# Interpretation and Aggregation of Marks in Classroom Learning Partner

by

**Kenneth D. Wu**

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

January 30, 2008

Author ______________________________________________
Department of Electrical Engineering and Computer Science
February 1, 2008

Certified by ______________________________________________
Kimberle Koile, Ph.D.
Research Scientist, MIT CSAIL
Thesis Supervisor

Accepted by ______________________________________________
Arthur C. Smith, Ph.D.
Professor of Electrical Engineering
Chairman, Department Committee on Graduate Theses

# Interpretation and Aggregation of Marks in Classroom Learning Partner

by

**Kenneth D. Wu**

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

Abstract:

This thesis explores the mark understanding problem in the context of a Tablet-PC-based classroom interaction system. It presents a novel method for interpreting digital ink strokes on background images, and aggregating those interpretations. It addresses complexity of mark interpreters and development and acquisition of a representation of a contextual background. It details the design, implementation, testing, and plans for future extension of a mark interpreter and aggregator in the Classroom Learning Partner, our classroom interaction system.

**Thesis Supervisor**: Kimberle Koile, Ph.D.
**Title**: Research Scientist

## Acknowledgments

Many people have assisted and inspired me during the creation of this thesis.

First and foremost, I would like to thank my supervisor Kimberle Koile, who has been a very patient and understanding advisor ever since I started to work with her, even despite her extremely busy schedule. Also, my fellow members in the group have brightened my time here with their cooperation and camaraderie: especially Kevin Chevalier, whose work was parallel to mine, and who spent much time with me discussing various concerns about both our projects; Adam Rogal, whose dry humor made sure there wasn't a dull moment in lab and whose expertise with the code base was invaluable to completing this project; Kah Seng Tay, who has been a great help and friend both inside and outside working at CLP; Michel Rbeiz, who laid the foundation for my work by teaching me much that I know about sketch recognition; and Capen Low, whose many hours of toil have resulted in the instructor authoring tool that is crucial to the success of this thesis.

I am very grateful to my friends over the last few years at MIT, who have kept me focused, sane, and sometimes, awake until very late in the night; to my various mentors, who have been a great influence on me and kept me on the correct path throughout life; and of course, to my parents for their support, understanding, and love.

## Table of Contents

## Table of Figures

# 1 Introduction

Classroom Learning Partner (CLP) is a prototype classroom presentation tool developed to facilitate and enhance the processes of teaching and learning by allowing instructors and students to receive live feedback from student in-class exercises.  Students complete the exercises on individual Tablet PC machines and submit their answers wirelessly and anonymously via CLP to the instructor during class.  CLP automatically interprets these answers, separates them into equivalence classes or bins, and displays to the instructor a histogram, detailing what kinds of answers were given, along with a representative answer for each bin. This automatic grouping of answers, similar to that produced by wireless polling systems (Duncan 2005), allows the instructor to view class results without being overwhelmed by a large number of answers.  Unlike polling systems, however, CLP allows students to write their answers rather than choosing from a predefined set.  This information empowers the instructor to determine on the spot whether students sufficiently understand the subject material, and whether to proceed to the next topic or to continue pursuing the current one.  When the instructor displays and discusses representative answers with the class, students also receive feedback about their own understanding.  CLP has been shown to successfully boost the performance of students in 6.001, the introductory computer science course at the Massachusetts Institute of Technology, using exercises that take alphanumeric strings as answers (Koile et al. 2007a, Koile et al. 2007b).

CLP's applicability to teaching computer science, as well as other fields such as chemistry and physics, however, is hampered by an inability to process specialized inputs, such as the box-and-pointer diagrams and environment diagrams that are commonly used to visualize data structures, variable scope, and program flow in Scheme, the language introduced in 6.001 (Abelson and Sussman 1996). Without the ability to read in and understand these specialized kinds of answers, CLP would be unable to support the wireless interaction described above, as the grouping of answers into equivalence classes is critically dependent on machine interpretation of the answers. This thesis presents a novel extension of CLP that allows it to handle exercises whose answers are in the form of markings on a background image, a type of input that we will hereafter refer to simply as marking. An example of a marked image in the fields of chemistry and physics can be seen in Figure 1 below.

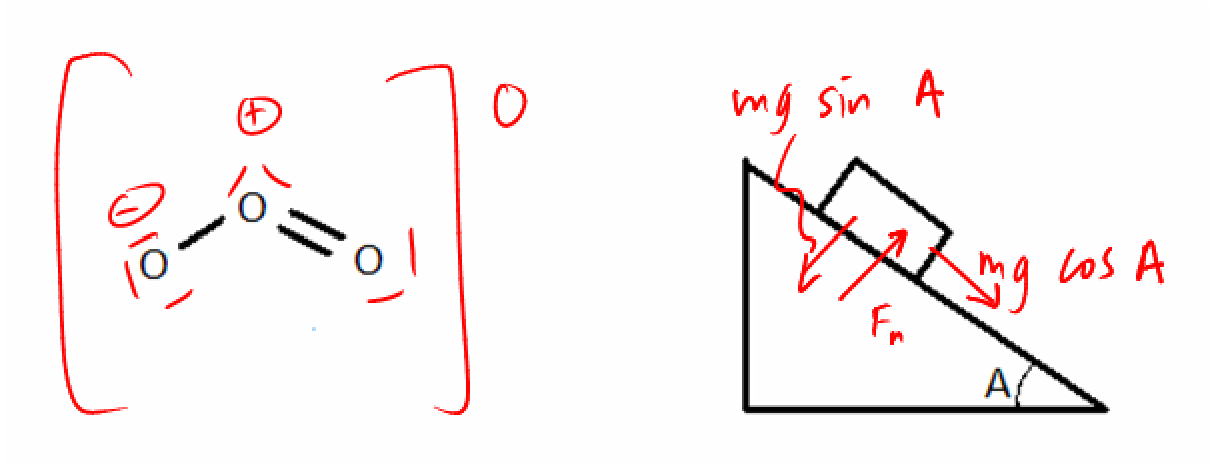


*Figure 1: An example of marking two sample in-class exercises in freshman science curricula, one in chemistry, and the other in physics.*

Marking can be seen as a subset of sketch recognition, depending on the broadness of the definition of what constitutes sketch recognition. Our definition of marking differs from the traditional model of sketch recognition in that, unlike sketch recognition systems, a marking understanding system need not understand the meaning of the background image; for some applications, ours in particular, it is sufficient for a system to recognize the location and extent of drawn marks. A true "understanding" of marks would have to rely upon a sketch recognition technique that understands the mark itself and contains a model of an underlying object associated with the mark.

This thesis aims to explore the mark understanding problem for a classroom interaction system. It is a prototype implementation of a solution to the problem in our classroom system, CLP, and presents ideas for future implementations of solutions in both CLP and other applications. This document is organized as follows: first, we define and examine the mark understanding problem, considering possible ways to implement a mark understanding system for different applications, determining how to classify different types of marks, and understanding what kind of tools we will need to tackle the difficulties. We identify what, if any, commonalities and differences might exist between different possible mark understanding methods, and describe a technique for use with CLP. If such commonalities exist, identifying them and listing them can provide a starting point by which others, whether in academia or the private sector, may approach mark understanding. By sorting out the differences, on the other hand, we can formulate ways to reconcile these differences and define a unified approach for implementing of a mark understander. We then provide a quick overview of CLP, explore how we apply our discussion of mark understanding to the specific domain of classroom interaction, and detail the steps along the way to integrating our solution into our prototype architecture. Last, we evaluate the success of the CLP implementation and discuss future extensions to both the CLP implementation and implementations of mark understanding in general.

# 2 The Mark Understanding Problem

Mark understanding can be useful in many applications, such as in games, business software suites, automatic form-fillers, and music composition tools for pen-enabled computers like the Tablet PC, or for specially enhanced pens like those by the Anoto Group (anoto.com). Because the kinds of things being marked in such applications differ greatly, different approaches are required for each application, whether in modeling the background or in deciding what exactly each mark signifies. In spite of these differences in approach, however, we have identified a common process among applications employing a marking understanding component. In this section, we discuss three possible marking applications, their varying needs for marking understanding tools, and the three steps in the process.

1. **Mark Analysis**. A mark detector uses data about the mark itself to narrow down the possibilities of what it could mean.
2. **Context Identification**. The meaning of a mark is tied to an area of the background data in some way. This area is the context of the mark, and it must be located.
3. **Context Analysis**. The context is further analyzed so that the meaning of the context can be understood sufficiently to fully understand the mark.

At the same time, applications will differ from each other in significant ways.

1. **Location of Mark Entry**. Marks may be input either directly on top of background data, or separately from background data. If they are placed on background data, then they are to be evaluated at a certain salient point or points of the mark with respect to the

background data. If they are separate from background data, there must be some mechanism by which a location on the background data can be selected and indicated.

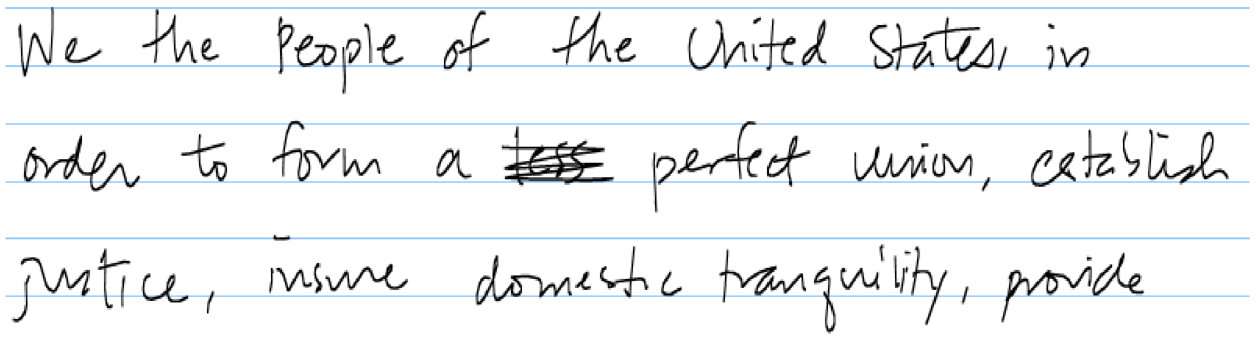
2. **Variation in Physical Mark Complexity and Mark Meaning**. Marks may not have a one-to-one correspondence with meanings, and they may consist of one stroke, several strokes, or may even be input images with an undetermined number of strokes. To deal with variation in types of marks, mark meanings, and mark complexity, one must then suitably adjust the complexity of the mark recognizer. The recognizer must be able to differentiate between different shapes of marks, decide on a meaning or number of meanings depending on the mark shape, and then pinpoint the location or locations on the background image to which the mark corresponds.

3. **Variation in Mark Context**. The underlying representation of the background data must be sufficiently sophisticated so that the meaning of the context can be understood, and this understanding must be sufficiently sophisticated enough for the mark in its entirety to be understood.

With these points in mind, we now proceed to examine the design of solutions for different applications.

