophic failure and the uncertainty bounds for the map scale. In addition, the framework can only include inputs that are measurable automatically in an image of a historical map, and one can implement image probes to evaluate the inputs feeding the uncertainty framework. We give experimental results pertaining to the uncertainty analysis in Chapter 5.
2 PREVIOUS WORK

In general, the specific application considered in this account appears to be novel, while some of the approaches used in various parts of the algorithms described herein either have been explored previously, or are closely related to other techniques which have been developed.

Note that the current account draws extensively from [2], which is a previous paper by the same authors describing the project.

2.1 Previous Work Related to Segmentation Algorithm

As can be observed in Figure 1, the regions targeted for segmentation—lake interiors—have relatively homogeneous color. As is noted in [3], a segmentation problem with this property is well-suited to a region-growing approach. An early survey of region-growing approaches to segmentation can be found in [3] with the current approach falling under what the authors designate as “Approach 1: Regional Neighbor Search.”

The authors of [4] developed a method conceptually similar to the template supervised segmentation approach used here. Much like the algorithm we present, the algorithm described in [4] attempts to segment an object from a given image which is similar in shape to a given query template. Their approach attempts to handle objects composed of multiple regions of uniform intensity existing in a certain special configuration (for example, a car, in which the windows would have different intensity from the metallic frame). They represent the spatial configuration of these regions using a graph structure which provides a signature for the decomposed object and also allows its constituent parts (such as the roof of a house vs. doors or windows) to be identified. In the current application, this functionality is not required, since the goal is not to divide the interior of a lake into semantically meaningful parts, but to consider shape properties of the interior as a whole.

The Hu invariant moments have been used extensively in previous object recognition applications. In [1], the original paper which introduced these moments, Hu describes their application to optical character recognition. A further example is provided by [5], in which the
authors describe the application of their efficient method for computing the moments to simple object silhouettes.

2.2 Previous Work Related to Map Scale Detection Algorithm

To the best of our knowledge, the problem of automatically analyzing dashed neatlines in order to determine map scale has not been explored previously. Some of the techniques we used in our approach to this problem have been researched. In particular, techniques for detecting peaks (or equivalently, valleys) in a noisy one-dimensional signal have been developed for use in many other contexts, such as the automated analysis of astronomical spectra as described in [6]. The peak-detection problem is central to the line detection and transversal detection subsystems described in Sections 4.3 and 4.6, respectively. As the authors of [7] observe, most work on this problem has been application specific. The reason for this is that the definitions of what constitutes a peak vs. what constitutes noise depend on the application under consideration. In keeping with this idea, the scale detection algorithms used in this experiment employ a thresholding technique involving application-specific, empirically chosen parameters. The authors of [7] argue for a more general approach which would eliminate the need for such empirical parameters. They present a method for peak detection which assumes known peak shape and noise characteristics, as well as non-overlapping peaks. While the approximate peak shape can be known in the present application, the latter two assumptions are problematic. Firstly, it is not known how to estimate the noise statistics, since the vast majority of noise is caused by the failure of the cartographer to achieve perfectly uniform foreground and background colors, and by the presence of clutter such as text, rather than by more tractable sources such as noise in the image capture device. Secondly, it cannot be assumed in the current application that adjacent peaks will not overlap; this problem is caused by a slight rotation of the image relative to its true orientation as will be seen in Section 4.3.2. Nonetheless, it might be profitable to attempt to develop a similarly systematic method which would apply to the current system in the future.
3 SEGMENTATION

3.1 Overview of Segmentation Algorithm

We have designed and developed a segmentation algorithm for extracting regions whose shape is most similar to that of a given example. The algorithm uses ball-based region-growing segmentation combined with the seven Hu moments to evaluate shape similarity. The ball-based segmentation places a circular region into a seed location and grows the region subject to color homogeneity and spatial contiguity constraints. Each resulting region is described by the Hu moments and compared to the Hu moments of a given example. The algorithm searches over a space of parameters including the region growing criteria (intensity threshold and ball size) and seed placement, attempting to find the closest result (using the Euclidean distance metric in the space of Hu moments) to the example. The algorithmic workflow is illustrated in Figure 2.

Figure 2. Segmentation algorithmic workflow.

3.1.1 Underlying Ball-Based Region Growing Segmentation Algorithm

The segmentation algorithm developed for this effort uses an underlying low-level segmentation algorithm based on a region growing approach. Beginning at a given seed pixel, the algorithm
recursively examines (four-connected) neighboring pixels, which are included in the foreground if they differ from the value of the seed pixel by less than a chosen threshold. To handle images in which object boundaries may contain gaps, an additional ball-size parameter is added. For non-trivial values of this parameter (i.e. values greater than one), a circular region of diameter equal to the ball size is examined at each candidate pixel. If every pixel within this region differs from the seed value by less than the threshold, then the entire region is labeled as foreground, and the neighbors of the candidate pixel are examined as before. Pseudocode for this algorithm is given below.

```
Ball = Disc of diameter BallSize
(SeedX, SeedY) = Coordinates of seed location
SeedValue = Average pixel value over a circular region of diameter BallSize at the seed location
Threshold = Intensity threshold defining range of foreground intensities relative to SeedValue
Intensity(): Returns the intensity of a given pixel

// Algorithm Start
Segment(SeedX, SeedY);

Segment(x, y){
    Place Ball at (x, y);
    for(Pixel ∈ Ball){
        if(|Intensity(Pixel) - SeedValue| > Threshold) return;
    }
    Label all pixels in Ball as foreground;
    Segment(x+1, y);
    Segment(x-1, y);
    Segment(x, y+1);
    Segment(x, y-1);
}
```
3.1.2 Shape Supervised Ball-Based Region Growing Segmentation Algorithm

As described earlier, the segmentation algorithm designed for this endeavor acts as a layer on top of the underlying segmentation algorithm detailed in the previous section. It uses a manually segmented example shape, typically extracted from a modern map, to guide the choice of parameters (threshold, ball size, and seed location) for the underlying algorithm. It is assumed that the more similar a given segmentation result is to the example shape, the more likely it is to represent the desired object, such as one of the Great Lakes.

3.1.3 Shape Features: Hu Moments

Similarity is measured using the seven Hu moments as features to describe both the example shape and each segmentation result. These features were chosen for their invariance under rotation, translation, and scaling, a property which is desirable in this application because resolution and positioning of the target object may vary between images, and even orientation may vary slightly due to differing notions of the ground truth between cartographers. The Hu moments are defined in terms of the standard image moments according to equations (1), where \( I \) is the binary image under consideration [1]:
\[ m_{pq} = \sum_r \sum_c (r - \bar{r})^p (c - \bar{c})^q I(r, c) \]

\[ \bar{r} = \frac{\sum_r \sum_c r I(r, c)}{\sum_r \sum_c I(r, c)} \]

\[ \bar{c} = \frac{\sum_r \sum_c c I(r, c)}{\sum_r \sum_c I(r, c)} \]

\[ \mu_{pq} = \frac{m_{pq}}{m_{00}^{\frac{1}{1+(p+q)/2}}} \]

\[ h_1 = \mu_{20} + \mu_{02} \]

\[ h_2 = (\mu_{20} - \mu_{02})^2 + 4\mu_{11}^2 \]

\[ h_3 = (\mu_{30} - 3\mu_{12})^2 + (3\mu_{21} - \mu_{03})^2 \]

\[ h_4 = (\mu_{30} + \mu_{12})^2 + (\mu_{21} + \mu_{03})^2 \]

\[ h_5 = (\mu_{30} - 3\mu_{12})[\mu_{30} + \mu_{12}]^2 - 3(\mu_{21} + \mu_{03})^2 + (\mu_{21} - \mu_{03})(\mu_{21} + \mu_{03})^2 \]

\[ h_6 = (\mu_{20} - \mu_{02})[\mu_{30} + \mu_{12}]^2 - (\mu_{21} + \mu_{03})^2 + 4\mu_{11}(\mu_{30} + \mu_{12})(\mu_{21} + \mu_{03}) \]

\[ h_7 = (3\mu_{21} - \mu_{03})(\mu_{30} + \mu_{12})[\mu_{30} + \mu_{12}]^2 - 3(\mu_{21} + \mu_{03})^2 - (\mu_{30} - 3\mu_{12})(\mu_{21} + \mu_{03})^2 \]

