# Interpretation and Clustering of Handwritten Student Responses



by

Kelsey Leigh Von Tish

B.S., Massachusetts Institute of Technology (2011)

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Engineering in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May, 2012



Author .....
Department of Electrical Engineering and Computer Science
May 21, 2012

Certified by.....
.....
Dr. Kimberle Koile
Research Scientist, MIT CECI
Thesis Supervisor
May 21, 2012

Accepted by .....
.....
Prof. Dennis M. Freeman
Chairman, Masters of Engineering Thesis Committee

# Interpretation and Clustering of Handwritten Student Responses

by

Kelsey Leigh Von Tish

Submitted to the Department of Electrical Engineering and Computer Science
on May 21, 2012, in partial fulfillment of the
requirements for the degree of
Master of Engineering in Electrical Engineering and Computer Science

## Abstract

This thesis presents an interpretation and clustering framework for handwritten student responses on tablet computers. The ink analysis system is able to capture and interpret digital ink strokes for many types of classroom exercises, including graphs, number lines, and fraction shading problems. By approaching the problem with both online and offline ink interpretation methods, relevant information is extracted from sets of ink strokes to produce a representation of a student's answer. A clustering algorithm is then used to group similar student responses. Overall, this approach makes it easier for teachers to view a set of responses and subsequently supply feedback to his or her students.

Thesis Supervisor: Dr. Kimberle Koile
Title: Research Scientist, MIT CECI

# Acknowledgments

First and foremost, I would like to thank my supervisor, Dr. Kimberle Koile, for making my research a great experience. Her passion for the Classroom Learning Partner as well as her constant encouragement allowed me to think creatively and explore my ideas. Without her deep commitment to the group, the project as well as this thesis would not have been possible.

I would also like to thank Andee Rubin and Lily Ko, our collaborators at TERC, for providing invaluable advice and insight about education. Their contributions continue to help shape our project from a simple piece of software to a beneficial teaching tool.

I am grateful for the efforts of Sherry Nichols and Renee Lockwood, who allowed us to test our software in their classrooms. Their patience and devotion to the project made it possible for us to make vital improvements to the system.

I owe my deepest gratitude to Steve Chapman, our lead developer, who truly made the project what it is today. Without his help both writing and integrating code, a final, working piece of software would not be possible. Through constant work and many late-night debugging sessions, he created a solid backbone for the system that others can build on.

I am very thankful for the other members of my group, Claire DeRosa, Jessie Mueller, and Eryn Maynard. Through many meetings and hours of brainstorming, they gave me inspiration for my work and made even long hours in lab enjoyable.

Finally, I would like to thank my parents, Drew Von Tish and Christine Staver, for their constant love and support. Their care packages and encouragement helped me through many difficult times at MIT.

# Contents

# List of Figures

# List of Algorithms

# Chapter 1

# Introduction

It is well-known and widely accepted that prompt, cogent feedback on student work facilitates better learning (e.g. [1, 2]). As class sizes increase, however, it becomes more and more difficult for teachers to provide sufficient attention to each individual. Feedback on student work is delayed, so it is challenging to diagnose when an individual is struggling and harder still to correct their mistakes so that they may overcome their difficulties. To ameliorate this issue, the process of communication between a teacher and her students must be streamlined. The time between writing down an answer to a question and receiving feedback about it should be significantly reduced.

Classroom Learning Partner is a system of wirelessly connected tablet computers that aims to solve this problem [8, 9]. Relying on a notebook metaphor, the system replaces a student workbook or exercise booklet. A teacher can create and present questions to her students by authoring a "digital notebook" containing exercises organized on "pages". Upon viewing these exercises on their own tablets, students write out answers and submit them to the teacher electronically. The teacher can then view the student work in real-time, enabling her to identify students who need help and to select student work to use as the focus of class discussion. This process of viewing student work significantly reduces the time between when a student solves a problem and when the teacher reviews the answer. As a result, it is much easier to give timely and effective feedback.

With the goal of expedited feedback in mind, this thesis builds upon previous work with new methods that further reduce communication delays between student and teacher. By utilizing automated interpretation techniques, the system is capable of robustly interpreting student answers for a wide variety of problems including graphs, number sentences, and fraction shading. This approach also makes use of clustering techniques to automatically group responses by both the obtained answer and the method used to reach that answer. These clustering techniques make student responses much easier to sort through and, as a result, encourage more effective communication with fewer delays.

## 1.1 Motivation

Studies have shown that tablet computers can aid in education. They not only enable more effective communication between students and teachers but also keep students more engaged and make it easier to stay organized (e.g. [9, 10]). However, these benefits can only be attained with a well-implemented system. Naturally, if the interface is difficult to use, it cannot provide the same advantages. If student responses are not well-organized, for instance, a teacher viewing them could easily become overwhelmed. As this student work piles up, it would be increasingly difficult to attend to each individual submission.

Classroom Learning Partner, currently being implemented and tested in upper elementary school classrooms, aims to tackle this usability issue in several ways. Prior research by Martyna Jozwiak suggests numerous methods for solving this problem from a user interface standpoint [6]. By providing several grouping options for student submissions in the UI, she makes student work much easier to sort through. However, there is still room for a backend solution to this issue. By automatically interpreting and clustering responses, a useful grouping method could be applied automatically. In addition to sorting student responses by name or time of submission, they could

be sorted by correctness or methodology. When attempting to quickly give students feedback on their work, these characteristics are extremely important. Developing methods to extract this information is the focus of this thesis.

## 1.2 System Overview

Classroom Learning Partner involves several distinct phases of usage that are all tied together using important objects called Page Objects. These classes, which all inherit from the same base class, represent the building blocks for all questions the teacher can design. Ranging from text boxes for writing out questions to grids for graph problems, Page Objects are generally responsible for displaying, storing, and capturing data. The stamps shown in Figure 1-1, for instance, capture ink strokes that are written on them. They then display the strokes and store information about them so that they can be copied, or "stamped", elsewhere on the page.

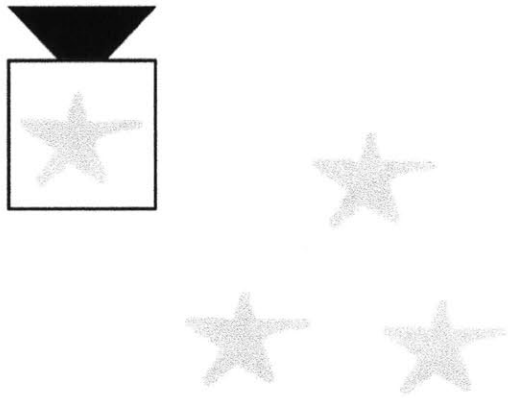


Figure 1-1: A stamp Page Object. Users can draw on the stamp and make copies of their drawing by clicking and dragging the black handle.

The first stage of usage, authoring, pertains to the creation of a notebook, which enables one to design the problems that will be presented in class. When in this mode, the teacher is able to add pages to a digital notebook and add Page Objects to

the pages. The next stage concerns the actual presentation of this notebook on the student and teacher tablet computers. In this phase, the authored notebook can be viewed and annotated with the addition of ink strokes or other actions such as dragging, clicking, or adding certain types of Page Objects. As a result, the Page Objects stored in the notebook are altered by the students' or teacher's actions. Finally, there is a submitting or sending phase in which the information in the notebook can be sent over a network to various locations. Students are able to submit their answers to the teacher, student responses can be sent to a database, or the teacher can send the answers to a machine hooked up to a projector for reviewing information with the entire class. The Page Objects are a critical part of this process because they encapsulate a large portion of the work that students do on their tablets, and they can be transmitted over a network through serialization and subsequent deserialization.

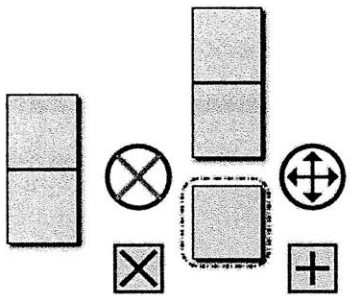


Figure 1-2: Snap tile Page Objects. The green tiles can be placed on the page and then snapped together vertically when they are dragged near each other. Page Object operators, shown at the corners of one of the tiles, appear when the tablet pen hovers over the tile.

My project utilizes the power of Page Objects for both interpretation and clustering. Making use of their ability to capture ink strokes, they feed stroke information to interpretation methods introduced in Section 1.2.1. My implementation then stores the result of the interpretation on the Page Object itself so that it can be transmitted

to the teacher or stored in the database. Necessary information for clustering is also extracted and stored so that student answers can be grouped as described in Section 1.2.2.

### 1.2.1 Interpretation

The focus of this thesis is purely on the interpretation of digital ink. While there are alternate ways for students to provide answers, such as the stamps pictured in Figure 1-1 and the snap tiles shown in Figure 1-2, thus far in our research ink has been the most common means of responding. Whether writing out a number sentence to solve a math problem, explaining an answer with handwritten text, or plotting points on a graph or number line, handwritten ink is an integral part of student answers in Classroom Learning Partner.

To focus my work, I concentrated on five specific question types that receive digital ink as input:

- Handwriting recognition for interpreting handwritten numbers, letters, equations, words, and sentences
- Fraction shading to extract what percentage of an area is shaded in by ink
- Data tables for organizing ink in a table structure and then running interpretation on each table cell
- Graphs for plotting points, lines, and curves in two-dimensional space
- Shape recognition to locate shape primitives drawn on a number line

Described in detail in Chapter 3, these types span a wide variety of questions that could be posed in an elementary school setting.

### 1.2.2 Clustering

To group student responses by similarity, I concentrated not only on clustering by correctness of the answer but also by the methodology used to achieve it. That is, my approach aims to ensure that all the paths to various answers are reflected in the resulting clusters. As with most clustering algorithms, this process consists of three major steps:

1. Feature extraction to reduce the critical components of a student response to a series of numbers
2. Re-weighting to weight each numerical feature by its overall importance
3. Clustering to group the sets of features by their similarity

These steps are described in Chapter 4.

## 1.3 Outline

Chapter 2 details prior contributions to sketch interpretation, clustering, and Classroom Learning Partner. Chapter 3 and Chapter 4 describe my implementation of the interpretation and clustering modules respectively. In Chapter 5, I discuss the testing results for the interpretation and clustering methods. Chapter 6 concerns future improvements to the system I developed. Finally, in Chapter 7 I summarize my contributions to Classroom Learning Partner.

# Chapter 2

# Previous Work

There is a great deal of research concerning both digital ink stroke interpretation and data clustering. Ink interpretation is not new to the Classroom Learning Partner, so I was able to build upon previous work such as the Recognition Grids described in Section 2.1.3 below. Student answer clustering, however, is not only new to Classroom Learning Partner but also a fairly novel application of standard clustering algorithms. As a result, I took inspiration from but did not completely rely on the methods described in Section 2.2.

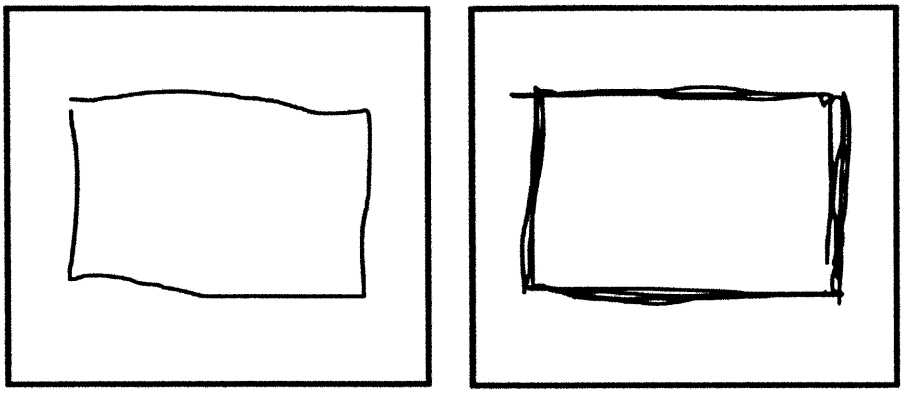
## 2.1 Sketch Interpretation

Ink interpretation methods come in two main varieties: online methods and offline methods. Online ink interpretation techniques, described in Section 2.1.1, take full advantage of all the information that is received when an ink stroke is being formed. The position, timing, and even pen pressure of points that make up the ink strokes can be used to find the meaning of digital ink. In contrast, offline methods such as those in Section 2.1.2 only use spatial information, usually in the form of a binary image, to perform interpretation. The Recognition Grid approach detailed in Section 2.1.3 attempts to hybridize these techniques.

### 2.1.1 Online Interpretation

One example of online sketch interpretation is presented by Tevfik Sezgin and Randall Davis in [13]. Their technique relies mostly on temporal information of a set of strokes in order to label drawn circuit diagrams with individual circuit components such as wires, resistors, and diodes. The initial set of strokes is segmented into a time-ordered set of geometric primitives, such as lines and arcs. They then use a Bayesian network approach to assign the most likely labels to parts of the sketched diagram based on the order in which the segments are drawn. For instance, two parallel lines drawn one after another likely represent a capacitor, so strokes drawn in this way will be labeled as such. The failure cases of this technique, however, illustrate some of the disadvantages of online interpretation. For example, two parallel lines that are drawn sequentially but far away from each other will still often be interpreted as a capacitor. Though they are not spatially related, this method only takes their temporal relationship into account. It can incorrectly classify parts of the diagram if they are not drawn in exactly the right order, so it is evident that for some domains this technique relies too heavily on temporal cues.

