EUROGRAPHICS Symposium on Sketch-Based Interfaces and Modeling (2012)
L. B. Kara and K. Singh (Editors)

Newtons Pen II: An Intelligent, Sketch-Based Tutoring System and its Sketch Processing Techniques

C. Lee[2] and J. Jordan[2] and T. F. Stahovich[1] and J. Herold[2]

[1]Department of Mechanical Engineering, University of California, Riverside, CA 92521, United States
[2]Department of Computer Science, University of California, Riverside, CA 92521, United States

Abstract

We present a pen-based intelligent tutoring system (ITS) for undergraduate Statics which scaffolds students in the construction of free body diagrams and equilibrium equations. Most existing ITSs rely on traditional WIMP (Windows, Icons, Menus, Pointers) interfaces, which often require the student to select the correct answer from among a set of predefined choices. Our system, by contrast, guides students in constructing solutions from scratch, mirroring the way they ordinarily solve problems, which recent research suggests is important for effective instruction. Our system employs several new techniques for sketch understanding, including a simple-to-implement stroke merging technique, a stroke clustering technique, and a technique that uses a Hidden Markov Model to correct interpretation errors in equations. Our tutoring system was deployed in an undergraduate Statics course at our university. Attitudinal surveys indicate that the tutoring system is preferable to traditional WIMP-based systems and is an effective educational tool.

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [H.5.2]: Information interfaces and presentation (e.g., HCI)—User InterfacesPrototyping

1. Introduction

We present a pen-based tutoring system that scaffolds students in solving Statics problems similar to how they ordinarily solve them with pencil and paper. The development of the system is motivated by recent research that has provided compelling evidence for the potential of natural user interfaces for instructional tools. The research by Oviatt [OAC06] showed 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."

Our Statics tutoring system, called Newton's Pen II (NP2), provides a natural user interface for students; it accepts digital input from a stylus and avoids common WIMP (Windows, Icons, Menus, Pointers) interactions that would be cumbersome to perform with a pen, such as menu scrolling and entering typed text.

NP2 guides students in constructing and labeling free body diagrams (FBD) and deriving the corresponding equilibrium equations. NP2 employs numerous stroke recognition and parsing techniques to understand students' hand drawn FBDs and equations. In the FBD workspace of the interface, NP2 identifies traced graphical objects, associates hand drawn labels with relevant components, verifies that all required components are included in the FBD boundary, and verifies that the student has correctly identified all necessary forces and moments. In the equilibrium equation workspace, NP2 recognizes and interprets multi-stroke mathematical equations. Here, we consider the design of the interpretation algorithms. For a discussion of the instructional design and educational effectiveness see [LS11].

To evaluate the educational effectiveness of NP2, a user study was conducted in an undergraduate Statics course at our university. A survey of student opinions regarding the usefulness and usability of the system was conducted.

The next section places this work in the context of related work. This is followed by an overview of the NP2 system, and then detailed description of the sketch understanding

DOI: 10.2312/SBM/SBM12/057-065

techniques employed in the FBD and equilibrium equation sections. Lastly, the user study used to evaluate NP2 is described and the results of the attitudinal survey taken at the end of the study are reported.

2. Related Work

Intelligent tutoring systems (ITS) have been developed for a wide variety of domains [RHC*03, SH04, VLS*05]. Almost all existing ITS employ WIMP interfaces. Our system utilizes a pedagogically-sound, pen-based interface. While conventional systems work with unambiguous input provided with a keyboard and mouse, our system is designed to work with ambiguous, hand-drawn input.

One of the more widely used pen-based instructional systems is the Classroom Presenter [AMS05]. This system assists in students' and instructor's communication during lectures by using a networked set of tablet computers. However, this system does not interpret what is written and provides no tutorial assistance.

Newton's Pen [LdSP*07] is a Statics tutoring system implemented on LeapFrog Inc.'s FLY^{TM} pentop computer. Newton's Pen is unable to provide instruction for complex frame and machine problems that involve multiple rigid bodies. Enabling NP2 to handle such problems required the development of new interpretation techniques and new user interface techniques.

The Mechanix system [KFH11] is another pen-based tutoring system for teaching statics. This system is limited to the analysis of trusses, and cannot handle the more complicated frame and machine problems that our system supports. Also, while our system can interpret and critique handwritten equilibrium equations, Mechanix cannot.

More recent examples of sketch-based tutoring systems include PhysicsBook [CL12] and LogicPad [KJ12], which can be used to tutor students in solving physics problems and constructing boolean algebra visualizations respectively. Newton's Pen II is distinguished from these systems in that it interprets users' hand drawn free body diagrams and equations and also provides scaffolding for students based on those interpretations.

Kirchhoff's Pen is a pen-based tutoring system for teaching Kirchhoff's Law [dSBL*07]. While Kirchhoff's Pen is also developed for the tablet PC, there are several distinct differences between it and NP2. For example, the two domains involve very different recognition tasks, and the clustering and error correction methods used are different.

