Mechanix: A Sketch-Based Tutoring and Grading System for Free-Body Diagrams

Stephanie Valentine, Francisco Vides, George Lucchese, David Turner, Hong-hoe Kim, Wenzhe Li, Julie Linsey, Tracy Hammond

■ *Introductory engineering courses within large universities often have annual enrollments that can reach up to a thousand students. In this article, we introduce Mechanix, a sketch-based deployed tutoring system for engineering students enrolled in statics courses. Our system not only allows students to enter planar truss and free-body diagrams into the system, just as they would with pencil and paper, but our system also checks the student's work against a hand-drawn answer entered by the instructor, and then returns immediate and detailed feedback to the student. Students are allowed to correct any errors in their work and resubmit until the entire content is correct and thus all of the objectives are learned. Since Mechanix facilitates the grading and feedback processes, instructors are now able to assign more free-response questions, increasing teacher's knowledge of student comprehension. Furthermore, the iterative correction process allows students to learn during a test, rather than simply display memorized information.*

Mechanical and civil engineering students learn fundamental engineering concepts through a course in statics. Statics problems usually require the student to draw planar truss and other free-body diagrams (FBDs).

In civil and mechanical engineering, visual and spatial thoughts are essential for the correct absorption of fundamental concepts. By relying on visual aids as opposed to only using the verbal channel of acquiring knowledge, the cognitive load becomes lighter and learning becomes more effective (Sweller 1994). Furthermore, learners should actively engage in the process in order to reach the highest levels of comprehension. The task of freehand sketching encourages and demands that learners are actively constructing their knowledge. In the civil and mechanical engineering domain, sketched planar truss and other free-body diagrams are particularly helpful to the understanding of statics concepts. This type of "forced active processing" ensures attention to visual information and helps learners attend to key elements of the visual system (Kozma 1994).

A planar truss diagram is simply a two-dimensional representation of a structure that is constructed from physical beams and joints. Joints, also referred to as nodes, are located at the intersection of two or more beams and are the location where external forces may act upon the object.

Furthermore, these external forces create member forces within each individual beam by tension or compression of the beam. Trusses are used as supports in many structures such as bridges, houses, and other buildings. An excellent foundation of how to construct a truss is critical for a student's success as an engineer in the future. While a truss is a type of free-body diagram, other nontruss free-body diagrams can be used to analyze all of the internal and external forces acting on a general object of any shape.

In the task of learning how to construct a correctly structured truss the learner must receive appropriate feedback on what he or she is doing in order to correctly assess the effectiveness of the learning process. Feedback helps learners identify misconceptions and guides the learner to a more accurate conception of the knowledge (Bangert-Drowns et al. 1991). It is fundamental that this feedback is both concordant with the student's learning stage and also that it is given in a correctly timed manner. If feedback is delayed for too long, the overall learning process becomes degraded, which gives us a preference for immediate feedback.

While immediate feedback is ideal, the large class sizes of introductory engineering courses (excess of 100 students per instructor) prevent detailed feedback on hand-drawn solutions from being given often because of the time commitment involved. In these courses, students are likely to be assigned only one or two hand-drawn assignments a semester. To combat these time constraints in testing environments, multiple choice questions are the primary evaluation method.

To stimulate the educational value of these courses, the need for a better method of grading hand-drawn truss diagrams is necessary. Hand-drawn homework problems, such as truss diagrams, afford themselves the use of sketch recognition as a solution.

Here, we introduce Mechanix, a sketch-recognition system that can recognize, correct, and provide real-time feedback on a student's hand-drawn truss diagram that is checked against a single instructor-entered key sketch. The aim of our deployed system is to advance the artificial intelligence of automated mechanical and civil engineering instruction, such that the automated instruction emulates the expert performance achieved by human instructors.

Prior Work

Sketch Worksheets (Yin et al. 2010) is a sketch-based system that allows instructors to create non-domain-specific sketched worksheets for students to complete. The system generates "facts" based on a spatial analysis of the sketch. Instructors choose which facts are important and define what advice should be given to the student. In order to do this, instructors must understand a somewhat complicated language based on spatial relationships. Both Sketch Worksheets and Mechanix allow instructor-sketched solutions and provide incremental feedback, but Mechanix requires much less of a learning curve. Additionally, because Mechanix is a specialized system, it understands the diagrams it recognizes and can even solve students' trusses without an instructor-provided sketch.

Some educational software packages specialized to the statics domain already act as tutors for students that are learning statics. These packages include WinTruss (Callahan et al. 1988), McGraw Hill Connect Engineering, and Bridge Architect, which are respectively stand-alone, web-based, and mobile phone engineering tutoring applications. Similarly to our deployed system they provide step-by-step instructions and provide some form of feedback on the input. However, none of them offer an opportunity for students to solve the complete problem by themselves; they all provide a partially completed solution and give overall feedback as to whether the student solution is correct or not. Additionally, none of these allow for hand-drawn input.

Two other related systems are the Andes physics tutoring system (Vanlehn et al. 2005) and the Free Body Diagram Assistant (Roselli et al. 2003), which allow students to create electronic solutions for homework assignments. Both systems were designed as alternatives to pen-and-paper homework assignments to make classroom adoption easy for professors. Additionally, both consist of a design palette where users can pick pieces and use a mouse to drag them on the workspace in order to build their solution. While this is an improvement to providing partially completed solutions to the student (as in the methods from the previous paragraph), the deployed system described in this article further improves on this by allowing students to use a stylus to hand-sketch their input. We provide a unique opportunity to use a more physical and traditional method of solving the problems, while assessing the correctness of the free-body diagram. In this way, students can acquire important skills that might prove to be valuable in their future careers even when they are away from a computer.

Some other systems also tackle the truss and free-body diagram teaching problem, allowing for freehand sketching as input. Newton's Pen (Lee et al. 2008) is a pen-based tutoring system for statics that runs on the FLY pentop computer (based on the Anoto digital pen-and-paper technology). The

recognition capability of Newton's Pen is limited by the hardware in the FLY pentop computer. Additionally, in order to facilitate recognition, Newton's Pen constrains the user to draw free-body diagram components in a very specific order. Unlike Newton's Pen, Mechanix offers truly free sketching in that the recognition is not dependent on the order in which the student draws the components of the solution.

To achieve this goal we rely on state-of-the-art techniques of sketch recognition. The most prominent research in this field can be categorized into three categories: gesture recognition (Rubine 1991; Wobbrock, Wilson, and Li 2007), vision-based recognition (Kara and Stahovich 2005; Miller, Matsakis, and Viola 2000), and geometric recognition (Hammond and Davis 2005). Geometric recognition has been explored and researched in various distinct domains. In this kind of recognition there is usually a bottom-up approach, and after preprocessing, there is a low-level recognizer that can identify primitive shapes such as lines, circles, or arcs. On top of primitive recognition there is a high-level recognizer that can use a set of constraints to determine whether the basic shapes and the relationships between them compose a more complex shape. This approach has been used successfully in domains such as military course of action diagrams (Johnston and Hammond 2010) and circuit diagrams (Alvarado and Davis 2004, Gennari et al. 2005). In all cases, a combination of primitive shapes under a set of known constraints results in the production of higher-level shapes that comply with certain standards, yet allow free sketching.