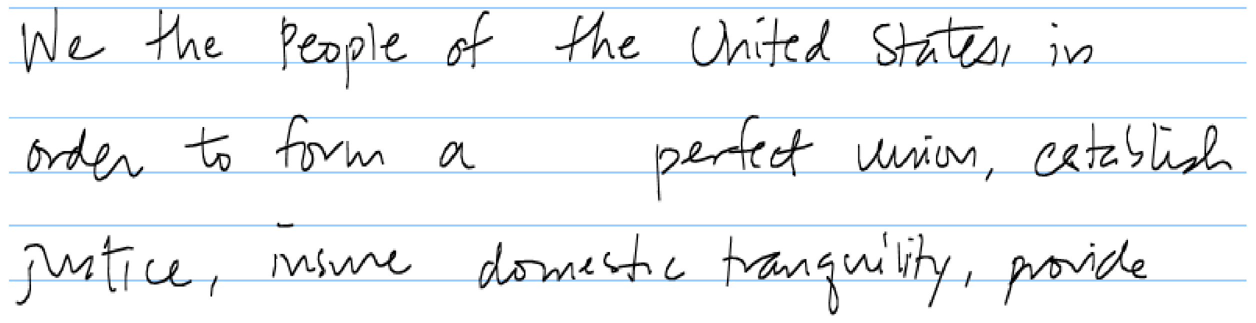
## *2.1 Gestures*

We first consider gestures, a type of interaction currently used in various kinds of software on Tablet PCs (Jarrett and Su 2003). With this kind of interaction, a user touches a pen-like stylus to the computer screen and moves the pen tip in some way to command the computer to carry out a task. Depending on which gesture is being performed, gestures can create visible strokes that are drawn, interpreted, executed, and then automatically deleted; or they may not be displayed on the screen for the user to see, but simply interpreted and executed. The obvious advantage of showing the user the mark lies in user feedback; that is, the user can see exactly what he or she drew on the screen, so that if the gesture is not read correctly, or not read at all, the user can modify the way in which he or she drew the gesture to improve the chance of its being interpreted and executed correctly. Gestures may be performed directly over the background upon which the command must act, or they may be performed in a separate input window. In both cases, though, the Tablet PC provides an infrastructure by which gestures are recognized and evaluated. The example in Figure 2 contains a gesture in Windows Journal, nicknamed the “scratch out” gesture, which deletes all the strokes through which the gesture crosses.



We the People of the United States, in order to form a ~~less~~ perfect union, establish justice, insure domestic tranquility, provide

*Figure 2a: The scratch out gesture over the word “less.”*



We the People of the United States, in order to form a perfect union, establish justice, insure domestic tranquility, provide

*Figure 2b: The Journal pad after the gesture is evaluated.*

Another example given in Figure 3 below shows the gesture that is equivalent to inserting a line break in Microsoft Word, this time while using the handwriting recognition feature of the Tablet PC.

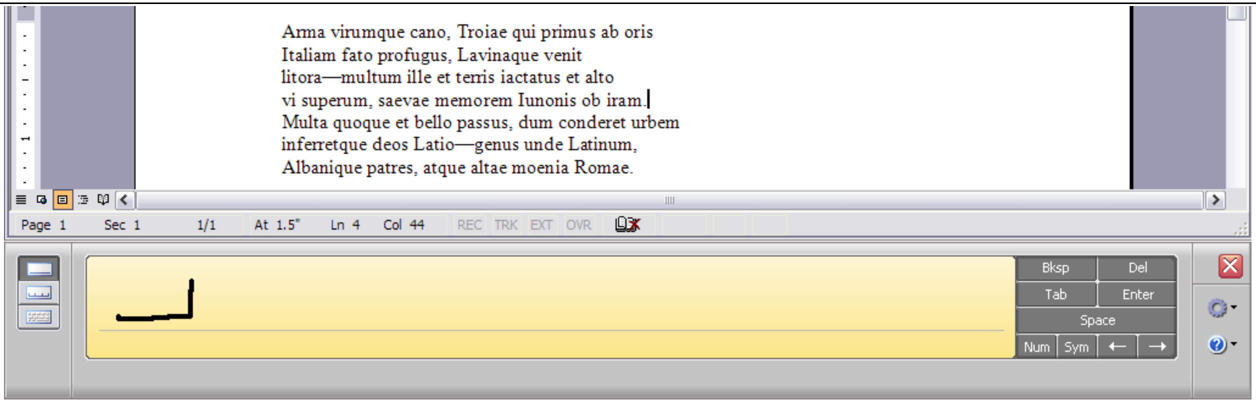

Arma virumque cano, Troiae qui primus ab oris
Italiam fato profugus, Lavinaque venit
litora—multum ille et terris iactatus et alto
vi superum, saevae memorem Iunonis ob iram.
Multa quoque et bello passus, dum conderet urbem
inferretque deos Latio—genus unde Latinum,
Albanique patres, atque altae moenia Romae.

*Figure 3: The gesture drawn in the input area below the main window will insert a line break at the cursor, at the end of the fourth line.*

Because the handwriting recognition on the Tablet PC is separate from the actual word processor itself, the gesture in the second example is different: the mark is not made on a previously created image. In the Journal application shown in Figure 2, the gesture was recognized to be a scratch-out gesture, and the command for deletion was evaluated on the previously created strokes that the gesture intersected. In the Word application in Figure 3, on the other hand, the gesture was recognized to be equivalent to an Enter key button press, and the handwriting recognizer was able simply to tell Word to create a line break. This difference, however, is superficial at best. In both of these applications of gestures, pen input is recognized and evaluated based on the state of the program, or more specifically, the gesture causes a change at a certain position on the screen. The only difference is that while screen position is not abstracted from gesture evaluation in the Journal example, it is abstracted in the Word example by the input window passing a command to the Word program, which inserts the line break at the current position of the cursor.

While the Journal implementation of mark understanding via gestures depends on traditional, well-documented handwriting and symbol recognition and interpretation methods, it also depends implicitly on a detailed representation of the underlying image to supply the contextual information concerning which commands must be executed and which objects on which to execute those commands. In Journal, the underlying representation for the image presumably contains far more information than could be contained in a pixel matrix of brightness values. This representation can easily be built up because the objects manipulated in Journal were also built with Journal, or imported from another application in an appropriate intermediate format from which Journal is able to deduce an internal representation. Additionally, this application of marking is a relatively simple command, in that there is a definite one-to-one correspondence between the mark and the meaning of the mark; that is, the scratch-out gesture is defined to signify a deletion, and the user who performs the gesture can only have one intended outcome by performing this gesture.

## *2.2 Editing Documents*

We now consider a more complex example, a document-editing program that has the capability to read more "intuitive" marks from the user when editing a pre-existing document, similar to the ones explored in (Rao 1998, Conroy et al. 2004, Rodríguez et al. 2006). These gestures are unlike the Journal case in that they are drawn on the screen and left there to be executed at a later time, presumably by the writer after the editing marks have been reviewed. In addition, these marks are much more complicated than a simple scratch-out gesture. In this case, the system needs to be able to differentiate between different types of marks, and have a complicated mark recognizer that is capable of doing so. We will now formulate a design for a new system similar to those aforementioned to evaluate these marks given an underlying representation that could be expected of a word processor. Examples of marks that could be used are shown in Figure 4. There is a very large variety of marks with differing levels of complexity, and one might expect a mark recognizer built for this program to be as complex, if not more so, than a handwriting recognizer.

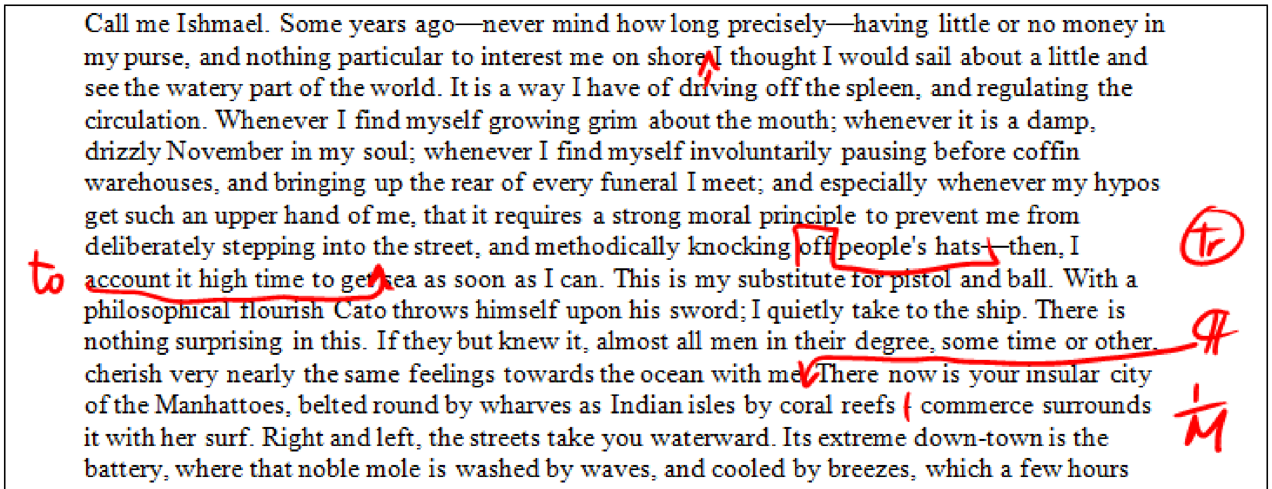

Call me Ishmael. Some years ago—never mind how long precisely—having little or no money in my purse, and nothing particular to interest me on shore, I thought I would sail about a little and see the watery part of the world. It is a way I have of driving off the spleen, and regulating the circulation. Whenever I find myself growing grim about the mouth; whenever it is a damp, drizzly November in my soul; whenever I find myself involuntarily pausing before coffin warehouses, and bringing up the rear of every funeral I meet; and especially whenever my hypos get such an upper hand of me, that it requires a strong moral principle to prevent me from deliberately stepping into the street, and methodically knocking off people's hats—then, I account it high time to get sea as soon as I can. This is my substitute for pistol and ball. With a philosophical flourish Cato throws himself upon his sword; I quietly take to the ship. There is nothing surprising in this. If they but knew it, almost all men in their degree, some time or other, cherish very nearly the same feelings towards the ocean with me. There now is your insular city of the Manhattoes, belted round by wharves as Indian isles by coral reefs ( commerce surrounds it with her surf. Right and left, the streets take you waterward. Its extreme down-town is the battery, where that noble mole is washed by waves, and cooled by breezes, which a few hours

Figure 4: Example of marks in a document-editing program.

Now we consider a way to evaluate the marks in a different model of program use. In this new model, the document-editing program is used by the editor to merely indicate minor changes and annotations to the document, and is completely separate from the program used to create the document, and therefore, the editing program has no extra information other than a mere image of the document. When a user initiates the changes based on mark evaluation, the program must still be able to make the changes as if it understood the underlying document. In this case, however, a program can merely apply a type of optical character recognition (OCR) algorithm to the image to come up with a representation of the document. This is possible because the program can make an assumption about the nature of the incoming data that should be true for all background images being annotated by the program; that is, this background image input to this program has one expected type, a document of text. Nevertheless, it must be noted that even the best OCR programs still cannot deliver 100% accuracy and may introduce errors to the text that were not there in the first place, which is unacceptable for this application (Rice et al. 1996).

