EUROGRAPHICS Workshop on Sketch-Based Interfaces and Modeling (2007)
M. van de Panne, E. Saund (Editors)

Newton's Pen - A Pen-based Tutoring System for Statics

WeeSan Lee,[1] Ruwanee de Silva,[2] Eric J. Peterson,[2] Robert C. Calfee,[3] and Thomas F. Stahovich[2]

[1]Computer Science Department, University of California, Riverside
[2]Mechanical Engineering Department, University of California, Riverside
[3]Graduate School of Education, University of California, Riverside

Abstract

We present Newton's Pen, a statics tutor implemented on a "pentop computer," a writing instrument with an integrated digitizer and embedded processor. The tutor, intended for undergraduate education, scaffolds students in the construction of free body diagrams and equilibrium equations. This project entailed the development of sketch understanding techniques and user interface principles for creating pedagogically-sound instructional tools for pentop computers. Development on the pentop platform presented novel challenges because of limited computational resources and a visually static, ink-on-paper display (the only dynamic output device is an audio speaker). We show that a system architecture based on a finite state machine reduces the computational complexity, and serves as a convenient means for providing context-sensitive tutorial help. Our pilot study suggests that Newton's Pen has potential as an effective teaching tool.

1. Introduction

Intelligent tutoring systems have been widely studied and have been applied to a variety of domains [SH04, FAR84, BDM06, RHC*03, VLS*05]. Most current systems are based on WIMP (windows, icons, mouse, and pointer) interfaces. However, research suggests that transfer of skills from training to testing is higher when the testing and training environments are similar [Mes05]. Thus, there is a clear benefit to creating tutoring systems with interfaces that match real-world problem-solving environments.

The long-term goal of our work, therefore, is to create computational techniques to enable natural, pen-based tutoring systems that scaffold students in solving problems in the same way they would ordinarily solve them with paper and pencil. This goal is consistent with recent research comparing student performance across different user interfaces showing that "as the interfaces departed more from familiar work practice..., students would experience greater cognitive load such that performance would deteriorate in speed, attentional focus, meta-cognitive control, correctness of problem solutions, and memory" [OAC06]. While that work used systems that provided no problem-solving assistance (i.e., they were not tutoring systems), the findings provide compelling evidence of the potential benefits of well-designed, pen-based instructional tools.

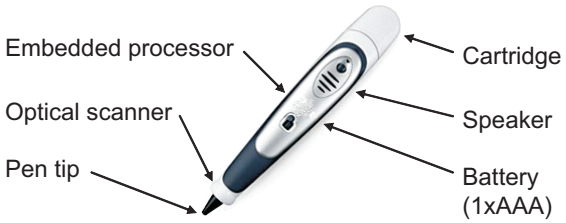


Figure 1: *The FLY pentop computer.*

As a step towards our goal, we have built Newton's Pen, a pen-based statics tutor designed for the LeapFrog FLY pentop computer. (Statics is the sub-discipline of engineering mechanics concerned with the equilibrium of objects subjected to forces.) The FLY, shown in Figure 1, is a writing instrument (pen) with an integrated digitizer and an embedded processor, and is used in conjunction with paper preprinted with a specially designed dot pattern. As one writes on this "digital paper," the digitizer uses the grid to locate the pen tip on the page and digitize the pen stroke. Newton's pen runs entirely on the FLY's embedded processor, which created significant challenges because of the limited memory and computational power available. Audio is the only form

of dynamic output on the FLY: the system has a speech synthesizer and can play recorded sound clips. The platform restrictions presented substantial user interface design issues.

Newton's Pen scaffolds students in the construction of free body diagrams and equilibrium equations. Problems are solved on digital paper preprinted with user interface objects, such as "HELP" and "DONE" buttons. Figure 3a shows a worksheet for drawing a free body diagram. To begin, the student taps the pen on the "START FBD" button, and the system prompts the student to draw the free body diagram in the space provided. After each graphical element is drawn, the system provides interpretive audio feedback. If the student makes a problem-solving error, or the input is not recognized, the system informs the student via synthesized speech. The student can tap "HELP" at any time for audio hints about what to do next, or for guidance after an error. When the student completes the free body diagram, he or she hits the "DONE" button, and is directed to write the equilibrium equations on the worksheet in Figure 3b.

The next section discusses related work. This is followed by an overview of Newton's Pen, and then the details of the system's design. Finally, results of a user study are presented and discussed.

2. Related Work

Intelligent tutoring systems have been developed for a wide variety of domains: medicine [SH04], computer programming [FAR84], electric circuits [BDM06], free body diagrams [RHC*03], and physics [VLS*05]. Nearly all of these systems are based on WIMP or keyboard interfaces. Our work, by contrast, is focused on building pedagogically-sound, pen-based interfaces for tutoring systems. While conventional tutoring systems work from unambiguous input provided with a keyboard and mouse, we focus on the challenges of working from ambiguous, hand-drawn input.