Recognition in hand-drawn sketches is difficult because of the ambiguity inherent in human hand writing. Several recognition techniques have been developed, including: a graph-based recognizer [LKS06], an image-based recognizer [KS04], a feature-based recognizer [Rub91], a drawing-order-dependent recognizer [SD05], and many more. NP2 employs a combination of the inverse-curvature recognizer [LdSP*08] and the Dollar recognizer [WWL07]. However, the latter is extended to facilitate better recognition of arrows.

Stroke clustering has been a long researched area in sketch understanding. Many prior recognition systems choose to avoid this task by restricting the user's input, such as recognizing single stroke objects [Rub91,LM01], completing one object before moving on to the next [GKSS05], and asking users to explicitly group the strokes together by either pressing a button or waiting for a timeout [HN05,FPJ02]. While our system utilizes some of these approaches, it also includes other innovative clustering methods, such as a feature-based method for clustering equations which is based on the work of Peterson [PSDA10].

3. System Overview

Figure 1 shows the NP2 user interface. The top portion of the program window contains the problem description and the FBD workspace where the student constructs and labels FBDs. The bottom portion contains the equation workspace where the student constructs the equilibrium equations for each FBD.

To construct a FBD, the student must identify the boundary of the system of interest. This is performed by tracing a boundary directly over the problem picture with the stylus as illustrated in Figure 2. After the student traces a boundary, he clicks the "Recognize Boundary" button to trigger interpretation. NP2 then identifies which parts of the problem have been included in the system boundary. Once the student has defined a suitable boundary, he can use the stylus to drag a beautified version of it to the FBD workspace.

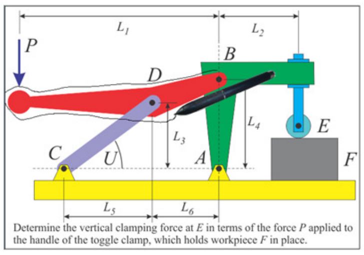
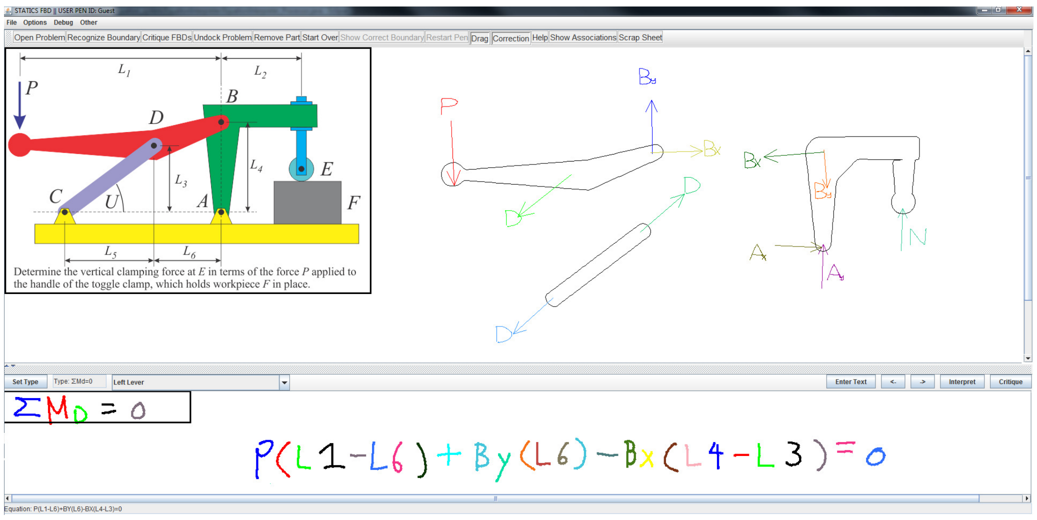

L_1
L_2
P
B
D
L_4
E
L_3
C
U
A
F
L_5
L_6
Determine the vertical clamping force at E in terms of the force P applied to the handle of the toggle clamp, which holds workpiece F in place.

Figure 2: *An example of a FBD boundary trace in progress.*

To complete the FBD, the student must draw the forces and moments acting on the boundary, using straight and curved arrows, respectively. During our user study, arrows were drawn with a single stroke. However, we have also implemented a recognizer to support multi-stroke arrows. If a pen stroke is recognized as a force or a moment, it is replaced by a machine-drawn arrow. The student must label each force and moment arrow; NP2 associates each arrow with the nearest label. To show the association, the arrow and its label are rendered in the same unique color.



Determine the vertical clamping force at *E* in terms of the force *P* applied to the handle of the toggle clamp, which holds workpiece *F* in place.

Equation: P(L1-L6)+BY(L6)-BX(L4-L3)=0

Figure 1: *The NP2 user interface. The top of the application window contains the problem description and FBD workspace. The bottom contains the equation workspace.*

To create additional FBDs for multi-body problems, the student simply repeats these steps. When all of the FBDs are completed, the student clicks the "Critique FBDs" button to get tutorial feedback. If all the FBDs are correct, the student is instructed to write the equilibrium equations. Otherwise, appropriate tutorial feedback is given to the student. Section 4 describes the methods NP2 uses to interpret FBD diagrams. See [LS11] for a description of the tutorial feedback and instructional design.