3.2 Optimization of Algorithmic Parameters

The need to optimize the selection of parameters in some way arises from the diverse characteristics of the map images being studied. The purpose of the segmentation algorithm is to capture the foreground of the desired object, for example a lake, while rejecting the background, which is the boundary of the lake. Since the intensity characteristics of the foreground and background may vary widely, there may be no single value of the threshold parameter which works in every case. Similarly, since the thickness of the object boundary and its contrast with the interior may also vary, the ball size parameter must be chosen adaptively as well.
3.2.1 Weights Associated with Hu Moments in Shape Similarity Metric

An optimization challenge which was encountered in using the Hu moments as features was the question of how these moments should be weighted. Currently, they are given equal weight in the algorithm. Since the higher order moments have much smaller magnitude than the lower order moments, however, this amounts to only using the first few moments to characterize shape similarity. Since it is not clear analytically how the weights should be chosen, a possible solution is to learn them from training data. Accordingly, a side experiment was conducted in which a classifier was designed to distinguish among four of the Great Lakes’ shapes. In this experiment, the AdaBoost algorithm was used to train classifiers distinguishing each lake shape from all other lake shapes, which were then combined to create a multi-class classifier. The results are given in Table 1. Although accuracies were significantly higher than 20%, the rate expected from random assignment of labels, they were not high enough to merit the application of this method to segmentation. If additional training data could be acquired, however, it might be worthwhile to revisit this approach.

Table 1. Results of AdaBoost classification side-experiment

<table>
<thead>
<tr>
<th>Lake</th>
<th>Erie</th>
<th>Huron</th>
<th>Ontario</th>
<th>Superior</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Instances</td>
<td>36</td>
<td>34</td>
<td>37</td>
<td>37</td>
<td>144</td>
</tr>
<tr>
<td>Number of Errors</td>
<td>24</td>
<td>11</td>
<td>16</td>
<td>18</td>
<td>69</td>
</tr>
<tr>
<td>Classification Accuracy</td>
<td>33%</td>
<td>68%</td>
<td>57%</td>
<td>51%</td>
<td>52%</td>
</tr>
</tbody>
</table>

3.2.2 Segmentation Seed Location

Apart from searching for the most similar result in the space of Hu moments, the algorithm also imposes size constraints on the extrema it discovers. Since the Hu moments are scale invariant, there is a possibility that a very small object (for example, an island) could appear to be similar in shape to the template. In order to account for this, we assume that the desired object will take up between 15% and 85% of the original cropped image. Any segmentation result with area outside of these bounds is disqualified from being a candidate for the optimal result. This assumption places a constraint on the human who crops the lake images, who will have to
perform a subjective estimate of the percentage of the image taken up by the lake in order to ensure that it falls within these bounds.

This condition creates the possibility that none of the shape similarity based minima found will meet the size condition. In practice, this condition might correspond to a case in which the seed lands inside a letter as shown in Figure 3. In this situation, there may be no choice of parameters which yields a reasonable result, since a threshold which is too low will capture only the letter, while one that is too high will escape the letter and also the lake boundary (both of which tend to be dark). This situation is dealt with by repeating the algorithm with a different seed location offset from the original location by a random displacement vector with length constrained to a certain range. If the seed lands inside a letter, for example, it will eventually escape the letter via this procedure, which resembles a two-dimensional random walk. This is illustrated in Figure 3, in which the first two seed locations fail due to the seed having landed on the letter “H,” while by the third attempt, the seed has escaped the letter, and the algorithm returns a reasonable result.

Figure 3. Methodology for correcting initial poor seed placement.
3.2.3 **Intensity Threshold and Ball Size Parameter Search**

To illustrate the method by which the segmentation algorithm chooses the best parameters, consider Figure 4, in which the target image is a modern map of Lake Ontario, and the example shape is derived from a different modern map. The six lower images represent each point in the parameter search at which a new minimum distance between the Hu features of example and target shapes has been found (relevant parameters are given below each result). We observe that the algorithm is able to reject the initial incomplete results, and adjust the intensity threshold to include the brighter region in the northeast of the lake. In this case, the optimal threshold was found to be 50 while the optimal ball size was found to be 2.

![Target Image](image1)

![Example Shape](image2)

![Results](image3)

Figure 4. Illustration of search over parameters of underlying ball-based segmentation algorithm: D denotes the distance from the example in Hu moment space, B denotes the ball size, and T denotes the threshold.

3.3 **Pre- and Post-Processing Operations**

For reasons which will be explained in the following sub-sections, it is desirable to prepare images for segmentation using morphological preprocessing and to perform post-processing to remove holes from results. The following sections detail these procedures and the reasons for them.
3.3.1 Pre-Processing Operation

A further refinement of the algorithm is the use of morphological operations to remove clutter from input images prior to processing. A major challenge in segmenting geographic objects using a region-growing approach is the presence of clutter in the form of latitude and longitude lines, boundary following hash patterns, or text. This clutter can sometimes form a barrier to region growing, as is the case in the image of Lake Michigan at left in Figure 5. In this example, the seed is placed in the southern part of the lake, and a latitude line forms a barrier which prevents the region from growing into the northern part of the lake. Using morphological closing with a vertically oriented 5x1 structuring element, the latitude line can be removed, allowing the region-growing algorithm to segment the entire lake. Similarly, a 1x5 structuring element can be used to eliminate the longitude line blocking the narrow strip of lake in the southeast. A description of morphological closing can be found in [8]. In the example of Figure 5, the region growing algorithm was initially blocked by the two graticule lines, but was able to capture the whole lake once morphological pre-processing had been introduced.

![Figure 5. Removal of latitude and longitude lines using morphological closing with 5x1 and 1x5 structuring elements, respectively: the original image and corresponding segmentation result are shown at left while the filtered image and improved segmentation result (using the same parameters) are shown at right.](image-url)
3.3.2 Post-Processing Operation

Note that since the objects of interest are assumed to be simply-connected, and initial segmentation results are not in general (due to intensity noise, background clutter including latitude and longitude lines, country demarcations, hashes along lake boundaries referring to shallow depth, or text written over regions), we perform a step of post-processing similar to morphological closing \([1]\), in which the ball-based region growing algorithm is applied to the initial result with a large ball-size and a seed location outside of the object, thereby eliminating “holes” in the shape. This procedure is shown in Figure 6.

![Raw segmentation result](image1.png) ![Result after post-processing](image2.png)

Figure 6. Post-processing to remove holes in segmentation results.

3.4 Pseudo-Code of Segmentation Algorithm

Pseudo-code for the segmentation algorithm—including the search for a minimum in the feature space, the size range constraints—and the morphological pre-processing is given below.
3.5 Segmentation Performance Evaluation

It is desirable to evaluate the performance of the segmentation algorithm, in order to obtain an understanding of how long it would take to process a large dataset, and how much the final reported lake areas are impacted by inaccuracies in segmentation. The following two sections examine these questions.