Research on pen-based interfaces is quite active at present. Examples of existing experimental applications include: a tool for simulating simple, hand-drawn, mechanical devices [AD01], a tool for sketching user interfaces [LM01], a UML diagram tool [HD02], a tool for interpreting hand-drawn equations [Mat99], and a tool for understanding military tactics [FFU01]. Likewise, there has been significant progress in sketching 3D shapes [ZHH96].

While sketch understanding techniques have been used for a wide range of applications, the impact on tutoring systems has been limited. [AMS05] describes a classroom interaction system that allows students and instructors to communicate wirelessly in lecture environments using tablet computers. However, this system does not interpret what is written, nor is it intended to provide tutoring capabilities.

Researchers have recently begun to explore issues related to the development of pen-based instructional tools. For example, Oviatt [OAC06] compared student performance using paper-and-pencil, the Anoto digital pen system (a digital pen that digitizes ink but does not run applications) [ano], and the Tablet PC. Students used these platforms solely as a recording medium for problem solving; no tutoring capabilities were provided. This work demonstrated that "as the interfaces departed more from familiar work practice..., students would experience greater cognitive load such that performance would deteriorate in speed, attentional focus, meta-cognitive control, correctness of problem solutions, and memory." This work speaks to the importance of good user interface design in creating effective educational systems. [AYK05] describes a study of methods for entering mathematical equations into a computer. The goal was to explore interface issues for tutoring systems. Although the user input was not interpreted, the study suggests that pen-input of equations is substantially more efficient than keyboard entry, and is greatly preferred by users.

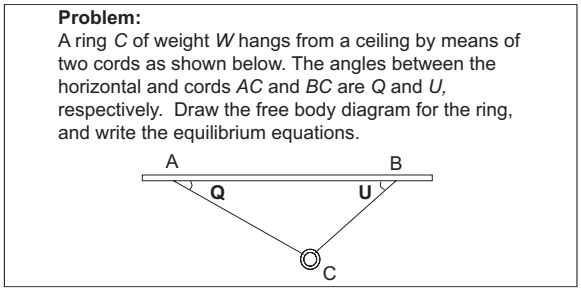
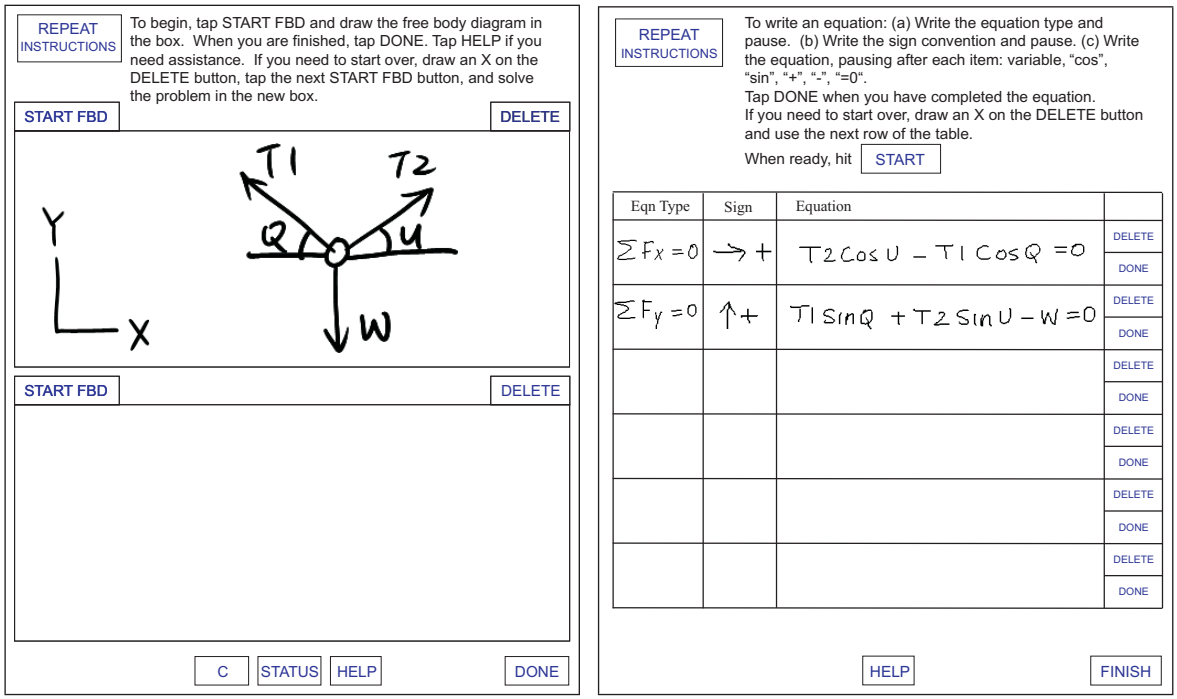