To construct an equilibrium equation for a particular FBD, the student first selects the name of the body from a drop-down list. The student then specifies the *equation type*, which is either a moment equilibrium equation or a force equilibrium equation in either the X- or Y-direction. For example, to indicate an equilibrium equation in the X-direction, the student writes $\Sigma F_x = 0$ in the equation type field (Figure 3.b). Next, the student writes the equation itself (Figure 3.c) and clicks the "Interpret" button to trigger recognition. NP2 displays the interpretation in the equation display area (Figure 3.d) and uses color coding to show how the pen strokes are grouped into characters. If there are recognition errors, the student may erase and rewrite that portion of the equation or select the correct interpretation for a specific character from a list of alternatives. Once the interpretation is correct, the student clicks the "Critique" button to receive tutorial feedback. Section 5 describes the techniques used for interpreting and critiquing equations. Again, [LS11] contains a thorough description of the tutorial feedback and instructional design.

4. Interpreting Free Body Diagrams

This section describes the techniques NP2 uses to interpret a FBD, including techniques for interpreting the system boundary, techniques for recognizing arrows, and techniques for interpreting force/moment labels.

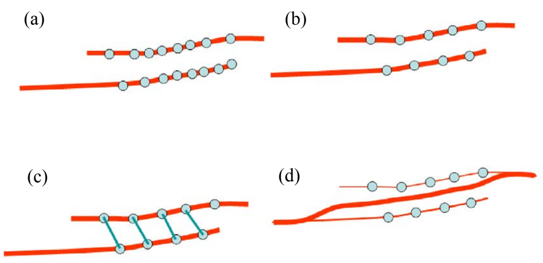
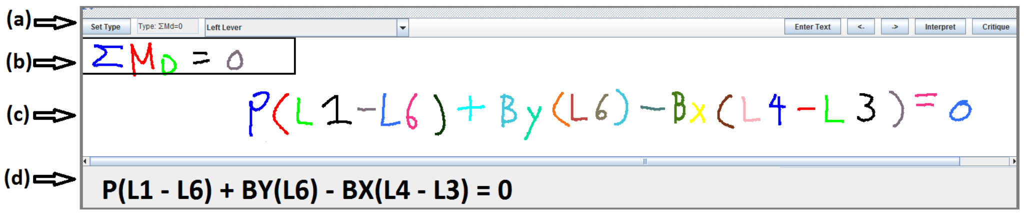


Figure 4: *Stroke merging. (a) raw data points. (b) Re-sampled data points. (c) Aligned points on one stroke with those on the other. (d) Merged stroke constructed by merging matched points.*

4.1. Stroke Merging

The notion of a system boundary is a critical part of the NP2's instructional design. Novice students often have difficulty defining a system, and may attempt to draw a free body diagram by simply redrawing the original problem figure. NP2 is designed to overcome such misconceptions by guiding the student to explicitly identify the boundary of the system. Having such a boundary helps the student distinguish between external forces which must be represented on the free body diagram, and internal forces which should not appear there.

To define a boundary, the student directly traces a closed contour around components or portions of components depicted in the problem picture. Students may construct the contour with multiple pen strokes. Thus, to interpret the



P(L1 - L6) + BY(L6) - BX(L4 - L3) = 0

Figure 3: *The equation workspace: (a) action buttons, (b) equation type area, (c) equation area, (d) equation interpretation area. The ink in the equation is color coded to display the grouping of the strokes into characters.*

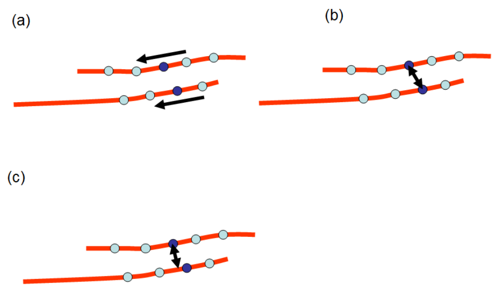


Figure 5: *Stroke merging criteria. (a) Tangent angle must differ by no more than* 10°*. (b) Euclidean distance must be less than 35 pixels. (c) Perpendicular distance must be less than 20 pixels.*

boundary, it is necessary to combine the pen strokes into a closed contour. While various contour identification techniques exist [MLYS05, BGLSS03], none are suitable for merging pen strokes.

Our pen stroke merging technique, illustrated in Figure 4, examines nearby strokes to determine if they nearly overlap, and thus should be merged together. To reduce computation, each stroke is first resampled to 128 data points. Additionally, only the first and last 10% of each stroke is considered for possible merging. To reduce noise, the strokes are also smoothed.

Figure 4(a) shows the start of one stroke (bottom) and the end of another (top). Part (b) of the figure shows the resampled ink, and part (c) shows the assignment of points on one stroke to the nearest points on the other. If a pair of assigned points satisfies the merging criteria, it is merged by creating a new point located at the average position of the two original points (Figure 4(d)).