At this point, one might observe that OCR provides the program with a representation that perhaps contains more information than is necessary, and that perhaps one could come up with a representation that does not have to exactly know each and every letter or word.

Although we do not need to know more than which pixels are not white to insert a comma or transpose words, though, consider the insertion of many words, a line break, or a new paragraph. In this case, subsequent line breaks and page breaks would have to be redone. If, for example, the document has instances of long words that fragmented between lines and therefore separated by hyphens, then in the re-evaluation of a line breaking algorithm, these must be taken into account, and hyphens removed from words that no longer require them, and added to words which may now require them. To run hyphenation algorithms on the new text, our editing program would require knowledge of all of the letters in all of the words following the original mark as to create acceptable hyphenations. At the same time, however, automatic hyphenation is a feature that many would consider not very important, so in practice, an editing program only would have to know information about where there was text and where there was not text, except for around the areas which are marked. Thus, we must note that the more complicated a representation of the background is, the harder it is to accurately represent it based on the background image alone; and also, we note that the complexity of the representation needed to accurately place, interpret, and evaluate a mark depends on the complexity of the task that the mark is designed to accomplish.

## *2.3 Digital Pen and Paper*

A careful balance between an accurate, simple representation and a less accurate, complex one can also be seen in the last marking example we will analyze, the automatic filling out of forms with digital pen and paper, as is done with Anoto functionality. In this type of paradigm, someone can write or draw on a special piece of paper and have the data sent to a computer automatically, eliminating the step where traditionally one must then enter the data on the completed form into one on the computer. While the pen uses actual ink, it also contains a camera to capture and process data on the paper; the paper used has a pre-printed proprietary pattern that allows the pen hardware to uniquely identify position and even the exact sheet of paper upon which the user is writing. Using this position data, the information is then sent to a computer in real time. An example of this functionality can be used for sending e-mail. Using a special form, a user can specify destination addresses and subject line, as well as scribble or draw a message. When a certain box on the form is marked with a check, the e-mail will be sent; based on the position of a recognized check mark, a computer program can find that the check mark was placed at a certain position on a certain piece of paper, and that this mark is equivalent to a command to sending an e-mail.

Another way Anoto functionality is used is called PaperPoint, which extends PowerPoint input devices to include pen and paper, where a new presentation can be drawn, or an existing presentation annotated on paper, and commands can be issued using a special "pidgets," or paper widgets, to move forward or backward in the presentation (Signer and Norrie 2007). The PaperPoint equivalent of clicking a GUI button in PowerPoint is simply touching the "paper button" with the pen tip. In this case, the underlying representation is the knowledge of what is expected to be drawn on each paper, and where on each paper (as Anoto technology can distinguish exactly which sheet the user of a pen is drawing on), and upon processing this position data, issue a command to PowerPoint, which then carries out the instruction using its own internal representation of the presentation. The marks here, however, are merely the touching of the pen to the paper: they are not complex and are the only action expected to be done in those areas by the user, so they do not have to be interpreted by a complex mark recognizer (although they still have to be detected). One might consider the paper annotations done on a special printout of an existing presentation to be marking, but because these marks are only displayed in the presentation and read by another human, this is not inside the problem of marking understanding. This is a situation where marking occurs, but the actual understanding is left up to the user to deduce, and no other functionality is required.

## *2.4 Similarities and Differences in Application*

We have examined three different applications for mark understanders. As mentioned in Section 2.1, all three of them contain a mark analyzer to find what kind of mark was inputted (if there was one at all), a context identifier which locates the relevant areas in the background image, and a context analyzer which builds a model of the background image so that the system can synthesize the information from the other two systems to figure out a response. At the same time, there exist stark contrasts: each system had different ways of locating the mark context; each system has unique requirements for how complex and varied the mark "dictionary" was, and therefore needs to have unique ways to analyze the marks; and each system has different information about the context of the mark, so a representation for the background image can have varying accuracy depending on how complex the representation has to be able to fully understand the meaning of each mark.

# 3 Classroom Learning Partner

Before describing our mark understanding system, we provide an overview of Classroom Learning Partner (CLP) and how the marking operation fits into our classroom interaction scenario.

## *3.1 Classroom Learning Partner Overview*

Classroom Learning Partner is a wireless presentation system that uses Tablet PCs to allow students to complete in-class exercises and have their answers assessed and reviewed quickly during class. Developed at the Computer Science and Artificial Intelligence Laboratory at MIT, CLP aims to enhance student performance and overall experience in large classes by increasing interaction between instructors and students (Koile and Singer 2006a, Koile and Singer 2006b, Koile et al. 2007a, Koile et al. 2007b). CLP is built on top of Classroom Presenter, a wireless Tablet-PC-based presentation system (Anderson et al. 2003, Anderson et al. 2004, Anderson et al. 2005). CLP extends Classroom Presenter by supporting interpretation and aggregation of student answers to in-class exercises so that an instructor and students can quickly get a sense for students' level of understanding. Using the Tablet pen, students can take digital ink notes on the presentation, work out exercise solutions, and submit these solutions anonymously via a wireless network to the instructor. The instructor can then examine individual solutions, select solutions for display on a public machine, and lead a class discussion of both correct and incorrect answers. In order to help instructors choose answers to display, CLP groups student answers into equivalence classes and presents the result to the instructor, who then can select representative examples from each of the classes. Our three controlled studies done in classes using Tablet PCs and this wireless feedback pedagogy suggest that use of a CLP and similar systems improves student learning, especially among students who would otherwise struggle with the material (Koile et al. 2007a, Koile et al. 2007b).

The bulk of the work in extending Classroom Presenter into CLP has been in creating interpretation and aggregation schemes for different expected types of student answer input methods. At the beginning of the project, the CLP group concentrated on interpretation and aggregation of answers containing alphanumeric characters (e.g. “480,” “define,” “a5-c4”), and sequences of several alphanumeric character groups (e.g. “(foo bar),” “(list 1 2 3),” “(set-cdr! x y)”) (Rbeiz 2006, Smith 2006). While research in this area continues (Tay 2008a, Tay and Koile 2008b), CLP has also been extended to interpret and aggregate a specialized kind of diagram often used in the old computer science introduction course, called a box-and-pointer diagram (Chevalier 2007, Pal 2007). This thesis extends CLP's answer types to include various kinds of marks, as described in later sections.

## *3.2 Classroom Learning Partner Architecture*

The architecture of CLP shown below in Figure 5 illustrates the different modules in CLP and where interpretation and aggregation fit in.

The following description of the interaction among CLP’s components is also presented in (Koile et al 2007b):

1. **Slides Saved**. Before class, the instructor creates the presentation and specifies the exercises to be presented in class using the Instructor Authoring Tool, or IAT. Exercise data is stored on a database, and slides are stored on a file server.

2. **Slides Retrieved**. At the beginning of class, the instructor retrieves the presentation from the database, or the slides can be placed on the machine beforehand.

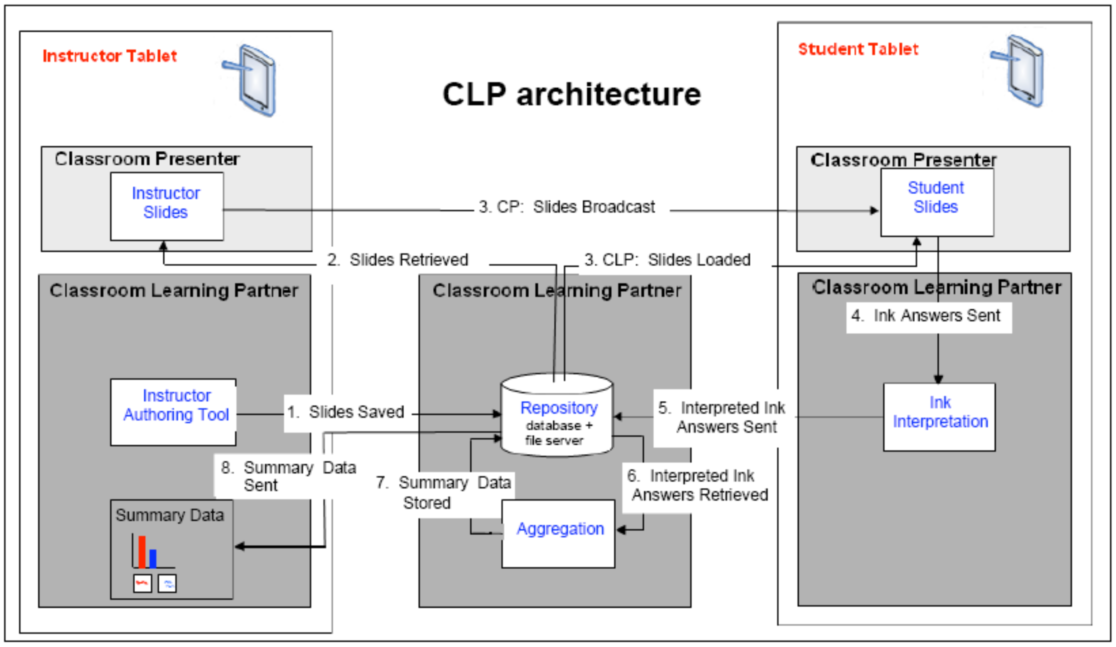


*Figure 5: Architecture of Classroom Learning Partner (Koile et al. 2007b).*

3. **Slides Broadcast / Slides Loaded**. A remnant of Classroom Presenter, the instructor can use multicast broadcasting to deliver the slides to individual student Tablets. However, in a large classroom setting, this may not be desirable, so in the CLP framework used at MIT, students automatically download the slides from the file server, or the slides may be manually loaded onto student machines. This generally allows a more error-free delivery of slides to student Tablets.

4. **Ink Answers Sent**. While going through the presentation in class, the instructor directs the students to answer in-class exercises. After the students have completed the exercise, they submit their student answers. In the process of submission, each student's submission is read by the interpreter on each individual student machine. Placing the interpretation step on individual student machines is an attempt to balance the workload among the machines. At this point as well, the actual ink answers may be broadcast over the ad-hoc network to the instructor machine, should the instructor also wish to see every answer.

5. **Interpreted Ink Answers Sent**. Using the same ad-hoc wireless peer-to-peer network, each student machine sends individual interpretation results to the database for storage.

6. **Interpreted Ink Answers Retrieved**. When the instructor clicks the "Aggregate" button on his or her own machine, ink answers are retrieved by the aggregator, which then divides these answers into bins, which constitute the summary data.

7. **Summary Data Stored**. The aggregator stores the finished bin data in the database.

8. **Summary Data Retrieved**. The instructor machine retrieves and displays summary data.