Figure 2: *A typical problem from Newton's Pen.*

Several recent research efforts have aimed at using pentop technology to build pen-based applications. The PapierCraft [LGH05] system enables users to edit documents by writing on paper printouts with an Anoto digital pen. Annotations and command gestures are captured by the digital pen, and are then uploaded to a PC to be interpreted and executed on a digital version of the document. The NISMap system [CM04] is a military planning tool, also based on the Anoto digital pen. As the user annotates a paper map, digitized pen strokes are wirelessly transmitted to a PC for processing. The pen input is processed in real time, but is done so on a traditional computer. The work in [LGL06] has begun exploring methods of providing real-time user feedback directly from a pentop using LEDs, voice coils, and audio speakers.

3. System Overview

Newton's Pen is deployed on the LeapFrog FLY pentop computer (Figure 1), which is based on Anoto [ano] digital pen and paper technology. "Digital paper" is ordinary paper printed with a special dot pattern. The pentop contains a digitizer (camera), which uses the dot pattern to locate the pen tip. The digitizer, which is near the tip of the pen, is activated when the pen is pressed against the paper. The FLY produces two kinds of output: it leaves ink on the paper, and can produce synthesized speech and sound clips through a speaker or headphone jack.



(a) (b)

Figure 3: *(a) Worksheet for drawing free body diagrams. (b) Worksheet for writing equilibrium equations.*

The Newton's pen system consists of software and specially designed worksheets. The software is contained on a flash memory cartridge and runs entirely on the FLY's embedded processor. The worksheets, which are printed on digital paper, contain instructions for using the system, user interface objects (i.e., buttons), and spaces for solving problems. Each problem is laid out in three pages: one page, on ordinary paper, for the problem description (Figure 2), one for the free body diagram (Figure 3a), and one for the equilibrium equations (Figure 3b).

Figure 2 shows a typical problem in which the student is asked to draw a free body diagram for a ring of weight W supported by two cords. Figure 3a shows a worksheet for drawing free body diagrams. The top of the page has instructions for using the system. As the student works, the system provides additional instructions via synthesized speech. The "REPEAT INSTRUCTIONS" button at the top of the page is used to repeat the last audio instruction. Below the printed instructions are three blocks for drawing free body diagrams (only two are shown in the figure), which allow the student to make three attempts at solving the problem. There are two buttons at the top of each block: "START FBD" and "DELETE". To begin, the student taps the pen on the "START FBD" button, and the system prompts the student to draw the coordinate system and then the body in the space provided. Once these are completed, the student is prompted to draw and label the forces, which can be drawn in any order. After each graphical element is drawn, the system provides interpretive audio feedback. If the student makes a problem-solving error, or the input is not recognized, the system informs the student by saying "try again." At any time, the student can tap the "HELP" button at the bottom of the page for audio hints about what to do next, or for guidance following an error.

If the student makes several mistakes while drawing a free body diagram, he or she can start over in the next block. Before doing so, the student marks an 'X' on the "DELETE" button to indicate that the current work should be discarded. The student then taps the "START FBD" button of the next block, and begins as before.

There are three additional buttons at the bottom of the page: "C", "STATUS", and "DONE". The "C" (continue) button is used for debugging purposes and causes the system to advance to the next step in the solution, as if the current step had been completed correctly. The "STATUS" button informs the student of the number of forces still to be drawn to complete the free body diagram. When the student taps the "DONE" button, the system analyzes the free body diagram. If it is complete, the student is congratulated. If forces are missing, the system notifies the student of the number of missing forces, just as the "STATUS" button would. The "STATUS" and "DONE" buttons do perform similar functions. However, the student uses the former to query for help, and the latter to indicate that the problem has been solved. This distinction will eventually be used in assigning credit.

Figure 3b shows the worksheet for writing equilibrium equations. It provides a table for the student to write equations, where each row contains a space for the equation type,

the sign convention, and the equation. The equation type denotes the generic equilibrium equation, and the sign convention indicates the positive coordinate direction. For example, "$\Sigma Fx = 0$" and "$\rightarrow +$" indicate an equilibrium equation for the x-direction, with positive defined to the right. Requiring the student to provide this information has several benefits: It supports meta-cognition, is consistent with the way statics is commonly taught, and allows Newton's Pen to provide focused feedback.