In our case, we rely on a powerful low-level recognizer called PaleoSketch (Paulson and Hammond 2008), or Paleo, which identifies primitive shapes such as lines, arcs, ellipses, or spirals. Paleo integrates several techniques such as corner finding and geometric perception to perform a series of prerecognitions over the supported shapes. It then uses a novel ranking algorithm to determine which of these shapes has a better fit. Although the current version of Paleo supports more than 10 basic shapes, we mostly rely on the recognition of lines, polylines, and dots for Mechanix. Paleo has a reported accuracy of more than 98 percent (Paulson and Hammond 2008).

Implementation

The Mechanix system provides a clean and intuitive interface in which students and instructors can interact. Figure 1 shows the Mechanix interface.

The text at the top center of the interface provides the description of the problem to be solved. The problem can be accompanied by an image that allows clicking to zoom in for more detail. The tool panel lies just below the problem description and contains useful functions to help edit the sketches. The buttons (from left to right) are undo, redo, clear, open, save, and erase. The checklist on the left edge of the interface provides step-by-step guides to assist students to finish the problem. A pullout notepad exists on the tab behind the checklist. Students can use this panel to make notes when they are working on any problem. The drawing panel is the center of the interface on which students draw their diagrams. Each pen-down motion is captured and processed by our software. Students use the equation panel (located below the drawing panel) to enter the required equations and values of reaction forces. Then the system compares these inputs with correct answers.

The system gives feedback by showing a helpful and informative message in the orange bar below the tool panel. In figure 1, the message shows that we have forgotten to draw an axis, which is necessary to process the force directions. These messages are an intuitive guide for students. The submit button is the green checkmark in the top right corner of the interface. Whenever students want to check their solutions, they can simply click the submit button to see whether their answers are correct.

Interface Modes

There are two distinct modes of interaction in Mechanix, the student mode and the instructor mode. Both modes provide a workspace to draw truss diagrams and enter metadata relevant to the problem. The student mode is the interface that a student sees when working on a homework assignment and was described in detail above.

The instructor mode allows instructors to create new assignments and corresponding sets of questions in a simple fashion. Instructors can add the problem statement in the form of text and explanatory images, and by using a similar interface to the one provided for students, they can sketch a solution diagram to each problem. This sketched solution is interpreted by the system and will be used to check the student-drawn sketches. Instructors are responsible for labeling nodes and forces. Mechanix then generates the appropriate system of equations and values for reaction forces. Certain additional required values can be input by the instructor, such as the factor of safety. After the instructor has finished creating a problem he or she can save it to the central server so students can access it. This same interface is also used to review detailed information about each student submission.

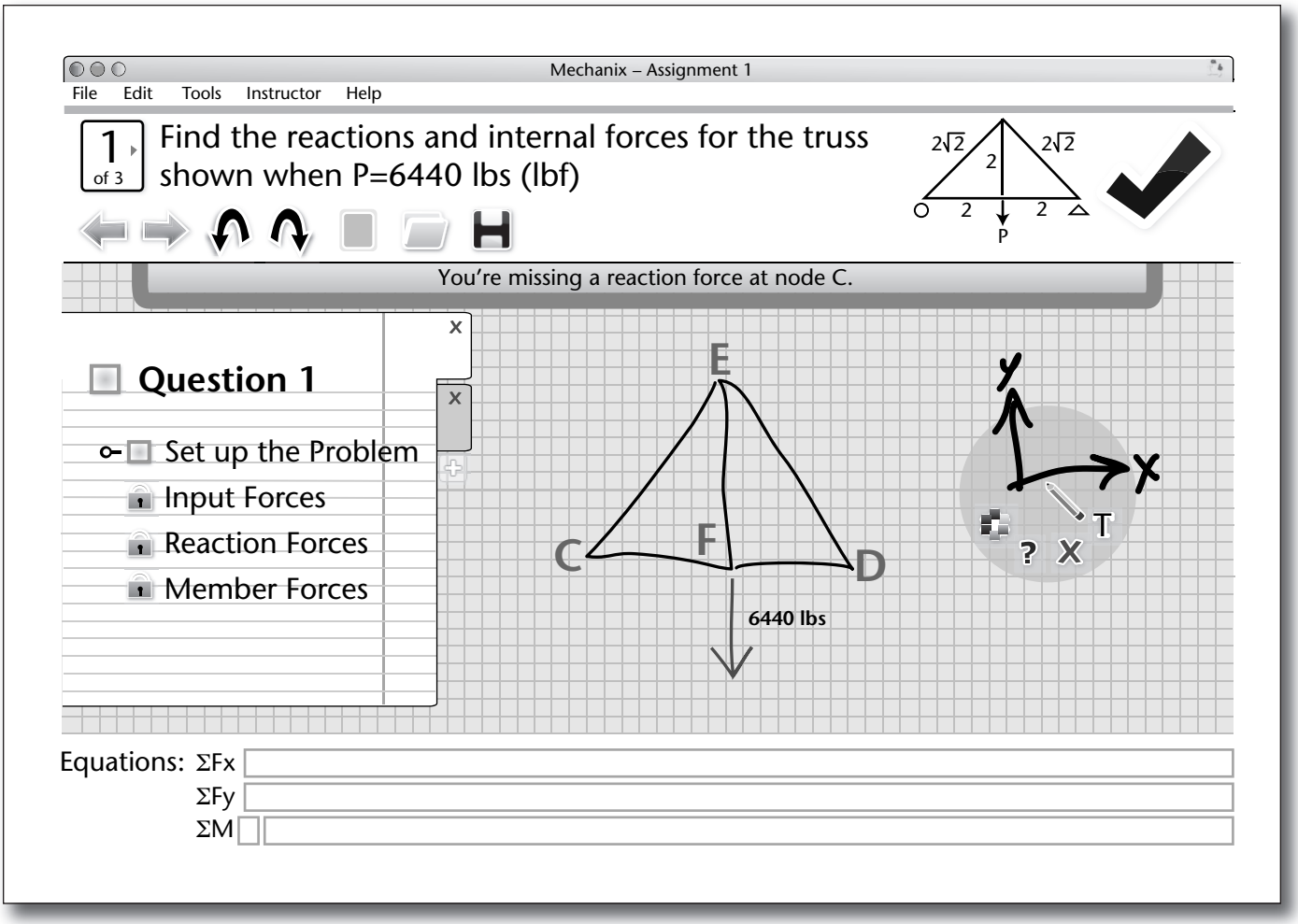

Figure 1. Mechanix Interface.

Interaction Methods

One great advantage of a sketch-based system is that it allows users to continually modify or edit their drawings as they would on pen and paper. The current Mechanix system provides such functionality through our round menu, buttons, and freehand erasure.

When we hover the mouse or pen on top of a stroke or recognized shape for a small time interval, the round menu will automatically appear. In figure 1, the round menu is shown providing options for an axis. Using this round menu, we can change the color of a shape, delete the highlighted shape, or label the shape.

There are several buttons on the tool panel, as we have shown in figure 1. Users can use these buttons to explicitly undo or redo or clear or erase their drawings.

