Mechanix: A Sketch-Based Tutoring System for Statics Courses *

Stephanie Valentine†, Francisco Vides†, George Lucchese†, David Turner†, Hong-hoe Kim†, Wenzhe Li†, Julie Linsey+, Tracy Hammond†
†Department of Computer Science and Engineering, +Department of Mechanical Engineering
Texas A&M University
valentine@cs.tamu.edu, fvides@tamu.edu, george_lucchese@tamu.edu, dturner@cs.tamu.edu, hhkim@cs.tamu.edu, liwenzhe@tamu.edu, jlinsey@tamu.edu, hammond@cs.tamu.edu

Abstract

Introductory engineering courses within large universities often have annual enrollments which can reach up to a thousand students. It is very challenging to achieve differentiated instruction in classrooms with class sizes and student diversity of such great magnitude. Professors can only assess whether students have mastered a concept by using multiple choice questions, while detailed homework assignments, such as planar truss diagrams, are rarely assigned because professors and teaching assistants would be too overburdened with grading to return assignments with valuable feedback in a timely manner. In this paper, 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 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 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 displaying memorized information.

Introduction

Freshmen, in their first semester as mechanical and civil engineering students, learn the fundamental engineering concepts through a course in statics. Statics problems usually require the student to draw planar truss and other free-body diagrams.

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 non-truss free-body diagrams can be used to analyze all of the internal and external forces acting on a general object of any shape.

In order to correctly assess the effectiveness of the learning process the learner must receive appropriate feedback on what he or she is doing. 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 hand-drawn solutions from being used often because of the time commitment involved in grading and providing feedback to the students. To combat these time constraints, multiple choice questions are the primary source of testing. In these courses, students are likely to receive only one or two hand-drawn assignments a semester.

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

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 expertise performance achieved by human instructors.

Prior Work

Specialized educational software packages act as tutors for students that are learning statics problems. These include WinTruss (Callahan et al. 1988), McGraw Hill Connect Engineering (Hill 2011), and Bridge Architect (Fissi 2011), 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 paper 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 (FBD) 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 dependant on the order in which the student draws the components of the solution.

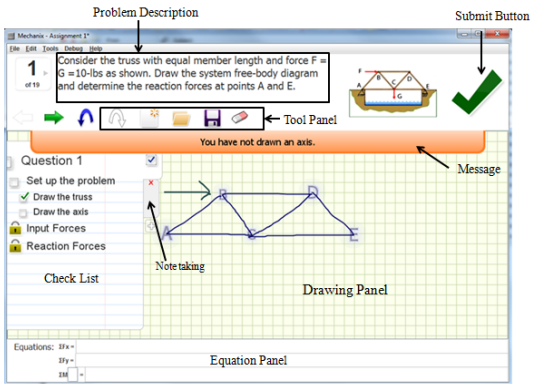

Figure 1: Mechanix interface. It can be divided into several different parts. a) Problem Description b) Drawing Panel. c) Tool Panel. d) Submit Button e) Check List f) Message g) Equation Panel h) Note Taking Panel

To achieve this eager 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 constrains to determine if the basic shapes and the relationship between them compose a more complex shape. This approach has been used successfully in domains such as military Course of Action (Johnston and Hammond 2010) diagrams, 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 pre-recognitions 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. Paleo has a reported accuracy of more than 98%.

Implementation

A Tour of the Interface

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

- **Problem Description.** The text shows 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.
- **Drawing Panel.** This is the panel on which students draw their diagrams. Each pen-down motion is captured and processed by our software.
- **Tool Panel.** This panel contains useful functions to help edit the sketches. The buttons (from left to right) are undo, redo, clear, open, save and erase. In addition to the erase button, the system also provides a scribble erasing functionality.
- **Submit Button**. Whenever students want to check their solutions, they can simply click the submit button to see whether their answers are correct.
- **Check List.** The checklist provides step-by-step guides to assist students to finish the problem.
- **Message.** The system gives feedback by showing a helpful and informative message. 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.
- **Equation Panel.** Students use this panel to enter the required equations and values of reaction forces. Then the system compares these inputs with correct answers.
- **Note Taking.** Students can use this panel to make notes when they are working any problem

Interface Modes

There are two distinct modes of interaction in Mechanix, the student mode and 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.