As would be expected of such a small device, the FLY has limited computational power and memory (RAM). Because of these limitations, computational complexity was of paramount concern in designing the system. For example, we employed manual rather than automatic sketch segmentation. (Segmentation is the task of decomposing a sketch into the individual objects that comprise it.) Like other applications for the FLY, Newton's Pen requires the user to pause after drawing each object. Following a pause, the system attempts to recognize the object, and then announces the result via synthesized speech. For example, if a force arrow is recognized, and is drawn in the correct location and orientation, the system says "force." If the input is not recognized as an arrow, or the force is drawn incorrectly, the system says "try again." The student can double tap the pen after drawing an object to cause immediate recognition, thus avoiding the need to pause.

As another means of reducing computation, Newton's Pen requires certain compound objects to be drawn in a particular order. For example, to draw a force not aligned with a coordinate direction, the student draws an arrow, the force label, a leader line, an arc to indicate the angle, and finally the angle label, all in this order, with a pause after each object. Figure 4 shows an example.

In addition to the normal problem-solving mode, Newton's Pen has a demonstration mode that familiarizes students with the system. The demonstration begins by providing step-by-step instruction for drawing and labeling forces. The student is then shown an example of a force and is asked to copy it. Next, the student is shown a sample problem, and the complete free body diagram. The system asks the student to copy the diagram, and provides step-by-step audio instruction for doing so. Likewise, the student is shown the equilibrium equations and is guided through the process of copying them. Before starting the demonstration, we typically ask students to practice by trying commercial FLY applications such as the calculator or music keyboard ("FLY Tones") applications. This helps students become familiar with basic interaction concepts, such as pausing after drawing and waiting for audio feedback.

4. System Design

The following sections describe how Newton's Pen performs its tasks. In particular the system architecture, recognition techniques, equation interpreter, and tutoring techniques are presented.

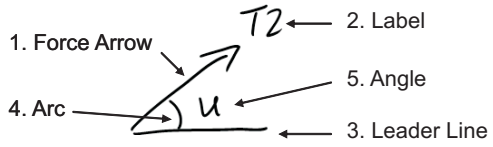


Figure 4: *A typical force. Numbers indicate required drawing order.*

4.1. System Architecture

Because of the limited computational resources on the FLY, we require the student to solve problems in a particular order. For example, when drawing a free body diagram, the student must first draw the coordinate system, then the body, and finally the forces. Also, as discussed in Section 3, there is a prescribed drawing order for drawing and labeling forces.

Because of the sequential nature of interaction with Newton's Pen, we employ a system architecture that is essentially a non-deterministic finite state machine. (Each problem actually has two finite state machines: one for the free body diagram, and one for the equilibrium equations.) In our case, a "state" is a C++ object that contains a list of legal inputs, a list of next states, and context sensitive help. Inputs to states consist of hand-drawn, graphical objects, such as arrows, text, leader lines, etc. The expected properties of each graphical object are encoded in the state. For example, a force arrow is associated with an expected orientation and location on the body. Each legal input for a state is associated with a specific recognizer that can identify that kind of input (see Section 4.2). After the student has drawn an object and paused (or double tapped), the system calls each of the recognizers associated with the current state to determine if the input is one of the legal inputs for that state. If so, the system advances to the appropriate next state. If not, the system uses synthesized speech to say "try again," and remains in the same state.

Consider the free body diagram on the worksheet in Figure 3a, describing a particle subject to three forces $T1$, $T2$, and W. A portion of the corresponding finite state machine is shown in Figure 5. At the instant shown, the student has drawn the coordinate system and body, and the system is in state *Forces*. There are three legal inputs to this state: Arrow1, the arrow corresponding to force $T1$; Arrow2, the arrow corresponding to force $T2$; and Arrow 3, the arrow corresponding to force W. If the student draws an arrow matching Arrow3, for example, the system will transition to state *Label_W* in which the only legal input is the text label "W". If the student writes a "W" while in this new state, the system will transition back to state *Forces* and wait for the other forces to be drawn. Thus, our C++ "state" object actually represents multiple logical states.