3.5.1 Computational Speed

In order to improve the computational performance of the segmentation, we have used the method proposed by Yang and Algretsen for speeding up calculation of the Hu moments [5].
This method uses a discrete version of Green’s theorem to compute general image moments by traversing the boundary of an object in a binary image, instead of the entire image, thereby speeding up the calculation of the Hu moments, which are based on the general image moments. Since this method works by following the object boundary instead of visiting every pixel in the image, the number of pixels visited during moment computation varies as the square root of the image area, for a given object shape. By contrast, the naive method visits every pixel in the image, giving linear dependence on the image area. As would be expected, this improvement in computation time is most visible for the larger images that were processed. For example, one image of lake Michigan containing about 0.6 million pixels took 164 seconds to completely process using the fast method, compared with 459 seconds using the naive method. The moment computation itself took an average of 24 milliseconds with the speedup and 2.8 seconds without, an improvement by a factor of more than 100. These numbers are quite typical and, as they suggest, the entire dataset took on the order of hours to process. Even though a larger dataset would take days to process, this may well be acceptable, since the results might not be needed urgently, and large amounts of labor would be saved even if there is little speed improvement over manual analysis.

### 3.5.2 Segmentation Accuracy Evaluation

Segmentation performance can be evaluated through comparison with manually segmented masks. The error between the manual and automatic results is computed as illustrated in Figure 7, and is then normalized by the foreground area of the manual mask to produce a percent error. Let $I_M(r,c)$ and $I_A(r,c)$ denote the manual and automatically generated binary images, respectively. The formula for the percent error is given by equation (2).

$$\text{percent error} = \frac{\sum_r \sum_c I_M(r,c) \text{ xor } I_A(r,c)}{\sum_r \sum_c I_M(r,c)}$$

The average percent errors can then be computed by averaging the percent errors for each lake over the entire dataset.
Once percent errors have been calculated for each instance, a segmentation result can be classified as successful if the corresponding percent error is less than 50%. This is the standard used to determine success in Table 2, which shows segmentation performance for each of the five lakes. Clutter such as text and latitude/longitude lines presented a significant challenge. Other obstacles included thick lake boundaries accompanied by hashing, as well as thin, faded boundaries in some maps.

Table 2. Segmentation results: percent errors are computed relative to manually segmented masks

<table>
<thead>
<tr>
<th>Lake:</th>
<th>Erie</th>
<th>Huron</th>
<th>Michigan</th>
<th>Ontario</th>
<th>Superior</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Maps</td>
<td>36</td>
<td>34</td>
<td>38</td>
<td>37</td>
<td>37</td>
</tr>
<tr>
<td>Average Percent Error</td>
<td>73%</td>
<td>57%</td>
<td>59%</td>
<td>76%</td>
<td>50%</td>
</tr>
<tr>
<td>Number Successful</td>
<td>16</td>
<td>22</td>
<td>22</td>
<td>21</td>
<td>22</td>
</tr>
<tr>
<td>Average Percent Error for Successful Results</td>
<td>31%</td>
<td>34%</td>
<td>32%</td>
<td>35%</td>
<td>31%</td>
</tr>
</tbody>
</table>
4 MAP SCALE DETECTION

4.1 Introduction

If map scale is unknown, then only intensive parameters—that is, only parameters which do not depend on the actual size of geographic objects under consideration—can be used to compare different maps. This is because resolutions may vary greatly between different map images, so that a given number of pixels may represent very different physical distances in miles. In order to compare extensive parameters such as object area or boundary length between maps, we must first determine the physical distance (i.e. the distance in miles or another physical unit) represented by a single pixel. In the case of the current experiment, this information allows computation of lake areas in square miles, which can then be compared to one another and to the more precise modern figure. Scale detection, the task of determining this conversion, will be addressed in the sections that follow.

4.1.1 Manual Calibration Methods

To manually extract the scale of a map, one can select two points on the map such that the distance between them is known, and compute the ratio of this distance to the distance in pixels obtained from the digital image. An example is shown in Figure 8, where the north-south extent of the Chesapeake Bay is used for calibration. The distance between two cities, such as Boston and New York, can also be used, since cities provide relatively localized points which are expected to be consistent across different maps.

Another approach is to examine the explicit legends which are present on some maps. These legends, of course, provide an ideal characterization of the cartographer’s intention regarding the scale of the map, but they are not always present, and may be expressed in archaic distance units.

Finally, map scale can be determined by examining the dashed neatline that runs along the border of many maps, and counting the number of dashes between successive intersections with latitude or longitude lines (see Figure 9). Due to mapmaking conventions identified by
examining the dataset under study, this number corresponds uniquely to the desired scale (the details of this correspondence will be described in Section 4.7).

![Image](image_url)

Figure 8. Manual calibration using Chesapeake Bay (left), boundary extraction (right).

![Image](image_url)

Figure 9. Intersections (circled in orange) of the dashed neatline with successive longitude lines.

### 4.1.2 Translation to Algorithmic Steps

The first two methods described in the preceding section (point-to-point distance and map legend based) are ill-suited to automation. The method using two points for calibration suffers from the drawback that a point may have an ill-defined location (such as the source or outlet of the Chesapeake Bay), may be absent from any given map (such as a city which had not yet been founded), or may have different labels depending on the time and the source of the map (French vs. British). The method relying on map legends is also problematic because many maps do not have such legends. Additionally their locations and format are not standardized, which complicates automated detection. The physical units they employ may vary as well. The third method, by contrast, is better suited to automation since dashed neatlines are present in most
maps and tend to have similar positions and formats. Accordingly, this is the method that was chosen to automate scale detection.

Neatline analysis can be broken down into several constituent steps including:

- Map boundary area selection to choose which side of the map to analyze
- Line detection to find the position of the neatline
- Line classification to distinguish the dashed neatline from any parallel solid lines
- Dash-length calculation to find the length in pixels of a single stripe of the dashed line
- Transversal detection to find intersections with latitude or longitude lines

The ratio of the distance in pixels between successive intersections to the length of a stripe in pixels will then produce the desired number. We will examine each of the steps in detail in the sections that follow. A system block diagram showing the relationships among the subsystems is shown in Figure 10.

![Figure 10. Map scale detection system block diagram.](image)
4.2 Map Boundary Extraction and Selection

Before attempting to find the neatline, it is useful to extract the four map boundaries in order to shrink the search space and reduce clutter. In order to do this, simple intensity thresholding is used to locate the end of the uniform map background (see the yellow arrows in Figure 11). The four boundaries are then cropped (see red lines in Figure 11) so that the end of the background is centered in each region.

Since only one boundary need be considered in the final analysis, the next step is to select the boundary with the least clutter. This can be accomplished by summing intensities perpendicular to each boundary, and taking the variance of the resulting signal. Since this signal would ideally be almost constant (each boundary should have roughly uniform cross-sections along its length), the boundary yielding the lowest variance is selected.

![Figure 11. Intensity thresholding in boundary extraction.](image)

4.3 Line Detection

Once a boundary area has been selected and segmented, the next task is to locate the dashed neatline within this boundary. The purpose of this step is to obtain an image of the dashed neatline alone, which is required for the dash length calculation subsystem, and to obtain an
image of the immediate neighborhood of the neatline, which is required by the transversal
detection subsystem.