We allow erasure by means of scribbling strokes, which can be faster and more natural for interaction than explicitly using buttons or menus. Keeping this purpose in mind, we integrated scribble gesture into the Mechanix system. Figure 2 shows how we can use the scribble gesture to remove either a complete shape or part of a shape. We recognize scribble shapes as combinations of strokes in which time intervals are within 400 milliseconds. If a scribble stroke intersects most of a shape, the scribble erases the entire shape. On the other hand, if the stroke intersects only one line of the shape, then the scribble deletes that single line. In the case of amorphous closed shapes, if the scribble is localized on the stroke, it deletes only that part.

We also allow certain interactions defined by the tapping of the pen or clicking of the mouse. Users can move the drawn shapes by clicking and holding until the cursor changes to the move cursor, then the user can drag the shapes around to the desired position. Items such as arrows or nodes can be labeled by the student either using the round menu as described above or by double clicking on the

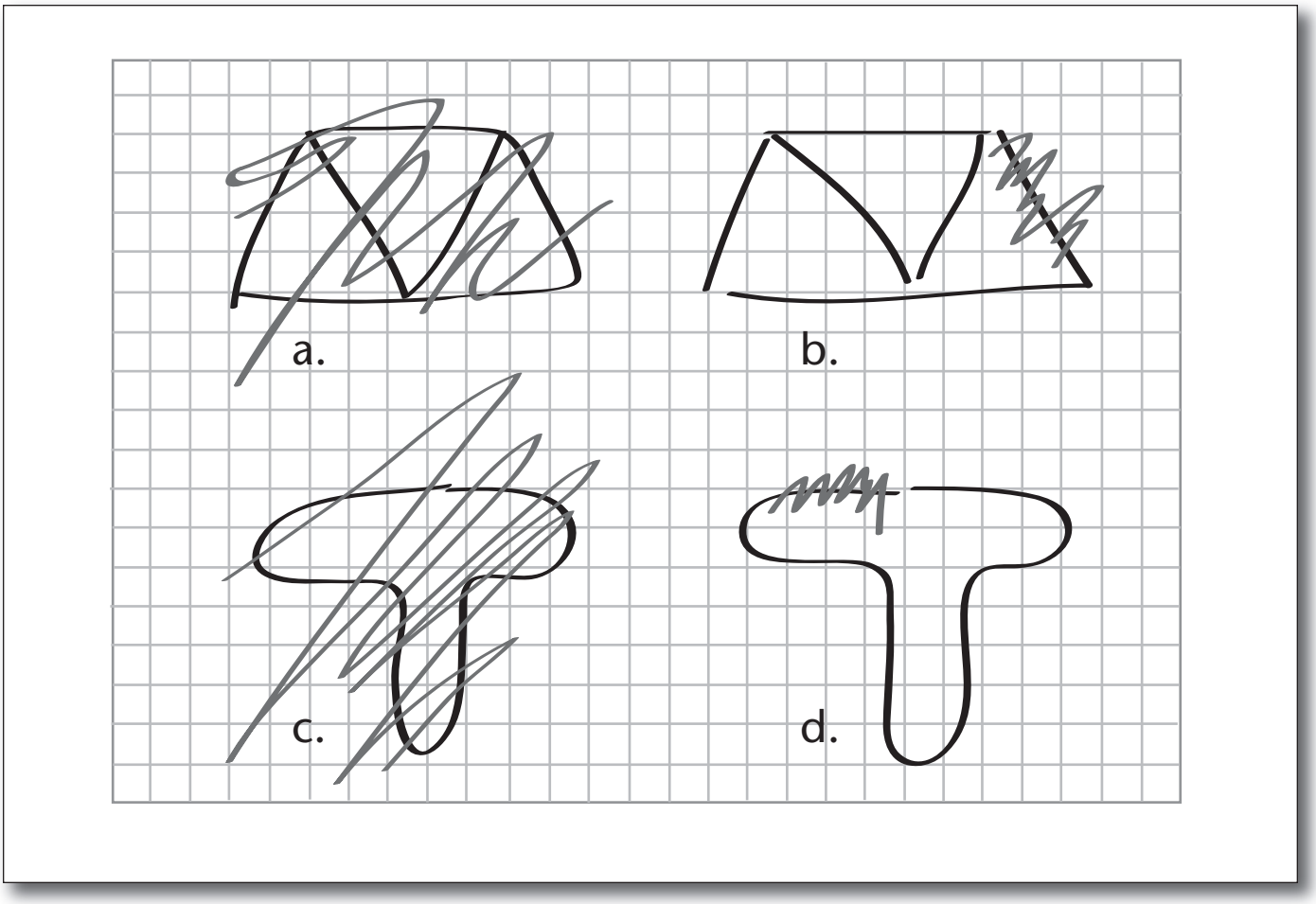

Figure 2. Examples of the Capabilities of the Scribble Erasure Feature.

shape and entering the text.

Visual Feedback

Primitive recognition is used not only as the basis for higher-level recognition but also for coloring drawn shapes as a feedback method. For example, a shape recognized as a force (an arrow) is colored lime green to indicate to the student that Mechanix recognized the force and that they may enter relevant metadata for that force, such as the magnitude.

Geometric Recognition

The hierarchical building blocks of our recognition are points, strokes, and shapes. The program generates a point for each movement of the mouse. It records the *x* and *y* coordinates and the current time. Strokes contain the group of points collected in the time between when the pen touches down on the tablet and when it loses contact with it. For example, a *y* character written in cursive will be one stroke, but a printed *y* will likely be two strokes. Primitive shapes contain at most a single stroke. Strokes are segmented using a cusp detector (Wolin 2010) and the primitive shapes are recognized by PaleoSketch (Paulson and Hammond 2008). Examples of primitive shapes are line segments, circles, arcs, curves, polylines (several line segments drawn in a single stroke), triangles, and others. Complex shapes, such as roller supports (described in figure 3), are built first of primitives and composed hierarchically to allow for more and more complex shapes. Mechanix creates complex shapes only after the member shapes pass our geometric-constraint-based recognizers.

Steps to Recognition

Our geometric shape recognition happens in six simple steps. First, we record points as the pen traces on the screen and add points to a new stroke. Second, Mechanix sends each new stroke to PaleoSketch to find its primitive shapes (line, circle, arc, polyline, and others.). Third, we add each new shape to the collection of all shapes. Fourth, we send various groupings of shapes to complex shape recognizers. Then, we apply a system of geometric constraints to recognize high-level shapes. Finally, we replace low-level shapes with new high-