This architecture has a number of advantages. First, it provides a convenient method for providing context sensitive help. Each state is associated with help messages that are announced to the student when the help button is pressed. Because the program's state mirrors the student's problem-

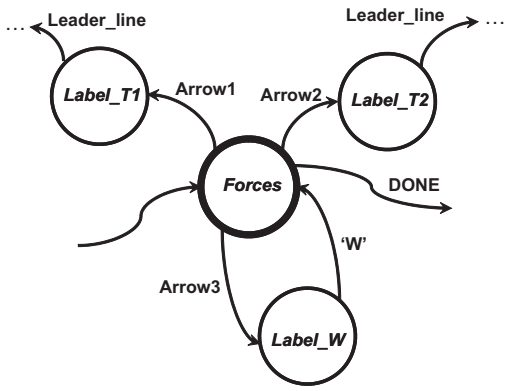


Figure 5: *A portion of the finite state machine corresponding to the free body diagram in Figure 3a. The system is currently in state* Forces.

solving progress, the help messages are always relevant. Second, the architecture allows new problems to be easily coded. A new problem is created by instantiating the appropriate set of state objects. Third, the architecture allows the system to be run in a demonstration mode that guides the student in the use of the system (see Section 3). Each state contains an explanation of what should be drawn in that state. In demonstration mode, when a state is entered, the explanation is announced to the student. Fourth, the architecture improves recognition accuracy by limiting the number of graphical objects that are expected at any given time, thus limiting opportunities for confusion.

4.2. Recognition

Newton's Pen must recognize a variety of hand-drawn objects including: text, force arrows, leader lines and arcs used to indicate angles, and bodies. In most recognition-based systems, the task is to determine which of many possible classifications should be assigned to an arbitrary symbol. By contrast, our task is to determine whether or not a given object matches the expected classification. For example, our system must determine if a symbol is or is not an arrow. Because of the nature of our task, we can use a set of efficient, special-purpose recognizers, each suited for specific types of symbols. The current state of the finite state machine determines which recognizer is used at any given time. This is less expensive than using a single, general-purpose recognizer. Also, because our task is typically to verify the class of a symbol, rather than to select the classification from multiple possibilities, our recognizers need not be as robust as those required for typical recognition-based interfaces. The following sections describe the various recognizers we use.

4.2.1. Character Recognizer

The FLY has an integrated character recognizer which we use to recognize force and angle labels, and the characters in equations. To improve accuracy while recognizing labels, we bias the character recognizer to the force and angle labels used in the problem. Biasing the recognizer helps to prevent situations when "T1" is misclassified as "TI", for instance. The character recognizer provides a confidence value for the recognition result. If the confidence is below a threshold, Newton's Pen considers the ink to be unrecognizable.

4.2.2. Image-Based Recognizer

We have implemented a computationally-inexpensive, image-based recognizer for recognizing arrows and the sign convention portion of an equation, such as "$\rightarrow +$". This recognizer is based on one previously developed by our group [KS05], but is modified for the limited computational resources of the FLY.

Symbols are internally represented as binary, bitmap images or "templates." To construct a template, the initial image is framed and sampled into a 17x17 square grid. To frame the image, a bounding box aligned with the screen axes is constructed and then scaled to form a square. Pen strokes often have sparse data points. When sampling the framed image, it is necessary to interpolate between consecutive points in the pen strokes.

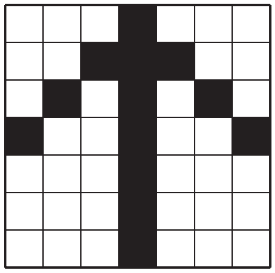
We use a dissimilarity score to compare an unknown symbol, U, to a definition symbol, D. The dissimilarity score, $h(U,D)$, is defined as:

$$h(U,D) = \frac{1}{N_U} \sum_{u \in U} \min_{d \in D} \|u - d\| \qquad (1)$$

where u is a black pixel in the unknown symbol, d is a black pixel in the definition symbol, N_u is the number of black pixels in the unknown, and $\|u-d\|$ is the Euclidean distance from u to d. This is similar to the Modified Hausdorff Distance (MHD) [DJ94], except that we consider a uni-directional rather than bi-directional match: $MHD(U,D) = max(h(U,D),h(D,U))$. A uni-directional match is adequate, in part, because we have few definitions. Also, the non-uniform scaling used to frame the unknown helps prevent some false-positive matches that could occur if the unknown were a subset of a definition. For example, when framed, a (nearly) vertical line becomes a diagonal line, and thus does not match the definition of a vertical arrow.

The computation in Equation 1 can be accelerated by transforming the definition template into a "distance map," in which each pixel encodes the Euclidean distance to the nearest black pixel in the definition template. Using the distance map, the $\min_{d \in D} \|u-d\|$ can be determined by simply reading the value encoded in the definition pixel that corresponds to u.

To reduce computation, we use integer math, rather than floating point math, whenever possible. Thus, distance maps are constructed with integer values. To maintain accuracy, distances are initially computed using floating point math, and are then scaled by 10 before converting to integers. The maps are constructed off-line (on a PC) and are compiled into the executable as "const" data.