## *3.3 Interpretation and Aggregation Paradigm*

Interpretation begins when on a student clicks the "Submit" button on his or her own machine. The ink interpreter runs on the background on the student machine, getting all the ink in the "exercise box" area pre-defined by the instructor when the instructor created the presentation using the IAT and proceeds to create and send to the database an internal representation of the ink to be processed later during the aggregation step. The student does not see the results of interpretation in class, as the student may be distracted by the added step, and undisturbed interaction between the student and the instructor is important for learning.

When the "Aggregate" button is pressed on the CLP instructor machine, aggregation takes place in the background of instructor machine. While aggregation does not have a time limit per se, generally, a time of less than 30 seconds is preferred in order to not delay the class discussion of submissions. Different aggregation techniques have been used in CLP for different expected types. The aggregator clusters the answers into up to a number of bins that can be pre-defined in two ways. While creating an exercise in the IAT, the instructor may specify a number of instructor answers. Instructor answers are example student answers, as they are inputted and interpreted in the same way that student answers are, and may represent correct or incorrect responses. An instructor may include incorrect responses in order to guide a class discussion about common misunderstandings of the concepts tested in the exercise. By specifying these instructor answers, the instructor can bias the aggregator to create equivalence classes centered about these instructor answers, thereby controlling the number and content of the bins. If the instructor does not specify these answers, the system creates a default number of bins, currently set to seven, and clusters student responses using similarity measures based on previously implemented answer types (Koile et al. 2007b). The bins are stored in the database and displayed on the instructor machine in the form of a histogram and representative answers for each bin. In a large class setting, the instructor would be overwhelmed by seeing all student answers, so the histogram provides important feedback about overall class understanding.

# 4 Formulation of the Framework

One of the goals of this project was to have it work on representative domain-independent marks, so it was decided to concentrate on a subset of marks—those used for selection. We identified two main classes of such marks, *location-based* and *region-based*. Location-based marks point out a certain location on a background image, whereas region-based ones delineate an entire region on a background image. Location-based marks can be used to select small boxes or buttons that correspond to text, as in multiple choice questions and surveys; to mark critical points of graphs on a provided set of axes; or to select small pre-defined regions on a map. Region-based marks are used to select regions on background images that are not pre-defined, such as areas of text or parts of a diagram.

Two commonly used and widely applicable location-based marks are the X mark and the check mark, whereas two common styles of region-based marks are shading in of a region, and encirclement of a region. Shading can be done in a variety of ways, three of which are shown in Figure 6: (1) totally covering an area with ink; (2) partially covering an area with ink (i.e. a sloppy attempt to totally cover); and (3) using spaced parallel lines.

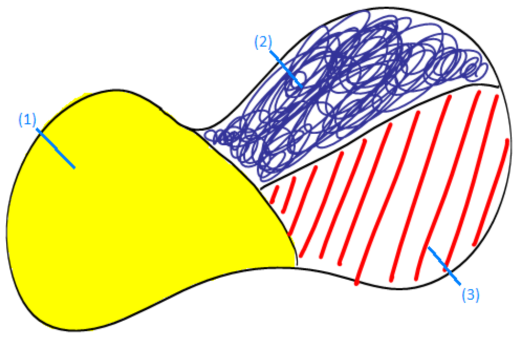


*Figure 6: Three methods of shading.*

Encirclement can be done using either one single stroke to encircle an area, or using a collection of more than one stroke to create an encircling polygon, as shown below in Figure 7.

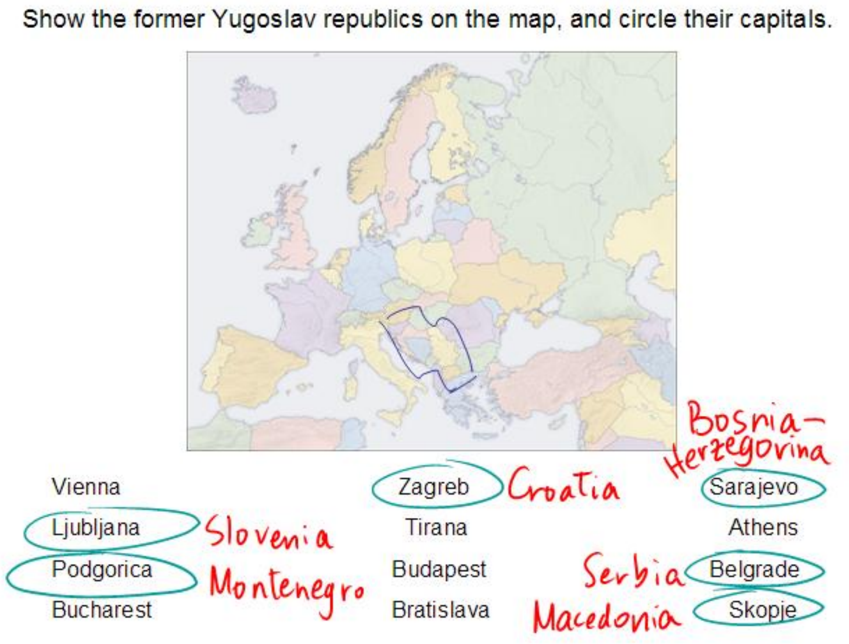

Show the former Yugoslav republics on the map, and circle their capitals.

*Figure 7: An example of a problem using encirclement.*

While the X mark and check mark are specific types of marks that sometimes do not share similar semantics, shading and encirclement each constitute different kinds of marking methods that lead to a semantically identical mark. Often, however, all four of these marking methods could technically be interchangeable in that no matter what kind of mark was used, someone could understand what was meant, but connotations of each of the marks can change which mark is used conventionally. To illustrate these ideas, we present seven canonical examples using these four mark types.

1. **Surveys**. Surveys often have boxes in which users mark their choices, as shown below. In census or medical forms, for example, one question might ask for a person’s gender, giving the choices of “male” or “female,” each with a small box next to them. One could

use either an X mark or a check mark to indicate one's choice. This kind of interaction could be used for multiple choice questions or true/false questions as well.

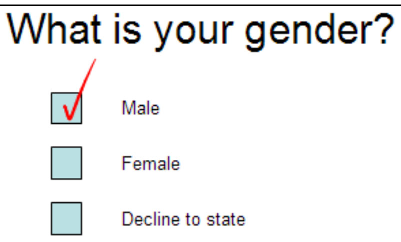


*Figure 8: An example of a survey using boxes for participants to mark.*

2. **Odd One Out**. A student might be asked to select the odd one out of a number of small pictures or diagrams. A logical way to do so would be to X out the diagram that is the odd one out. A check mark here is much less conventional, though, since the X mark carries the connotation of incorrectness, whereas the check connotes correctness.

3. **Identifying Countries on a Map**. In a geography class, a student may be asked to identify a country on a map of a continent, given an unlabeled map of the continent. For this type of question, the student generally would shade in the country on the map, but an instance where a student might circle the country or countries in question is shown in Figure 7 above.

4. **Parts of the Brain**. A student participating in a psychology or physiology class may be asked to demonstrate his or her knowledge of the structure of the human brain by shading in a certain area of the brain on an unlabeled diagram of the brain. This example is similar to the Example 3, but involves a region that is not pre-delineated, which can cause much more variation in the region that is actually shaded, and therefore make answer equivalence determination more difficult. In Figure 9, for example, the highlighted area could extend into the middle of the image and still be considered correct, as long as it was mostly touching the lower-right side of the diagram.

Where is the occipital lobe?

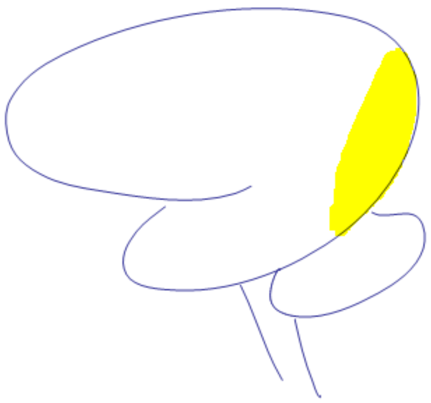
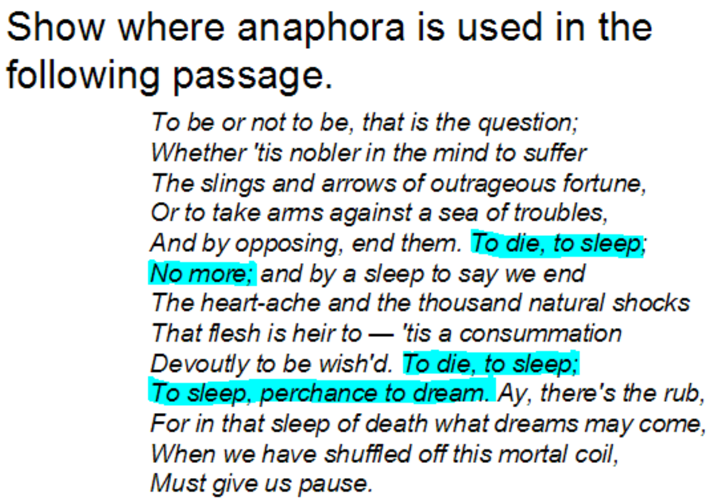


Figure 9: Shading in a part of the brain.

5. **Highlighting Text**. In an English class, a student may be instructed to examine a passage of literature to identify portions of the text that are rhetorical devices, unusual syntax, or grammatical imperfections. An intuitive way to do this would be to highlight the text, which constitutes the shading of an area. With text, areas to be shaded are not delineated because there are no absolute boundaries of a region (for example, one might include extraneous words in a correct answer); however, these areas are not completely unbounded either, because the highlight marks should be centered on or at least include certain portions of the text, as shown in Figure 10.

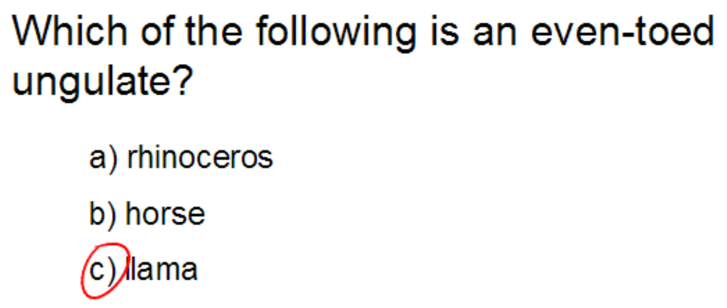


Show where anaphora is used in the following passage.

*To be or not to be, that is the question;*
*Whether 'tis nobler in the mind to suffer*
*The slings and arrows of outrageous fortune,*
*Or to take arms against a sea of troubles,*
*And by opposing, end them. To die, to sleep;*
*No more; and by a sleep to say we end*
*The heart-ache and the thousand natural shocks*
*That flesh is heir to — 'tis a consummation*
*Devoutly to be wish'd. To die, to sleep;*
*To sleep, perchance to dream. Ay, there's the rub,*
*For in that sleep of death what dreams may come,*
*When we have shuffled off this mortal coil,*
*Must give us pause.*

Figure 10: Highlighting a portion of text.