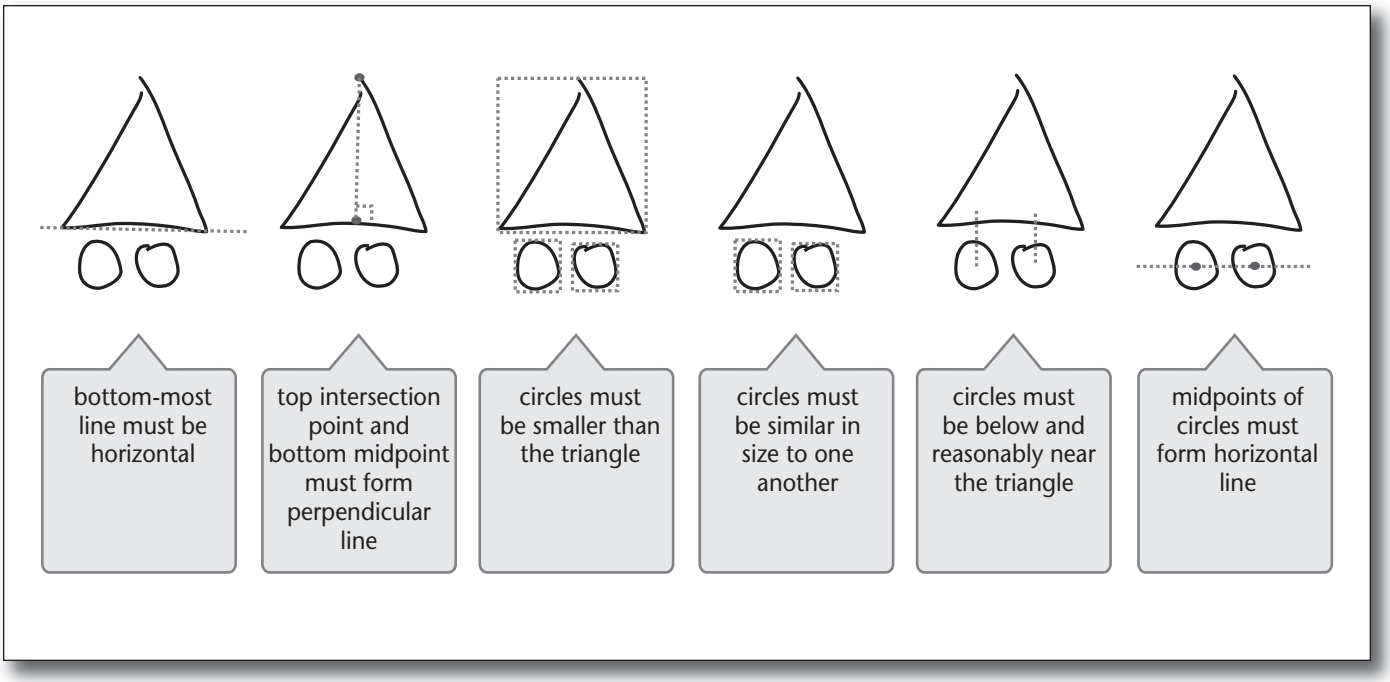

Figure 3. Example Geometric Constraints for a Roller Support.

level shapes, return to step 3, and cycle until no more complex shapes can be found. As an example of a system of geometric constraints, the recognizer for the roller support given in figure 3 requires a triangle and two circles as components. The roller support recognizer checks each of the conditions given in figure 3.

If the component shapes meet all of the constraints for a specific configuration, the recognizer combines them into a new complex shape. Upon positive recognition, the recognizer assigns the new shape a label that acts as a name tag to other shapes. The recognizers test new groupings made with that new shape to ensure the most complete and thorough recognition possible before the program returns to the first step (gathering points and making strokes). All recognition executes in real time.

Truss Recognition

Trusses are one of the main structures or symbols for the Mechanix system to recognize. Trusses are basic structures used in applications such as bridges, airplane wings, and buildings. Truss diagrams allow students to learn the way forces react on beams. Rather than attempt to define each valid truss individually, we use a general definition to identify any trusses the instructors or students may want to draw. We collected 589 data sets and achieved an accuracy of 94.6 percent.

We define a truss as a collection of convex polygons, each of which shares at least one side with another polygon in the truss. If we can find two polygons that share an edge, then we have found a truss.

Figure 4 shows two examples of shared edges. Once we have built the connected graph, we consider each edge AB as potentially a shared edge. To find out if the edge AB is a shared edge, we remove the edge from the graph. Then the system tries to find another path to go from A to B using a breadth-first algorithm. If we can find another path, then the edge AB is recognized as a shared edge. As shown in figure 4, the breadth-first search (BFS) algorithm will first find the blue path and remove all of its edges from the graph. At that point, the red path tries to find its way from A to B. In figure 4 (1), the red path can reach from A to B, and our system identifies the edge AB as a shared edge. However, in figure 4 (2), the red line cannot reach from A to B, so our algorithm will not detect the edge AB as a shared edge. The algorithm can be found in Field et al. (2011).

Students receive subtle feedback that leads them to draw correctly recognized trusses or bodies. For example, when any shape is recognized, hovering over any part of that shape will highlight all of the component strokes in unison. If a shape is not recognized, the strokes will highlight individually. Additionally, when the truss is recognized correctly in student mode, the nodes that the instructor defined will appear. With little to no training, students know that in the rare case that the truss or body was misrecognized and the nodes do not appear, they need to redraw the truss.

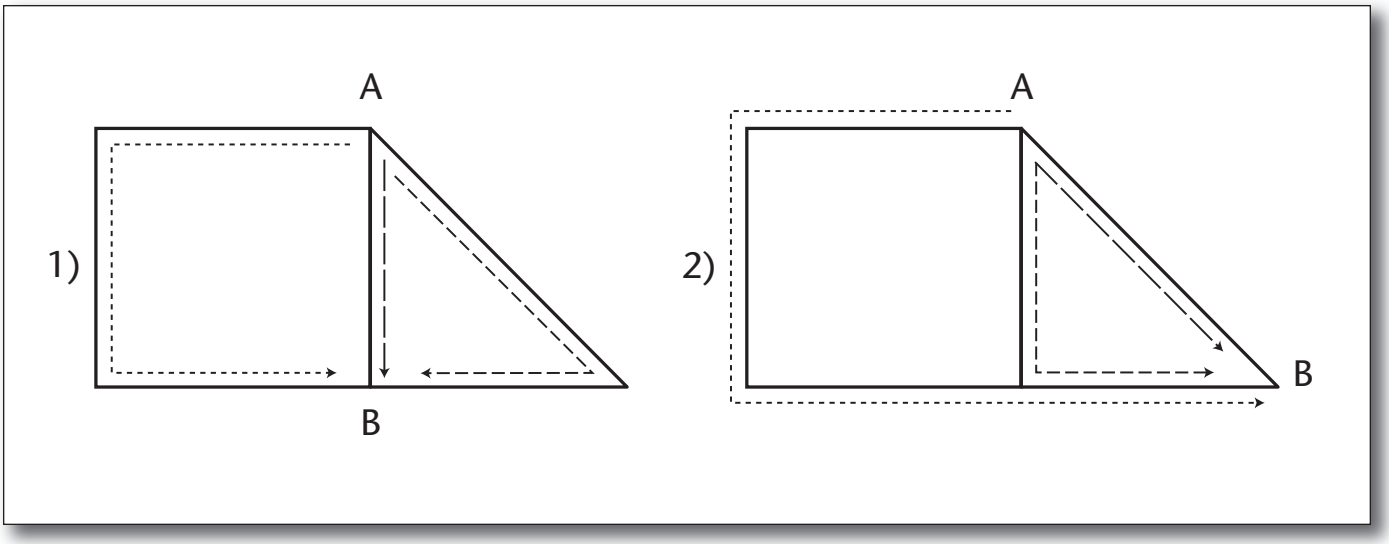

Figure 4. Examples of Shared Edges in Trusses.

Answer Checking

Mechanix automatically checks the students' assignments as they are submitted. If students submit incorrect answers, Mechanix provides beneficial feedback to help students reach the correct answers. Mechanix initializes the answer checker whenever a student clicks the green check button in the upper right corner of the interface. In order to grade the student's submission, Mechanix compares the student's sketch with the instructor's sketch. Mechanix checks the assignments by first ensuring that the student's truss has the correct configuration. Then, it checks that the student's sketch contains an axis. Then it ensures all forces are present and in the correct direction. Finally, Mechanix checks that the student's force values are equivalent to the instructor's force values.