22	14	10	0	10	14	22
14	10	0	0	0	10	14
10	0	10	0	10	0	10
0	10	10	0	10	10	0
10	14	10	0	10	14	10
20	20	10	0	10	20	20
30	20	10	0	10	20	30

(a) (b)

Figure 6: *(a) A 7x7 binary bitmap of an arrow. (b) The corresponding distance map.*

We use a 17x17 grid to reduce storage. With a grid this size, the largest-possible, scaled, integer distance (the distance between diagonally opposed corners) can be represented with a single byte. We use a grid with odd, rather than even, numbers of rows and columns so that we can accurately represent shapes that have a line along the center of the template, such as a horizontal or vertical arrow.

Figure 6 shows an example of a distance map for a 7x7 image of an arrow. (Here we show a 7x7 rather than 17x17 grid for clarity of illustration.) The distance map values start at 0 for pixels that are themselves black, and increase as one moves farther into the white space. For example, the distance transform value of the lower left white pixel is 30 because the nearest black pixel is 3 units away.

Our library of definitions includes eight arrows oriented at 0^o, 45^o, 90^o, 135^o, 180^o, 225^o, 270^o, and 315^o. It also includes two symbols for sign convention: "$\rightarrow +$" and "$\uparrow +$". (In the future, we will allow more general sign conventions, but this was adequate for our initial set of problems.)

An unknown symbol is classified by the definition that matches with the minimum dissimilarity score computed with Equation 1. However, if the minimum score is greater than 26, the symbol is considered to be unrecognizable. The distance map is scaled by 10, thus the threshold is actually 2.6 drawing units, or about 10% of the length of the diagonal of the 17x17 grid.

4.2.3. Body Recognizer

For the problems we have considered, bodies are either circles or polygons. A circular body must be drawn with a single pen stroke, and is recognized as a closed contour whose bounding box is approximately the same size as the body in the problem description. The edges of a polygonal body must be drawn with separate pen strokes. Each edge is characterized by a relative length and an approximate angle. The relative length of an edge is the ratio of the length of that edge to the sum of the lengths of all of the edges. For example, the relative length of each edge in a square is 0.25. Angles are specified with a resolution of 45^o. If a set of edges form a closed polygon, and are within tolerance of the expected angles and relative lengths, those edges are a match for the body.

4.2.4. Angle-Indicator Recognizer

The angle of a force is indicated with a leader line and arc as shown in Figure 4. A leader line is a pen stroke that is roughly straight, and approximately horizontal or vertical. Here we make use of the "straightness ratio," defined as the ratio of the distance between the first and last points of a stroke, to the total length of the stroke. If the straightness ratio is less than or equal to 0.2, the stroke is considered straight. A line is assumed to be horizontal if its vertical extent is less than 30% of its horizontal extent. Vertical lines are defined analogously. A pen stroke is considered to be an arc if the straightness ratio is greater than 0.2.

4.3. Equation Interpreter

Our current system can interpret only a limited class of equations appropriate for the set of problems we have considered thus far. Currently, the system can interpret force equilibrium equations, but not moment equilibrium equations. The latter are unnecessary for particle equilibrium problems. Also, equations cannot have parentheses.

We use FLY's character recognizer to recognize the individual characters in an equation. We then use a simple tokenizer to identify the specific terms in an equation, including: variable names, single digits, "SIN", "COS", "+", "-", and "=". A variable name is a single capital letter, optionally followed by a single digit. If the program finds any variable names that are not in the problem description, an error is reported to the student. Otherwise, the equation is parsed using a simple parser.

The results of parsing are stored in a matrix representation. For the class of problems we consider, a term in an equilibrium equation will consist of a sign, a force label, and a force component. The component is either the sine or cosine of an angle, or 1.0. Thus, all possible terms in an equation can be represented by a matrix in which the rows correspond to the possible force labels, and the columns correspond to the sines and cosines of the possible angles, and 1.0. A "1" in the matrix indicates that the term exists in the equation. A "-1" indicates that the term exists, and has a negative sign. A "0" indicates that the term does not exist in the equation.

Consider, the problem in Figure 3a which has three forces, $T1$, $T2$, and W, and two angles, U, and Q. Equilibrium equations for this problem can be represented by the 3x5 matrix shown in Table 1. Equation 2 shows one of the two equilibrium equations for this problem. The first term in the equation, "T1 SIN Q", is represented by the "1" in the upper left corner of the matrix. The second term, "T2 SIN U", is represented by the "1" in row two, column three. The third term, "$-W$", is represented by the "-1" in the lower right corner.