6. **An Intuitive Multiple Choice Response**. Multiple choice questions can be asked in a number of ways. A student, for example, might be asked to the write the letter of his or her response in a box, or, as in Example 1, to check one or more boxes that correspond to choices. Nevertheless, an equally intuitive way to answer a multiple choice question is by encircling the letter that corresponds to the choice one makes, as is commonly done on paper multiple choice exercises. Figure 11 shows an example of this style of asking and answering multiple choice questions.



Which of the following is an even-toed ungulate?

a) rhinoceros

b) horse

[✓ c)] llama

*Figure 11: An intuitive way of asking and answering a multiple choice question.*

7. **Identifying Parts of an Environment Diagram**. In 6.001, the introductory course to computer science at MIT, students analyze a structure known as an environment diagram, which provides a model for reasoning about scoping rules. In analyzing a pre-existing diagram, students may be asked to identify parts of the diagram that are incorrectly drawn, or that correspond to certain snippets of code, as seen below in Figure 12. Often, these parts are not easily enclosed shapes, so they may be encircled with tight polygons consisting of multiple strokes, or wide single-stroke loops that contain large areas of blank or extraneous space. The disparity between the region encircled by these marks leads to a challenge in designing an accurate system.

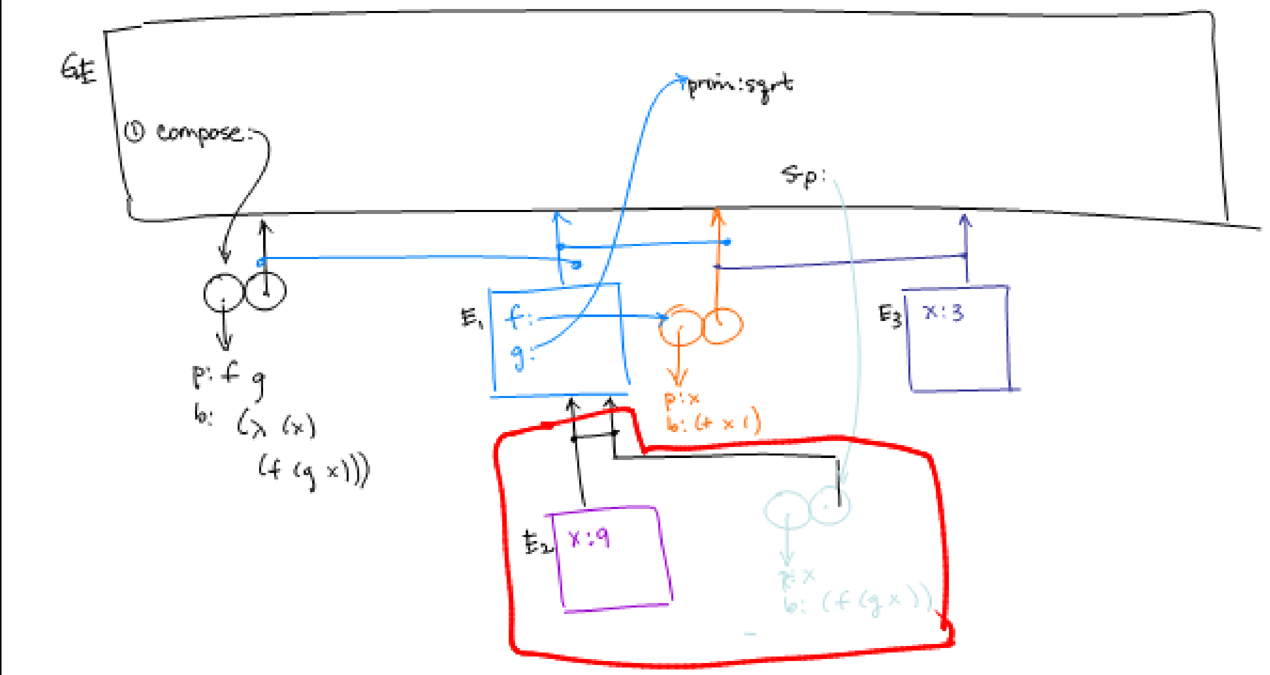


Figure 12: An example of an environment diagram with two parts enclosed.

We now describe the general framework for integrating the interpretation and aggregation of marks on exercises similar to canonical examples. When the instructor uses the instructor authoring tool (IAT) to create a presentation, the central database receives information for each exercise, which includes the expected type of mark for the exercise and instructor answers for the exercise. As shown by the variation in the canonical examples above, we cannot expect that expected mark type will be sufficient to understand the context of marks, so we must rely on instructor answers for contextual information. This means that the system does not "understand" context to the point of being able to reason about the semantics of the background image—it "understands" merely the type and location of the mark. As discussed in the next section, this representation is sufficient for our application; we need not tackle the difficult question of extracting semantics from a background image.

Once the instructor has inputted expected mark type and answers, interpretation, then, is a task of deciding which strokes in the students' answers constitutes marks, and then translating the digital ink of those marks into a representation of the locations or regions indicated by those marks. Upon identifying these locations and regions, the aggregator uses similarity measures between student answers and instructor answers to place each student answer in an appropriate bin.

This system supports the three marking understanding steps identified previously: mark analysis, context identification, and context analysis.

1. **Mark analysis** is done in CLP by the interpreter. The interpreter receives the ink and must decide whether or not each stroke in the ink is part of a mark. Then, it decides which mark is being made, and sends a representation of the marks that were made to the database. This representation is merely a location for each location-based mark, or a region for each region-based mark.

2. **Context identification** can be said to be done in the interpreter because it decides what the relevant location or region is.

3. **Context analysis** can be said to be done in the IAT in that the instructor inputs instructor answers. The meaning behind each answer is later recalled by the instructor when viewing the summary data returned by the aggregator, so it need not be supplied to CLP.

# 5 Design

## 5.1 *Interpretation of Marks*

The mark interpreter must perform mark analysis, which translates incoming ink into a set of locations and regions. Actual detection of marks could be done in many ways. It might seem that a mark recognizer from another system could be easily imported and extended to fit the CLP system. Three different systems were considered.

1. **Gesture Recognizer**. The gesture recognizer is a compact way of recognizing small pre-defined marks such as the check or X. A gesture recognizer, however, is also designed with relatively tight direction and speed parameters which a user can be trained to satisfy with continued user feedback, and CLP does not support this, as one of its goals was to not require students to spend time training or being trained by the system. Moreover, a gesture recognizer is not easily customizable (Jarrett and Su 2003).

2. **LADDER**. The LADDER shape recognizer can be used to recognize simple shapes (Hammond and Davis 2005). LADDER's main drawback, though, is that it is a large system written in Java, and tends to make CLP runtime long and sluggish (Chevalier 2007). CLP needs to be able to interpret ink quickly so as not to interrupt the flow of the class. In addition, LADDER's representation contains much more information than our extension needs.

3. **Reverse Graphics**. Reverse Graphics has much potential for use as a marking interpreter, but it is currently patented (Rao 1998).

As all three of these systems have issues for integration with CLP, it was decided that a simple mark interpreter should be implemented. The resulting mark interpreter recognizes four types of marking methods by analyzing all of the strokes and looking for a certain type of mark.

1. **X Marks**. The mark interpreter tries to look for two strokes that intersect at an angle within some angle error $\theta$ of their expected intersection angle $\phi_x$, and that are relatively straight. We say that a stroke is straight enough if a certain percentage *p* of the length across the stroke is close to the stroke's overall direction, which is calculated between the endpoints.[1] To ascertain that each X mark is not part of another mark or an extraneous mark, we make sure that neither stroke intersects with other long strokes near their intersection. The location returned is the intersection of the two strokes.
2. **Check Marks**. The mark interpreter looks for strokes with one large sharp turn in the middle by looking for cusps in the stroke, and then divides the strike in two at the cusp. Then, it checks that the two divided parts are relatively straight using the method above and are within the angle error $\theta$ of their expected orientations, $\phi_1$ and $\phi_2$.[2] If both of the sections are of acceptable orientation and straightness, it then makes sure that the ratio of the lengths of the two sections, *r* is within a certain range, or that $r_{low} \leq r \leq r_{high}$. If this is so, then the mark interpreter returns the calculated cusp as the check mark's location.

[1] Because we are using a discrete representation of the data, our model of the stroke can be viewed as a linear piecewise approximation of what was actually drawn. Using this approximation, we can calculate exactly what percent of the length of each stroke is going in the general direction of overall stroke.

[2] This assumes the check is perfectly upright. To allow for a check of arbitrary orientation, we could use a method called extended circular images (Horn 1986). However, checks are almost always drawn close to upright, so such an orientation-blind system is unnecessary, at least for the canonical example that involves checks.

3. **Shading**. Previously, we viewed shading as consisting of three distinct methods. Completely and partially filled-in areas, however, can both be translated using the same method, so only two interpretation techniques are required.

   a) **Completely and Partially Filled-In Areas**. When a user completely or partially fills in an area, strokes mostly consist of many cusps and self-intersections, and are extremely long. Strokes that do not fill this requirement and that are still part of the shading mark take up relatively little space compared to the strokes that are longer. They also are generally drawn by the user to beautify the image and smooth the image, so they are situated near the edges and are highly redundant with previously drawn strokes, and therefore are less relevant. We thus can still get a very good idea of the shaded region by looking only at the long strokes with many cusps and self-intersections. The mark interpreter identifies all strokes with at least *c* cusps, *s* self-intersections, and ℓ length. To find the shaded region, the mark interpreter finds the outermost boundary of each one of these strokes, and fills in the region. Filling in this region will in effect cover the entire area in the representation, whereas it might not have been covered in the hand-drawn ink.

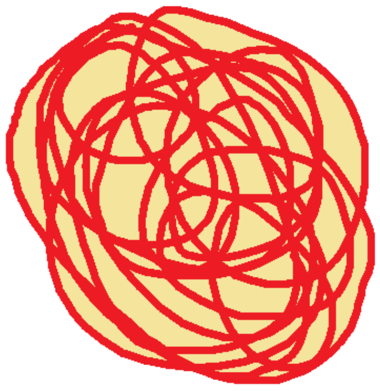


*Figure 13: Running the shading detection algorithm returns even the blank regions within each stroke that were not actually filled in.*

b) **Shading By Parallel Lines**. The process the interpreter uses to translate parallel line shading into a region is illustrated in Figure 14. (a) The mark interpreter first isolates all the strokes that are relatively straight and measures their slopes. (b) Then, it creates a graph in which each stroke is represented by a vertex. If two strokes have similar slopes then it creates an edge between the two vertices which represent those two strokes with edge weight equal to the distance between the two strokes. If the edge weight is greater than a certain value $d_1$, then it is not included. (c) On the resulting graph, it isolates individual connected subgraphs, and runs Kruskal's algorithm to find minimum spanning trees for each subgraph (Cormen et al. 2001). (d) It then constructs almost-trapezoids by connecting the endpoints of all pairs of strokes whose representative vertices are connected by an edge in the minimum spanning trees, and adds the interior of each trapezoid to be part of the region that is said to be selected by the shading.