To check whether the student's truss is similar to the instructor's hand-drawn truss, we first create a graph data structure composed of connected nodes and beams for both trusses. Students have different styles of drawing trusses. For example, when an assignment requires a truss such as the ones given in figure 1, some students draw a big triangle first and then draw a line down the middle, while other students draw the two triangles individually. To make the number of beams equal between all cases, we implemented a beam intersect mechanism. If a beam intersects another, both beams are segmented at the intersection point and a node is formed between them. After all intersections (nodes) have been found, we use basic graph isomorphism to determine whether the graphs are similar.

After the answer checker determines that the truss is correct, the system checks to see if the sketch includes an axis. Axes are necessary to specify what is considered positive.

To check the force values, we first check the type of force, which can be either a reaction force or an input force. An input force means that it has a value or constant for its label, and a reaction force has a label that starts with *F* or *R* and its direction *X* or *Y* at the end of the label (for example "Fay"). After Mechanix checks the types of forces, the system checks whether the forces are attached to appropriate nodes and have been given the correct values. Finally, if the submitted sketch has any missing or incorrect answers, Mechanix gives beneficial feedback such as "You are missing an input force at Node B" or "You have not drawn an axis".

Other Problem Types

Nontruss free-body diagrams are a second type of problem in the statics domain. These free-body diagrams depict the forces acting on an arbitrary physical body, such as a table or an escalator stair. The body given in the problem statement could be any bubblelike shape, such as those seen in figure 5. Because we as programmers cannot predict and write geometric recognizers for all of the possible bodies, a generic closed-shape comparison technique is necessary. An instructor simply sketches the desired closed-shape body in the answer key. All student closed shapes are compared with the instructor's and a binary similarity decision is made.

First, we recognize the shape bodies. We take a collection of primitive line shapes drawn in one or more strokes and attempt to traverse them and return to the first point of the first line. Two lines

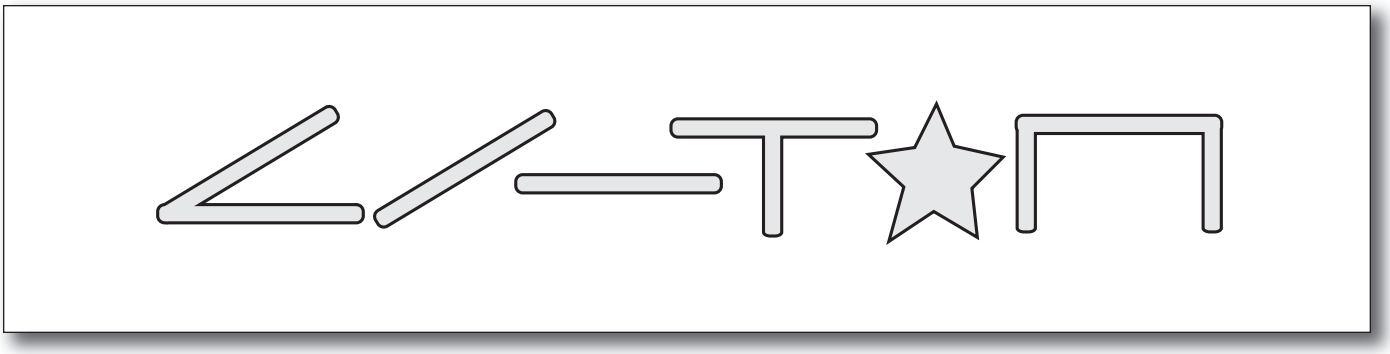

Figure 5. Example Closed-Shape Bodies.

are traversible if their end points are within 9 percent of the total path length of all the strokes combined. This gap allowance allows for bodies to be made of multiple strokes, without excessive care to perfectly line up end points.

To begin comparing the two shapes (the student's shape and the instructor's shape), we use the method defined by Wobbrock, Wilson, and Li (2007). We resample both shapes to contain 64 evenly spaced points. We then scale the shapes down to a 40 by 40 window so we can measure distances accurately.

We use three of the similarity metrics proposed by Kara and Stahovich (2005), the Hausdorff distance, a modified Hausdorff distance, and the Tanimoto coefficient. To find the Hausdorff distances between the two bodies A and B, we initialize two distance vectors DA and DB , each of size 64 (the number of points in each shape) such that each entry represents the closest distance from each point to the closest point in the other shape. The maximum value in both DA and DB is the Hausdorff distance. The modified Hausdorff distance is the average of the DA and DB values combined. Because the Hausdorff distance (the maximum minimum distance between two points in A and B) and the modified Hausdorff distance (the average minimum distance between two points in A and B) are distance measures, we convert them to a similarity measure by dividing the distance by 20 and subtracting that value from 1.

We chose the value 20 because it represents half of the width of the bounding window. Any two shapes that contain points whose nearest neighbor points are more than half the width of the bounding window apart cannot be deemed similar. The final measure used to determine body similarity is the Tanimoto coefficent. This is simply the ratio of points that "overlap" (number of points that have distance values in DA and DB less than or equal to 4.0) to the total number of points in DA and DB.

Finally, we take the three measures (Hausdorff distance similarity measure, modified Hausdorff similarity measure, and Tanimoto coefficient) and average their values (Kara and Stahovich 2005). If the resulting average similarity measure is greater than 0.65, we consider the student's and the instructor's shapes to match. A full description of the algorithm can be found in Field et al. (2011).

As an indication to the student that a match has been found, the nodes from the instructor's sketch are automatically revealed to the student. (Instructors add nodes by drawing small dots and labeling them with the desired letter in instructor mode.)

Creative Response

A creative response problem is a problem that is open ended for the student to solve. Instead of the normal problem where the student has to draw a truss to match the teacher's truss, the student has to draw a truss to meet a certain set of design constraints. A typical problem is:

> A village needs a bridge to connect it to the city marketplace across a chasm. The bridge should span between 7 and 8 feet as measured from the end supports and should be able to carry a load of 56 pounds applied to the center top of the span. The maximum load for each member is 70 Newtons.

This allows the students to design any bridge they desire to accomplish the task at hand. There is no example truss that is used for comparison. Instead, the Mechanix system uses artificial intelligence to check the student-created truss (figure 6).