The program derives the correct equation matrices for each problem directly from the problem description. When the student's equations have been parsed, the resulting matrices are compared to the correct ones to detect errors. By

comparing the matrices term by term, the program can identify missing terms, extra terms, and sign errors.

$$T1SINQ + T2SINU - W = 0 \quad (2)$$

	SINQ	*COSQ*	*SINU*	*COSU*	1
$T1$	1	0	0	0	0
$T2$	0	0	1	0	0
W	0	0	0	0	- 1

Table 1: *The matrix representation of Equation 2.*

4.4. Tutoring

Our system's finite state machine architecture enables it to provide context sensitive help. As discussed in Section 4.1, each state contains a set of help messages that are announced to the student with a speech synthesizer when the "HELP" button is tapped. Because the program's state mirrors the student's problem-solving progress, the help messages are always relevant. The first time a student taps the "HELP" button in a given state, he or she receives general help. Successive taps of the button result in more specific help. For example, when the current state requires the student to draw the body for the free body diagram in Figure 3a, the sequence of help messages is: "draw the body", "the body is a ring", and "draw a circle ."

The equation interpreter, described in Section 4.3 also provides some tutorial feedback. In particular, it notifies the student of errors in the equilibrium equations.

5. User Study and Results

We conducted a pilot study to assess the educational value of Newton's Pen and to obtain a preliminary evaluation of the usability of its user interface. The study included nine volunteer test subjects from an introductory physics class at the University of California, Riverside. The subjects had just completed lectures on free body diagrams, and had been assigned homework problems, but had not yet begun solving them. During hour-long sessions, students were given a pre-test problem to be solved with ordinary pen and paper, a model-problem using Newton's Pen in demonstration mode, a transfer problem using Newton's Pen (i.e., with scaffolding), and a "re-test" of the pre-test using Newton's Pen. The problems were comparable to the one in Figure 2.

At the time of the pilot study, the free body diagram interpreter was implemented. The equation interpreter, and the portion of the demonstration mode in which the student practiced drawing an isolated force, were simulated with a Wizard-of-Oz approach. A member of the research team read a script to simulate the audio feedback the system would have provided had this functionality been implemented.

From an educational perspective, the results were clear and encouraging: Every student was stymied by the pretest. Every student then worked carefully through the model problem in demonstration mode. Every student successfully solved the first transfer problem, and was then successful when retested on the pretest (one student did not do the retest). The "diagram time" – the time from presentation of the problem to completion of the free body diagram – decreased steadily from the model to the pretest-retest. Thus, learning with evidence of transfer occurred over the course of an hour.

The pilot study suggested that the user interface could use refinement. Most students eventually became proficient with the interface, but initially needed assistance in using the system. For example, some students initially needed reminders about pausing at the right time or about drawing compound objects in the correct order. Nevertheless, students were extremely enthusiastic about the system. They made comments such as "it answered all of the questions I would have asked the professor" and "it was like having the professor in my hand."

6. Discussion and Future Work

Currently, Newton's Pen can handle a select class of problems having to do with the equilibrium of particles. We are working to extend the system to a much broader class of problems, including finite bodies requiring moment equilibrium equations, friction, springs, multi-body devices, etc. This extension will require expanding the system's knowledge of free body diagrams and the range of equations it can interpret.

We are also working to improve the user interface. Synthesized speech is efficient for development, but can become irritating. We plan to use recorded audio clips in the future. Also, we plan to allow more flexibility in the order in which objects are drawn. We currently enforce specific drawing orders because this reduces computational complexity – the system has only a few interpretations to consider at any given time. However, a rigid drawing order can be unnatural. For example, some users prefer to draw all of the force arrows first, and then label the angles. Others like to completely specify one force before considering the next. Additionally, we currently require each edge of a polygonal body to be drawn with a separate pen stroke. We plan to use a pen stroke segmenter we developed [Sta04] to allow multiple edges, or even entire polygons, to be drawn with a single stroke.

Newton's pen currently has a simple tutorial help system, which was adequate for our initial prototype. Our technique of providing successively more concrete help with each tap of the "HELP" button has proven to be an effective instructional design. However, we need to improve the content of the tutorial help. For example, if a student draws a force in the wrong direction, the system currently reports that the force is incorrect. A better approach would be to explain what the likely misconception is. For example, if the student draws the weight force inclined from vertical, and perpendicular to a surface on the body, the system could report that

the student has confused weight and normal forces: weight forces are vertical while normal forces are determined by the surface normal. In this sense, our improved tutorial help system will be a form of the "buggy rules" approach to intelligent tutoring [FAR84].