There are other online methods that strike a balance between spatial and temporal information. Thomas Stahovich, for example, summarizes an ink stroke segmentation technique in [15] that utilizes pen speed to calculate critical points. He explains that important points in a stroke, or key points, are those that have low speed and high curvature. That is, they occur where there are large bends in the stroke that are drawn slowly and carefully. While this technique balances temporal and spatial information, it also has flaws when used to classify sketches and shapes. Figure 2-1 illustrates one such issue. While both sketches clearly represent rectangles, (a) is composed of only one ink stroke while (b) has a lot of over-stroking. Though they are the same shape, the segmentation result for the second rectangle will contain many more key points and segments. It is not always trivial to dispose of duplicate segments, so it is clear that this technique is not right for all applications. Despite these flaws, however,



(a) A rectangle with one stroke. (b) A rectangle with over-stroking.

Figure 2-1: Multiple ways to draw a rectangle. While example (a) is easy for online methods, the line tracing in (b) causes problems.

this method influences my stroke segmentation and graph interpretation algorithms summarized in Sections 3.1.5 and 3.2.4.

### 2.1.2 Offline Interpretation

Offline interpretation methods can be advantageous for many reasons. Most notably, they are often more efficient than online methods and not as susceptible to the issues with over-stroking shown in Figure 2-1. These improvements result from ink strokes being discretized to a grid. The result is a binary image that has no stroke segmentation issues and is easy to compare to a set of example sketches using image comparison techniques.

Kara and Stahovich detail three very common image comparison techniques that could be used with offline ink interpretation methods [7]. The first, Hausdorff distance, measures how far away two sets of points are. The formal definition, shown below, finds the maximum of the distance between closest point pairs, a and b, across two images, A and B.

$$H(A,B) = max(h(A,B), h(B,A)),$$
$$h(A,B) = max_{a \in A}(min_{b \in B}(||a - b||))$$

The Tanimoto coefficient, another image comparison metric, involves the proportionality of overlapping pixels between two images. Its mathematical definition is the following:

$$T(A,B) = \frac{n_{ab}}{n_a + n_b - n_{ab}}$$

In this definition, $n_a$ is the number of black pixels in the first image, $n_b$ is the number of black pixels in the second image, and $n_{ab}$ is the number of overlapping black pixels. It gives a general measurement of the similarity between pixel placement in two bitmaps. The last metric, the Yule coefficient, follows a very similar approach. However, it also takes into account the white pixels in the image. It can be summarized as follows:

$$Y(A,B) = \frac{n_{11}n_{00} - n_{10}n_{01}}{n_{11}n_{00} + n_{10}n_{01}}$$

In the equation above, $n_{11}$ and $n_{00}$ are the overlapping black and white pixels respectively, and $n_{10}$ and $n_{01}$ are the unmatched black and white pixels between the images. Like the Tanimoto coefficient, the Yule coefficient tries to determine how similar two pictures are by matching pixel locations.

What is important to note about these image comparison techniques is that they are only compatible with problems in which one image must be compared to a set of known template images. That is, they are only useful in classification problems where the set of classes are known ahead of time. This is not the case when clustering student responses because the answers students might give are not that predictable. As a result, these sorts of template-based techniques are not compatible with the problem this thesis aims to solve. Instead, I developed a method that extracts the properties of each individual student answer and then group answers based on their similarity to each other rather than their similarity to a set of templates.

### 2.1.3 Recognition Grids

Recognition Grids are an ink processing module developed for Classroom Learning Partner by Neil Chao [3]. Combining both online and offline techniques, this method

was designed to be used with every kind of ink-based interpretation problem in Classroom Learning Partner. Specifically, it is meant to process the ink strokes for handwriting recognition, fraction shading, data tables, graphs, and number lines. It is a generic solution that aims to be extensible to many situations where ink processing is necessary.


Figure 2-2: A recognition grid. Green grid squares are those that contain stroke segments.

As the name suggests, Recognitions Grids are built around the concept of discretizing ink strokes to a grid. Much like the offline methods discussed above in Section 2.1.2, strokes are divided amongst the grid squares they overlap as illustrated in Figure 2-2. Instead of simply forming a binary image that shows which grid squares are covered by ink, however, the Recognition Grid technique actually saves the overlapping portion of the stroke in each respective grid square. Hence, on a large scale Recognition Grids act as an offline ink interpretation technique where a bitmap representation can be used. Within each grid square, however, online methods can be applied to the segmented strokes.

Though Recognition Grids are versatile because of their hybridized ink interpreta-

tion, they unfortunately fall short in both accuracy and efficiency. Too much critical information is thrown away in the discretization process to make them applicable to all possible types of Classroom Learning Partner problems. For instance, the geometry of an entire stroke, as opposed to its discretized segments, is often important for accurately extracting shape information. An example is shown in Figure 2-3. Though three separate elliptical shapes were drawn to form the image, the amount of overlap between them makes the binary bitmap representation ambiguous. The discretized stroke segments also cannot accurately reconstruct the original shapes. In order to get a robust representation of what was drawn, the ordering and geometry of the initial input ink strokes must be known, and the Recognition Grid fails to capture this information.

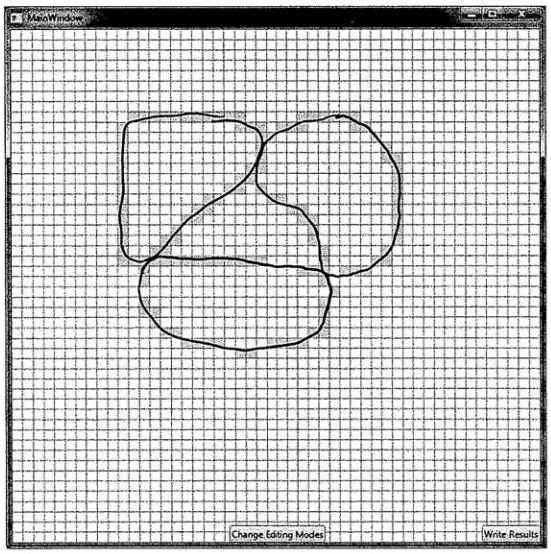



Figure 2-3: A type of question that Recognition Grids cannot handle. The shapes drawn in the image are ambiguous.

Recognition Grids are well-suited for applications in which high level shape information is not important. For example, they are able to accurately determine what percentage of the grid is shaded by comparing the number of shaded and unshaded squares in the binary bitmap representation. Recognition Grids would seem to provide a good interpretation method for fraction shading problems used with Classroom

Learning Partner, but they are not efficient at interpreting these problems due to their generality.

The goal of a single, unified ink interpretation technique was a worthy one, but the cost in time and resources has proven to be too high. Instead of going with a generic approach such as the Recognition Grid, the research in this thesis focuses on analysis methods that are tailored to particular types of problems with the goal of creating accurate and efficient solutions. This research suggests that this method, though not as immediately extensible to future problem types, ensures a better and more usable product. Furthermore, the interpretation techniques summarized in Chapter 3 are designed as building blocks than can be reused and combined to solve more complex problems. By building up a toolkit that draws from both the online and offline methods described above, the system not only achieves the desired accuracy and efficiency but can be easily manipulated and extended to achieve versatility.

## 2.2 Clustering

The problem of clustering is not a trivial one, for it requires the unsupervised partition of a dataset into a natural grouping. That is, without the aid of predefined classes, a clustering algorithm needs to successfully associate similar data. Nearly all clustering procedures can be broken down into two steps. First, a feature vector must be extracted from the items that are being clustered. This feature vector numerically describes the fundamental properties of the objects. This vector is then fed into the clustering algorithm itself, in which a distance metric is used to group the input vectors based on their proximity. In this fashion, whether grouping images, web pages, or any other data, everything turns into a distance problem.

The specific clustering techniques vary. K-means, however, is likely the most popular due to its ease of implementation. It is a relatively simple center-based clustering algorithm that partitions a dataset based on a number of centers or centroids [5]. In

essence, this method attempts to find an optimal partitioning of the data by assigning each point to its nearest centroid. The number of centroids, k, is known and used as input to the algorithm. Mathematically this can be expressed as a minimization of the following formula:

$$KM(X,C) = \sum_{i=1}^{n} \min_{j \in \{1....K\}} ||x_i - c_j||^2$$

Here $x_i$ represents each point and $c_j (j \in \{1...K\})$ are the centroids. The final result is minimal variance within each of the k clusters [5].

There have been many proposed improvements to the k-means algorithm. Asserting that the optimization function above does not always lead to an ideal partitioning of the dataset, alternate methods such as Gaussian expectation-maximization, fuzzy k-means, and k-harmonic means propose novel functions to minimize. The fuzzy k-means algorithm, for instance, utilizes a soft membership function to improve results. In other words, instead of assigning a point to a distinct cluster during the optimization process, a single point can belong partially to several clusters. In mathematical terms:

$$FKM(X,C) = \sum_{i=1}^{n} \sum_{j=1}^{k} u_{ij}^{r} ||x_i - c_j||^2$$

The $u_{ij}$ parameter is a weight that specifies what portion of the point $x_i$ belongs to the centroid $c_j$. In the end, points are assigned to the cluster in which they are the most heavily weighted. The r parameter, which must be greater than or equal to one, dictates the "fuzziness" of the algorithm. Higher values place more emphasis on the point weights, whereas values close to one make it behave much more like the standard k-means algorithm. The end result is an algorithm that doesn't treat all data points equally. Outliers do not have the same weight as those that conform with the rest of the dataset, so results can often be more accurate [5].

While k-means and its variations are well-suited for many clustering problems, the concept is fundamentally not applicable to grouping student responses. This

problem is caused by the inclusion of the parameter k, which defines how many clusters there are. This number is not known for student answers because it is impossible to know in how many different ways a group of students will respond to a question. Trying to estimate this k parameter would likely lead to a false grouping even with the improved methods described above. As a result, my research involved finding a clustering algorithm that does not require prior knowledge about the final number of clusters. The solution arrived at, a DBSCAN-based clustering algorithm, is discussed in Chapter 4.

# Chapter 3

# Ink Interpretation

As described above in Section 2.1.3, my approach to interpretation involves several distinct ink analysis routines that can be chained together to perform more complex interpretation tasks. To accomplish this, I created a static library capable of executing five fundamental types of analysis: handwriting recognition, shape recognition, discretization to many grid squares, discretization to a single grid square, and stroke segmentation. When used in isolation or in combination, the methods I developed can analyze handwriting recognition, fraction shading, data table, graph, and number line problems. Future combinations and extensions of these methods are discussed in Chapter 6.

To integrate the analysis routines with Classroom Learning Partner, I turned each of the problem types listed above into a type of Page Object. Like the stamp Page Object mentioned in Section 1.2, the various interpretation-based Page Objects accept ink strokes. Whenever the Page Objects are serialized, which occurs when a notebook page is either saved or submitted to the teacher, one or several of the analysis routines described in Section 3.1 act on the strokes in order to find the interpreted answer. This answer is then stored on the Page Object itself. An overview of these Page Objects is presented in Section 3.2.

## 3.1 Fundamental Analysis Routines

The following sections describe the analysis routines included in the static ink interpretation library. No matter how the architecture of Classroom Learning Partner changes in the future, these methods will likely still be usable. They all accept a collection of ink strokes as an input parameter and output an analyzed representation of the strokes.

### 3.1.1 Handwriting Recognition

To achieve a robust handwriting recognition routine, I relied on Microsoft's built-in handwriting recognizer defined in the Ink Analyzer class. This method is advantageous because the system is already trained on many examples of handwriting and, as a result, achieves high recognition rates without much effort. Furthermore, the analyzer can be customized in several ways to add more accuracy to the system. This customization is achieved through specifying what is called an Analysis Hint Node. This class allows for the following modifications to the InkAnalyzer:

- Prioritize single words or multiple word results in a given area
- Allow partial dictionary terms (portions of a word such as "hel" instead of "hello")
- Specify what type of input is expected (a single character, a single digit, a number, a date, one of several specified words, etc.)

The most important modification is the ability to specify the expected input. By narrowing down the answer to a digit, number, or word, the accuracy of the system is greatly increased [1]. For this reason, the handwriting recognition routine accepts a set of ink strokes as well as an analysis type parameter. For the purposes of Classroom Leaning Partner, the set of possible types is limited only to numbers, digits, words,

[1] Research with previous versions of Classroom Learning Partner bears out this result (e.g. [8, 11, 16])

and the default analysis, which can accept all input types. The increased accuracy this yields can be observed in Section 5.1.

### 3.1.2 Shape Recognition

For shape recognition I also used the built-in functionality of Microsoft's ink analysis routines. Along with recognizing handwriting, the Ink Analyzer is able to detect simple geometric shapes including squares, rectangles, triangles, circles, and ellipses. The analysis routine makes use of this capability to not only extract what shapes are in a set of ink strokes but also to determine what particular strokes make up each of those shapes. As a result, this analysis routine is extremely versatile because it is trivial to find where each shape is located by simply locating the corresponding ink strokes. This algorithm is applicable in the following situations:

1. Determining how many instances of particular shapes have been drawn
2. Finding where particular shapes have been drawn on a number line or graph to easily determine the student's answer
3. Circling items on a page with ink strokes and determining which items are circled

The first two applications are accomplished by the number line Page Object discussed in Section 3.2.5. The third application is listed in Chapter 6 as an extension of this thesis.

### 3.1.3 Discretization to Many Grid Squares

The need for a routine that discretizes ink strokes to grid squares is motivated by the offline interpretation routines detailed in Section 2.1.2 and the Recognition Grids discussed in Section 2.1.3. In many situations, such as the fraction shading problems in Section 3.2.2, having a binary bitmap representation makes the analysis and feature extraction process much easier. As a result, I devised a simple, efficient algorithm

that successfully divides strokes up amongst a grid of a specified width and height. Unlike the Recognition Grid approach, however, my algorithm does not go through the extra computation of dividing and saving the ink stroke segments. That information is discarded, and all that is returned is the binary bitmap result that states whether a square contains or does not contain a stroke. A summary of this algorithm is shown in Algorithm 1.