4.3.1 Line Detection Algorithm

Line detection can be automated by summing along the length of the boundary in order
to determine a signal such as that shown by the green line in Figure 13. Observe that the valleys in
this signal correspond to the locations of the two vertical lines: the solid dark line and the dashed
neatline. In order to detect these valleys, a moving average (red line) is used, with a moving
threshold a fixed number of standard deviations below (blue line). Each contiguous region for
which the signal is below the threshold is hypothesized to correspond to a line.

A block diagram for the line detection subsystem is shown in Figure 12. Since lines in the
boundary image may be rotated slightly from the vertical, it is necessary to de-rotate the image
prior to taking column sums. This step is required because if the length of the line far exceeds its
width, then a rotation by even a slight angle can result in significant spreading of the valleys in
the final signal, which is undesirable for detection purposes. Derotation is discussed in more
detail in the next section. Next, the width of a valley is estimated by thresholding the signal half
way between its mean and minimum, and the result is used to compute the window size for
filtering (and potentially for adaptive thresholding). The valley depth estimation block estimates
the depth of the deepest valley so that the parameter $\alpha$ can be chosen relative to this depth. Let
the estimated valley width be $v$. We use the lowpass filter defined by $h(n) = \left\lfloor \frac{1}{[v/5]} \right\rfloor_{n=1}^{N}$. Let the
length-$N$ column sum signal be denoted by $s(n)$. Then

$$\alpha = \frac{\mu_s - \min_n s}{5\sigma_s}$$

where $\mu_s$ and $\sigma_s$ are the mean and standard deviation of $s(n)$, respectively. Although $w$
could be computed as a function of $v$, it was found that a value of $N/10$ works in practice. Note that the constants in the
preceding equations are empirical, and were chosen on the basis of observed results. We observe
in Figure 12 that the candidate lines produced by the algorithm correspond to the three vertical
lines present in the input image.
Figure 12. Line detection block diagram.

Figure 13. Windowed thresholding for line detection.
4.3.2 Boundary Image Derotation Overview

Prior to performing thresholding in order to determine line positions, it is important to ensure that neatline images are oriented so that all lines are as parallel as possible to image borders. The rationale for this is illustrated in Figure 14, which shows a slightly rotated neatline image (left), and the same image after derotation (right). From the corresponding column sum signals shown below each image, it is clear that derotation significantly increases (negative) peak sharpness, giving better resolution in determining line positions and boundaries, which is beneficial in future stages of the algorithm.

Figure 14. Derotation applied to the neatline image at left achieves sharper peaks in the column sum signal.
4.3.3 Derotation Algorithm Description

The derotation subsystem works by examining the same column sum signals whose quality it strives to improve. Specifically, it searches for the angle of rotation which will yield the smallest minimum in the column sum signal (which is expected to correspond to the angle yielding the sharpest peaks). The initial image is assumed to have near-perfect orientation, so that only a small range of angles needs to be searched. Nevertheless, the correct angle must be determined with high precision, especially for images in which the length of the neatline greatly exceeds its width. For such images, even finding the correct angle to the nearest degree was often found not to suffice in practice. Because of this, a small step-size must be used when searching for the best derotation angle, which increases computation time. Therefore, a hierarchical search is used, in which the correct angle is localized by successive searches at different levels of precision. For example, the best angle may first be found to the nearest degree, and then to the nearest tenth of a degree within this smaller range, and so on. This approach assumes that the function mapping angles to column-sum minima will be well-behaved in the sense that its global minimum will be located within a one-step-size range around the minimum found at a given level and using a given step-size. Although there is no guarantee that this will be true, in practice the algorithm performed well as long as the step-size was small enough. Additionally, any discrete search for the extrema of a continuous function for which a closed-form expression is not known will necessarily make similar assumptions.

The pseudocode for the derotation algorithm is given below.
4.4 Line Classification

Given the lines found in the previous step, the next task is to decide which one is the sought-after neatline. To do this, we designed a classifier to distinguish solid lines from dashed lines. The first step in the classification is to generate signals such as those shown in Figure 15. The signals represent sums of pixel intensities perpendicular to the lines as a function of position along the lines. Note that the signal corresponding to the dashed line resembles a noisy square-wave as one might expect, while the other signal has no discernable structure. In order to differentiate between these two signals, we compute their autocovariances as shown in Figure 16. The energy (squared area) of the autocovariance from the origin to the first zero-crossing divided by the total energy provides a feature which has been found experimentally to provide good separation between the classes. Let \( l \) be the image of the line, \( x(n) \) the length \( N \) signal of sums perpendicular to the line, \( \mu_x \) the mean of \( x \), \( c(n) \) the autocovariance, and \( f \) the feature. Suppose
the line image is oriented horizontally, without loss of generality, and has width $W$. Then the feature is defined by equations (3).

$$f = \frac{\sum_{k=0}^{x} |c(k)|}{\sum_{k=0}^{N-1} |c(k)|}$$

$$c(n) = \sum_{i=-\infty}^{\infty} \bar{x}(l) \bar{x}(l + n), n \in \{0, 1, \ldots, N - 1\}$$

$$x(n) = \frac{1}{W} \sum_{r} I(r, n)$$

$$z = \min_{c(n)=0} n, \bar{x}(n) = x(n) - \mu_x$$

From the plots in Figure 16, it is clear that this ratio should be much lower for dashed lines than for solid lines in general.

![Solid Line](image1.png)
![Dashed Line](image2.png)

Figure 15. Column-sum signals for line classification.
4.5 Dash Length Calculation

After the classification step, the neatline has been segmented from the image. One of the two quantities required for the final calculation is the number of pixels per dash. This can be determined by again generating a signal of sums perpendicular to the line after filtering to remove noise, and thresholding the signal about its mean. These steps are shown at left in Figure 18 (p. 30). Contiguous sequences of ones and zeros in the resulting binary signal are taken to be dashes, and their median length can be computed, giving an estimate of the true dash length. The median is used in preference to the mean due to its better rejection of small numbers of extreme outliers, which may be caused by localized artifacts in the line image. A block diagram of the dash length calculation system is shown in Figure 17.
4.5.1 Adaptive Thresholding for Dash Length Estimation

Since local intensity can vary over the length of the neatline, thresholding about the global mean may be problematic, as can be seen at the right end of the signal at left in Figure 18. Due to locally higher intensity in this region, the threshold barely contacts a local minimum in the signal, producing noise in the corresponding binary sequence shown underneath. In order to obtain a better estimate, it is desirable to threshold the signal about a local mean. The local mean can be computed using a moving average with window-size equal to one period of the square wave so that intensity variations at the fundamental frequency of the signal cancel out. Note that the window-size used in this second pass is computed using the dash length estimated in the first pass. The adaptive thresholding approach is illustrated in the right half of Figure 18, along with the resulting binary sequence, which can be observed to approximate dash boundaries more precisely. The median dash length calculated from this second pass is used as the final estimate.