7. Conclusion

We have presented Newton's Pen, a statics tutor implemented on the LeapFrog FLY pentop computer. A pentop is a writing instrument with an integrated digitizer and embedded processor. Our tutor, intended for undergraduate education, scaffolds students in the construction of free body diagrams and equilibrium equations for a selected class of problems. This project has entailed the development of sketch-understanding techniques and user-interface principles for creating pedagogically-sound instructional tools for pentop computers. Development on the pentop platform presented novel challenges because of limited computational resources and a visually static, ink-on-paper display (the only dynamic output device is an audio speaker). We have demonstrated that a system architecture based on a finite state machine reduced the computational complexity, and served as a convenient means for providing context-sensitive tutorial feedback. Newton's Pen is a prototype, and there is more work to be done, including improving its user interface and expanding its tutorial knowledge base. Nevertheless, our pilot study suggests that Newton's Pen has potential as an effective teaching tool.

8. Acknowledgments

The authors are grateful to LeapFrog Enterprises, Inc. for their support for this work.

References

[AD01] ALVARADO C., DAVIS R.: Resolving ambiguities to create a natural sketch based interface. In *IJCAI'01* (2001), pp. 1365–1371.

[AMS05] ANDERSON R., MCDOWELL L., SIMON B.: Use of classroom presenter in engineering courses. In *FIE'05* (2005), pp. T2G–13–18.

[ano] Anoto group AB. http://www.anoto.com/.

[AYK05] ANTHONY L., YANG J., KOEDINGER K. R.: Evaluation of multimodal input for entering mathematical equations on the computer. In *CHI '05* (2005).

[BDM06] BUTZ B. P., DUARTE M., MILLER S. M.: An intelligent tutoring system for circuit analysis. In *IEEE Transactions on Education* (2006), vol. 49, pp. 216–223.

[CM04] COHEN P. R., MCGEE D. R.: Tangible multimodal interfaces for safety-critical applications. *Communications of the ACM 47*, 1 (2004).

[DJ94] DUBUISSON M.-P., JAIN A. K.: A modified hausdorff distance for object matching. In *12th International Conference on Pattern Recognition* (1994), pp. 566–568.

[FAR84] FARRELL R. G., ANDERSON J. R., REISER B. J.: An interactive computer-based tutor for LISP. In *AAAI'04* (1984), pp. 106–109.

[FFU01] FORBUS K. D., FERGUSON R. W., USHER J. M.: Towards a computational model of sketching. In *6th International Conference on Intelligent User Interfaces* (2001), pp. 77–83.

[HD02] HAMMOND T., DAVIS R.: Tahuti: A geometrical sketch recognition system for UML class diagrams. In *AAAI Spring Symposium on Sketch Understanding* (2002), pp. 59–68.

[KS05] KARA L. B., STAHOVICH T. F.: An image-based, trainable symbol recognizer for hand-drawn sketches. *Computers & Graphics 29*, 4 (2005), 501–517.

[LGH05] LIAO C., GUIMBRETIERE F., HINCKLEY K.: Papiercraft: a command system for interactive paper. In *UIST '05* (2005), pp. 241–244.

[LGL06] LIAO C., GUIMBRETIERE F., LOECKENHOFF C. E.: Pen-top feedback for paper-based interfaces. In *UIST '06* (2006), pp. 201–210.

[LM01] LANDAY J. A., MYERS B. A.: Sketching interfaces: Toward more human interface design. *IEEE Computer 34*, 3 (2001).

[Mat99] MATSAKIS N.: *Recognition of Handwritten Mathematical Expressions*. Master's thesis, MIT, Cambridge, MA, 1999.

[Mes05] MESTRE J. P. (Ed.): *Transfer of learning from a modern multidisciplinary perspective*. Information Age Publishing, Greenwich, CT, 2005.

[OAC06] OVIATT S., ARTHUR A., COHEN J.: Quiet interfaces that help students think. In *UIST '06* (2006), pp. 191–200.

[RHC*03] ROSELLI R. J., HOWARD L., CINNAMON B., BROPHY S., NORRIS P., ROTHNEY M., EGGERS D.: Integration of an interactive free body diagram assistant with a courseware authoring package and an experimental learning management system. In *ASEE'03* (2003).

[SH04] SUEBNUKARN S., HADDAWY P.: A collaborative intelligent tutoring system for medical problem-based learning. In *IUI '04* (2004), pp. 14–21.

[Sta04] STAHOVICH T. F.: Segmentation of pen strokes using pen speed. In *AAAI Symposium, Making Pen-Based Interaction Intelligent and Natural* (2004).

[VLS*05] VANLEHN K., LYNCH C., SCHULZE K., SHAPIRO J. A., SHELBY R., TAYLOR L., TREACY D., WEINSTEIN A., WINTERSGILL M.: The andes physics tutoring system: Lessons learned. *International Journal of AI in Education 15*, 3 (2005).

[ZHH96] ZELEZNIK R., HERNDON K., HUGHES J.: SKETCH: An interface for sketching 3D scenes. In *SIGGRAPH '96* (1996), pp. 163–170.