**Algorithm 1** Discretization to many grid squares.

**function** $DiscretizeMany(strokes, w, h, space)$
  $rows = floor(\frac{h}{space})$
  $cols = floor(\frac{w}{space})$
  $result = bool[cols, rows]$
  **for** (*each stroke S in strokes*) **do**
    **for** (*each point P in S*) **do**
      $pointX = P.X$
      $pointY = P.Y$
      $indexX = floor(pointX/\frac{width}{cols})$
      $indexY = floor(pointY/\frac{height}{rows})$
      **if** ($indexX > 0$ and $indexY > 0$ and $indexX < cols$ and $indexY < rows$) **then**
        $result[indexX, indexY] = true$
      **end if**
    **end for**
  **end for**
  **return** *result*

The parameter "space" dictates the approximate spacing between grid squares. That is, the number of rows and columns in the grid is computed using this parameter as well as the pixel width, "w", and height, "h", of the entire grid. This spacing is critical to the accuracy of the system because it dictates how large an area a single ink stroke covers in the final implementation. If the granularity is too coarse, distinct answers will not be dissimilar enough to distinguish from each other. In contrast, if it is too fine, the minor imperfections in an answer, such as cursory shading that does not completely cover the area that was meant to be colored in, will negatively impact the final interpreted answer. The effects of changing this parameter are discussed further in Section 5.1.2.

After establishing the correct number of rows and columns using the "space" parameter, the algorithm iterates over all of the points that compose all of the strokes and discretizes them to the grid. If the discretized point lies within a valid grid square, the value of the binary bitmap at that square is set to true. The final bitmap is returned in the end, and this output is currently used by the fraction shading Page Object discussed in Section 3.2.2.

Building upon this core algorithm, I also added an option to filter out ink strokes that are completely horizontal or completely vertical. In order words, if the standard deviation of an ink stroke's x or y coordinates is not large enough, the stroke is ignored during the discretization process. This feature was added specifically to increase the accuracy of interpreting the fraction shading problems described in Section 3.2.2. The motivation behind this addition is discussed in greater detail there.

### 3.1.4 Discretization to One Grid Square

I created a process for discretizing an ink stroke to a single grid square so that writing can be entered into the cells of table. The algorithm is similar to the method of discretizing to many grid squares above and to Recognition Grids but changes some critical aspects. The task of writing entries in tables is fundamentally compatible with the discretization process on which the Recognition Grids rely. That is, it is necessary to discretize the ink strokes into grid squares—the squares of the table—to interpret what is written in each cell. The Recognition Grid technique, however, assumes that an ink stroke can belong to multiple grid squares, and as a result, it divides the stroke amongst several cells. In the case of data tables, an ink stroke can only belong to a single square because data should only be entered in one cell at a time. Hence, the processing step can be refined to take this difference into account. Instead of segmenting an ink stroke into all of the squares it crosses, this technique finds a single square to which a stroke belongs.

This process is summarized by Algorithm 2. Once again, "w" and "h" represent the width and height of the entire grid. Unlike the method for discretizing to many squares, however, this algorithm also accepts a specified number of rows and columns instead of a spacing parameter. This difference is because the grid squares in this situation correspond to the actual grid squares in a data table whereas in the method in Section 3.1.3 the grid squares were simply an invisible mechanism used to create a binary bitmap representation of the input. This discretization technique then calculates the centroid of each stroke and uses that information to decide to which individual grid square it belongs.

**Algorithm 2** Discretization to one grid square.

**function** *DiscretizeOne*$(strokes, w, h, rows, cols)$
  $result$ = *a list of points*
  **for** (*each stroke* $S$ *in strokes*) **do**
    $centroidX = 0.0$
    $centroidY = 0.0$
    **for** (*each point* $P$ *in* $S$) **do**
      $centroidX+ = P.X$
      $centroidY+ = P.Y$
    **end for**
    $centroidX/ =$ *total points*
    $centroidY/ =$ *total points*
    $indexX = floor(centroidX/\frac{w}{cols})$
    $indexY = floor(centroidY/\frac{h}{rows})$
    **if** ($indexX > 0$ and $indexY > 0$ and $indexX < cols$ and $indexY < rows$) **then**
      *Add Point(indexX, indexY) to result*
    **end if**
  **end for**
  **return** *result*

Processes using this discretization method can utilize the output to store the ink strokes that are inside of each square of a grid. As a result, further stroke processing steps, such as handwriting recognition or shape recognition, can be used on the strokes that lie in each cell. The data table Page Object described in Section 3.2.3 takes advantage of this fact to serve as a robust data entry component.

### 3.1.5 Stroke Segmentation

The final method of stroke processing is inspired entirely by the online techniques mentioned in Section 2.1.1. Unlike the other ink interpretation techniques described above, this approach attempts to extract critical shape information and develop a robust numerical description of what a set of strokes looks like. Instead of labeling ink strokes with specific shape types as the approach in Section 3.1.2 does, this algorithm extracts a set of important points and describes the geometry between the key points with their curvature as described in [15]. In this paper, Stahovich claims that the most critical points in an ink stroke, or key points, are those with high curvature and low pen speed. In other words, key points are those that are created where a stroke changes direction and the user is writing carefully.

Due to the limitations of the stroke objects within the implementation language, C#, I had to make several changes to the system. Most notably, I removed any calculations based on pen speed from the algorithm proposed in [15]. This change was necessary because the points that make up an ink stroke contain no timestamp or any other timing information other than their ordering, which is the order in which the points were formed. As a result, I only rely on curvature values to extract key points and describe the geometry of a set of strokes. By tuning the curvature threshold value to find points of high curvature, however, the exclusion of the speed parameter did not result in a loss of accuracy. An overview of this method is shown in Algorithm 3.

This method accepts a set of strokes as input and calculates the curvature at each point, which is estimated as the change in angle of the tangent to the curve divided by the change in arc length of the stroke. Local maxima with high curvature are then extracted as key points and returned from the algorithm along with the calculated set of curvatures. The combination of these two pieces of data allows for the calculation of the average curvature between two key points. As described in Section 3.2.4, this

**Algorithm 3** Stroke segmentation.

**function** *Segment(strokes)*
  **for** (*each stroke S in strokes*) **do**
    *Calculate curvature at each point:*

    1. *Calculate arc length at each point:*
       $arclen(j) = \sum_{i=1}^{j} ||S.points[i] - S.points[i-1]||$, $arclen(0) = 0$
    2. *Calculate slope of the tangent line to the curve at each point:*
       $slope(j) =$ *slope of* $LinearRegression(S.points[j-5 : j+5])$
    3. *Calculate the angle of the tangent at each point:*
       $angle(j) = atan(slope(j))$
    4. *Calculate the curvature (change in angle over change in arc length):*
       $curvature(j) = \frac{\delta(angle(j))}{\delta(arclen(j))}$ *(estimated with linear regression)*

    **for** (*j=0; j < length of S.points; j++*) **do**
      **if** (*if* $|curvature(j)|$ *is a local max and* $> 0.05$) **then**
        *Add to the set of key points*
      **end if**
    **end for**
  **end for**
  **return** *key points and curvatures*

technique is well-suited for describing the geometry of a set of strokes in a graph.

## 3.2 Integration and User Interface

Once the set of core interpretation algorithms described above were in place, I created several Page Objects to utilize this functionality within Classroom Learning Partner. Each Page Object corresponds to a specific type of problem and contains a user interface component that is visible to the user, a model that holds the interpretation and captured stroke information, and interaction logic that allows the UI and model to communicate. Most of the user interface elements pertain to Classroom Learning Partner's editing mode, in which the teacher is able to design problems by placing Page Objects in a notebook. Interpretation-based Page Objects, which are Page Objects that can accept and interpret ink, can be placed where ever the teacher would like to capture and interpret inked answers on a page.

### 3.2.1 Handwritten Problems

Handwritten problems are a major part of Classroom Learning Partner. Often when providing a final answer to a question, students need to specify their response as a handwritten number, mathematical expression, word, or combination of these things. As a result, I developed a recognition region that can be placed on a page in editing mode and that can interpret any handwritten answers that are provided on in that region.

As the name suggests, the handwriting recognition Page Object relies on the handwriting recognition routine, which is described in Section 3.1.1. Shown in Figure 3-1, the Page Object itself is actually invisible to anyone but the teacher and can only be seen in editing mode. The dashed line illustrates the boundaries of the interpreted area, the word in the upper right corner represents the type of interpretation that is being performed, and the string in the upper left shows the interpretation result that is stored in the model of the Page Object. The icon on the lower right is simply a means of editing the interpretation region so that the type of handwriting recognition can be altered.

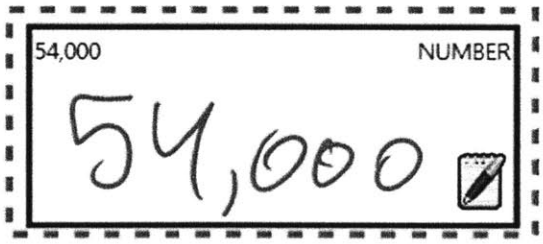



Figure 3-1: An example of a handwriting recognition Page Object. The dashed line, analysis type, analysis result, and edit button are only visible in editing mode.

When creating or editing a handwriting recognition Page Object, the dialog in Figure 3-2 appears. As to be expected, this dialog allows the user to select the type of handwriting interpretation that is passed to the analysis routine as a parameter. As mentioned in Section 3.1.1, this information can significantly increase the accuracy

of the interpretation over the default interpretation method.

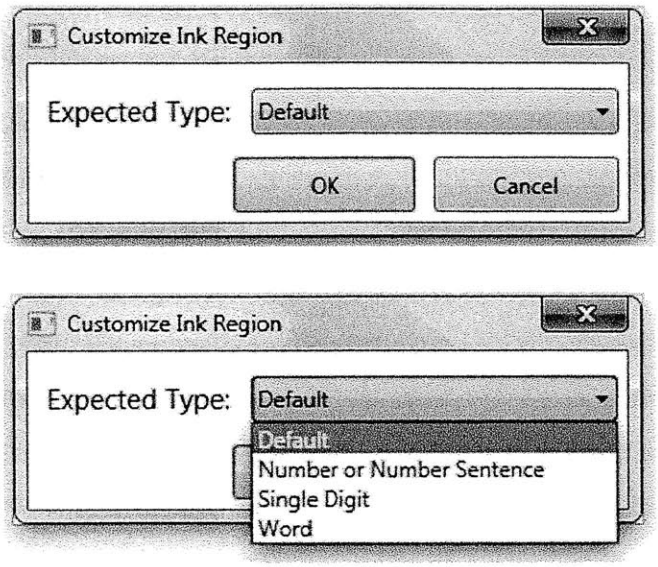



Figure 3-2: The customization dialog for handwriting region Page Objects.

### 3.2.2 Fraction Shading

Fractions are an important part of the elementary school math curriculum, so it is crucial that there are ways to use and visualize fractions in Classroom Learning Partner. I constructed a fraction shading Page Object to fulfill this requirement. This object is able to accept ink strokes and determine what percentage of the designated area is filled in. As a result, students can draw a visual representation of a fraction, as shown in Figure 3-3 and the software can check whether their answer is correct or incorrect.

This interpretation region utilizes the ink stroke discretization technique summarized in Section 3.1.3. This routine ensures that each stroke is discretized to one or more squares of an invisible grid that lies beneath the full area of the region. Once this binary bitmap representation of the image is formed, the number of filled in squares is divided by the total squares in order to obtain the filled in percentage.

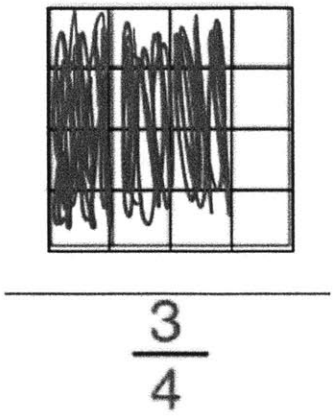



Figure 3-3: An example of a fraction shading Page Object.

This number is then stored on the Page Object for later use. The visible grid, shown in Figure 3-3, is a purely visual element that the teacher can add as a guideline for the students when they solve the problem.

As mentioned in Section 3.1.3, the shading region object utilizes a filtering feature within the discretization algorithm to ignore purely horizontal or purely vertical strokes. The inclusion of this feature was motivated by the common use of guideline strokes when solving this sort of problem. As pictured in Figure 3-4, students will often divide up the area with strokes when no predefined guidelines exist. If these strokes were included in the final answer, the returned percentage would clearly be greater than the portion the student actually intended to shade in. As a result, the algorithm makes use of the option to filter out these strokes to arrive at a final answer.

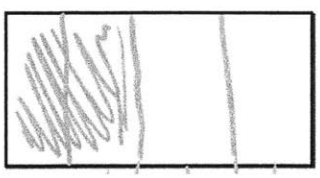


Figure 3-4: An example of a fraction shading problem with guideline strokes.

When customizing a fraction shading region, the author of a notebook is pre-

sented with the dialog in Figure 3-5. Here she can select the number of rows and columns that are displayed by the Page Object. As stated above, these grid lines are purely visual and have nothing to do with the discretization step performed by the interpretation algorithm. They are provided as guidelines for students when the teacher wishes to simplify a problem. Instead, a predefined spacing parameter is used to facilitate the formation of the binary bitmap representation of the set of strokes.