The merging criteria are illustrated in Figure 5. For two points to be merged, the drawing directions must be the same (Figure 5(a)). We have observed that users typically draw contours with a consistent orientation. For example, when drawing a square, users typically draw all strokes clockwise or all counterclockwise. Thus, two strokes must have the same drawing direction for them to be part of a contiguous path. In the example of the square, two strokes forming one side of the square would have the same drawing directions and could be merged. However, strokes from opposite sides would have opposite drawing directions and would not be merged. We measure drawing direction using the tangent to the pen stroke. To be considered for merging, the tangents of two points cannot differ by more than 10°.

For two points to merged, they must be near each other. The Euclidean distance between them must be less than 35 pixels (Figure 5(b)). Additionally, the perpendicular distance between the two strokes, measured near the points, must be less than 20 pixels (Figure 5(c)). Finally, to prevent neighboring points on a single stroke from being merged, the Euclidean distance between the points must be at least twice as large as the path length between the points. This allows two ends of a single stroke to be merged to form a closed contour, as the path length between the ends is relatively long.

4.2. Identifying Selected Components

NP2 uses polygon intersection techniques to identify which components in the problem are included in the student's boundary. To determine if a set of components is selected, the program merges the boundaries of the components to produce a "nominal" polygon. A larger ("maximum") and smaller ("minimum") polygon are constructed as uniform offsets of the nominal polygon. Figure 6 shows an example in which the set of components consists of a single rectangular-shaped part.

A set of components is considered to be selected if three conditions are satisfied: (a) the area of the student's boundary must differ from the area of the nominal polygon by no more than 10%, (b) the boundary must be contained entirely inside the maximum polygon, and (b) the minimum polygon must be contained entirely inside the boundary. These computations are performed using the polygon intersection technique in [Bri].

When interpreting the boundary, the program considers all combinations of components in the problem. Also, the program handles flexible components (e.g., cables and ropes) using special techniques: it is possible for the boundary to select only a portion of a flexible element.

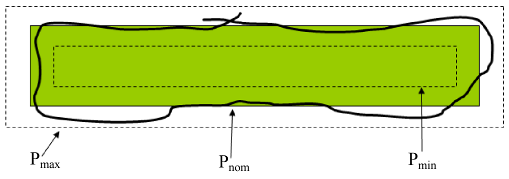


Figure 6: *For a set of components to be selected, the student's boundary must be contained entirely in the maximum polygon P_{max}, and must be outside the minimum polygon P_{min}.*

4.3. Force and Moment Arrow Recognition

After the FBD boundary has been dragged into the FBD workspace, the student draws arrows to represent forces and moments, with straight arrows for the former and curved ones for the latter. NP2 is capable of recognizing arrows drawn tail to head with either one or two strokes. The system accomplishes this by combining two different arrow recognizers: an *inverse-curvature* recognizer and a *dollar arrow* recognizer.

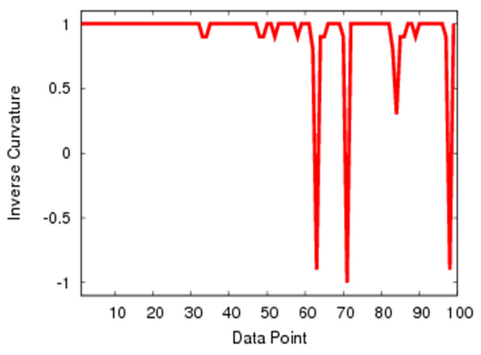


Figure 7: *Inverse curvature of an arrow. The three corners of the arrowhead are the three smallest values of inverse curvature. Image from [LdSP*08].*

The inverse-curvature recognizer identifies arrows by the three corners typically found on arrowhead. With this technique, the stroke is first resampled to 128 points to reduce computation time and then smoothed to remove noise. Next, line segments are constructed between each pair of consecutive points. The "inverse curvature" at a point is defined as the cosine of the angle between the two adjacent segments. If the adjacent segments are nearly collinear, the inverse curvature value is close to one. If the segments make a sharp angle, the inverse curvature value is much smaller than one. In the case of a 180° bend, for instance, the inverse curvature is negative one. The three corners of an arrowhead can be detected by their small inverse curvature values (Figure 7).

The inverse-curvature recognizer uses a neural network to classify pen strokes. The 128 inverse curvature values for the resampled pen stroke serve as the features to the classifier. The neural network was trained with approximately 100 arrows and 600 non-arrows (letters and numbers) from 5 different users. (The complete details can be found in [LdSP*07].)

The Dollar Arrow recognizer is a modified version of the Dollar recognizer [WWL07]. The Dollar recognizer has difficulty recognizing arrows, because the shafts may be arbitrarily long. We have modified the algorithm to ignore most of the shaft. To do this, the Dollar Arrow recognizer uses inverse curvature to identify the three corners of the arrowhead. The corners are used to compute the length of the line segments forming the arrowhead. If the shaft (i.e., the "non-arrowhead" portion of the arrow) is longer than twice this average length of the arrowhead segments, the shaft is clipped to this length, otherwise the entire shaft is used. After this preprocessing, recognition continues with the usual Dollar recognizer approach.