Consider again the block diagram in Figure 17. Note that since the raw signal generated by summing perpendicular to the neatline tends to be very noisy, as shown in the output of the “Sum Along Rows” block, the signal is preprocessed using a median filter. Median filtering was chosen due to its edge-preserving property, which leads to better localization of dash boundaries than could be expected if a linear lowpass filter were used. Since the number of pixels per dash is not yet known at the time of preprocessing, the window size used in the filtering is refined in a similar manner to the window size used in adaptive thresholding. That is, an initial guess is made, and the resulting estimate is used to refine this guess. Choosing the window size of the filter is important because a size which is too small will fail to provide adequate noise rejection, while a size which is too large will distort the underlying desired signal; in practice, a window size of one half the length of a dash was used.
The second quantity required for the calculation of the final result is the distance in pixels between two successive intersections of latitude or longitude lines with the neatline. These intersections can be located by moving a point along the length of the neatline. For each position of the point, pixel intensities can be summed along lines with a range of angles passing through that point. If the current position of the point corresponds to an intersection, there will exist an angle such that the sum of intensities along a line at that angle is very low (i.e. when this line aligns with the latitude or longitude line involved in the intersection). By finding the minimum over all angles in the specified range of the line-sums of intensities as a function of position along the neatline, we get a signal such as that shown in Figure 19.
The mathematical description of this signal is given below, where \( I \) is the image, \( (r, c) \) is the point at which the line-sum is evaluated, \( \theta \) is the angle of the line, which varies between \( \theta_{\text{min}} \) and \( \theta_{\text{max}} \), and the factor of \( \cos \theta \) in the denominator is intended to normalize the sum so that equal weight is given to sums at any angle.

\[
\text{sig}(r, c) = \min_{\theta \in \theta_{\text{min}} \ldots \theta_{\text{max}}} \sum_{(r', c') \in I} \frac{I(r - \lfloor (c' - c) \tan \theta \rfloor, c')}{\cos(\theta)}
\]  

Note that since we fix \( r \) to correspond to the center of the neatline, the signal becomes a function of \( c \) only. Applying the same method as was used for line detection, we obtain the positions of the intersections, and hence the distances between them, which can be averaged to produce the desired result.

Finally, using the fact that intersections with latitude or longitude lines always occur at dash boundaries, combined with the fact that these boundaries are known approximately from the dash-length calculation step, we can increase computational efficiency and robustness to spurious peaks caused by clutter by only calculating the minimum line-sum signal in the neighborhoods of dash boundaries. This refinement is responsible for the periodic flat intervals visible in Figure 19.
Figure 19. Signal for transversal detection.

4.7 Lookup Table for Converting Number of Dashes to Miles per Pixel

After calculating the length of a single dash and the distance between two successive intersections (both in pixels), we compute the ratio of these quantities rounded to the nearest integer. This ratio gives the number of dashes between two successive intersections. Due to cartographic practices, this number maps to the number of degrees between two successive intersections of the neatline with graticule line; a complete set of mappings for the dataset under study is given in Table 3. The mappings in Table 3 were derived from empirical observations made by collaborating humanists. Once this is known, the scale can be determined using the length of a dash in a North-South running neatline combined with the fact that one degree of latitude corresponds to approximately 69 miles.

Let $S$ be the scale in miles per pixel, $l_D$ the length of a dash in pixels, $l_I$ the distance between intersections in pixels, $N$ the number of dashes between intersections, and $d_I$ the number of degrees between intersections. Mathematically, the final scale calculations are performed as
shown in equations (5) below. Observe that the units of \( S \) come out to be miles per pixel, as desired.

\[
N = \frac{l_i}{l_d}, d_i = \text{LookupTable}(N) \\
S = \frac{69 \left[ \frac{\text{miles}}{\text{degrees}} \right] d_i[\text{degrees}]}{Nl_d[\text{pixels}]} \tag{5}
\]

Table 3. Mapping between number of dashes between intersections and number of degrees between intersections

<table>
<thead>
<tr>
<th>Dashes between Intersections ((N))</th>
<th>Degrees between Intersections ((d_i))</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>10</td>
<td>5</td>
</tr>
<tr>
<td>12</td>
<td>2</td>
</tr>
<tr>
<td>8</td>
<td>4</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
</tr>
</tbody>
</table>

In order to test the validity of this method of scale estimation, manually and automatically computed scales were compared for 13 maps selected from among the successful results of the automatic algorithm (Table 4). Automatic results were found to differ from manual results by an average of 14\%, suggesting that the automatic results are meaningful. Note that it is not clear a priori which method yields a more accurate estimate of the intended scale, since manual results are subject to sources of error, including ill-defined positions of the two points used in calibration.
Table 4. Comparison of automatically and manually calculated scales (in miles/pixel)

<table>
<thead>
<tr>
<th>Automatic</th>
<th>Manual</th>
<th>Percent Difference (relative to manual)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.56</td>
<td>0.42</td>
<td>33.6</td>
</tr>
<tr>
<td>0.27</td>
<td>0.30</td>
<td>8.7</td>
</tr>
<tr>
<td>0.55</td>
<td>0.52</td>
<td>5.3</td>
</tr>
<tr>
<td>0.46</td>
<td>0.46</td>
<td>0.7</td>
</tr>
<tr>
<td>1.08</td>
<td>1.16</td>
<td>7.1</td>
</tr>
<tr>
<td>0.45</td>
<td>0.59</td>
<td>24.1</td>
</tr>
<tr>
<td>0.80</td>
<td>1.12</td>
<td>28.4</td>
</tr>
<tr>
<td>0.23</td>
<td>0.25</td>
<td>8.0</td>
</tr>
<tr>
<td>0.23</td>
<td>0.23</td>
<td>2.0</td>
</tr>
<tr>
<td>0.46</td>
<td>0.46</td>
<td>0.0</td>
</tr>
<tr>
<td>0.72</td>
<td>0.51</td>
<td>40.9</td>
</tr>
<tr>
<td>0.80</td>
<td>0.76</td>
<td>5.6</td>
</tr>
<tr>
<td>0.34</td>
<td>0.28</td>
<td>20.2</td>
</tr>
</tbody>
</table>

4.8 Map Scale Detection Performance Evaluation

Map scale detection accuracy can be evaluated using the following standard: if the calculated number of dashes between two intersections rounds to the correct integer, then the algorithm has succeeded; otherwise, it has failed. Out of 25 maps which were used to test the scale detection algorithm, 16 (64%) were successful. Factors which caused the other maps to fail included severe clutter, which hampered transversal detection, and extremely low contrast, which hampered line detection.
5 UNCERTAINTY ANALYSIS OF MAP SCALE

It is desirable to attempt an analysis of the uncertainties involved in scale detection. The purpose of the analysis is to estimate for each computed scale the degree to which that scale reflects the true scale intended by the cartographer. In the case of scale detection, it turns out that this estimate can be naturally broken down into two separate quantities: (1) the probability of catastrophic failure, and (2) the uncertainty given that catastrophic failure has not occurred.

The reason for this is the lookup table used to obtain the number of degrees per dash. Even if there was significant error in the estimate of the number of dashes between successive intersections, this error vanishes as long as the estimate rounds to the correct integer, so that the appropriate element in the lookup table is used. On the other hand, if the error is so great that the estimate does not round to the correct integer, then a spurious element in the lookup table is accessed, causing the final result to be meaningless. Hence, errors in the inputs to the lookup table either cause no error in the output, or lead to catastrophic failure, so that it is meaningful to associate a probability of catastrophic failure with the computed output. Given that such failure has not occurred, the only source of error in the final result will be uncertainty in the length of a dash.

Note that we do not need to consider failures in the boundary extraction/selection, line detection, or line classification systems, since the algorithm halts and signals error if no dashed line is found, so that such failures are always detected. The only exception to this would be a false-positive in the line classifier, which would allow the algorithm to proceed beyond this point without reporting failure. Such a false-positive, however, has never been observed, so the probability of such a situation is assumed to be negligible.