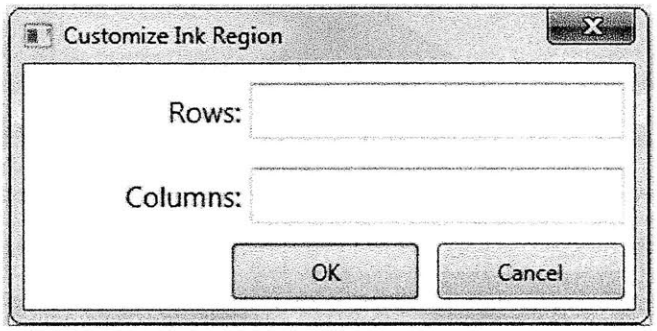



Figure 3-5: An fraction shading Page Object creation dialog box.

### 3.2.3 Data Tables

Classroom exercises often necessitate some sort of organized data entry. To accommodate this, I developed a data table Page Object that is able to divide ink strokes up amongst its cells and then interpret each cell's contents separately. The discretization step is accomplished with the technique detailed in Section 3.1.4. In this case, ink strokes that are written on top of the Page Object are placed in grid squares based on the location of their centroids. The ink strokes are then associated with the individual cell to which they belong so that the handwriting recognition method in Section 3.1.1 can act on each grid square separately. The final interpreted result for each grid square is then stored on the Page Object.

An example data table Page Object is shown in Figure 3-6. In this case, the written numbers 40 and 8 are placed in their respective cells of the table with the

discretization algorithm, and the handwriting recognizer interprets the contents of each cell individually. While the current version of Classroom Learning Partner only utilizes the handwriting recognizer to interpret the ink strokes located in each cell, this method is certainly not the only way data tables could be used. Any of the other interpretation methods described in Section 3.1 could be used to perform the second interpretation step. The shape recognizer, for example, could be used on the strokes in each grid square to determine what shapes are drawn in the cells. This property makes the data table Page Object a very useful stepping stone to future varieties of class exercises.

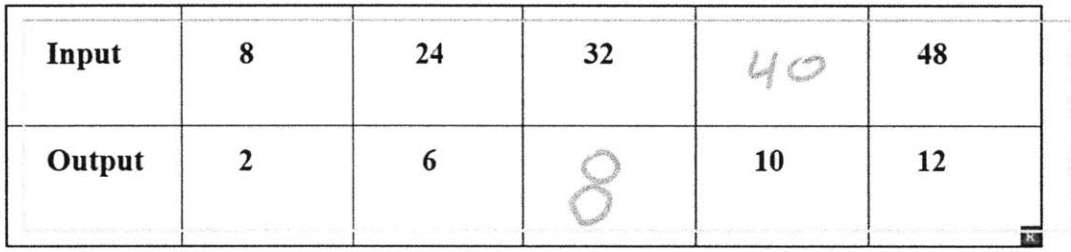



Figure 3-6: A data table Page Object.

Data tables can be customized with the dialog shown in Figure 3-7. Much like shading regions, the number of rows and columns is specified in order to create a visible grid. This grid is not purely visual, however, for its dimensions are actually fed into the interpretation algorithm to discretize the ink strokes to the cells. A customization option is also available for the type of handwriting recognition was it as with the pure handwriting recognition region. This option allows the handwriting analysis step for the data table to receive the same boost in accuracy.

### 3.2.4 Graphs

It is sometimes necessary to extract more sophisticated shape information from a set of strokes than written text or simple shape primitives. In the case of plotted graphs,

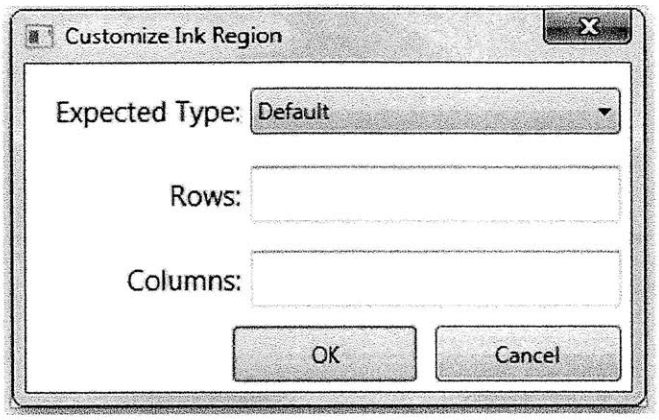



Figure 3-7: The dialog box for data table customization.

for instance, it is important to accurately summarize graph geometry in order to determine the accuracy of an answer. To accomplish this, I built a graph Page Object that uses the stroke segmentation algorithm in Section 3.1.5.

Shown in Figure 3-8, this object uses the visual elements of the fraction shading and data table Page Objects but utilizes different interpretation logic. Much like the fraction shading object, the grid lines are purely a visual element that does not have an effect on the interpreted answer. The final result of the interpretation—the extracted key points—are visualized in editing mode with the set of red dots. It is easy to see that the points are indeed locations with high curvature as described in the process in Section 3.1.5. The geometry between these key points can be easily calculated by taking an average of the stored curvature values along the strokes between each set of key points. The final result is a simplified, numerical representation of the visual elements of the graph. This information can then be compared against a model to determine the accuracy of the graph.

The process of adding a graph Page Object to a notebook page is identical to the creation of a fraction shading Page Object. In fact, the dialog that appears is the same dialog that is shown in Figure 3-5. This dialog allows the user to customize the number of visible grid lines on the interpretation object. Though they do not alter

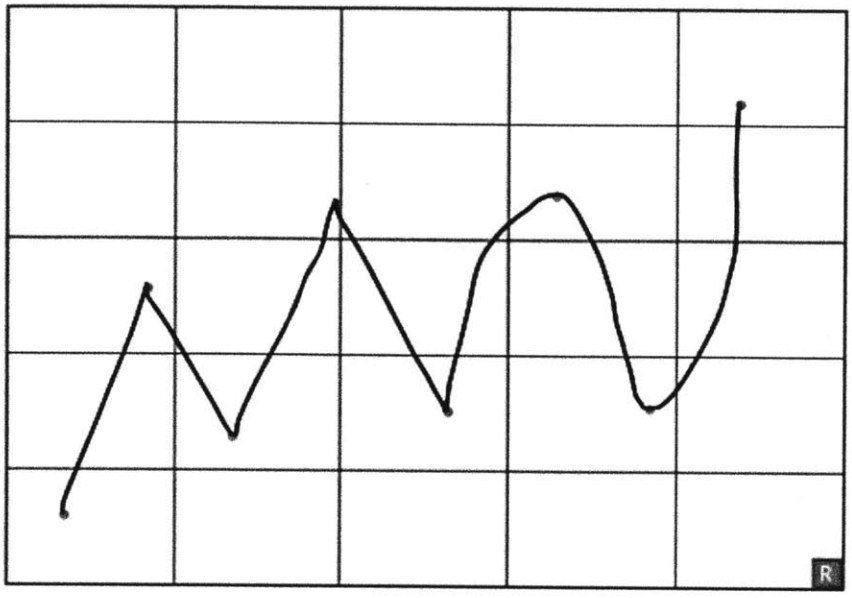


Figure 3-8: The dialog box for data table customization.

the interpretation, the visual guidelines make the process of plotting a graph much easier and more user-friendly.

### 3.2.5 Number Lines

One of the major obstacles when performing interpretation on student answers is sorting out which ink strokes are scratch work and which are part of the final answer. This issue is probably most visible in number line problems such as the one shown in Figure 3-9. In these types of questions, students must mark points on a line and will often label these marks with text, numbers, or both. In this situation, many ink strokes are crammed into a small space, and it is extremely difficult to segment them. Student answers are extremely unpredictable, so dividing the ink strokes with spatial, color, or timing information often proves futile.

Though constraining the problem is necessary to reach a simple interpretation solution, it is undesirable to limit the student work too aggressively. Forcing students to use specific colors to designate different parts of their response would certainly make

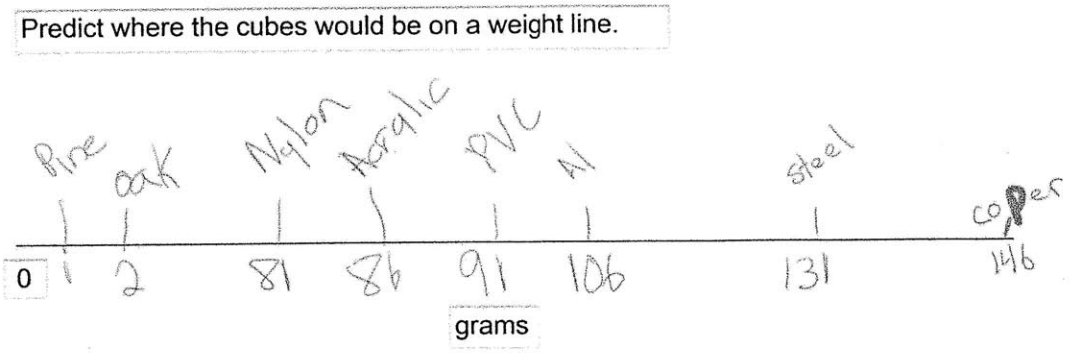



Figure 3-9: A sample number line problem. It is difficult to segment the labels and points.

the task of interpretation easier, but it makes their work unnecessarily tedious and inhibits creativity. Instead, I developed an algorithm that uses shape as a defining factor for interpreting number line problems. In this case, students plot points on the number line using specified shapes so that their final answer can be distinguished from any labels or other scratch work. Though this limitation is still not ideal, it does not require a change in the pen mode as the color-based method does. As a result, providing an answer in this fashion could prove more fluid and less tedious.

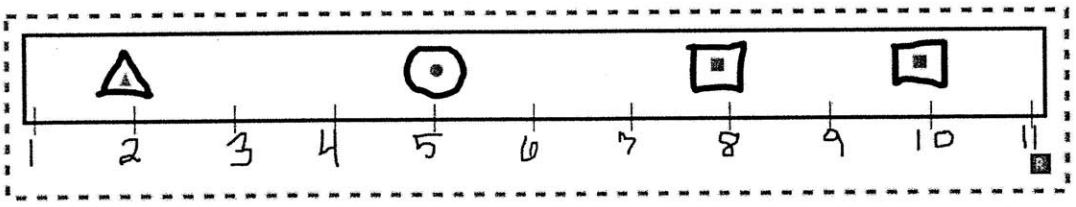



Figure 3-10: An example of a shape recognition Page Object for a number line problem. The red shapes represent the shape and location of the detected ink shapes and are only visible in editing mode when the interpretation button in the bottom right corner is pushed.

An example number line Page Object is illustrated in Figure 3-10. This object simply the shape recognition code detailed in Section 3.1.2 to extract the locations of various shape primitives. After determining what types of shapes are drawn and what ink strokes make up those shapes, the centroid of each set of ink strokes is used to

pinpoint the locations. In editing mode, these extracted locations are visualized with red shapes. When extracting the final answer, the x coordinates of these locations are taken into account to determine where students plotted points along the number line.

# Chapter 4

# Clustering

Grouping student responses is a challenging problem. For any given question, it is impossible to predict all the ways students could approach it, so there is no way to create template representations of all of the types of answers. As a result, a machine learning approach based on classification, or grouping based on a known set of classes, is not compatible. Instead, I approached this as a clustering problem where student responses are grouped based on their similarity to each other.

On top of developing a clustering technique, it was necessary to extract relevant features from the preprocessed sketch data. This task could have been accomplished with a top-down approach, where student work is analyzed by hand to identify important features from an educational standpoint. This task is extremely time consuming, however. Instead, I opted for a bottom-up approach that is initialized with many features and then performs feature selection to automatically extract the most relevant ones. An overall depiction of the architecture of my program is shown in Figure 4-1.

## 4.1 Feature Extraction

The first step of the algorithm involves extracting a large number of features from the student answers. Whether these features end up being relevant or irrelevant to classification does not matter because the algorithm later performs re-weighting through

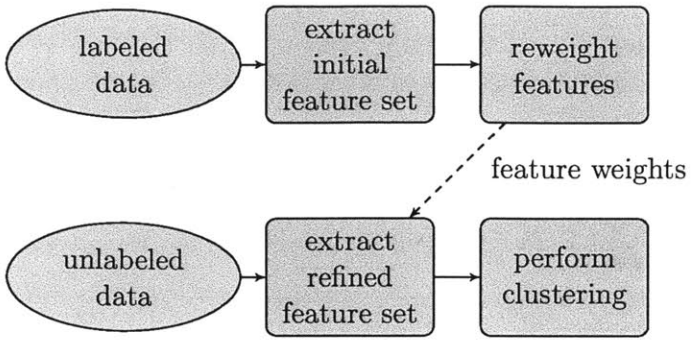



Figure 4-1: The architecture of my clustering module.

a feature selection algorithm and discard features without a sufficiently large weight applied to them.

To test the efficacy of the algorithm, I concentrated specifically on fraction grid problems, summarized in Section 3.2.2. In these questions, students must fill in a specified portion of a bounded area. The sketch preprocessing module turns a set of strokes into a binary bitmap as described in Section 3.1.3. As a result, I analyze the results as if they were binary images and extract the following set of features:

- The percent of filled in squares in the bitmap
- The x and y centroids of filled in squares (average x and y locations of filled in squares)
- The distance between the x and y centroids and the center of the grid
- The sum of the distances between the x and y centroids and the center of the grid
- The minimum x and y values of filled in squares
- The length, width, and area of a bounding box around the filled in squares

For each student response in a set of data labeled by anticipated cluster assignment, I extract these eleven values and place them in a vector. I then normalize each

number by its maximum possible value to weight all features equally and use them as input to my feature selection module explained in Section 4.2 below.

## 4.2 Feature Re-weighting

To extract and upweight the most important features for clustering, I used a variation of the Adaboost algorithm introduced by [12]. Originally used to turn a set of weak classification algorithms into a stronger one, Adaboost up-weights classifiers that correctly categorize the most data points when used in isolation. As described in [14], this approach makes it ideal for use as a feature selection algorithm. Each input feature can be formulated as a simple classifier, and the most accurate and important features will be emphasized through up-weighting.