Both of these recognizers accept a stroke as input and returns a confidence value between $[0,1]$ indicating the likelihood that the stroke is an arrow. The confidence values of the two recognizers are combined by taking their average. If this average is greater than 0.8, the stroke is classified as an arrow. Both the inverse-curvature and Dollar Arrow recognizer are designed for single stroke arrows. To use these recognizers for multi-stroke arrows, the end of the first stroke is joined to the start of the second stroke. Arrows with more than two strokes can be processed in a similar fashion, but in this case, it is useful to consider all permutations of stroke order when appending strokes.

To determine if an arrow is a force or a moment, NP2 fits a linear least squares line to the shaft of the arrow. If the error of fit is less than a threshold, the arrow is considered to be a force, otherwise it is considered to be a moment.

4.4. Force Labels

Forces and moments can be labeled by drawing directly on the workspace, or by using a text entry window. In the former case, the characters are recognized with the Microsoft Handwriting Recognizer (MHR) [Pit07] after a user-adjustable timeout. The ink is then replaced with its interpretation, which is rendered with a hand-drawn character font. We use this approach, rather than using a traditional computer font, to preserve the informality of a student's handwritten work. The student can correct misinterpretations by erasing and rewriting the text, or by pressing the button on the barrel of the stylus and circling the ink, triggering the display of alternative interpretations from which the student may select the intended characters.

To initiate the use of the text entry window, the student simply draws an "L"-shaped pen stroke, causing the window to appear. The text entry window contains two character entry boxes, as force labels are limited to two characters. After each character is written, it is recognized using MHR and replaced with its interpretation. If the text is misinterpreted, the student can select the correct interpretation from the provided lists of alternatives. Once text entry is complete, the final interpretation is displayed in the workspace with a handwritten font.

5. Interpreting Equilibrium Equations

Equations consist of multiple characters written in-line. Once the student completes an equation, he clicks the “interpret” button to trigger recognition. Clicking the “critique” button produces appropriate tutorial feedback. This section describes the recognition process including stroke grouping, recognition, and error correction.

5.1. Stroke Grouping

The first step in interpreting an equation is grouping the pen strokes into characters. We do this using a technique based on the grouping technique in [PSDA10]. Our technique uses a C4.5 decision tree to determine if two strokes belong to the same character.

The classifier employs six pairwise features: five are spatial and one is temporal. The spatial features characterize the relative positions of the strokes. These features include: the minimum distance between the strokes, the maximum distance between them, the distances between their centroids, and their horizontal and vertical overlap (Figure 8). To achieve size-invariance, the spatial features are normalized by the height of the equation. The temporal feature is the difference in start times of the two strokes.

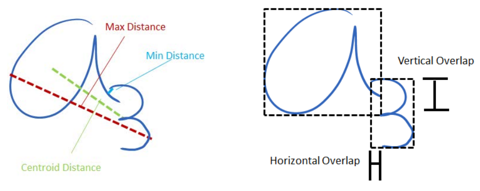
It is unlikely that a single character will be comprised of strokes that are far from each other. Thus, two strokes are considered for grouping only if the horizontal distance between them is less than half the equation height. This both speeds up the grouping process and increases accuracy.

To build the training data set, equations were originally collected from members of our research lab. Later NP2 was retrained using real world data taken from users of the tutoring system. This data included 221 equilibrium equations and 189 equation type samples. To prevent biasing the data towards any particular user, at most three equations from each user were used for training.

5.2. Character Recognition

Once the pen strokes have been grouped into characters, recognition begins. We use MHR for most characters, however, it has difficulty recognizing equal signs. Thus, we use a special-purpose recognizer for equal signs.

MHR uses a lexicon to achieve high recognition accuracy



(a) Distance Features (b) Overlap features

Figure 8: *Pairwise spatial features used by the grouping classifier.*

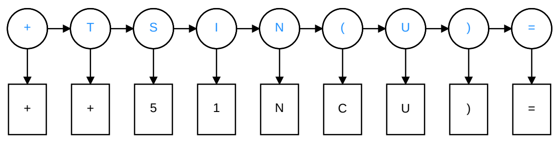


Figure 9: *The handwriting recognition HMM uses hidden states that correspond to the symbols a user intended (blue text) and observations that correspond to the recognized characters (black text).*

S_i	S_{i+1}
’ ’, **alpha_***	’ ’
’(’	’(’, ’-’, **digit**, ’C’, ’S’, ’T’, ’P’, ’N’, ’D’, ’U’
op	**digit**, ’C’, ’S’, ’T’, ’P’, ’N’, ’D’, ’U’
digit	’(’, ’)’, **op**, ’.’, **digit**, ’=’, ’C’, ’S’, ’T’, ’P’, ’N’, ’D’, ’U’
’=’	’0’
’C’	"CO"
"CO"	"COS"
"COS"	’(’, **digit**, ’C’, ’S’, ’T’, ’P’, ’N’, ’D’, ’U’
’D’	"D1", "D2"
’N’	’(’, ’)’, **op**, **digit**, ’=’, ’C’, ’S’, ’T’, ’P’, ’N’, ’D’, ’U’