The relationships among the variables that contribute to failure and uncertainty in the scale estimation system can be represented by the graph shown in Figure 20. The meanings of the variables shown are given on p. vi, and the relationships shown will be explored in the sections that follow.
5.1 Variables Affecting Transversal Detection

In the transversal detection subsystem, we obtain a vector of ‘inter-transversal’ lengths using the algorithm described previously. In order to evaluate uncertainty in this calculation, one might attempt to compute the variance of this vector. The problem with this approach is that there are typically only a small number of intersections between the neatline and graticule lines in a given boundary image, so that the variance may not be a meaningful measure of uncertainty. For example, one might encounter a case in which only two intersections are detected, giving only one length measurement. Assuming that one of these is a false positive, we might report an uncertainty of zero in a case where the algorithm has failed completely. Because of this, a better approach is to examine common causes of failure in this subsystem, and attempt to detect their presence or absence in a given image. Section 5.2 examines failure caused by the presence of clutter in the vicinity of the neatline, while Section 5.3 examines failure due to poor contrast of graticule lines relative to the background.
5.2 Background Clutter

One of the primary mechanisms of catastrophic failure is clutter impinging upon the neighborhood of the neatline. This clutter often takes the form of text, rivers, or other objects which may lead to false positives in the transversal detection subsystem. Figure 21 shows an example of clutter in the neighborhood of the neatline (top image) vs. the absence of such clutter (bottom image). In the cluttered image, artifacts such as rivers may produce false positives in the transversal detection subsystem. If these false positives are too numerous, then the wrong entry in the lookup table may be used, leading to catastrophic failure.

In order to predict whether failure will occur in a given case due to clutter, it is first necessary to quantify the amount of clutter present in the image of the neatline neighborhood. This can be done by considering the vector of means perpendicular to the neatline (i.e. along columns in the images below), and taking the variance of this vector. For a horizontally oriented neatline, the score is given by equation (6).

\[
\sigma_{cl}^2 = \text{var} \left( \sum_r I(r, c) \right)
\]  

(6)

Since the vector should be nearly constant in the absence of significant clutter, a higher variance indicates larger amounts of clutter. As an example, the score of the top image in Figure 21 according to this measure was approximately ten times that of the bottom image.

Figure 21. Top: high clutter (score = 310.2), bottom: low clutter (score = 31.0).
Since it is not clear how to evaluate the impact of this score analytically, an experiment was conducted in which clutter scores were calculated for 22 maps and the success or failure of each was recorded for the transversal detection system only. From the results, shown in Figure 22, it is clear that all but one of the maps with clutter score in excess of 200 failed, while all but one of those with scores lower than 200 succeeded. Therefore, we can estimate the relationship of the clutter score to the probability of catastrophic failure as shown in Table 5, where $T_{CL}$ is the threshold beyond which failure is predicted (200 in this case). These estimates could be improved by performing the same experiment on a larger dataset.

Table 5. Estimated relationship between clutter score and failure probability

<table>
<thead>
<tr>
<th>Clutter Score Condition</th>
<th>$P(success)$</th>
<th>$P(failure)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\sigma_{cl}^2 \geq T_{CL}$</td>
<td>$P(success</td>
<td>\sigma_{cl}^2 \geq T_{CL}) \approx 1/7$</td>
</tr>
<tr>
<td>$\sigma_{cl}^2 &lt; T_{CL}$</td>
<td>$P(success</td>
<td>\sigma_{cl}^2 &lt; T_{CL}) \approx 14/15$</td>
</tr>
</tbody>
</table>

Figure 22. Use of clutter score to predict failure in transversal detection on 22 maps (15 successful, 7 failed).
5.3 Graticule Line Contrast

A second factor that can contribute to failure in the transversal detection system is the contrast between graticule lines and the map background. If the contrast is poor, then the corresponding peaks in the signal of the type discussed in Section 4.6 may not be detected by the adaptive threshold. Examples of low and high contrast and the effect on the corresponding detection signals are shown in Figure 23. Another situation which has similar effects occurs when graticule lines may not have low contrast, but have less extent within the chosen neighborhood, thereby resulting in shallower peaks as though contrast had been low.

Figure 23. Low contrast (top) and high contrast (bottom) graticule lines and corresponding detection signals.
Contrast between graticule lines and the background (or the extent of such lines where applicable) can be measured by considering the depth of peaks in the corresponding transversal detection signal below the chosen threshold. If a given peak extends far below the detection threshold, then it is reasonable to be more confident that this peak represents something significant, which could either be a graticule line or high-contrast clutter. If a peak barely extends beyond the threshold, on the other hand, then the chance of it being due to noise is greater. Additionally, such peaks may be symptomatic of a poorly chosen threshold, and one might suspect that even if the peak does correspond to a graticule line, then other slightly less prominent peaks may have been missed. Therefore, the existence of peaks which do not extend far beyond the threshold may indicate the presence of false positives and/or missed detections in the system.

Let $I_j$ be the $j$-th interval in which a given transversal detection signal of the kind shown in green in Figure 23 lies below the threshold, $\sigma$ the standard deviation of the signal, and $thresh(c)$ the threshold evaluated at column $c$. As in Section 4.3.1 $\alpha$ denotes the number of standard deviations between signal and threshold. We define the contrast score of the signal as in equations (7).

$$
\epsilon_{CO}^{DL} = \text{median}_{j}\left\{\frac{\mid sig(c_j^*) - thresh(c_j^*)\mid}{\alpha \sigma}\right\}
$$

$$
c_j^* = \arg\min_{c \in I_j} sig(c)
$$

(7)

Note that $c_j^*$ is the column in the interval $I_j$ at which the signal attains its minimum value.

The contrast score can be plotted against the outcome (success or failure) as in the case of clutter. Such a plot is given in Figure 24 for the same dataset as before. Here again, the blue data points denote success, while the red data points denote failure. The three yellow data points represent cases in which clutter was the major cause of failure (the four red data points often exhibited both clutter and low contrast). The four false positives, on the other hand, are due to the fact that the system can sometimes produce the correct result in spite of low contrast. This is largely due to the fact that the median is used to calculate the reported transversal spacing so that a small
number of missing or spurious peaks may not impact the end result. Similar to the case when the effects of clutter were analyzed, we can define a threshold, $T_{CO}$, below which we predict failure. Table 6 gives failure probabilities for scores above and below $T_{CO}$. As in Section 5.2, we note that additional data would be needed to derive a more accurate threshold, although the same methodology could be used.

Table 6. Estimated relationship between contrast score and failure probability

| $e_{CO}^{DL} \geq T_{CO}$ | $P(\text{success}|e_{CO}^{DL} \geq T_{CO}) \approx 10/11$ | $P(\text{failure}|e_{CO}^{DL} \geq T_{CO}) \approx 1/11$ |
|---------------------------|------------------------------------------------|------------------------------------------------|
| $e_{co}^{DL} < T_{CO}$   | $P(\text{success}|e_{CO}^{DL} < T_{CO}) \approx 4/7$ | $P(\text{failure}|e_{CO}^{DL} < T_{CO}) \approx 3/7$ |

Figure 24. Use of contrast score to predict failure in transversal detection on 22 maps (15 successful, 7 failed).

5.4 Synthesis of Transversal Detection Failure Modes

Probabilities associated with the two failure modes discussed in Section 5.2 and Section 5.3 can be combined by plotting the clutter and contrast scores against one another as shown in Figure
25. From this figure, we can derive the combined estimated failure probabilities given the contrast and clutter scores. These probabilities are given in Table 7.

![Success or Failure vs. Clutter and Contrast Scores](image)

Figure 25. Success or failure of transversal detection vs. clutter and contrast scores.