To use Adaboost in my clustering module, I had to overcome a minor obstacle. Adaboost, like most machine learning techniques, is most commonly used for binary classification. That is, in its typical implementation, it can only say that a data point does or does not belong to a certain class. As a result, I had to extend it to a multi-class representation such that it could be used to distinguish more than two clusters. The multi-class Adaboost technique I use closely resembles that described in [17]. In essence, it turns each feature into a multi-class classifier and compares the classification rate to random guessing, $1/N$, where $N$ is the number of clusters, to achieve the appropriate weights for each feature.

To convert each feature into a multi-class classifier, the algorithm uses a centroid-based technique. Consider, for example, the bounding box area feature. If my labeled data consists of three different clusters, I calculate the average bounding box area for each of the three clusters. Then, when testing the accuracy of this feature for the Adaboost algorithm, I label each data point by the average, or centroid, to which it is closest. If this assigned label matches the anticipated label from the ground truth, the classifier is correct for that particular data point.

Once this accuracy rate is obtained, the rest of the algorithm follows the standard mutli-class Adaboost technique described in [17] and depicted in pseudocode in Algorithm 4. The algorithm undergoes M iterations, where M is the total number of features that are being assigned weights. On each of these iterations, the feature/classifier with the lowest error rate based on the current weights assigned to each data point is chosen and re-weighted based on this error rate. Normally the data points themselves are then re-weighted to emphasize points that had been previously misclassified as described in steps (c) and (d). However, I found that this re-weighting technique is not compatible with the purpose of feature selection because it erroneously favors the features that are chosen first by the algorithm. As a result, I left these steps out of my final implementation. The final output is a weight for each input feature. A threshold can be applied at this point to eliminate features without a sufficiently large weight applied to them. I then use the final set of features and weights in my refined feature extraction algorithm for the final clustering process.

**Algorithm 4** Multi-class Adaboost.

1. *Initialize weights for each data point to* $w_i = 1/n, i = 1, 2, ...n$
2. *For* $m = 1$ *to* $M$*, the number of features*
   (a) *Find the feature/classifier* $T^{(m)}(x)$ *with the lowest error on the data points, each weighted by* $w_i$*, where:*
   $err^{(m)} = \sum_{i=1}^{n} w_i \amalg (c_i \neq T^{(m)}(x_i))$
   (b) *Compute the weight of the feature based on its error with:*
   $\alpha^{(m)} = \log\left(\frac{1-err^{(m)}}{err^{(m)}}\right) + \log(K-1)$ *(*$K$ *= the number of clusters)*
   (c) *Reassign weights to the data points, emphasizing those that were incorrectly classified:*
   $w_i \leftarrow w_i * \exp(\alpha^{(m)} * \amalg(c_i \neq T^{(m)}(x_i))), i = 1, ..., n.$
   (d) *Normalize weights such that* $\sum_{i=1}^{n} w_i = 1$
3. *Return feature weights* $\alpha^m, m = 1, ..., M$

## 4.3 Clustering Method

To perform the clustering process itself, I used DBSCAN, or Density-Based Spatial Clustering of Applications with Noise [4]. This particular clustering algorithm is beneficial because, unlike other methods such as k-means clustering, it does not require *a priori* knowledge of how many total clusters there will be. Furthermore, as the name suggests, it is able to handle situations with noisy data. Consider, for example, a situation in which a student does not follow directions and scribbles an answer to a question that is unlike all the other answers being clustered. While other clustering techniques would attempt to fit this noisy answer with the rest of the data, DBSCAN is able to easily recognize it as noise.

DBSCAN performs clustering based on the concept of density reachability. In essence, this means that it finds groups of points that are no more than a given distance, $\epsilon$, from each other and surrounded by sufficiently many points. These two parameters, $\epsilon$ and the minimum number of surrounding points, must be supplied to the algorithm along with the set of points being clustered. An overview of the process is shown in Algorithm 5.

The algorithm iterates through all the unvisited points in the dataset and determines how many neighbors they have within their epsilon-neighborhood. If this number is less than the required minimum number of points, the point is considered to be noise. Otherwise, a new cluster is formed with the point, its neighboring points, and any other points that are density reachable from the originating point. In the end, each point is either assigned to a cluster or labeled as noise. What is great about this approach is that no *a priori* knowledge about the number of clusters or their content is necessary to conceptually group student responses. Though machine learning techniques for classification are used to re-weight the input features for the clustering algorithm, the clustering itself is dynamic and specific to each individual question instead of being based on a group of known, template-based classes. The

**Algorithm 5** DBSCAN.

```
function DBSCAN(D, eps, minPts)
  currentCluster = 0
  for (each unvisited point P in D) do
    P.visited = true
    N = P.neighbors(eps)
    if (N.length < minPts) then
      P.cluster = NOISE
    else
      currentCluster++
      expandCluster(P, N, currentCluster, eps, minPts)
    end if
  end for
function expandCluster(P, N, C, eps, minPts)
  P.cluster = C
  for (each point P' in N) do
    if (!P'.visisted) then
      P'.visited = true
      N' = P'.neighbors(eps)
      if (N'.length ≥ minPts) then
        N = N.join(N')
      end if
    end if
    if (P'.cluster == NONE) then
      P'.cluster = C
    end if
  end for
```

results from using this approach are detailed below in Section 5.2.

# Chapter 5

# Evaluation

To evaluate Classroom Learning Partner as a whole, the software was deployed and used in actual fourth grade classrooms several times throughout this past year. During this time, I was able to collect and analyze data to determine the accuracy of my various ink interpretation and clustering techniques. Overall the results I obtained are very promising and justify the approach to both interpretation and clustering. They showcase both the strengths and the weaknesses of my implementation, however, and highlight areas that require improvement. Nonetheless, when the algorithms fail, it is clear what is causing the issue. Hence, as described in Chapter 6, it is possible to use this information to build a stronger system in the future.

## 5.1 Ink Interpretation

To determine the accuracy of the ink interpretation algorithms, I ran tests on data obtained the fourth grade classrooms as well as some of my own samples. In most cases, I obtained a quantitative representation of efficacy of each approach. A lack of examples for graph and number line problems, however, lead me to only analyze the data qualitatively with data generated outside of the classroom.

### 5.1.1 Handwriting Recognition

Handwritten student responses are often messier than adult writing and can include spelling mistakes as well. As a result, the handwriting recognition object I created was put through some very rigorous tests during our classroom trials. Overall my tests for handwriting recognition revealed three critical qualities of the interpretation routine:

1. Longer words, series of words, and numbers are generally interpreted more accurately.
2. Handwriting in the absence of stray ink strokes achieves high accuracy rates
3. Specifying a type of interpretation generally improves the analysis result.

Figure 5-1 illustrates the benefits of providing a lengthy response. In this particular case, the provided word, "multiplication", is fairly long and, as a result, will not often be confused with other words. Utilizing an English dictionary to solve for the correct interpretation result, the Microsoft handwriting recognizer is even able to solve the spelling mistake that this student made. Because they often reduce the ambiguity of the final result or provide more context clues, lengthier responses tend to be interpreted with a higher accuracy rate.

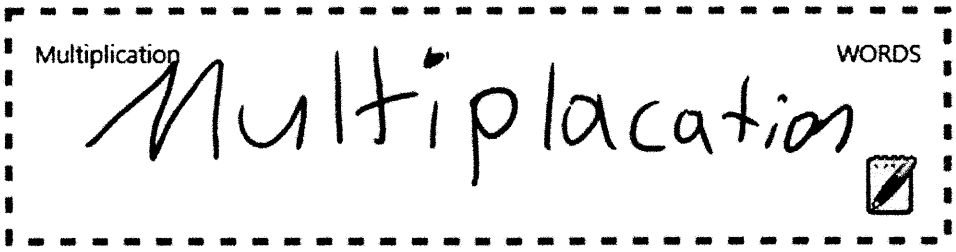



Figure 5-1: A handwriting recognition sample from classroom trials. The interpretation routine is able to reach the correct result despite the spelling mistake the student made.

When student work is isolated from other ink strokes on the page, the handwriting recognizer achieves very accurate results. As shown in Figure 5-2, numbers written in

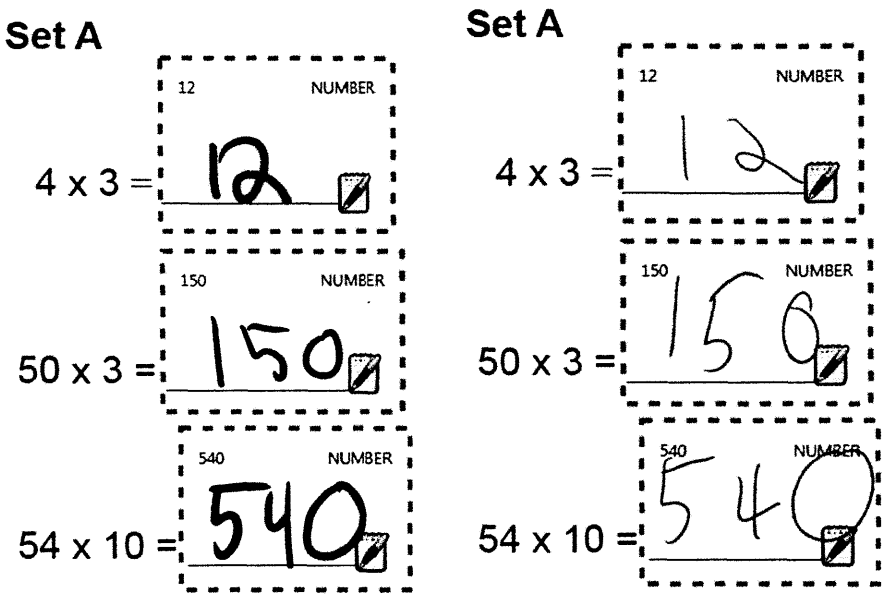



Figure 5-2: Examples showing the accuracy of the handwriting recognition system when the student answer is isolated.

the absence of any stray ink strokes are correctly interpreted by the system. Averaged over all of the student submissions, all three handwriting recognition Page Objects in Figure 5-2 achieve at least 80% accuracy. The center interpretation region, which receives "150" as input, interpreted 87.5% of the total student submissions correctly.

The results in Figure 5-3 demonstrate the power of specifying an expected input type for a handwriting interpretation region. In the exercise related to this question, students had to write a fraction, $\frac{3}{8}$, as their final answer. While numbers and some math-related characters such as addition and multiplication symbols are often recognized by the default classifier, horizontal lines used to designate fractions are usually not understood. Furthermore, the default classifier often fails to recognize the numbers that compose the fraction. By specifying number as the interpretation type, the division symbol can sometimes be recognized more easily. This specification also makes it much easier to extract the numerical components of the answer.

| | Correct | Correct Except for Division Symbol | Completely Incorrect |
|---|---|---|---|
| Default | 4 (20%) | 7 (35%) | 9 (45%) |
| Number | 5 (25%) | 8 (40%) | 7 (35%) |

Figure 5-3: A demonstration of the boosted accuracy obtained by specifying an expected input type for a fraction-based handwriting problem.

Finally, the results in Figure 5-4 show the various weaknesses of the handwriting interpreter. The topmost image represents a case in which the student's handwriting is fairly irregular and difficult to read. Since it likely does not match up with the training examples used to train the Microsoft ink interpretation library, the multiplication symbol is misinterpreted as an 8. The second example demonstrates the effects of extraneous marks within an interpretation region. The recognizer is not able to separate the vertical divided the student drew from the mathematical sentence. As a result, it is interpreted as if it were part of the final answer and throws off the final result. Lastly, the bottom image emphasizes the difficulty of interpreting results when they are stacked vertically. The handwriting recognizer is often not capable of associating parts of an answer when they are not laid out horizontally. Suggestions for eliminating these weaknesses are discussed in Section 6.1.4.

### 5.1.2 Fraction Shading

A quantitative representation of the accuracy of the fraction shading process is depicted in Figure 5-5. In a fraction shading problem where students had to shade in 3/8 of an area, I created a histogram of the shaded fractions that were actually extracted by the interpretation routine. Most results are extremely close to the target value of 0.375. Furthermore, the answers that did not fall within this range were often inaccurately shaded or contained numerous extraneous marks or guideline ink strokes that were not successfully filtered out by the analysis step. As a result, it is evident that this fraction shading approach provides adequate results.

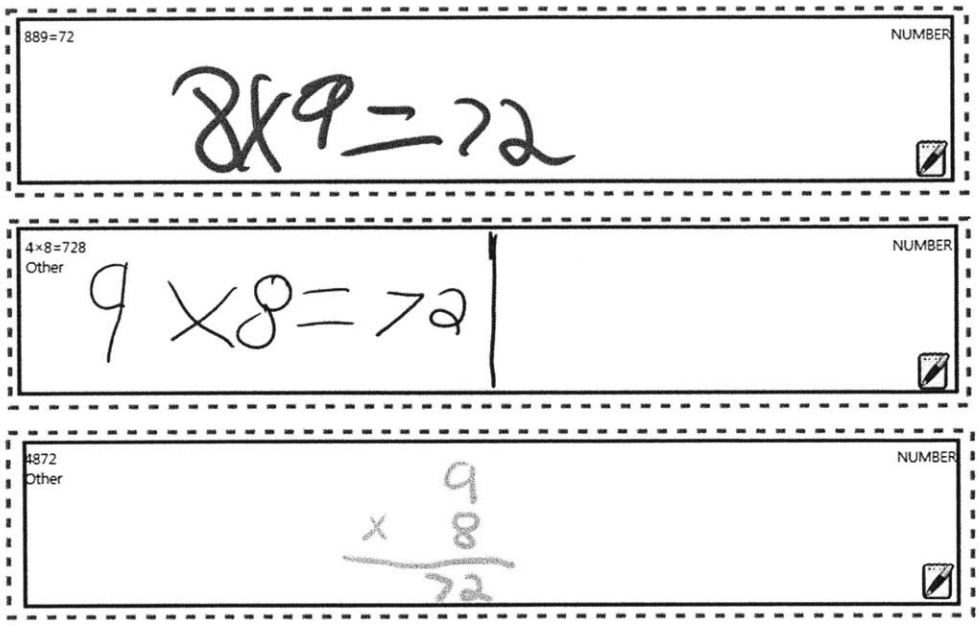



Figure 5-4: Examples illustrating the various weaknesses of the handwriting recognition system.