It creates a linear system of equations with three parts. The first part of the system is the summation of all forces along the *x*-axis in figure 6 (Rax and 10 lbs). It uses the arrow recognizer and compares the slope with the axis to determine the direction, which gives the equation

$$\sum F_x = R_{ax} + 10$$

After adding all the forces that reside along the *x*-axis, it does the same process along the *y*-axis, which ends up with

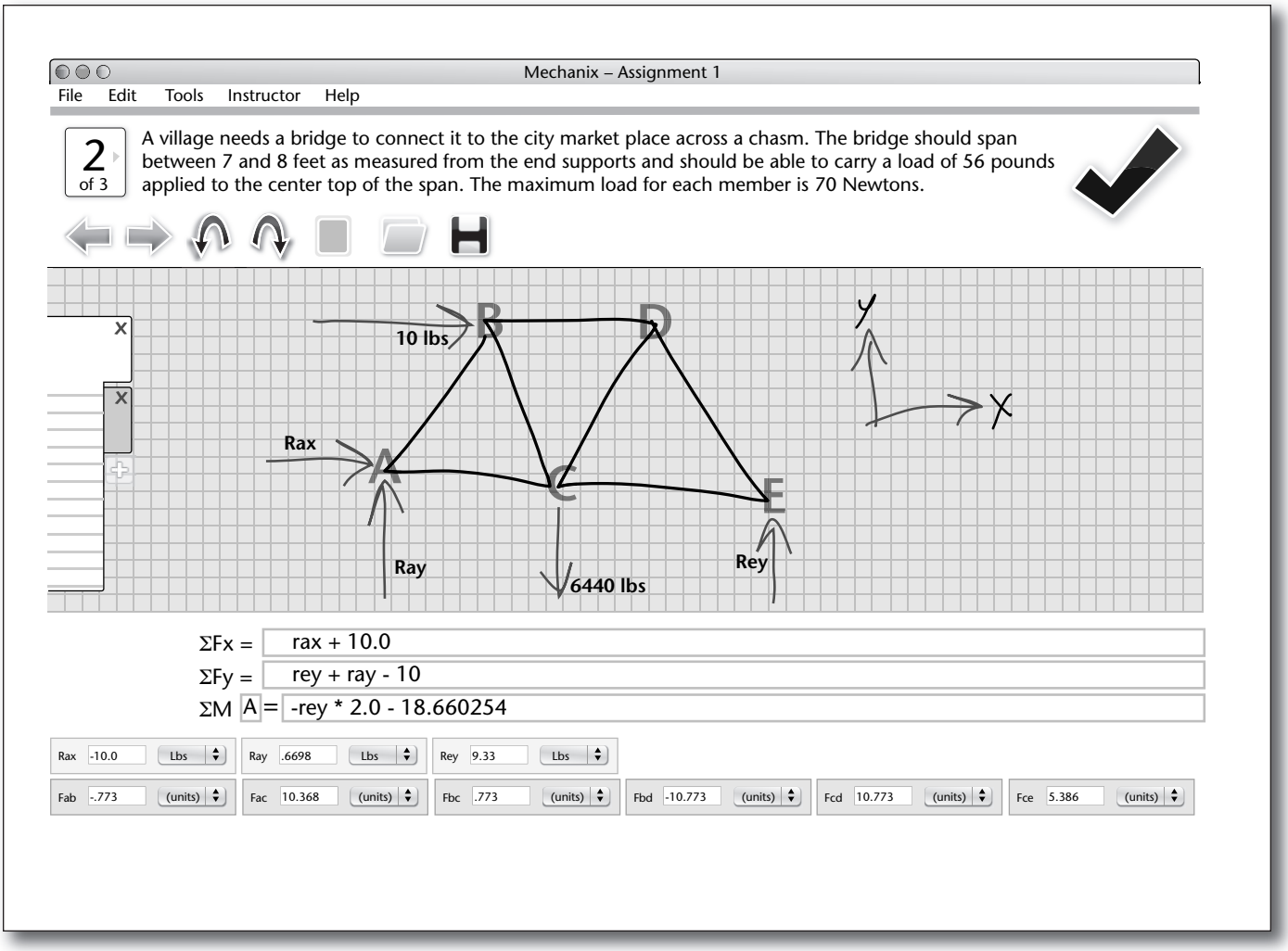

Figure 6. A Creative Response Truss.

$$\sum F_y = R_{ey} + R_{ay} - 10$$

The forces in the same direction have the same sign. For the final equation, Mechanix chooses a node with the most reaction forces around it. Then it does a summation of the moments iterating through all of the other forces computing the cross product between the forces and the distance to the node. In figure 6, node A is chosen to compute the cross product and results in the following equation:

$$\sum M = R_{ey} \times 2 - 10 \times 1 - 10 \times 1 \sin(\frac{\pi}{3})$$
$$= \sum M = R_{ey} \times 2 - 18.660$$

After finding all the external forces, a system of equations is formed using the method of joints for each beam in the truss. An example of this is with node A:

$$\sum A_x = 1F_{ac} + 0.5F_{ab} - 10.0 = 0$$

$$\sum A_y = F_{ab} \sin(\frac{\pi}{3}) + R_{ay} = 0$$

After creating all values from the student-drawn truss, Mechanix will use the values to compare to a set of constraints that the instructor previously entered, such as the length of the bridge or maximum force. The student will receive helpful feedback on whether the truss he or she drew adhered to the constraints.

Distributed Architecture

Students begin assignments by downloading and starting a client application. The client communicates to the server, which stores not only the student's final solution but also a log of mistakes that the student made while solving the problem. These logs may be used for assigning grades or, more importantly, determining problem areas where stu-

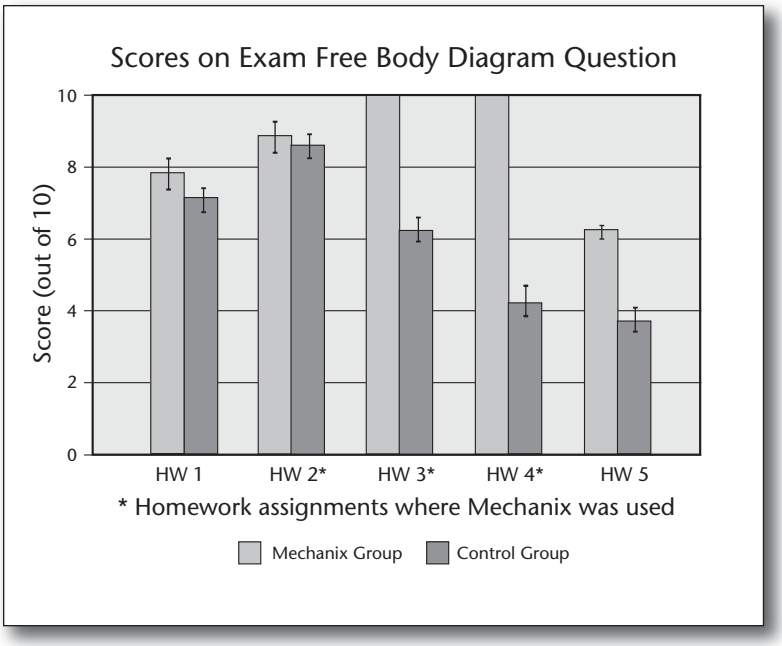

Figure 7. Students Using Mechanix Score 40 Percent Higher on Homework Assignments.

dents struggle with understanding the material. If many students make the same mistake, the instructor knows he or she should probably spend more class time explaining that particular concept.

To prevent the possibility of cheating, the answer sketches drawn by the instructor are never sent to the student's Mechanix client application. This means that all answer checking must be performed in a secure server application before feedback can be sent back to the Mechanix client. Initially this was handled by a single server, but the load of recognition for one problem submission is nontrivial, and when 30 plus students would submit sample problems simultaneously our server would become inoperable.