Table 1: *Subset of the legal state transitions for an equation HMM for a free body diagram with labels D1, D2, and N.* ***op*** *= ’+’, ’-’, ’*’, and ’/’;* ***digit*** *= digits from ’0’ to ’9’;* ***alpha_**** *= all uppercase letters except ’C’, ’S’, ’T’, ’P’, ’N’, ’D’, and ’U’*

on text. This approach is not suitable for our application, as it would bias equation recognition towards correct solutions, hampering the system’s educational effectiveness. Instead, NP2 uses a grammar for valid mathematical expressions and knowledge of the common recognition errors to improve recognition accuracy. With this approach, for example, the program is able to identify that a set of characters recognized as “C05” was likely intended to be “COS”.

Our error correction technique utilizes the Hidden Markov Model (HMM) shown in Figure 9. A HMM has five components: a set of hidden states which are unknown, a set of known observations, the probability of transitioning from each state to every other state, the probability of seeing each observation in each hidden state, and the probability of beginning in a particular hidden state. (For an overview of HMMs see [Rab89].)

In our problem, the initial recognition results for the equation characters comprise the observations. The hidden states are the intended meanings of the characters. There are 45 states that represent single characters: 26 uppercase letters (A-Z), 10 digits (0-9), the 4 basic arithmetic operators (+, -, /, *), the equal sign, left and right parentheses, the decimal point, and the null state. To facilitate recognition of multi-character symbols, such as “FX” or “SIN”, the program employs compound states. For example, in the latter case, there are three hidden states representing “S”, “SI”, and “SIN”.

The transition probabilities are based on a grammar of legal mathematical expressions. For each state, we identify the set of possible next states that can occur in a legal mathematical expression. Transitions to any of these legal states is assumed to be equally likely, with the total probability of transitioning to these states summing to 90%. The remaining 10% of the probability is distributed equally to the remaining states.

The set of legal states for a given problem depends on the variables (e.g., force labels) used in that problem. Thus, the transition probabilities are dynamically generated for each problem. Also, one HMM is used for interpreting equations, and another is used for interpreting equation types. Tables 1 and 2 give examples of legal transitions.

The observation probability matrix, $B_j(k)$, encodes the likelihood of witnessing observation k with hidden state j, i.e., the likelihood of the MHR producing a particular recognition result for a given character. This is computed as:

$$B_j(k) = 0.6B_j(k_T) + 0.4B_j(k_E) \qquad (1)$$

$B_j(k_T)$ is the “theoretical observation probability”, which assigns 36% probability to the observation matching the hidden state, 18% distributed equally over all observation that are reasonably similar to the state (e.g., “S” observed in the state “5”), and 6% probability distributed equally over all remaining states. $B_j(k_E)$ is the “empirical observation probability” and is computed directly from the confusion matrix of the handwriting recognizer.

The Viterbi algorithm is used to determine the most likely sequence of hidden states corresponding to the recognizer’s initial results. This sequence of hidden states is then used as the final interpretation of the equation.

6. System Evaluation

NP2 was integrated into a sophomore-level Engineering Statics course at our university in the winter quarter of 2010, in which just over 100 students were enrolled. Students worked in pairs due to the limited number of tablet PCs available. Students used NP2 in their discussion sections, which lasted 50 minutes, four times during the quarter (tablet PCs running NP2 were provided). The first session taught students how to construct FBDs for problems involving the analysis of a single body, the second session focused on construction of both FBDs and equilibrium equations for single bodies, the third session addressed FBDs for multi-body frame and machine problems, and finally the fourth session included both FBDs and equilibrium equations for frame and machine problems.

S_i	S_{i+1}
’ ’, **alpha_efm**, **op**, **digit**, ’(’, ’)’, ’.’	’ ’
’=’	’0’
’E’	’M’, ’F’
’F’	"FX", "FY"
"FX"	’=’
"FY"	’=’
’M’	"MP", "MB", "MA"

Table 2: *Legal state transitions for an “equation type” HMM. **op** = ’+’, ’-’, ’*’, and ’/’; **digit** = digits from ’0’ to ’9’; **alpha_efm** = all uppercase letters except ’E’, ’F’, and ’M’*

6.1. Attitudinal Survey

At the end of the quarter, students were asked to complete a 25 question survey of their opinions about NP2. The survey questions and responses are listed in Table 3. The first 11 questions were related to the usability of the system, and considered ease of drawing, recognition accuracy, and overall usability. In one question, students were asked to compare the interface to a traditional WIMP interface. Their answer was based on their general familiarity with WIMP interfaces as the students were not provided with a WIMP-based Statics tutoring system. The next 10 questions focused on the usefulness of the system for learning various statics concepts and the overall instructional usefulness. The final four questions concerned the students’ overall reaction to the system.

All questions were answered on a scale from one to ten, with one being the most negative response and ten being the most positive. The median response on all questions was on the positive side of the scale for all but two of the questions. Students experienced problems with the system’s recognition and interpretation functionality, which likely led to some frustration. We observed that some of the frustration was due to the system’s conventions. For example, some students drew arrows head to tail rather than tail to head as the program expected. Similarly, some students wrote equations in unexpected ways, for example, writing the word “SUM” rather than the symbol “Σ” for the equation type.