The fraction shading problems that proved to be difficult to interpret were the result of diversity in the way students supplied their answer. While some children meticulously filled every pixel in a portion of the grid, others supplied a cursory set of scribbles that barely covered the area they mean to shade. This difference can be observed in the clustering examples in Figures 5-14, 5-15, and 5-16, and it would be ideal to extract the same shaded fraction for all similar examples.

To accomplish this goal, I ran several tests to refine the spacing parameter discussed in Section 3.1.3. This value dictates the number of rows and columns in the grid that the strokes are discretized to, so it was necessary to optimize this value and ensure that both densely colored and sparely colored answers result in the same final fraction. One of these tests, summarized in Figure 5-6, uses the average error in the returned result for various spacing values to pick an ideal value. Averaged over many responses to a problem in which the students had to shade in 2/4, or 1/2, of one region and 3/8 of another, this technique centers in on 15 pixels as an ideal value. As a result,

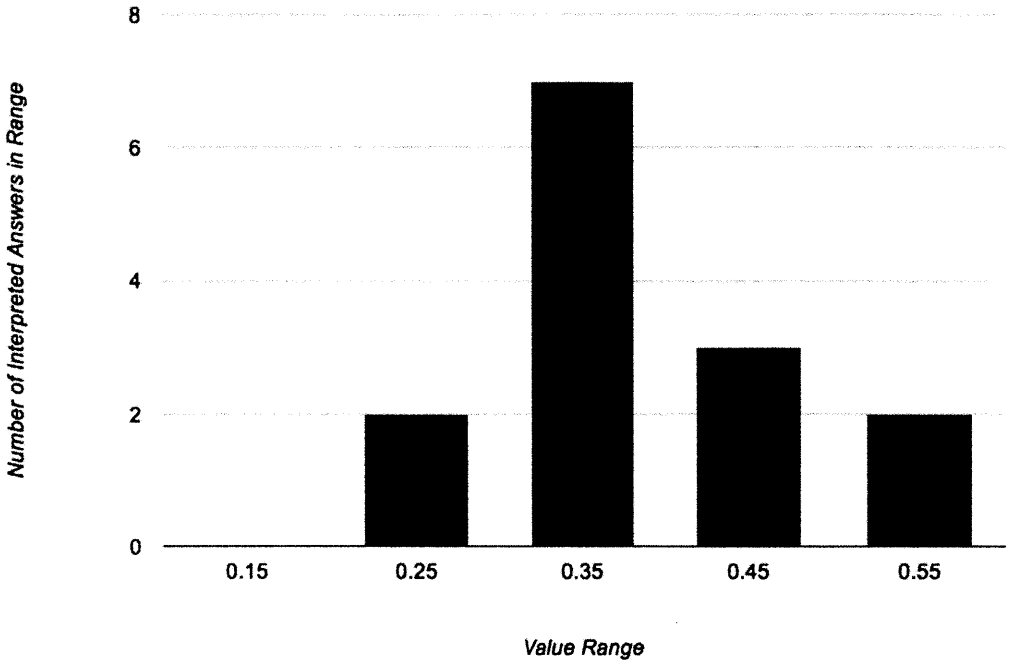



Figure 5-5: A histogram of values returned by a fraction shading objects representing 3/8. The peak is in the range 0.35 to 0.45, which is close to the target value of 0.375.

the rest of my fraction shading tests use 15 as the designated spacing parameter value.

| Spacing | Error for 2/4 | Error for 3/8 | Average Error |
|---|---|---|---|
| 5 px | 0.233 | 0.168 | 0.200 |
| 10 px | 0.094 | 0.053 | 0.074 |
| 15 px | 0.024 | 0.020 | 0.022 |
| 20 px | 0.023 | 0.055 | 0.039 |

Figure 5-6: A summary of the error in fraction shading when using various pixel spacings.

### 5.1.3 Data Tables

To evaluate the efficacy of data tables, I took note of the accuracy of the discretization step that places ink in the grid squares as well as the final handwriting interpretation

result for each cell. Figure 5-7 shows one example problem used for evaluation. For this question, students were required to write the numbers 40 and 8 in the appropriate table cells.

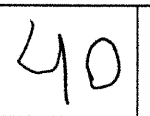
| | | | | | |
|---|---|---|---|---|---|
| **Input** | **8** | **24** | **32** |  | **48** |
| **Output** | **2** | **6** |  | **10** | **12** |

Figure 5-7: A test data table problem. Students had to write the numbers 40 and 8 in the designated cells of the table.

Figure 5-8 shows both the discretization and interpretation accuracy. While the handwriting recognizer struggled to interpret the 8 in many instances, the overall discretization result and interpretation results for the cell containing the number 40 were very accurate. In fact, not a single ink stroke in the entire set of student answers was placed in the wrong grid cell. This illustrates that data tables are a very effective and accurate means of dividing ink strokes. It is an efficient, simple, and useful pre-processing technique that makes further interpretation steps easier. It also illustrates the problems inherent in recognizing fourth graders' handwritten single digit numbers.

| Discr. Accuracy | Interp. Accuracy (40) | Interp. Accuracy (8) |
|---|---|---|
| 100% | 92.9% | 57.1% |

Figure 5-8: A summary of the accuracy in both discretization and interpretation for a sample data table problem.

### 5.1.4 Graphs

Though graph problems were not evaluated within the classroom, I was able to perform a series of tests to check the validity of the segmentation algorithm I employed.

The most critical property that the segmentation algorithm must possess is the ability to reproduce the same set of key points on two graphs with similar geometry, for this allows two graphs to be accurately compared using their sets of key points and the geometry between them. To verify that the segmentation algorithm I implemented has this quality, I drew sets of similar graphs side by side and evaluated the number of key points they have in common.

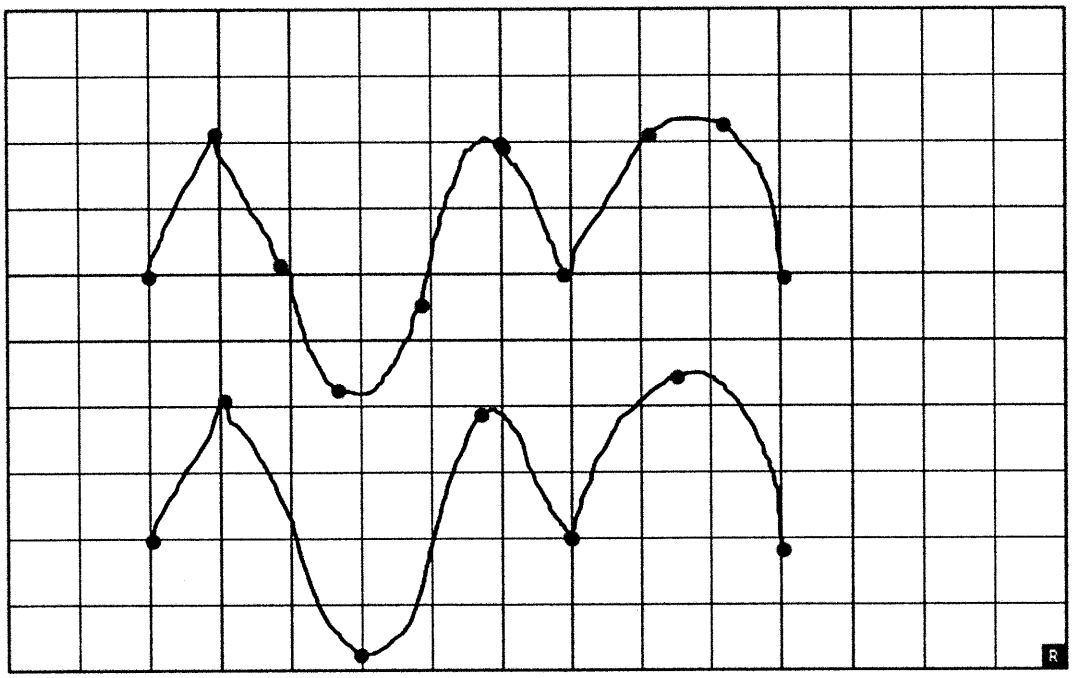


Figure 5-9: A comparison between the key points extracted from two instances of very similar graphs.

Figure 5-9 shows one example of this key point comparison technique. While the top example of the graph contains more extracted points than the bottom example, many critical points are in the same locations in both. As a result, a simple algorithm can be used to find the optimal subset of points shared by both graphs. The average curvature between the sets of shared points can then be used to evaluate if they are geometrically similar.

Another example employing this key point comparison strategy is pictured in Figure 5-10. Once again, there is a definite list of points that coincide between the two graphs. Thus, this implementation of ink stroke segmentation shows promise as a way of analyzing and comparing graph problems. Continuing to implement such a system will likely lead to a much more varied set of problems in Classroom Learning Partner.

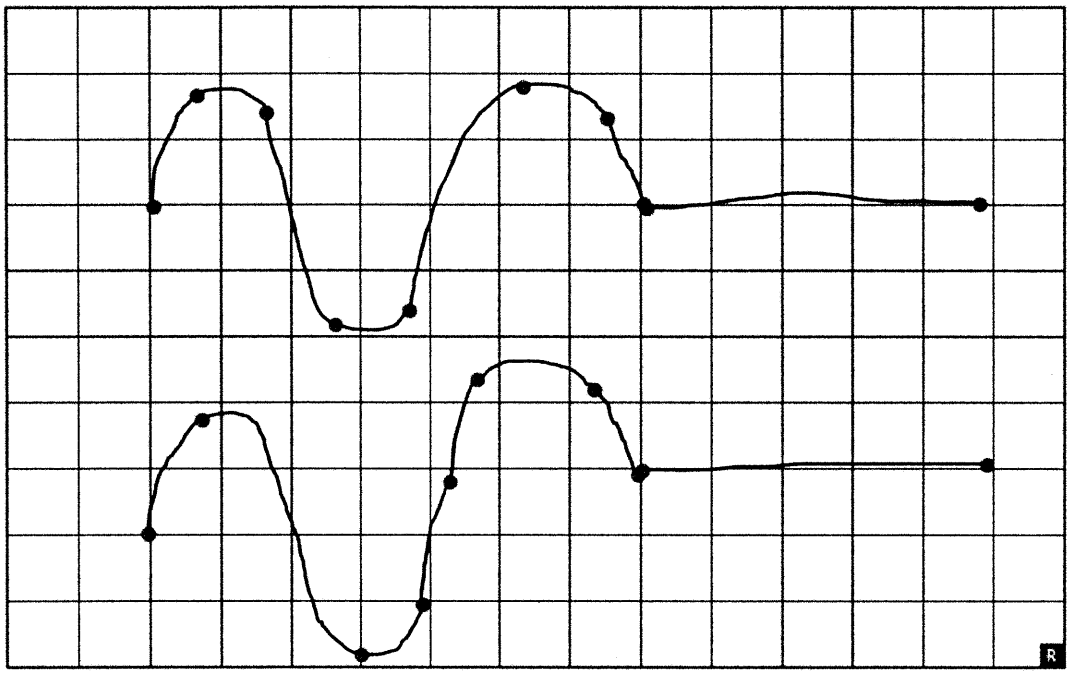


Figure 5-10: Another comparison between the key points extracted from two instances of very similar graphs.

### 5.1.5 Number Lines

Number lines problems have proven to be the most challenging of the problem types chosen for interpretation. Though classroom trials did not yield a significant number of these kinds of problems, this approach fails on even simple examples. Below I illustrate the difficulties inherent in interpreting number lines, and in Section 6.1.4 I discuss several methods for improving this interpretation routine in the future.

In the example in Figure 5-11, three shapes are drawn in the shape interpretation region to signify points along a number line. However, only two of these shapes are successfully interpreted and pinpointed. It appears that this misinterpretation is because the square, drawn in the center of the region, is not perfectly shaped. When shapes are drawn sloppily, the algorithm cannot detect them. Requiring perfectly drawn shapes is not acceptable for use in elementary school classrooms because children's handwriting often leads to imperfect drawings. A more robust algorithm must be found to accomplish this task, perhaps one that can anticipate expected shapes and use that information to coerce an incorrect interpretation into a correct one.

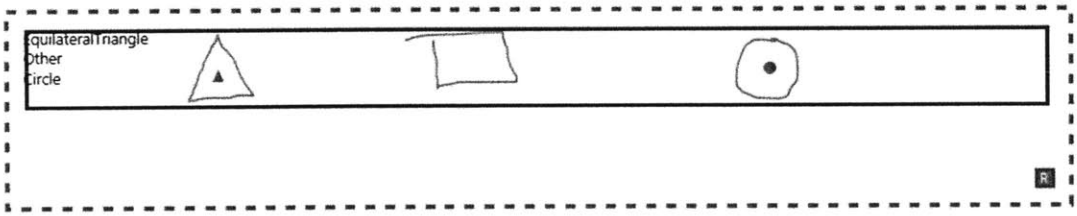



Figure 5-11: An example number line problem where shapes are drawn in isolation.