Table 7. Combined failure probabilities taking clutter and contrast scores into account

<table>
<thead>
<tr>
<th>Condition</th>
<th>( \sigma_{CL}^2 &lt; T_{CL} )</th>
<th>( \sigma_{CL}^2 \geq T_{CL} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \epsilon^{DL}<em>{CO} \geq T</em>{CO} )</td>
<td>( P(failure</td>
<td>\epsilon^{DL}<em>{CO} \geq T</em>{CO}, \sigma_{CL}^2 &lt; T_{CL}) \approx 1/10 )</td>
</tr>
<tr>
<td>( \epsilon^{DL}<em>{CO} &lt; T</em>{CO} )</td>
<td>( P(failure</td>
<td>\epsilon^{DL}<em>{CO} &lt; T</em>{CO}, \sigma_{CL}^2 &lt; T_{CL}) \approx 0/5 )</td>
</tr>
</tbody>
</table>

5.5 Variables Affecting Dash Length Calculation

Several variables contribute to error in dash length calculation including resolution, contrast, sharpness, and inherent imprecision. We will denote upper bounds on these error contributions by \( \epsilon_R \) (resolution), \( \epsilon_{CO} \) (contrast), \( \epsilon_{SH} \) (sharpness), and \( \epsilon_L \) (inherent dash length) respectively, and the upper bound on the overall error in scale \( \epsilon^C \). Thus, we have \( \epsilon^C \leq \epsilon_R + \epsilon_{CO} + \epsilon_{SH} + \epsilon_L \).
Although these contributing variables (with one exception) need not be estimated in order to perform uncertainty analysis, as will be explained below, it is nevertheless instructive to consider their impact on the dash length calculation subsystem, so that the causes of uncertainty are better understood.

5.6 Resolution

Since the map images under study are digital, the accuracy of any calculated length is necessarily limited by the image resolution. As shown in Figure 26, two analog boundaries between light and dark regions (such as the boundaries between two neighboring dashes) may be discretized to the same digital boundary even though their exact positions differ by as much as one pixel width. We denote the maximum error due to this variable by $\epsilon_R$. Clearly, we have $\epsilon_R = 1$.

![Figure 26. Uncertainty due to finite resolution: analog dash boundaries (left), digital boundary (right).](image)

5.7 Inherent Imprecision

Even given an image with distinct and well-localized boundaries, the lengths of different dashes may still differ due to the fact that they differ in the physical copy of the map as well. This may be due to limitations in the skill of the cartographer, or in the precision of his drawing tools. Because we are attempting to measure the length of a dash as intended by the cartographer, this variable, denoted by $\epsilon_L$, represents an inherent source of uncertainty in our ability to do so, since it is impossible to know which dash length should be taken as the true one. This idea is illustrated in Figure 27.
Contrast

The contrast between light and dark dashes can affect the ability of the algorithm to successfully estimate the length of a dash, since it increases susceptibility to noise and to local brightness variation in the image. As shown in Figure 28, a one-dimensional signal derived by summing perpendicular to a line with low contrast has less difference between its peaks and troughs (compare to Figure 18). Because of this, variations in the envelope of the signal (which corresponds to local brightness in the image) may have comparable magnitude to variations in intensity between light and dark dashes, which is problematic for thresholding. Additionally, noise such as what can be seen near the middle of the signal below becomes more of a problem when the contrast is lower (this can be viewed as a lower signal-to-noise ratio). We denote this variable by $\epsilon^{DL}_{CO}$, where the superscript serves to distinguish it from the contrast score for transversal detection.

Figure 28. Uncertainty due to low contrast between light and dark dashes: example 1D signal.
5.9 Sharpness

Since transitions between light and dark dashes may not be instantaneous, the exact position of the boundary between a light and a dark dash may be difficult to define precisely. Although it is reasonable to define transition as the point when pixel intensity reaches half way between the intensities of light and dark dashes, even these intensities are not constant, and there is no guarantee that the thresholding algorithm will follow this criterion. It is clear that this variable can introduce an error of at most twice the width of a transition region (see Figure 29).

![Figure 29. Uncertainty due to ill-defined transition: dashed line and corresponding 1D signal.](image)

5.10 Dash Length Uncertainty Measurement

Fortunately, in the case of the dash length estimation subsystem, we have access to a significant number of different dash length measurements in the form of the vector of dash boundaries obtained using the adaptive thresholding described previously. This allows us to directly estimate $e^C$ by computing the variance, $\sigma_{DL}^2$, of this set of measurements. In order to obtain an estimated upper bound on the error in computed dash length, we have plotted the absolute difference between automatically and manually calculated dash lengths vs. this variance.
It can be seen from the plot in Figure 30 that twice the standard deviation of dash lengths gives a good upper bound on the empirical error for the 19 instances used in the experiment. Since each dash length measurement is a measurement of a discrete quantity, we must also take $\epsilon_R$ into account. This can be accomplished by using $2\sigma_{DL} + 1$ as the bound on dash length error, instead of $2\sigma_{DL}$. We will denote this bound by $\epsilon'_{DL}$.

Once an upper bound on the dash length uncertainty is known, it is still necessary to translate this into an upper bound on the scale uncertainty, $\epsilon^C$. Recalling that the calculated map scale is denoted by $S'$ and $l'_d = l_d + \epsilon_{DL}$ is the calculated dash length with $S$, and $l_d$ being the true scale and dash length, respectively, and $\epsilon_{DL}$ the true error in dash length, this can be done as shown by equations (8).
\[
\epsilon^C = |S' - S| = \left| \frac{69 \, d_i}{Nl'_d} - \frac{69 \, d_i}{Nl_d} \right| = \frac{69 \, d_i}{N} \left| \frac{1}{l'_d} - \frac{1}{l_d} \right| = \frac{69 \, d_i}{N} \left| \frac{l_d - l'_d}{l_d l'_d} \right| \\
\leq \frac{69 \, d_i}{N} \frac{\epsilon'_{DL}}{l'_d(l'_d - \epsilon'_{DL})} \\
\epsilon'_{DL} = 2\sigma_{DL} + \epsilon_R = 2\sigma_{DL} + 1
\]

We assume that \( l'_d > \epsilon'_{DL} \), which should hold for any interesting cases. The case in which this does not hold will be considered in Section 5.11.

In order to illustrate the bound derived above, a second experiment was performed, in which automatically calculated scales were compared with scales calculated from manual measurements. For a dataset of 20 maps (all selected so that \( l'_d > \epsilon'_{DL} \)), the absolute error between these quantities, \(|S' - S|\), was plotted vs. the bound (see Figure 31). In this plot, \( 2\sigma_{DL} \) is used instead of \( \epsilon'_{DL} \), since \( S \) was also calculated using measurements from digitized images. Since the data points all lie above the red line, we see that the bound holds for this dataset. Note that two points far above the line were omitted for clarity of presentation.
Figure 31. Absolute difference between manual and automatic scale vs. derived error bound for 20 maps. The red line corresponds to the bound derived in equations (8), with $2\sigma_{DL}$ used in place of $\varepsilon'_{DL}$.

Since all of the quantities used to compute the bound ($d_i, \sigma_{DL}, N, l'_d$) can be calculated at runtime, this gives a method for estimating an upper bound on the uncertainty in calibration, given that catastrophic failure has not occurred.