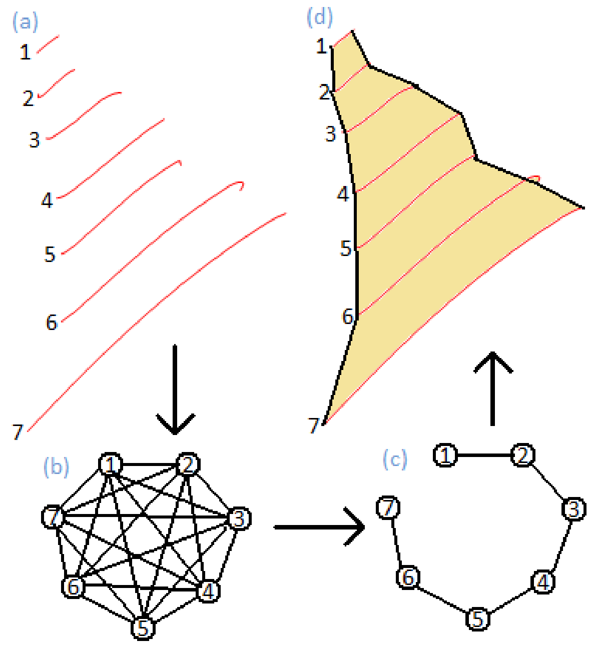


*Figure 14: An illustration of the process used by the mark interpreter to translate parallel shading into a selected region.*

4. **Encirclement**. To translate an encirclement mark, the mark interpreter must find single-stroke or multiple-stroke loops and return the region or regions that they form the boundary of. To do this, the mark interpreter uses a method called proximity linking (Mahoney and Fromherz 2002), which creates a link between two strokes if an endpoint on one stroke is within distance $d_2$ of an endpoint on another stroke. Assuming that the resulting graph is not complex, we can quickly find the loops that maximize the area encircled, and select those loops as the boundaries of the regions to be returned.[3] To bias the interpretation to find single-stroke loops, we can choose to enforce a constraint that if a stroke is linked to itself during proximity linking, it must be part of its own loop. This constraint also speeds up the mark interpreter if there are many single-stroke loops in an answer.

## 5.2 Aggregation of Marks

After student answers are interpreted and the locations and regions of marks are saved to the database, the aggregator can retrieve the information and begin to group the student answers into bins based on the interpretation of instructor answers. At this stage, we are left with only two kinds of data with which to work, locations and regions.

1. **Locations**. The simplest way of aggregating locations is by finding the nearest neighbor. Since locations are represented as ordered pairs, the aggregator measures every student answer's distance from every instructor answer, and places each student answer in the same bin as the instructor answer that is closest to it. In effect, each bin

[3] This can get slow very quickly, though, since the runtime can be worse than exponential in the number of strokes if they are drawn in such a way that there are many links and many possible ways to create loops. If $d_2$ is chosen in such a way that limits the number of links created by proximity linking, then this problem can be alleviated.

corresponds to a region on a Voronoi diagram of the instructor answers, and all student answers in the bin fall within that region. This visualization is shown in Figure 15. If there are multiple locations marked within the same answer, the aggregator enforces the rule that a student answer can only go in a bin with an instructor answer if the two answers have the same number of locations. In this case, it finds a set of pairings between locations in student answers and locations in prospective instructor answers such that the sum of the distances between all pairs in the set is the minimum over all possible pairings. To allow for student answers that were not foreseen by an instructor, the aggregator also enforces a maximum distance threshold $d_{\text{loc}}$ on each student answer such that if a student answer is farther away from every instructor answer than that threshold, it is placed in a miscellaneous bin.

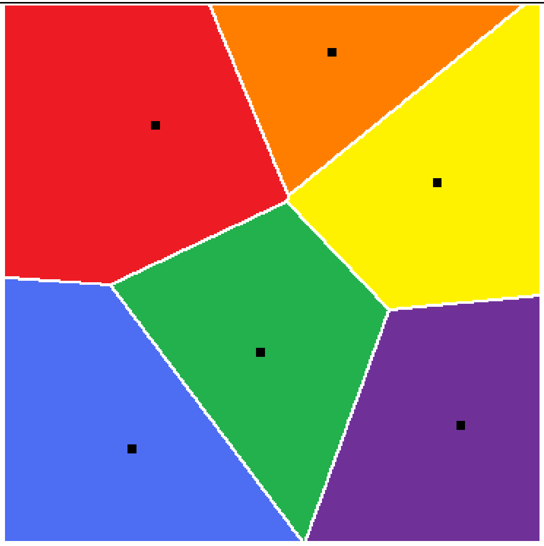


*Figure 15: An example Voronoi diagram for aggregation of locations. Black dots can represent the locations of instructor answers; white lines can represent the boundaries of the answer bins.*

2. **Regions**. Similarity between regions can be evaluated by observing the overlap between the student answer and instructor answer. We can juxtapose these answers together directly on top of each other like a Venn diagram, as shown below in Figure 16. Call the region selected by the student S, and the region selected by the instructor N. If

both the areas that are in S and N alone are small compared to the area that is both S and N—that is, their ratios are lower than some constants $r_s$ and $r_n$—then the aggregator can place the student answer in the same bin as the instructor answer. If a student answer fits these criteria for multiple instructor answers, then the aggregator places it in the instructor answer for which an average of the two ratios is smallest.

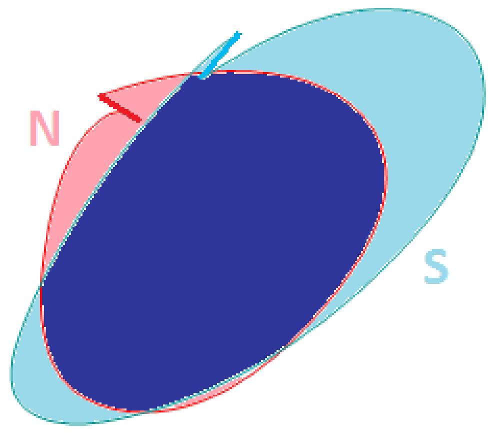
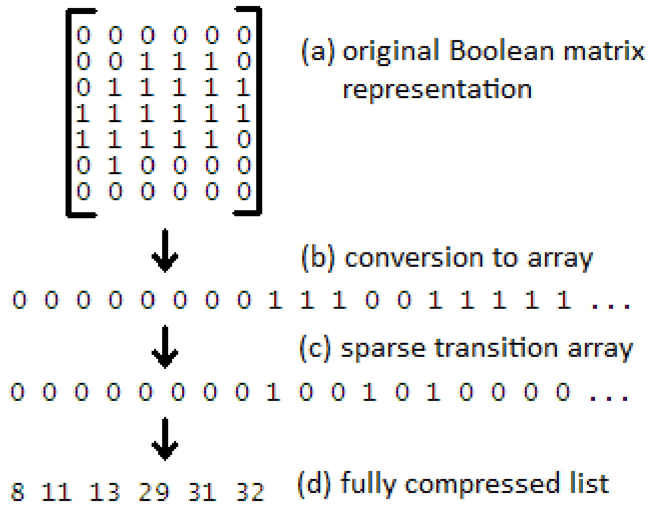


*Figure 16: An example showing an instructor answer N and a student answer S for encirclement. N is shown in light red, and S is shown in light blue, while their overlap is shown in dark blue. It is the comparison of overlapping area to non-overlapping areas that determines the similarity between N and S.*

## 5.3 Intermediate Representation of Regions

The interpreter returns a list of selected locations or regions to be inputted to the aggregator. Locations are represented as ordered pairs, whereas regions are represented as matrices of Booleans. This grid matrix corresponds to every possible combination of ordered pair in the box, and with the size of units used in the CLP, is fine enough such that student submissions can be over 6000 units tall and 10000 units wide. Even if the internal representation of a region of this size is simply a matrix of 1-bit values where a cell is true if the corresponding square unit is part of a selected region, then not only will the database quickly run out of disk space, but the wireless ad-hoc network will be flooded with data and begin to drop packets. A method to compress region data, then, was developed to address this problem.

First of all, the units used for the rest of CLP were much too fine for something like hand drawn images which are prone to great variability and can be drawn quite carelessly. Therefore, it was permissible to downsample the resulting matrix by a factor of 10. Even in doing so, though, the matrix was still quite large. For most Boolean matrices representing regions, however, the nature of selection of regions makes the matrices have true and false values clumped together; because there are only two values, we can instead represent these matrices by reshaping each matrix into a long array, and then making another array that has a value of true when there is a transition between the two values in the original array. This new representation should be sparse, allowing us to further condense the array by simply keeping the indices where the array has value true. Using this method, we can transform a large non-sparse matrix into a short list of values and save space at the expense of the time needed to compress and decompress each region matrix. An illustration of this method is shown below.



$\begin{bmatrix} 0&0&0&0&0&0\\ 0&0&1&1&1&0\\ 0&1&1&1&1&1\\ 1&1&1&1&1&1\\ 1&1&1&1&1&0\\ 0&1&0&0&0&0\\ 0&0&0&0&0&0 \end{bmatrix}$
(a) original Boolean matrix representation
(b) conversion to array
0 0 0 0 0 0 0 0 1 1 1 0 0 1 1 1 1 1 . . .
(c) sparse transition array
0 0 0 0 0 0 0 0 1 0 0 1 0 1 0 0 0 0 . . .
8 11 13 29 31 32
(d) fully compressed list

*Figure 17: The steps taken in compressing a large Boolean matrix representing a selected region. The original matrix is converted to an array; a new transition array is calculated; and the transition array is compressed into a list of positions.*

# 6 Testing

Our methods used to interpret and aggregate marks required many instances of threshold values. Choosing an appropriate threshold value is crucial to getting the maximum accuracy in each individual algorithm; therefore, the next step in implementing the CLP mark extension was to find the best threshold values. The values could easily be gathered empirically by running the system on a set of test data, each time changing one threshold value: by observing which threshold value resulted in the best accuracy in aggregated data, we can find the most appropriate threshold value for each case.

Our test data, consisting of both student and instructor answers, needed to effectively and accurately reflect the canonical examples discussed in Section 4. The IAT was used to create a presentation of 7 exercises (one for each canonical example) with many possible answers. For each exercise, 5 different instructor answers were given, and 11 people gave three student responses each, for a total of 33 student responses. These ink responses were then interpreted and aggregated using the methods described above, first using approximate guesses for each mentioned constant. After determining the accuracy values, the constant was changed a small step in one direction and the experiment repeated. By using a "human performed" moderated gradient descent over each threshold value with the target of a high accuracy, we were able to achieve good results.