This approach also suffers a loss in accuracy when stray marks, labels, and scratch work are drawn close to the shapes. In Figure 5-12, letters, numbers, and tick marks are added to the graph, and the previously obtained interpretation results are lost. This result is unfortunate because it undermines the purpose of using a shape recognizer in the first place. This step was meant to extract only the crucial information from a student answer by separating it from scratch work and other extraneous ink strokes. This algorithm does not accomplish this task, so modifications much be made to achieve the desired functionality.

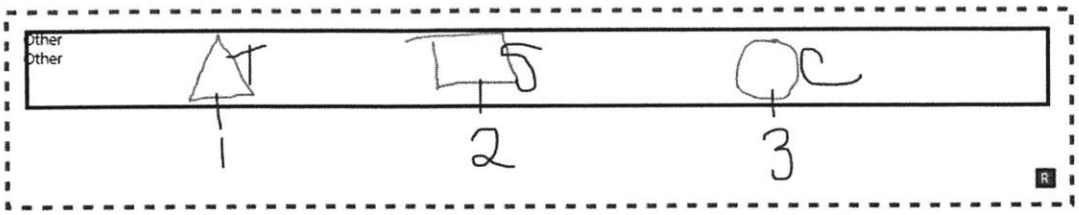



Figure 5-12: An example number line problem where labels and other markings are used.

## 5.2 Clustering

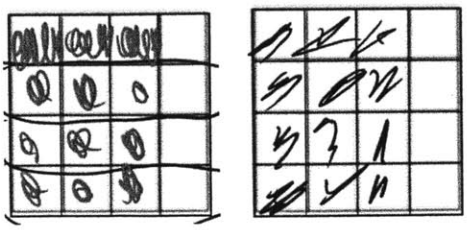
I used several fraction shading problems from the classroom trials to test my clustering approach. I first created a ground truth for a set of answers to one problem by manually grouping them by similarity. By feeding this information to the feature selection algorithm in Section 4.2 along with extracted feature vectors, I discovered a set of optimal feature weights. I then used these weights to run the DBSCAN clustering algorithm on the data used to create the ground truth as well as a distinct dataset. This method gives a good indication of the accuracy of the approach.

The ground truth used as input to the feature selection algorithm is shown in Figure 5-14. The groupings shown were created by hand and indicate which solutions are similar to each other. There are also answers that have no close matches and are not assigned to a distinct cluster. The Adaboost-based reweighing scheme outputs the set of weights shown in Figure 5-13 when the ground truth dataset is used as input.

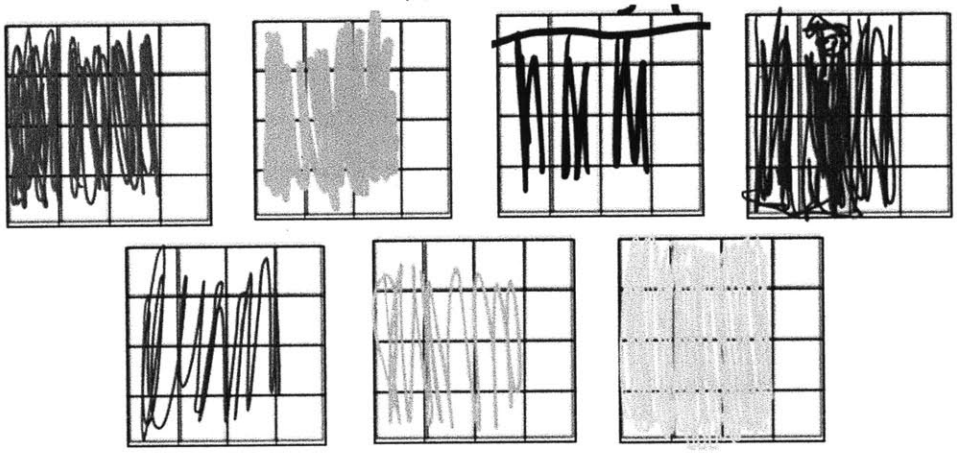
| Percent Filled | Centroid X | Centr. Diff. X | Centroid Y | Centr. Diff. Y |
|---|---|---|---|---|
| 2.15 | 1.91 | 1.91 | 1.91 | 1.91 |
| Diff. X + Diff. Y | B. Box Width | B. Box Height | B. Box Area | Min X |
| 0.92 | 0.00 | 0.92 | 0.25 | 0.25 |
| Min Y | — | — | — | — |
| 0.25 | | | | |

Figure 5-13: The weights output by Adaboost.

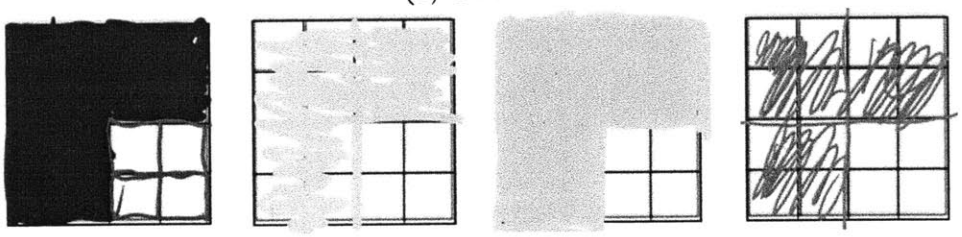
The DBSCAN clustering algorithm achieves the results displayed in Figures 5-15 and 5-16. Though the divisions between groups are not always clear-cut, answers that seem to be out of place are marked with a gray outline. Overall, this results in an accuracy rate of 65% for the ground truth data and 80% for the additional dataset. The results depicted in each of the figures demonstrate that the most popular answers are clustered together with relative ease. It is the edge cases that are sometimes inappropriately assigned to a group. Overall the formed clusters still have very clear visual characteristics in common, so the result is a grouping of student responses that is much easier to glance through. With further refinement of the system, such as the addition of more features to re-weight, this accuracy would likely increase even further. Other possible improvements are proposed in Section 6.2



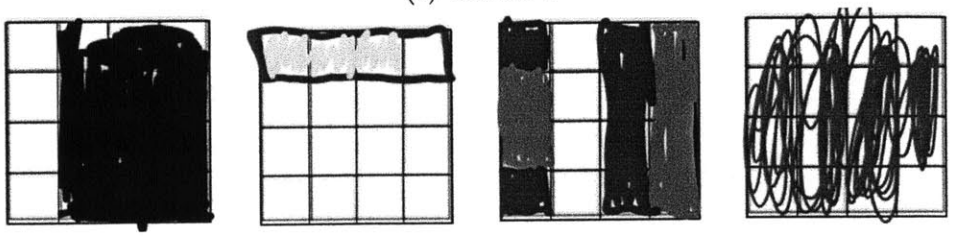
(a) Cluster 1



(b) Cluster 2

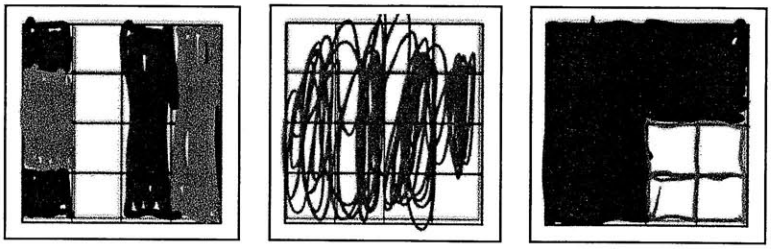


(c) Cluster 3

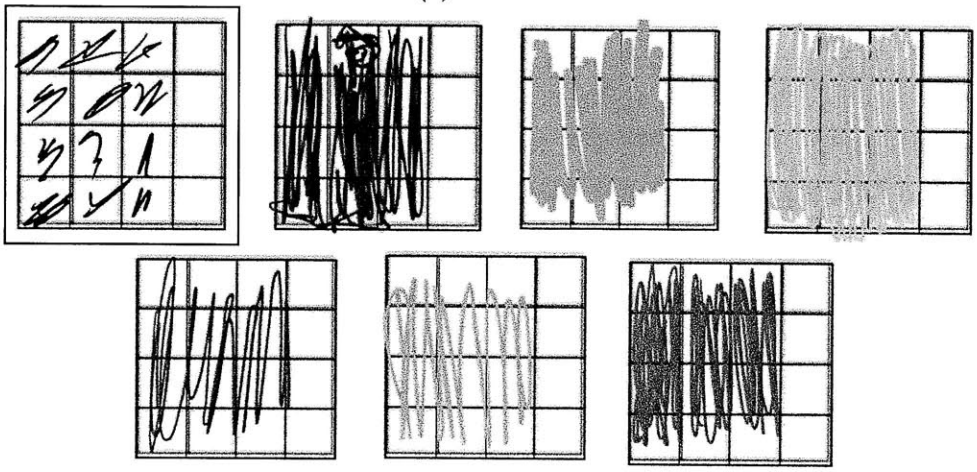


(d) No assigned cluster

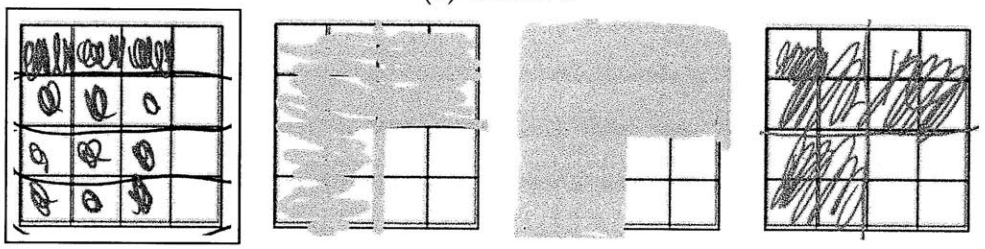
Figure 5-14: The ground truth for the feature selection algorithm.



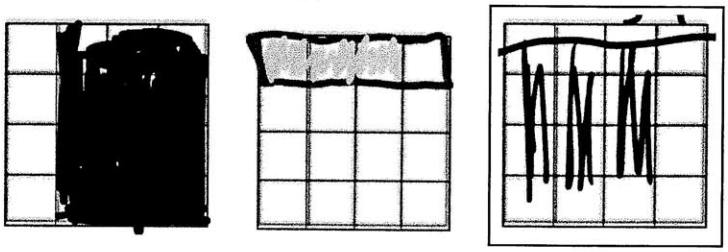
(a) Cluster 1



(b) Cluster 2

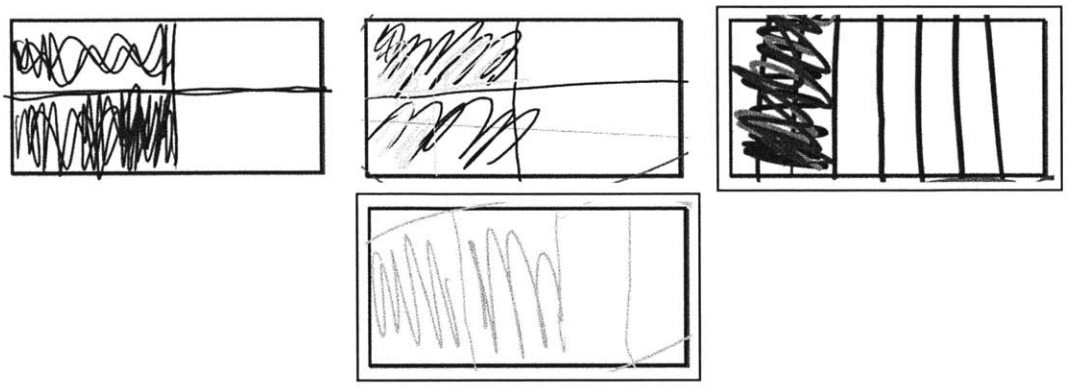


(c) Cluster 3

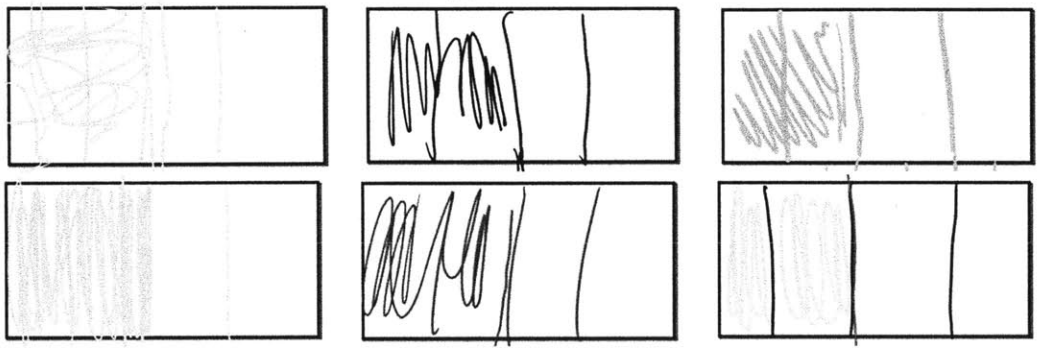


(d) No assigned cluster

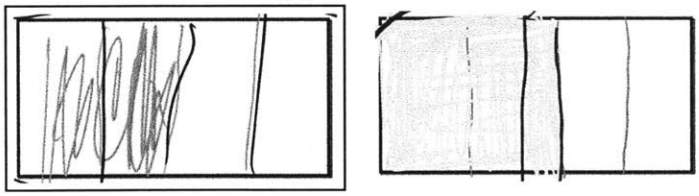
Figure 5-15: The clustering result for the ground truth.



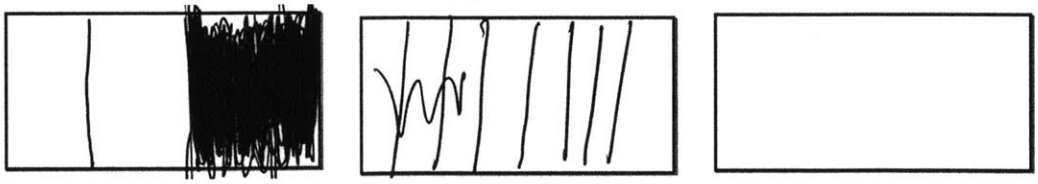
(a) Cluster 1



(b) Cluster 2



(c) Cluster 3



(d) No assigned cluster

Figure 5-16: The clustering result for an alternate dataset.