Because this was the first deployment of NP2, students

	Survey Question	**Scale**	**Ave**	**Med**
1	How easy is it to trace and drag system boundaries?	Hard (1) — Easy (10)	6.8	7
2	How easy is it to draw force arrows?	Hard (1) — Easy (10)	6.2	6
3	How easy is it to label force names?	Hard (1) — Easy (10)	5.6	6
4	How easy is it to draw correctly-interpreted equation types?	Hard (1) — Easy (10)	6.4	7
5	How easy is it to write correctly-interpreted equations?	Hard (1) — Easy (10)	5.5	6
6	How easy is it to correct interpretation errors in equations?	Hard (1) — Easy (10)	5.9	6
7	Are the program's interpretation capabilities sufficiently accurate?	Insufficient (1) — Sufficient (10)	5.2	5
8	How easy is it to learn to use the program?	Hard (1) — Easy (10)	7.1	8
9	What is your overall opinion of the usability of this program?	Not usable (1) — Very usable (10)	6.1	6
10	How similar is this system to working with paper and pencil?	Very Dissimilar (1) — Very similar (10)	5.3	5
11	Is this interface preferable to WIMP interfaces?	Prefer traditional UI (1) — Prefer pen-based UI (10)	6.5	7
12	Is this program's tutorial feedback presented in a usable form?	Not usable (1) — Very usable (10)	6.8	7
13	How useful is this for learning selection of system boundaries?	Not useful (1) — Very useful (10)	6.9	7
14	How useful is this for identifying forces on free body diagrams?	Not useful (1) — Very useful (10)	7.1	7
15	How useful is this for application of Newton's 3rd Law?	Not useful (1) — Very useful (10)	6.8	7
16	How useful is this for analysis of two-force members?	Not useful (1) — Very useful (10)	7.0	7
17	How useful is this for force equations?	Not useful (1) — Very useful (10)	6.4	7
18	How useful is this for moment equations?	Not useful (1) — Very useful (10)	6.5	7
19	What is your overall opinion of the usefulness of this program in learning to solve equilibrium problems?	Not useful (1) — Very useful (10)	6.5	7
20	How likely would you be using this system to study for Statics courses?	Unlikely (1) — Likely (10)	6.1	7
21	How likely would you be using this kind of system for other subjects?	Unlikely (1) — Likely (10)	6.2	7
22	Rate your overall reaction to the system.	Terrible (1) — Wonderful (10)	6.2	6
23	Rate your overall reaction to the system.	Difficult (1) — Easy (10)	6.7	7
24	Rate your overall reaction to the system.	Frustrating (1) — Satisfying (10)	5.4	5
25	Rate your overall reaction to the system.	Dull (1) — Stimulating (10)	6.4	7

Table 3: *Survey questions and responses. Questions are answered on a scale from one to ten, with one being the most negative and ten being the most positive. The average and medium values are listed.*

also discovered some bugs in the recognition algorithms. These issues were quickly addressed resulting in releases of improved versions of the system throughout the quarter. We believe that the survey results were negatively affected by students' experiences with the earlier versions of the system. We expect that future deployments of the improved system will result in more favorable opinions.

7. Conclusion

We have presented Newton's Pen II, a pen-based tutoring system for Engineering Statics. The system scaffolds students in the construction of free body diagrams and equilibrium equations of one or more rigid bodies.

This pen-based system employs several new sketch understanding techniques including: a natural pen-based tutorial interface, a method for merging strokes, an extension of the Dollar recognizer for arrows, a stroke grouping algorithm designed for equations, and a dynamically-generated HMM-based error corrector for equations.

We evaluated the system in the context of an undergraduate Statics course with an enrollment of just over 100 students. In a formal survey, students responded with favorable opinions about the user interface design and the usefulness of the system for learning Statics. While there is clear room for improvement in the system, design we are encouraged by the students' generally favorable response to the system.

8. Acknowledgements

This material is based upon work supported by the National Science Foundation under Grant No. 0735695.

References

[AMS05] ANDERSON R., MCDOWELL L., SIMON B.: Use of classroom presenter in engineering courses. In *Proceedings of Frontiers in Education '05* (2005), pp. T2G–13–18. 2

[BGLSS03] BAREQUET G., GOODRICH M. T., LEVI-STEINER A., STEINER D.: Straight-skeleton based contour interpolation. In *SODA '03: Proceedings of the fourteenth annual ACM-SIAM symposium on Discrete algorithms* (Philadelphia, PA, USA, 2003), Society for Industrial and Applied Mathematics, pp. 119–127. 4

[Bri] BRIDENBECKER D.: General poly clipper algorithm in java. http://www.seisw.com/GPCJ/GPCJ.html. 4

[CL12] CHEEMA S., LAVIOLA J.: Physicsbook: A sketch-based interface for animating physics diagrams. In *Proceedings of* *the 2012 International Conference on Intelligent User Interfaces* (2012), pp. 51–60. 2