The following two tables show the accuracy values received after three iterations. The first table separates test data by the canonical mark type represented in the exercises, while the second table separates test data simply by the mark type that was employed to give an answer. The final accuracy is the percentage of test data that was correctly classified at the end of aggregation after three iterations were completed.

| Canonical example | Type(s) of mark allowed | Final accuracy value |
|---|---|---|
| Survey | checks, X marks | 100% |
| Odd one out | X marks | 100% |
| Countries on a map | shading, encirclement | 78% |
| Parts of the brain | shading, encirclement | 43% |
| Highlighting text | shading | 91% |
| Multiple choice | encirclement | 100% |
| Environment diagrams | encirclement | 64% |

| Type of mark | Final accuracy value |
|---|---|
| check | 100% |
| X mark | 100% |
| completely filled shading | 82% |
| partially filled shading | 76% |
| parallel line shading | 68% |
| one-stroke encirclement | 100% |
| multiple-stroke encirclement | 42% |

The table in Appendix A shows initial value guesses, step sizes, and final values for each threshold mentioned.

# 7 Discussion and Future Work

In this section, we summarize and discuss issues, improvements, and extensions to our mark understanding system.

## 7.1 *Improving Accuracy of Interpretation and Aggregation*

To increase the accuracy of mark interpretation and aggregation, we would employ a larger test set than used in this study and continue with the threshold-setting process. We would aim for 90% accuracy across all types, as we deemed this value sufficient to give instructors and students meaningful feedback data (that is, 10% error is small enough such that it should not change the shape of the overall histogram of summary data). If this value could not be reached by adjusting thresholds, then adjustments must be made to the actual interpretation and aggregation algorithms.

The accuracy results, shown in Section 6, have some occurrences of 100% accuracy, while this cannot be guaranteed to be true in general. This result illustrates a problem with the testing methodology previously described in Section 6. With only a sample size of 11 people giving a total of at least 33 answers, it is entirely possible that an accuracy of 100% could result. For checks, there was a whole range of values for the orientation angles $\phi_1$ and $\phi_2$ over which there was 100% accuracy. This phenomenon can be attributed to the little variation in the way that these answers were drawn, perhaps because the test subjects drew the checks extremely carefully. With a large value for the allowed angle error $\theta$, these test student answers were all easily accepted as checks for a wide range of values for $\phi_1$ and $\phi_2$. Therefore, it was hard to tell what the correct threshold values would be because there were no border-line cases among the student answers to be classified. These border-line cases are extremely important for finding good and exact values for threshold constants; therefore, the constants found during the testing may not apply to a different group of less carefully drawn examples.

Another factor in testing that might have contributed to a bias in accuracy results was that the instructor answers used for this test were extremely distinct because they were all drawn by the same person. This bias may be good for all presentations created by that same person and can be extended in the future to allow for “instructor calibration” to further bias aggregation results. It however does not give generally applicable values for application thresholds. For example, let us assume that one instructor often colors outside the lines of a pre-delineated background region while shading, while another does not bother to shade all the way to the edges of the region, but only the middle portion. Both these kinds of errors can change the optimal values of the region overlap thresholds $r_s$ and $r_n$ greatly. To get around this problem, we could require instructors to create instructor answers in a standard, more rigid and careful fashion. A more desirable option would be to allow calibration of internal aggregator threshold values on a per-instructor basis.

In addition, our testing was insufficient to produce estimates for $d_{\text{loc}}$, $r_{\text{low}}$, and $r_{\text{high}}$, so they are not included in Appendix A. This insufficiency stems from all answers lacking marks that are not expected by the instructor and ink strokes that are not part of any mark but were accidentally left on the slide. In true cases of exercises, these mistakes should appear with some degree of frequency; however, during the test, the care with which the test subjects completed the exercises and submitted slides meant that insufficient “miss” data was collected. Similar problems occurred with $p$, $c$, $s$, $\ell$, $d_1$, $d_2$, and $d_{\text{loc}}$, in that the threshold was only approached from one side, causing an inability to tell if resulting threshold values were underfit or overfit. There was, for example, no way to find a “good” value for $d_1$ because there were no parallel lines that were not part of the shading mark. Therefore, the final value for $d_1$ is simply the maximum distance seen on a minimum spanning tree of the parallel lines for the set of student answers, even though this might be generally quite long if "distractor" lines were present.

One final way to improve threshold values is to change the method by which they are obtained. First of all, the method described in Section 6 was executed three times, and should actually be done over even more iterations because the choice of thresholds may not be independent. For example, the optimal value for the angle error $\theta$ may depend on the value used for the central intersection angle $\phi_x$ for an X mark. Over many iterations, the parameters may converge to local extrema. The main reason multiple iterations were not done is because they are time-consuming to do by hand. Therefore, future research in this area may benefit greatly from having an automated method for doing gradient descent.

Another way to improve accuracy is to change the values that are taken into consideration by the mark interpreter and aggregator in processing the data. For instance, one might look for a minimum intersection angle and a maximum intersection angle for an X, or look at the actual directions of the two strokes used to make an X. Another property is that proper threshold values might change with the size of the overall mark; for example, a check in a large box may have a smaller head-to-tail length ratio than a check in a small one. Also, there exist other, less common ways of shading which were not covered by this thesis; examples are shown in Figure 18. To tackle these problems, more values will have to be taken into account for thresholding. Finding which values are most relevant for the purposes of interpretation and aggregation of marks may be hypothesized, implemented, and tested, as was done in this thesis; on the other hand, they can also be done automatically, using a support vector machine or other kinds of machine learning techniques, which would make the interpreter and aggregator easier to change and extend in the future.

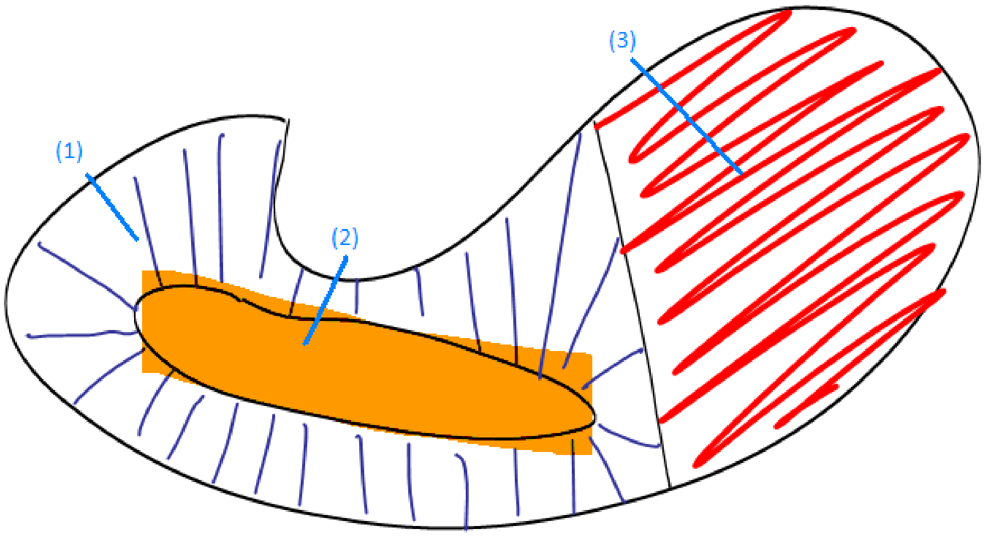


*Figure 18: Alternate ways of shading unimplemented in the current version of the mark interpreter: (1) individual straight lines that may not necessarily be parallel; (2) a large stroke of highlighter used to roughly shade in a thin area; and (3) a zigzag pattern with no intersections.*

## *7.2 Expanding the Scope and Reach of CLP*

So far, we have focused on implementing marking functionality in CLP that is sufficient for supporting seven canonical examples, as described in Section 4. In reality, however, these canonical examples need to be adjusted and expanded before being widely applicable to different domains that use slightly different types of exercises. An immediate way solution would be to simply add more types of expected marks and implement new interpretation and aggregation methods, in a process similar to that described in previous sections, or by using another system such as LADDER or Reverse Graphics as mentioned before. For a few more mark types, this approach may be enough; as the number of mark types grows, however, this approach may prove unwieldy and tedious and will require other methods of automating more of the process.

As concluded in Section 2, the greatest obstacle to tackling the mark understanding problem lies in the complexity and variation in both the marks and the context in which these marks are placed. Although we were able to sidestep the context variation problem in CLP, we see that even for the moderately simple set of marks presented in Section 4, a rather complex interpreter is needed to successfully interpret them. Before CLP's mark understanding component can be considered more than a prototype, an extremely varied and complex set of expected marks would be needed.

The need for a larger set of marks is extra incentive to use machine learning to aid in the design of an interpreter and aggregator, allowing new kinds of marks to be quickly added. (Tay and Koile 2008) posits that for alphanumeric inputs, an approach using machine learning may even eliminate the need to make assumptions about expected type. For marking, however, this question remains to be answered. A major drawback, of course, to machine learning, is that a very large test data set must be compiled. In some cases, this data set may be as unwieldy as reasoning about each individual case and hard-coding a solution, depending on how complicated each kind of mark and corresponding exercise is. (Mahoney and Fromherz 2002) suggests certain research areas which could lead to a solution to this problem by finding a way to incrementally build accurate models of contextual information and not rely on large data sets of sample answers.

Future versions of CLP with many different kinds of marks for many different kinds of specialized exercises may require different kinds of instructor answers than the current architecture of CLP supports. In Section 2 we explored the importance of having an adequate representation such that contextual analysis can be done, and in Section 4 we decided that an instructor answer that was the same as a student answer, as well some a priori knowledge about the expected type of the mark, should for our purposes provide enough contextual information for aggregation to proceed. Figure 19, however, shows that this conclusion is clearly not always true, as the boundaries generated by the instructor answers do not necessarily correspond to the true boundaries. For aggregating selected regions, this problem can be manifested in the environment diagram example as there is no change in what an answer means no matter how much blank space or extraneous material is included in the selected region. Thus, to successfully find the right boundaries in this example, an instructor would have to enter a model of the actual background data into the IAT to be used by the aggregator. Under this new paradigm, an instructor would have to select not only by the type of mark that was expected, but also by the type of the problem being presented—in this case "selection of pre-delineated regions by locators" would be the expected type. This information can be very useful in the aggregation stage, as selecting a pre-delineated region will generally result in less variation among student answers than one in the case where regions are not pre-delineated at all. Since this distinction was not made in this version, accuracy results suffered.