To overcome this limitation, we use web application load-balancing techniques. All data is transferred between the clients and servers as XML over HTTP. We use an HTTPS proxy server to encrypt all incoming and outgoing data to protect authentication information and student confidentiality. All incoming HTTP requests are then routed using an HTTP load balancer, HAProxy, to several machines, each running the Mechanix server software. Each Mechanix server runs an embedded instance of Jetty, which we use to handle HTTP communication and user session information. Finally the XML request body is parsed, we perform answer checking on the resultant sketch object, and return the necessary feedback as an XML HTTP response. This architecture allows us to use off-the-shelf software to achieve simple and practical scalability to support larger groups of students.

Deployed System Results

We have deployed Mechanix in the classroom for three consecutive semesters. The participating course, Engineering 111 (Foundations of Engineering), covers basic statics, visualization and CAD tools, Newton's laws, unit conversions, and others. Thus far, a total of 111 students (interesting coincidence given the course number) have used Mechanix to submit homework assignments that otherwise would be submitted on paper.

In the fall semester of 2010, Mechanix was first deployed in an honors section of ENG 111. Students in the course were given the option to use Mechanix or pencil and paper for two homework assignments. For the first assignment, all 33 students chose to use Mechanix. For the second assignment, 22 students chose to continue using Mechanix. In the first semester, our purpose was mainly debugging and refinement; we found it promising that 22 students chose to use the software again. Note that students were required to use Mechanix in a computer lab and could not access the software from home. This may have affected participation.

In the second semester of deployment (spring 2011), 20 of 64 student volunteers from a regular section of ENG 111 used Mechanix, and the remaining used traditional pencil and paper for comparison. Mechanix was used for three homework assignments. For the first of the Mechanix assignments, the grades were similar between the Mechanix and control groups. On the second and third assignments, however, the Mechanix group scored an average of more than 40 percent higher than the control group, as detailed in figure 7. More details regarding this deployment can be found in Atilola et al. (2011).

Note that in the spring 2011 semester, the student pool was made up significantly of at-risk students who either did not have the necessary math skills their first semester or were transfer students. Much of the increase in scores was due to the students in the Mechanix group having higher homework completion rates. We believe this to be a good measure of engagement. The honors sections from the other two semesters did not show this increase, which is most likely due to the fact that the honors students were already highly engaged, and thus already turning in complete homework.

For the third semester of deployment, 122 student volunteers from both honors and regular sections participated in the study. In total, 58 students were assigned randomly to the Mechanix group, and the remaining used pencil and paper. Again, three homework assignments were given. Partway through the semester, we found our server could not handle the increased student load (a jump from 20 to 58). Not all test students contin-

ued using Mechanix after the first assignment. These students were removed from our data.

A force concept inventory quiz was given before and after the lectures associated with free-body diagrams and forces. The students that used Mechanix for all three assignments showed substantially more improvement than the other students, as seen in figure 8. Students that used Mechanix scored higher on the free-body diagram question on the exam, as seen in figure 9. These findings indicate that Mechanix fosters learning of statics concepts better than traditional pen and paper. More detailed descriptions of the third semester's deployment can be found in Atilola et al. (2012).

In postexperiment focus groups each semester, students offered many constructive comments on Mechanix. Students found the instant feedback feature very helpful, which encouraged them to choose to use Mechanix over pencil and paper. Students were also impressed that the program could recognize even badly drawn trusses. This made them think more about the problem and less on trying to draw a perfect truss. One other feature that students mentioned was the checklist that told them the order in which to solve the problem. They thought it helped ensure they were solving the problem correctly. In the focus groups some students requested the use of Mechanix on exams in addition to homework. This implies confidence in the software and a willingness to continue to use it. Some students also mentioned that using Mechanix encouraged them to move on to another problem after finishing the first. Many students expressed that they thought of Mechanix more as a learning tool that helps teach the process of solving these problems than just another way to turn in homework.

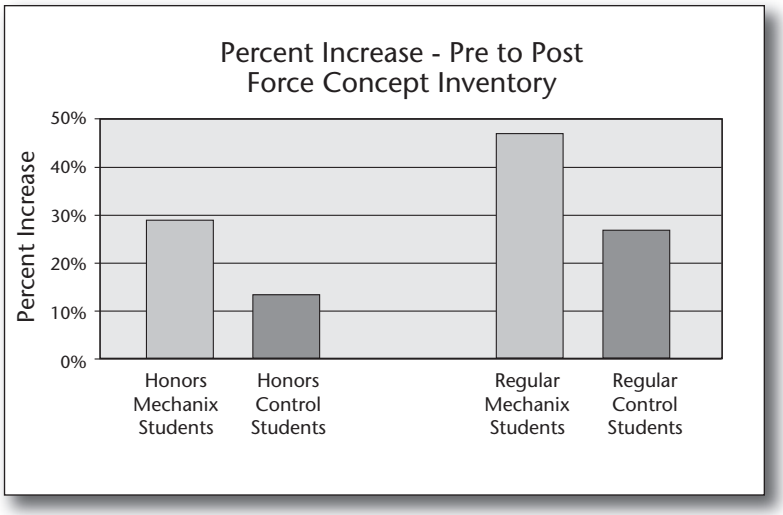

Figure 8. Students Using Mechanix Proved a Better Improvement in Knowledge of Forces than Nonusers

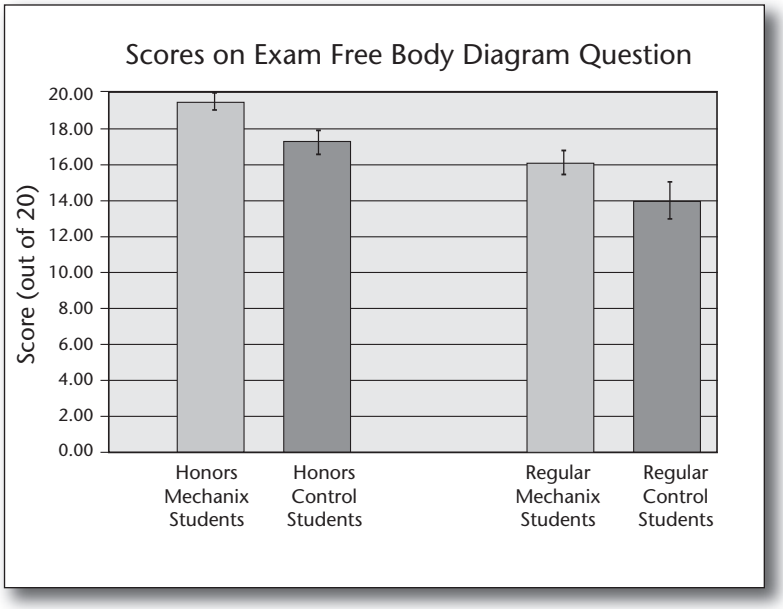

Figure 9. Students Using Mechanix Scored Higher on the Free-Body Diagram Exam Question.

Conclusion

In this article we described Mechanix, a deployed system, and its use of artificial intelligence to aid teachers and students with the learning process. Mechanix is built with a number of recognition techniques that give it many features to help students become successful in the classroom. It has been able to interpret students' answers in real time to provide instant feedback in a transparent environment. The goal of Mechanix is to be a fluid system that can provide instant feedback while still allowing students to hand-draw their solutions.

Acknowledgements

The authors thank the many other contributors to Mechanix, specifically Martin Field and Chris Aikens. This research is funded in part by Google and the National Science Foundation under Grant Nos. 0935219 and 0942400.