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 inputted 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.

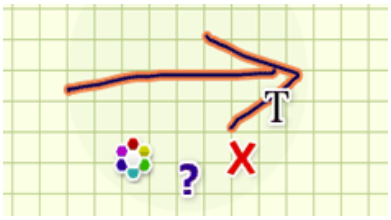

Figure 2: An example round menu and some basic options

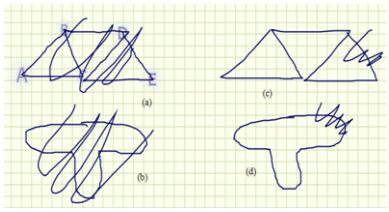

Figure 3: Examples of the capabilities of the scribble erasure feature.

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 free-hand erasure.

- **Round Menu**. Figure 2 shows the round menu. 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. Using this round menu, we can change the color of a shape, delete the highlighted shape, or label the shape.
- **Buttons**. There are several buttons on the tool panel, as we've shown in Figure 1. Users can use these buttons to explicitly undo/redo/clear/erase their drawings.
- **Gesture Recognition**. We allow erasure via 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 3 shows how can we 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.
- **Pen or mouse interaction**. 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 shape and entering the text.

- **Visual feedback**. Primitive recognition is used as not only 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 dark green to indicate to the student that Mechanix recognized the force and that they may enter relevant metadata for that force.

Geometric Recognition

Building Blocks

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, etc.
- **Complex shapes**, such as roller supports, 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 five simple steps:

1. Record points as the pen traces on the screen and add points to a new stroke.
2. Send each new stroke to PaleoSketch to find its primitive shapes (line, circle, arc, polyline, etc.).
3. Add each new shape to the collection of all shapes.
4. Send various groupings of shapes to complex shape recognizers.
5. Apply a system of geometric constraints to recognize high-level shapes.
6. Replace low-level shapes with new high-level shape, 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 4 requires a triangle and two circles as components. The roller support recognizer checks each of the conditions given in Figure 4.

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.

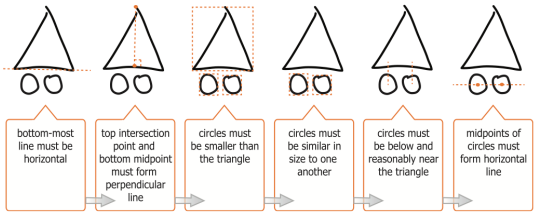

Figure 4: An example of the geometric constraints on a simple roller support.

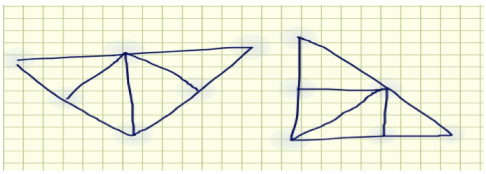

Figure 5: Examples of trusses recognized in Mechanix.

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 517 data sets and achieved an accuracy of 89%. Figure 5 shows examples of trusses recognized in Mechanix.

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 6 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 6, the 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 6 (a), the red path can reach from A to B, and our system identifies the edge AB as a shared edge. However, in Figure 6 (b), 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).

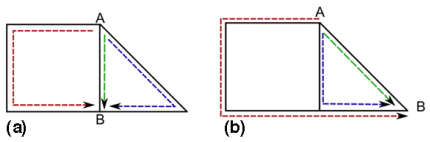

Figure 6: Two examples of shared edge. Only (a) will be recognized as a shared edge by our algorithm.

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-hand corner of the interface. In order to grade the student's submission, Mechanix compares the student's sketch with instructor's sketch. Mechanix checks the assignments as follows:

- The student's truss has the correct configuration.
- The student's sketch contains an axis.
- All forces are present and in the correct direction.
- The student's force values are equivalent to the instructor's force values.