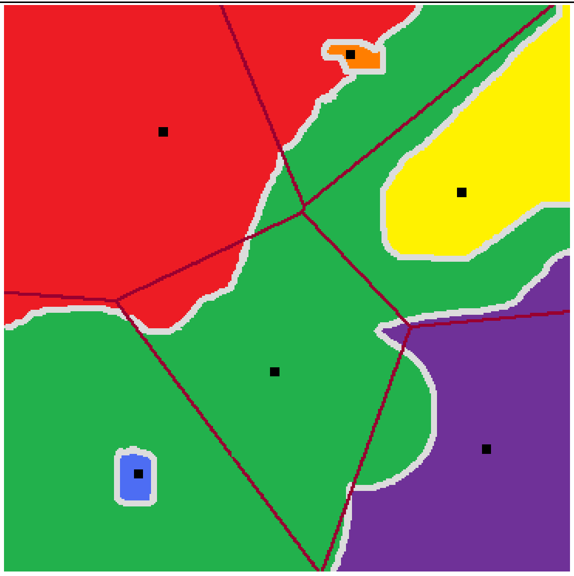
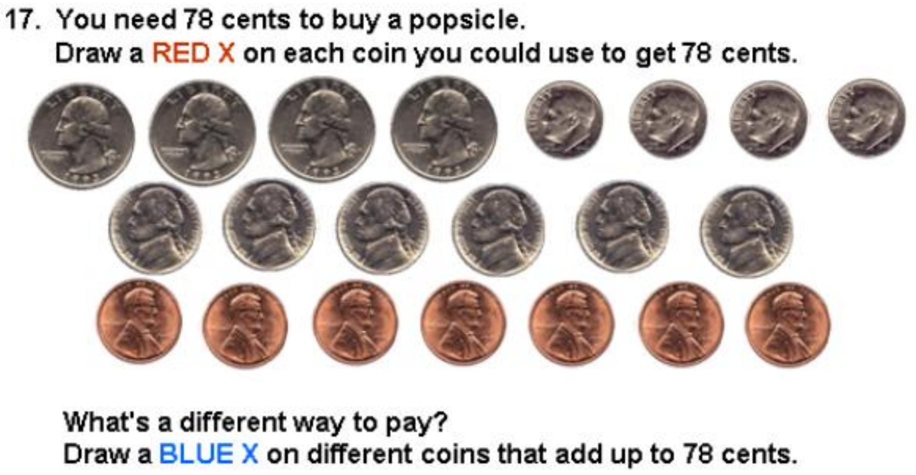


*Figure 19: Voronoi diagram showing boundaries generated by the aggregator in maroon, and actual boundaries on the background image in light gray. Discrepancies between these boundaries mean that the aggregator will misclassify student answers.*

This example shows the growing need in CLP for an instructor to input image semantics to improve accuracy results. Accuracy values differ, for example, among different kinds of canonical examples, as well as between different types of expected marks. Therefore, while in this version of CLP the interpreter and aggregator only have explicit a priori knowledge about the kind of mark expected, it would be of great help to know the type of the in-class exercise itself: there is a big difference, after all, between the one-stroke encirclement of a delineated or somewhat delineated region seen in a multiple choice question or selection of countries on a map and the totally non-delineated regions seen selecting regions of the brain or of oddly shaped regions as seen in environment diagrams. At the same time, improved semantics can lead to better ways to reason about regions; for example, an aggregator with a sufficient semantic model of background information can omit blank space in encircled areas, or discount shading outside the lines of pre-delineated regions, which were the two main problems encountered during testing, and which resulted in rather low accuracy values. This disparity between different types of exercises using the same type of mark can be allayed in future versions by allowing more distinction between different exercise types, exemplified by different canonical examples.

Another complex example illustrates that a semantic model for the background image is necessary in some circumstances. Consider an example in which a student is shown a picture of many different kinds of coins and is asked to select a combination of coins that will add up to a certain amount of money, as shown below in Figure 20. In this example, which was used in a second grade class, there are many different student answers selecting many different combinations of locations that would semantically amount to the same thing. In this case, an aggregator without this kind of understanding and which classified based on location of marks alone would fail to present the instructor with meaningful data. The kind of expected type for this mark is somewhat unclear, however, and being able to classify exercises such as this one may itself prove an interesting research topic.


17. You need 78 cents to buy a popsicle.
Draw a RED X on each coin you could use to get 78 cents.

What's a different way to pay?
Draw a BLUE X on different coins that add up to 78 cents.

*Figure 20: Example where semantic representation is extremely important as multiple combinations of X marks should be grouped together by the aggregator as having the same meaning.*

One could foresee that increased complexity in dealing with different types of backgrounds could lead to a situation in which the instructor generating the presentation becomes so overwhelmed by classification of models and inputting model data that usage of CLP would become a hassle. While the inputting problem could be solved by having different kinds of information about each stroke inputted in different fashions, such as changing the color or pen tip type to indicate a special property of the stroke, the instructor is still tasked with understanding a perhaps very intricate set of requirements and specifications to be able to classify the model being input. To alleviate this problem, a system could be devised that uses image recognition (or sketch recognition) methods to try to automatically decipher what the background image. At this point, however, CLP's marking component would exit the realm of marking and become a layered recognition and modeling problem, which currently, while being the subject of extensive research, does not yet have a solution that can reliably serve the needs of CLP (i.e. off-line processing, no student training or verification of interpretation results).

Overall, the approach taken in this thesis demonstrates a kind of paradox about marking. Marking lies in the gray area between a simple sketch recognition problem and a layered sketch recognition problem, as the problem of understanding the background image is tempered by *a priori* knowledge, which, in CLP, is provided by the instructor. With this principle in mind, we have provided a good foundation for further research into the nature of mark understanding and its applications both within and outside CLP.

# 8 References

Abelson, H. and Sussman, G.J. *Structure and Interpretation of Computer Programs, Second Edition*. Cambridge, MA: MIT Press, 1996.

Anderson, R., Anderson, R., McDowell, L., and Simon, B. Use of Classroom Presenter in Engineering Courses. In *Proceedings of ASEE/IEEE Frontiers in Education Conference*, 2005.

Anderson, R., Anderson, R., Simon, B., Wolfman, S., VanDeGrift, T., and Yasuhara, K. Experiences with a Tablet-PC-Based Lecture Presentation System in Computer Science Courses. In *SIGCSE*, 2004.

Anderson, R., Anderson, R., VanDeGrift, T., Wolfman, S., and Yasuhara, K. Promoting Interaction in Large Classes with Computer-Mediated Feedback. *CSCL*, 2003.

Chevalier, K. Interpretation of Box-and-Pointer Diagrams in Classroom Learning Partner. Massachusetts Institute of Technology EECS M.Eng. Thesis, 2007.

Conroy, K., Levin, D., and Guimbretière, F. ProofRite: A Paper-Augmented Word Processor. Submitted to *UIST*, 2004.

Cormen, T., Leiserson, C. E., Rivest, R. L., Stein, C. *Introduction to Algorithms, second edition*. Cambridge, MA: MIT Press and McGraw Hill, 2001.

Duncan, D. Clickers in the Classroom: How to Enhance Science Teaching Using Classroom Response Systems. San Francisco, CA: Addison Wesley, 2005.

Hammond, T., and Davis, R. LADDER, a sketching language for user interface developers. Elsevier, *Computers & Graphics* 29, 2005.

Horn, B. P. K. *Robot Vision*. Cambridge, MA: MIT Press, 1986.

Jarrett, R., and Su, P. *Building Tablet PC Applications*. Redmond, WA: Microsoft Press, 2003.

Koile, K., Chevalier, K., Low, C., Pal, S., Rogal, A., Singer, D., Sorensen, J., Tay, K. S., and Wu, K. (2007a) Supporting Pen-Based Classroom Interaction: New Findings and Functionality for Classroom Learning Partner. In *Proceedings of First International Workshop on Pen-Based Learning Technologies*, 2007.

Koile, K., Chevalier, K., Rbeiz, M., Rogal, A., Singer, D., Sorensen, J., Smith, A., Tay, K. S., and Wu, K. (2007b) Supporting Feedback and Assessment of Digital Ink Answers to In-Class Exercises. In *Proceedings of IAAI*, 2007.

Koile, K., and Singer, D. (2006a) Improving Learning in CS1 with Tablet-PC-based In-Class Assessment. In *Proceedings of ICER*, 2006.

Koile, K., and Singer, D. (2006b) Development of a Tablet-PC-based System to Increase Instructor-Student Classroom Interactions and Student Learning. *The Impact of Pen-based Technology on Education; Vignettes, Evaluations, and Future Directions*, ed. Berque, D., Gray J., and Reed, R. Purdue University Press, 2006.

Mahoney, J. V., and Fromherz, M. P. J. Three main concerns in sketch recognition and an approach to addressing them. In *Proceedings of AAAI Spring Symposium on Sketch Understanding*, 2002.

Pal, S. *Aggregation of Sketched Box and Pointer Diagrams in Classroom Learning Partner*. 2007.

Rao, S. Visual Routines and Attention. Massachusetts Institute of Technology EECS Ph.D. Thesis, 1998.

Rbeiz, M. Semantic Representation of Digital Ink in the Classroom Learning Partner. Massachusetts Institute of Technology EECS M.Eng. Thesis, 2006.

Rice, S. V., Jenkins, F. R., and Nartker, T. A. *The Fifth Annual Test of OCR Accuracy*. Las Vegas: Information Science Research Institute, 1996.

Rodríguez, J.A., Sánchez, G., and Lladós, J. Automatic interpretation of proofreading sketches. In *Proceedings of EUROGRAPHICS Workshop on Sketch-Based Interfaces and Modeling*, 2006.

Signer, B., and Norrie, M.C. PaperPoint: a paper-based presentation and interactive paper prototyping tool. In *Proceedings of TEI*, 2007.

Smith, A. Aggregation of Student Answers in a Classroom Setting. Massachusetts Institute of Technology EECS M.Eng. Thesis, 2006.

Tay, K. S. Massachusetts of Technology EECS M.Eng. Thesis to be completed, 2008.

Tay, K. S., Koile, K. Improving Digital Ink Interpretation through Expected Type Prediction and Dynamic Dispatch. Submitted to *AAAI*, 2008.

http://www.anoto.com/. Accessed on January 1, 2008.

## Appendix A

The following table shows the various threshold values discussed and information about them. Initial values were set before the first iteration. The final values were obtained after doing three iterations of the data using the method described in Section 6. All angle or orientation values are in radians.

| | Description | Initial value | Step size | Final value |
|---|---|---|---|---|
| $\theta$ | permissible angle error | 0.3927 | 0.0087 | 0.8901 |
| $\phi_x$ | expected X intersection angle | 1.571 | 0.0087 | 1.571 |
| $p$ | straightness percentage threshold | 80% | .5% | 74% |
| $\phi_1$ | check first part expected orientation | 5.236 | 0.0087 | 4.939 |
| $\phi_2$ | check second part expected orientation | 0.7854 | 0.0087 | 0.8552 |
| $r_{\text{low}}$ | check length ratio low threshold | 0 | .01 | .21 |
| $r_{\text{high}}$ | check length ratio high threshold | 1 | .01 | .73 |
| $c$ | minimum cusps for shading strokes | 5 | 1 | 13 |
| $s$ | minimum self-intersections for shading strokes | 5 | 1 | 7 |
| $\ell$ | minimum length for shading strokes | 1000 | 50 | 26150 |
| $d_1$ | maximum edge weight for parallel shading | 1000 | 50 | 1750 |
| $d_2$ | proximity linking threshold | 1000 | 10 | 470 |
| $d_{\text{loc}}$ | maximum Voronoi diagram cell radius | 1500 | 10 | 2140 |
| $r_s$ | maximum ratio of area in S alone to area in both S and N | .1 | .005 | .265 |
| $r_n$ | maximum ratio of area in N alone to area in both S and N | .1 | .005 | .085 |