[dSBL*07] DE SILVA R., BISCHEL D. T., LEE W., PETERSON E. J., CALFEE R. C., STAHOVICH T. F.: Kirchhoff's pen: a pen-based circuit analysis tutor. In *SBIM '07: Proceedings of the 4th Eurographics workshop on Sketch-based interfaces and modeling* (New York, NY, USA, 2007), ACM, pp. 75–82. 2

[FPJ02] FONSECA M. J., PIMENTEL C., JORGE J. A.: CALI-an online scribble recognizer for calligraphic interfaces. In *AAAI Spring Symposium on Sketch Understanding* (2002), pp. 51–58. 2

[GKSS05] GENNARI L., KARA L. B., STAHOVICH T. F., SHIMADA K.: Combining geometry and domain knowledge to interpret hand-drawn diagrams. *Computers & Graphics 29*, 4 (2005), 547–562. 2

[HN05] HSE H. H., NEWTON A. R.: Recognition and beautification of multi-stroke symbols in digital ink. *Computers & Graphics 29*, 4 (2005), 533–546. 2

[KFH11] KEBODEAUX K., FIELD M., HAMMOND T.: Defining precise measurements with sketched annotations. In *Proceedings of the Eighth Eurographics Symposium on Sketch-Based Interfaces and Modeling* (New York, NY, USA, 2011), SBIM '11, ACM, pp. 79–86. 2

[KJ12] KANG B., J. L.: Logicpad: A pen-based application for visualization and verification of boolean algebra. In *Proceedings of the 2012 International Conference on Intelligent User Interfaces* (2012), pp. 265–268. 2

[KS04] KARA L. B., STAHOVICH T. F.: An image-based trainable symbol recognizer for sketch-based interfaces. In *AAAI 2004 Fall Symposium: Making Pen-Based Interaction Intelligent and Natural* (2004). 2

[LdSP*07] LEE W., DE SILVA R., PETERSON E. J., CALFEE R. C., STAHOVICH T. F.: Newton's pen - a pen-based tutoring system for statics. In *2007 Eurographics Workshop on Sketch-Based Interfaces and Modeling* (August 2007). 2, 5

[LdSP*08] LEE W., DE SILVA R., PETERSON E. J., CALFEE R. C., STAHOVICH T. F.: Newton's pen: A pen-based tutoring system for statics. *Computers & Graphics 32*, 5 (2008), 511 – 524. 2, 5

[LKS06] LEE W., KARA L. B., STAHOVICH T. F.: An efficient graph-based symbol recognizer. In *2006 Eurographics Workshop on Sketch-Based Interfaces and Modeling* (2006), pp. 11–18. 2

[LM01] LANDAY J. A., MYERS B. A.: Sketching interfaces: Toward more human interface design. *IEEE Computer 34*, 3 (2001). 2

[LS11] LEE C.-K., STAHOVICH T. F.: A pen based tutoring system. In *ASEE Annual Conference and Exposition* (2011). 1, 3

[MLYS05] MORSE B. S., LIU W., YOO T. S., SUBRAMANIAN K.: Active contours using a constraint-based implicit representation. In *SIGGRAPH '05: ACM SIGGRAPH 2005 Courses* (New York, NY, USA, 2005), ACM, p. 252. 4

[OAC06] OVIATT S., ARTHUR A., COHEN J.: Quiet interfaces that help students think. In *Proceedings of the 19th annual ACM symposium on User interface software and technology (UIST '06)* (2006), pp. 191–200. 1

[Pit07] PITTMAN J. A.: Handwriting recognition: Tablet pc text input. *Computer 40* (2007), 49–54. 5

[PSDA10] PETERSON E. J., STAHOVICH T. F., DOI E., ALVARADO C.: Grouping strokes into shapes in hand-drawn diagrams. In *AAAI* (2010), Fox M., Poole D., (Eds.), AAAI Press. 2, 6

[Rab89] RABINER L. R.: A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE 77*, 2 (1989), 257–286. 7

[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 *Proceedings of American Society for Engineering Education Annual Conference & Exposition* (2003). 2

[Rub91] RUBINE D.: Specifying gestures by example. *Computer Graphics 25* (1991), 329–337. 2

[SD05] SEZGIN T. M., DAVIS R.: HMM-based efficient sketch recognition. In *International Conference on Intelligent User Interfaces (IUI'05)* . (2005). 2

[SH04] SUEBNUKARN S., HADDAWY P.: A collaborative intelligent tutoring system for medical problem-based learning. In *Proceedings of the 9th International Conference on Intelligent User Interfaces (IUI '04)* (2004), pp. 14–21. 2

[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 Artificial Intelligence in Education 15*, 3 (2005). 2

[WWL07] WOBBROCK J. O., WILSON A. D., LI Y.: Gestures without libraries, toolkits or training: a $1 recognizer for user interface prototypes. In *UIST '07: Proceedings of the 20th annual ACM symposium on User interface software and technology* (New York, NY, USA, 2007), ACM, pp. 159–168. 2, 5