# Chapter 6

# Future Work

This thesis introduces many new features to Classroom Learning Partner. While the methods I have designed, implemented, and tested show promise, much more work must be done to provide deeper integration into the system, develop a more intuitive user interface, and ensure interpretation and clustering accuracy. If the following improvements are made to the current system, the result will likely be a very intuitive, useful, and robust ink analysis framework.

## 6.1 Interpretation

The current implementation of ink interpretation in Classroom Learning Partner has two weaknesses. First, the interpretation results returned by the ink analysis routines are not used to their fullest extent. That is, the interpreted answer of an ink region is not displayed in an intuitive way in the student or teacher user interface. Secondly, the interpretation methods themselves need to be refined to improve accuracy and to work in combination with other input methods such as stamps. Fixing these issues using the five improvements suggested below will lead to a much more useful and robust system.

### 6.1.1 Determining Correctness with Constraints

To make better use of the interpreted results coming from the ink analysis routines, the system requires an easy means of specifying a correct answer to an ink-based problem as well as a reliable way to check interpreted results against this answer. This problem is far from trivial because it is difficult not only from an ink analysis standpoint but also from a user interface perspective.

Consider, for example, a simple math question where the answer is the expression $6 * 5 = 30$. It would be simple for the teacher to specify this particular answer as the correct answer if it is the only valid way to solve the problem. Depending on the question being asked, however, the commutative property of multiplication could allow $5 * 6 = 30$ to be an equally correct answer to the question. If the specific operator used in the final expression is not important to the question, something such as $30 \div 5 = 6$ or $30 \div 6 = 5$ may also constitute a correct response. Clearly, providing all the different ways a question could be answered is tedious and far from an ideal solution. As a result, a correct answer must actually be specified as a set of constraints. This example problem could, for instance, have the following set of constraints when checking for correct answers:

1. Uses the numbers 6, 5, and 30
2. Uses the multiplication operator
3. The expression is mathematically correct

This set of constraints would allow for only $6*5 = 30$ and $5*6 = 30$ to be provided as correct answers. In cases where there are many different correct answers to a question, this method of using constraints takes much less effort. Integrating such a system into Classroom Learning Partner's user interface may prove challenging, however. The process of creating and adding these constraints to the interpretation Page Objects described in Section 3.2 must be intuitive and efficient in order to avoid confusion and facilitate fluid problem development. Furthermore, a robust constraint

checking system needs to be built to compare the interpreted result against the set of constraints. Both of these tasks are very challenging and will require a great deal of effort. Nonetheless, taking these steps will make interpreted ink much more useful.

### 6.1.2 Real-Time Interpretation

Another step toward increasing the usefulness of interpreted ink involves running the interpretation methods in real-time. In other words, the analysis routines outlined in Chapter 3 should be running in the background as the interpretation Page Objects receive ink strokes instead of only when student work is being submitted or saved. This approach is advantageous because it would make it possible to give students meaningful feedback about their work as they are writing their answer. Instead of submitting responses to the teacher and waiting for her to review the interpreted answers, the interpreted result could be used on the student's machine. For instance, the system could warn students if they are supplying a wrong answer and even give suggestions for fixing it. As mentioned in Chapter 1, it is important to provide fast and accurate feedback to facilitate effective learning, and providing feedback in real-time based on interpreted results would certainly fulfill this requirement. Effort would need to be invested in tailoring this feedback so that it is helpful without being distracting or invasive, but the overall result could be extremely helpful from an education perspective.

Accomplishing this task is also technically challenging due to the sometimes sluggish nature of the ink analysis methods. Running these routines on the main program thread would lock the user interface, so it is necessary to use multithreading. Performing the analysis on a separate thread allows users to continue writing and interacting with the program while ink is interpreted seamlessly in the background. Multithreading, however, always presents obstacles and challenges. In experiments run to test a multithreaded approach, the interpretation code did not interact well with the process of submitting work to the teacher machine. Future efforts focused on making this process more stable could lead to a very interesting and effective use of interpretation results.

### 6.1.3 User Interface Integration

Overall the problems mentioned in Section 6.1.1 and Section 6.1.2 emphasize a need for better integration of interpretation routines with Classroom Learning Partner's user interface. Though the analysis routines are obtaining and storing useful results, this information is not yet being displayed in to teachers or students. While the interpretation-based Page Objects have a means of viewing the interpreted result within editing mode on the teacher machine, this was mainly implemented to quickly gather accuracy results. It is not intended or well-suited for use in an actual classroom situation.

Once important properties such as correctness information can be extracted from student answers using the constraint-based method in Section 6.1.1, time should be spent adding ways to display this information on the teacher's machine. All extracted data should be shown in a concise and easily viewed fashion so that the teacher can quickly and easily use ink interpretation results and sort through student responses faster. Again, this addition is extremely important because it is critical for expediting feedback about student responses. By making submitted student answers easier to read through, the process of sorting and responding to each submission is much more efficient for the teacher.

### 6.1.4 Refining Interpretation Methods

Though all of the interpretation routines developed provide adequate results in most situations, there is room for increased accuracy. In particular, the handwriting recognition method described in Section 3.1.1 and the shape recognition approach in Section 3.1.2 do not function well when used in non-ideal situations. Whenever such regions receive stray marks or scratch work, they are often incapable of providing any useful interpretation information. Because student work is often unpredictable and difficult to read, these methods are less than ideal for use in Classroom Learning Partner.

This issue can be dealt with in several ways. The first and most simple solution involves adding a stroke preprocessing module to the current methods. This preprocessing step could sort through the input strokes and attempt to distinguish scratch work from the final answer. That way, once strokes are finally sent to the handwriting or shape recognition algorithms, the input is more refined and less likely to contain noisy or irrelevant data. Such a task could be accomplished by filtering strokes by shape, pen pressure, position, or size. A machine learning approach might even be applicable. A classifier could be trained on examples of strokes critical to the answer and scratch work, and then it could be used to extract the relevant components of a set of strokes.

The second and much more work-intensive approach would require discarding the current handwriting and shape recognition techniques altogether and starting from scratch. Both methods rely on Microsoft ink interpretation libraries, so changing the methods would require that the entire system be built from the ground up. While this might be possible for the shape recognition system due to the relative simplicity of the possible shapes, creating a handwriting recognition system from scratch would be much more difficult.

### 6.1.5 Combined Interpretation Methods

While ink-based responses make up the majority of the student answers we've seen with Classroom Learning Partner, they are not the only types of answers students can provide. As mentioned in Section 1.2 and shown in Figures 1-1 and 1-2, stamps and snap tiles are also important means of responding to a question. Furthermore, ink is often used in conjunction with these methods to supply a final answer. For instance, several stamps or tiles can be circled by an ink stroke to indicate they are grouped together. In the future, it will be important to combine ink interpretation methods with information about stamps and tiles to arrive at a combined interpretation result.

As stated in Section 3.1.2, a shape recognition method is well-suited for this purpose. To determine what stamps or tiles have been circled by an ink stroke, for instance, the shape analysis routine can extract sets of ink strokes that compose circular or elliptical shapes. An intersection checking routine can then be run on the stamps or tiles on the page to determine which ones fall within each circle or ellipse. Adding this functionality will expand the variety of problems that can be interpreted and open up possibilities for additional question types. As stated above, however, the shape interpretation method will have to made much more robust to be utilized in this sort of situation.

## 6.2 Clustering

The clustering technique summarized in Chapter 4 produces good results for fraction shading problems. However, there are many possible extensions to this system that will make the system easier to work with and more useful. As with the proposed improvements to the ink interpretation system above, all of the clustering improvements involve better user interface integration and extended functionality.

### 6.2.1 Displaying Clustering Results

The results that are obtained from the clustering system are not yet tied into the teacher's user interface. As a result, the benefits from having such a system in place, such as making a set of student responses more readable, are not yet taken advantage of. In the future, the output of the clustering algorithm should be utilized to physically group the student responses in the user interface. The framework for accomplishing this task already exists within the Classroom Learning Partner code, so achieving this goal should be straightforward. It is a matter of properly hooking up the clustering module with the user interface.

### 6.2.2 Extension to More Problem Types

Though I concentrated on getting accurate results for clustering on fraction shading problems, there are clearly many more types of questions that should also be clustered. Handwriting problems, data tables, graphs, and number lines require a grouping method as well. The most challenging aspect of applying the same clustering method to these other varieties of problems is properly extracting a feature vector that numerically describes the answer. The features taken from the binary bitmap representation of a shading region are clearly not applicable to the key points and curvatures of a graph. As a result, time must be invested to develop a new list of important features for each type of problem. These features can then be fed into the feature selection algorithm detailed in Section 4.2 to extract and upweight the most important components. This compromise between a top-down and bottom-up approach should develop a useful set of features for each type of problem without too much effort.

### 6.2.3 Dynamic Clustering

When it comes to clustering student responses, there is often more than one way to divide up the group. If the teacher cares about certain aspects of the answers more than others, an alternate clustering result can be formed. For instance, a teacher may only be interested in finding all students that set up an math problem correctly but obtained the wrong answer in the end. When going through this search, only a fraction of the characteristics of the student answers are important. This situation is analogous to assigning more weight to certain features of the extracted feature vectors. Features that pertain to what the teacher is looking for could be assigned more importance in order to change the outcome of the final clustering. In this fashion, teachers could not only view automatically clustered results but dynamically change them based on the features they find important. If successfully implemented, such a system could make the process of sorting through student answers even more efficient.

# Chapter 7

# Conclusion

Interpreting the meaning of digital ink strokes is an incredibly challenging task, and, when working with elementary school children, the unpredictability of student work adds another layer of difficulty. This thesis has made significant progress toward accurate interpretation methods that gracefully handle this unpredictability. This work expands and improves upon the Recognition Grid approach that was previously used in Classroom Learning Partner. A new library of varied interpretation methods inspired by both offline and online ink analysis techniques modularizes the interpretation and will prove useful to future efforts. By approaching the problem with several different strategies, this work establishes a framework that can be expanded to many different types of problems in the future.

This work also lead to a compelling approach for automatically clustering student responses. By combining machine learning techniques with a clustering algorithm, the approach is guided by training examples but can group data dynamically without templates. While there is room to improve the system further, the foundation created exhibits the potential to improve the efficiency of Classroom Learning Partner. With the alterations suggested for both interpretation and clustering, it will be much faster and easier to give students constructive feedback on their work.

# Bibliography


[1] P. Black and D. William. *Inside the Black Box: Raising Standards Through Classroom Assessment.* King's College, London, 1998.

[2] J. Bransford, A. Brown, and R. Cocking. *How People Learn: Brain, Mind, Experience, and School.* National Academy Press, Washington D.C., 2000.

[3] N. Chao. Analyzing the impact of structure in handwriting recognition software. Master's thesis, Massachusetts Institute of Technology, May 2011.

[4] M. Ester, H. Kriegal, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining*, 1996.

[5] G. Hamerly and C. Elkan. Alternatives to the k-means algorithm that find better clusterings. In *Proceedings of the Eleventh International Conference on Information and Knowledge Management*, pages 600–607, 2002.

[6] M. Jozwiak. Identification and presentation of student answers to in-class exercises in classroom learning partner. Master's thesis, Massachusetts Institute of Technology, May 2011.

[7] L. Kara and T. Stahovich. An image-based, trainable symbol recognizer for hand-drawn sketches. *Computers and Graphics*, 29:501–517, August 2005.

[8] K. Koile, K. Chevalier, C. Low, S. Pal, A. Rogal, D. Singer, J. Sorensen, K.S. Tay, and K. Wu. Supporting pen-based classroom interaction: New findings and functionality for classroom learning partner. In *First International Workshop on Pen-Based Learning Technologies*, 2007.

[9] K. Koile, K. Chevalier, M. Rbeiz, A. Rogal, D. Singer, J. Sorensen, A. Smith, K.S. Tay, and K. Wu. Supporting feedback and assessment of digital ink answers to in-class exercises. In *Proceedings of the Nineteenth Conference on Innovative Applications of AI*, July 22-29, 2007.

[10] K. Koile and D. Singer. Improving learning in cs1 via tablet-pc-based in-class assessment. In *Proceedings of ICER 2006 (Second Internation Computing Education Research Workshop).* University of Kent, Caterbury, UK, September 9-10, 2006.

[11] M. Rbeiz. Semantic representation of digital ink in the classroom learning partner. Master's thesis, Massachusetts Institute of Technology, May 2006.

[12] R. Schapire. A brief introduction to boosting. In *16th Internatonal Joint Conference on Artificial Intelligence*, 1999.

[13] M. Sezgin and R. Davis. Sketch interpretation using multiscale models of temporal patterns. *IEEE Computer Graphics and Applications*, pages 28–37, Jan 2007.

[14] P. Silapachote, D. Karuppiah, and A. Hanson. Feature selection using adaboost for face expression recognition. In *The 4th IASTED International Conference on Visualization, Imaging, and Image Processing*, 2004.

[15] T. Stahovich. Segmentation of pen strokes using pen speed. *AAAI Fall Symposium Series: Making Pen-Based Interaction Intelligent and Natural*, 2004.

[16] K.S. Tay. Improving digital ink interpretation through expected type prediction and dynamic dispatch. Master's thesis, Massachusetts Institute of Technology, May 2008.

[17] J. Zhu, S. Rosset, H. Zou, and T. Hastie. Multi-class adaboost. *Statistics and Its Interface*, 2:349–360, 2009.