References

Alvarado, C., and Davis, R. 2004. Sketchread: A Multi-Domain Sketch Recognition Engine. In *Proceedings of the 17th Annual ACM Cymposium on User Interface Software and Technology,* 23–32. New York: Association for Computing Machinery.

Atilola, O.; Field, M.; McTigue, E.; Hammond, T.; and Linsey, J. 2011. Evaluation of a Natural Sketch Interface for Truss FBDs and Analysis. In *Proceedings of the IEEE 2011 Frontiers in Education Conference (FIE)*. Piscataway, NJ: Institute of Electrical and Electronics Engineers.

Atilola, O.; Osterman, C.; Hammond, T.; and Linsey, J. 2012. Mechanix: The Development of a Sketch Recognition Truss Tutoring System. In *Proceedings of the American*

Society for Engineering Education 2012 Annual Conference. Washington, DC: American Society for Engineering Education.

Bangert-Drowns, R. L.; Kulik, C. L. C.; Kulik, J. A.; and Morgan, M. 1991. The Instructional Effect of Feedback in Test-Like Events. *Review of Educational Research* 61(2): 213–238.

Callahan, J.; Hopkins, D.; Weiser, M.; and Shneiderman, B. 1988. An Empirical Comparison of Pie Versus Linear Menus. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems,* 95–100. New York: Association for Computing Machinery.

Field, M.; Valentine, S.; Linsey, J.; and Hammond, T. 2011. Sketch Recognition Algorithms for Comparing Complex and Unpredictable Shapes. In *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence*. Menlo Park, CA: AAAI Press.

Gennari, L.; Kara, L. B.; Stahovich, T. F.; and Shimada, K. 2005. Combining Geometry and Domain Knowledge to Interpret Hand-Drawn Diagrams. *Computers and Graphics* 29(4): 547–562.

Hammond, T., and Davis, R. 2005. LADDER, A Sketching Language for User Interface Developers. *Computers & Graphics* 29(4): 518–532.

Johnston, J., and Hammond, T. 2010. Computing Confidence Values for Geometric Constraints for Use in Sketch Recognition. In *Proceedings of the Seventh Eurographics / ACM Sketch-Based Interfaces and Modeling Symposium,* 71–78. Aire-la-Ville, Switzerland: Eurographics Association.

Kara, L. B., and Stahovich, T. F. 2005. An Image-Based, Trainable Symbol Recognizer for Hand-Drawn Sketches. *Computers & Graphics* 29(4): 501–517.

Kozma, R. B. 1994. Will Media Influence Learning Reframing the Debate? *Educational Technology* 42: 7–19.

Lee, W.; de Silva, R.; Peterson, E. J.; Calfee, R. C.; and Stahovich, T. F. 2008. Newton's Pen: A Pen-Based Tutoring System for Statics. *Computers & Graphics* 32(5): 511–524.

Miller, E. G.; Matsakis, N. E.; and Viola, P. A. 2000. Learning from One Example Through Shared Densities on Transforms. In *Proceedings of the 2000 IEEE Conference on Computer Vision and Pattern Recognition,* 464–471. Piscataway, NJ: Institute of Electrical and Electronics Engineers.

Paulson, B., and Hammond, T. 2008. PaleoSketch: Accurate Primitive Sketch Recognition and Beautification. In *Proceedings of the 13th International Cnference on Intelligent User Interfaces,* 1–10. New York: Association for Computing Machinery.

Roselli, R. J.; Howard, L.; Cinnamon, B.; Brophy, S.; Norris, P.; Rothney, M.; and Eggers, D. 2003. Integration of an Interactive Free Body Diagram Assistant with a Courseware Authoring Package and an Experimental Learning Management System. In *Proceedings of the American Society for Engineering Education.* Washington, DC: American Society for Engineering Education.

Rubine, D. 1991. Specifying Gestures by Example. In *SIGGRAPH '91: Proceedings of the 18th Annual Conference on Computer Graphics and Interactive Techniques,* 329–337. New York: Association for Computing Machinery.

Sweller, J. 1994. Cognitive Load Theory, Learning Difficulty, and Instructional Design. *Learning and Instruction* 4(4): 295–312.

Vanlehn, K.; Lynch, C.; Schulze, K.; Shapiro, J. A.; Shelby, R.; Taylor, L.; Treacy, D.; Weinstein, A.; and Wintersgill, M. 2005. The Andes Physics Tutoring System: Lessons Learned. *International Journal of Artificial Intelligence Education* 15(3): 147–204.

Wobbrock, J. O.; Wilson, A. D.; and Li, Y. 2007. Gestures Without Libraries, Toolkits, or Training: A $1 Recognizer for User Interface Prototypes. In *Proceedings of the 20th Annual ACM Symposium on User Interface Software and Technology,* 159–168. New York: Association for Computing Machinery.

Wolin, A. 2010. Segmenting Hand-Drawn Strokes. Master's thesis, Texas A&M University, College Station, TX.

Yin, P.; Forbus, K. D.; Usher, J.; Sageman, B.; and Jee, B. D. 2010. Sketch Worksheets: A Sketch-Based Educational Software System. In *Proceedings of the 22nd AAAI Conference on Innovative Applications of Artificial Intelligence*. Menlo Park, CA: AAAI Press.

Stephanie Valentine is a Ph.D. student at Texas A&M University. She works as a research assistant in the Sketch Recognition Lab under the guidance of Tracy Hammond. Her research interests include geometric sketch recognition, educational software, child-computer interaction, and cyber-safety for preteens and early adolescents.

Francisco Vides is currently working as a software engineer at PayPal. He received a Master's degree in computer science at Texas A&M University where he also worked as a graduate researcher in the Sketch Recognition Lab under the direction of Tracy Hammond. His research interests are in computer-human interaction, artificial intelligence, and intelligent tutoring systems.

George Lucchese is a Master's student at Texas A&M University and a software developer with IBM. He works as a research assistant in the Sketch Recognition Lab under the guidance of Tracy Hammond. His research primary interests are in computer human interaction, sketch recognition, pattern analysis, and distributed systems.

David Turner is a junior undergraduate student at Texas A&M University. He is a research assistant in the Sketch Recognition Lab under Tracy Hammond. He designed and implemented the creative response functionality of Mechanix over the past two summers.

Hong-hoe Kim is a Ph.D. student in the Sketch Recognition Lab at Texas A&M University under the supervision of Tracy Hammond. His research is in the area of human-computer interaction and artificial intelligence, including the design of pervasive computing applications, sketch-based educational applications, and pattern analysis of human's drawings. He holds a Bachelor's degree in computer science from Soongsil University in Korea.

Wenzhe Li is currently working as a software development engineer at Amazon.com. His research interests are in machine learning and computer human interaction. He received an MS in computer science at Texas A&M University, and a BS in computer science at Nankai University.

Julie Linsey is an assistant professor in the Mechanical Engineering Department at Texas A&M University. Her research focus is on design methods, theory, and engineering education with a particular focus on innovation and conceptual design.

Tracy Hammond is the director of the Sketch Recognition Lab and an associate professor in the Department of Computer Science and Engineering at Texas A&M University. Hammond earned a BA in mathematics, a BS in applied mathematics, an MS in computer science, and an MA in anthropology as well as an FTO (finance technology option) and a Ph.D. in computer science from MIT under Randy Davis.