### 5.11 Catastrophic Failure due to Dash Length Calculation Error

It is possible for error in dash length calculation to cause catastrophic failure if the calculated length is far enough from the true value to cause $N$ to be calculated incorrectly. Starting from the previously stated formula $N = \frac{l_i}{l_d}$, and neglecting the error in $l_i$, we can use reasoning analogous to that shown above to conclude that the error in $N$ due to error in $l_d$ is given by equation (9).
\[ \varepsilon^N = |N - N'| \leq \frac{\varepsilon_{DL}'l_i}{l'_d(l'_d - \varepsilon_{DL}')}, \]  

(9)

where \( N' \) is the estimated number of dashes between two intersections before rounding. Hence, we arrive at the condition for catastrophic failure given in equation (10).

\[ \frac{\varepsilon_{DL}'l_i}{l'_d(l'_d - \varepsilon_{DL}')} \geq 0.5 \]  

(10)

This condition would cause an incorrect choice of \( N \), so that an incorrect element in the lookup table would be accessed, making \( d_i \) invalid.

The preceding analysis assumes that \( l'_d > \varepsilon_{DL}' \). When this is not the case, it means that the maximum uncertainty in dash length has exceeded the dash length itself. In this case, it is clear that the result should not be trusted. Therefore, we report catastrophic failure whenever \( \varepsilon_{DL}' > l'_d \).

### 5.12 Catastrophic Failure and Uncertainty Synthesis

Combining the results derived in the previous sections, we predict catastrophic failure if one of the following conditions in equations (11) is satisfied:

\[ \varepsilon_{DL}' > l'_d \]

\[ \frac{\varepsilon_{DL}'l_i}{l'_d(l'_d - \varepsilon_{DL}')} \geq 0.5 \]  

(11)

If neither condition is satisfied, we use to Table 7 to compute the failure probability due to errors in transversal detection.

Given that catastrophic failure does not occur, we predict an upper bound of the error in \( S \) to be as given in equation (12).

\[ \frac{69 \ d_i \varepsilon_{DL}'}{Nl'_d(l'_d - \varepsilon_{DL}')}, \]  

(12)


6 RESULTS

Using the segmentation results which were classified as successful above, lake areas were calculated, and the errors of these calculated values relative to the modern estimates were averaged for each lake. Errors were calculated as the (absolute) difference between the calculated figure and the modern figure, as a percentage of the modern figure. The results are given in Table 8, along with other pertinent data such as mean creation dates (we expect maps created at later dates to be more accurate than earlier ones). Note that these results were generated using manual scale detection, so that the analysis given in Chapter 5 is not applicable. The errors reported in the table should be seen as a result of errors in segmentation and manual scale estimation as well as inaccurate knowledge on the part of the cartographer, the latter being the variable of interest. While we see from Table 2 that segmentation errors were often substantial, one would expect these errors (as well as errors in scale estimation) to impact British and French maps approximately equally, so that average percent errors can still be meaningfully compared given enough data points.

Table 8. Area calculation results

<table>
<thead>
<tr>
<th>Lake</th>
<th>Erie</th>
<th>Huron</th>
<th>Michigan</th>
<th>Ontario</th>
<th>Superior</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Maps (Overall)</td>
<td>16</td>
<td>22</td>
<td>22</td>
<td>21</td>
<td>22</td>
</tr>
<tr>
<td>Number of Maps (French)</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>9</td>
<td>10</td>
</tr>
<tr>
<td>Number of Maps (British)</td>
<td>7</td>
<td>12</td>
<td>11</td>
<td>12</td>
<td>12</td>
</tr>
<tr>
<td>Mean Creation Date (Overall)</td>
<td>1773</td>
<td>1769</td>
<td>1767</td>
<td>1768</td>
<td>1774</td>
</tr>
<tr>
<td>Mean Creation Date (French)</td>
<td>1762</td>
<td>1749</td>
<td>1746</td>
<td>1746</td>
<td>1758</td>
</tr>
<tr>
<td>Mean Creation Date (British)</td>
<td>1788</td>
<td>1785</td>
<td>1788</td>
<td>1785</td>
<td>1788</td>
</tr>
<tr>
<td>Actual Area (square miles)</td>
<td>9940</td>
<td>23010</td>
<td>22400</td>
<td>7540</td>
<td>49305</td>
</tr>
<tr>
<td>Average Percent Error (Overall)</td>
<td>99%</td>
<td>117%</td>
<td>70%</td>
<td>97%</td>
<td>83%</td>
</tr>
<tr>
<td>Average Percent Error (French)</td>
<td>102%</td>
<td>148%</td>
<td>97%</td>
<td>146%</td>
<td>144%</td>
</tr>
<tr>
<td>Average Percent Error (British)</td>
<td>94%</td>
<td>92%</td>
<td>44%</td>
<td>60%</td>
<td>32%</td>
</tr>
</tbody>
</table>

Figure 32 shows a plot of calculated areas of Lake Ontario vs. time, thereby giving an example of how the area data can be analyzed in order to compare British and French geographic knowledge over time. From this plot, for example, we have inferred that although the French
occupied the Great Lakes region sooner, French maps do not indicate more accurate depictions of Lake Ontario than British maps. Figure 33 through Figure 36 contain similar graphs for the other four Great Lakes. It would be desirable in the future to test the algorithms on a larger dataset in order to obtain more statistically meaningful results.

![Calculated Areas Over Time](image)

Figure 32. Calculated areas of Lake Ontario in French and British historical maps over time. Note that this graph was generated from a previous run of the algorithm with 14 British maps and 12 French maps.
Figure 33. Calculated areas of Lake Superior in French and British historical maps over time.

Figure 34. Calculated areas of Lake Erie in French and British historical maps over time.
Figure 35. Calculated areas of Lake Huron in French and British historical maps over time.

Figure 36. Calculated areas of Lake Michigan in French and British historical maps over time.
7 CONCLUSION

In automating extraction of cartographic information from images of historical maps, we had to overcome two automation obstacles: segmenting cartographic objects of interest and estimating map scale to report characteristics in physical units, as well as the problem of estimating upper bounds on the uncertainty of calculated scales.

The segmentation problem was approached using a template-based region growing algorithm, which searches for an object in the target image which is similar to a template in the 7-dimensional space of Hu moments. Future work on the segmentation subsystem should focus on increasing the accuracy of results. Possible methods of doing this include trying different features, such as wavelet descriptors, instead of the Hu moments, and developing a method to learn appropriate weights for each of the seven moments from training data.

The scale estimation problem was approached primarily by reducing it to the problem of detecting valleys (equivalently, peaks) in one-dimensional signals generated using various methods. Future work might seek to develop an improved peak detection algorithm which does not rely on a choice of empirical parameters (such as a threshold). Additionally, since clutter in the neighborhood of the neatline was the most common cause of failure, methods of increasing robustness to clutter should be developed.

Our analysis of the uncertainty associated with scale estimation provides groundwork for estimating how far automatically calculated scales may differ from the true values intended by the cartographer as a consequence of various sources of error. Additionally, it provides a method for estimating the probability of catastrophic failure when the algorithm is applied to a specific map, which is intended to eventually provide humanists with feedback regarding how trustworthy a given output is. This information is important to prevent spurious conclusions from being drawn as a result of algorithm failure. Future work on uncertainty analysis should focus on developing a similar framework for the segmentation subsystem. If successful, this would allow error bounds and failure probabilities to be derived for the area figures themselves, effectively placing error bars on the graphs given in Chapter 6, and providing support for any conclusions drawn from them.
Finally, our work combines results from the segmentation and scale estimation subsystems to estimate Great Lake areas, thereby providing a quantitative measure of regional geographic knowledge, and allowing historians to draw conclusions from a comparison of these results between colonial powers. This application would become especially feasible if improvements could be made in algorithm accuracy, and/or if a more comprehensive uncertainty analysis could be performed.
REFERENCES