To check if the student's truss is similar to the instructor's hand-drawn truss, we first create a graph data structure comprised 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 one given in Figure 5, some students drew a big triangle first and draw other small triangles. On the other hand, some students only drew three small triangles. 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 if 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 the primary direction of forces.

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 label (for example "Fay"). After Mechanix checks the types of forces, the system checks if 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".

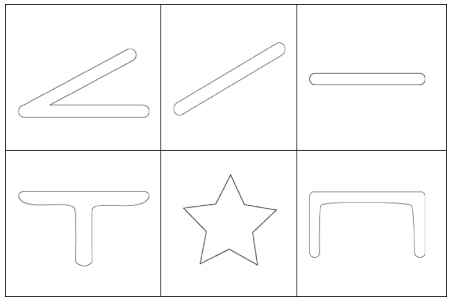

Figure 7: Example closed-shape bodies.

Other Problem Types

General Free-Body Diagrams

Free-body diagrams are a second type of problem in the statics domain. Generalized 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 bubble-like shape, such as those seen in Figure 7. 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 are traversible if their endpoints are within 9% of the total path length. This gap allowance allows for bodies to be made of multiple strokes, without excessive care to perfectly line up endpoints.

To begin comparing the two shapes (the student's shape and the instructor's shape), we first resample the shapes to ensure they have the same number of points. Using 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 40x40 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, each of size 64 (the number of points in each shape) D_A and D_B such that

$$D_A = \left\{ \min_{b \in P_B} \left(|a - b| \right), a \in P_A \right\}$$

P_A contains minimum distances for points in A, and P_B contains minimum distances for points in B. The values in D_B are similarly found. The maximum value in both D_A and D_B is the Hausdorff distance. The modified Hausdorff distance is the average of the D_A and D_B 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 with

$$P\left(\text{match}\right) = 1 - \frac{H}{20}$$

We chose the value 20 because it represents half of the width of the bounding window. Any two shapes that contain points whose nearest neighbors are more than half the width of the bounding window apart cannot be deemed similar. Therefore, two shapes resulting in a Hausdorff distance of 20 is considered 0% similar. Any two shapes with a Hausdorff distance greater than 20 receive a similarity measure of 0.

The final measure used to determine body similarity is the Tanimoto Coefficent. This is simply the ratio of pixels that "overlap" (number of points that have distance values in P_A and P_B less than or equal to 4.0) to the total number of pixels.

$$T\left(A, B\right) = \frac{n_{AB} + n_{BA}}{n_A + n_B}$$

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 left 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 constraints. A typical problem is:

> "A village needs a bridge to connect it to the city market place 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 student 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 solve the student created truss (Figure 8).

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. It uses the arrow recognizer and compares the slope with the axis to determine the direction, which gives the equation

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

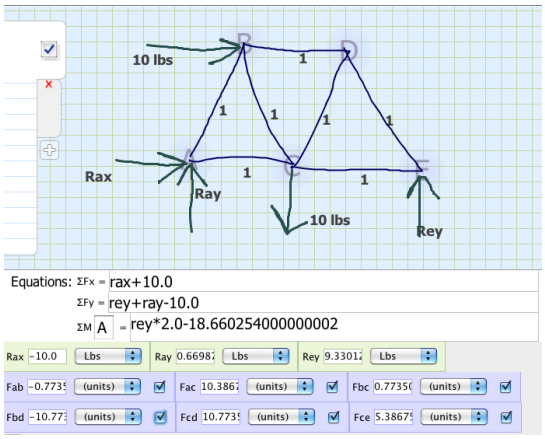

Figure 8: A Creative Response Truss

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

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

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 iterates through all of the other forces computing the cross product between the forces and the distance to the node. In Figure 8, node A is chosen to compute the cross product and results in the equation below.

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

$$= \sum M = R_{ey} \times 2 - 18.660$$

After finding all the external forces, a system of equations is formed 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 they drew adhered to the constraints.

Distributed Architecture

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 non-trivial, 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 a 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[1] , to several machines, each running the Mechanix server software. Each Mechanix server runs an embedded instance of Jetty[2] , 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 amounts 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, etc. 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.

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. The first semester, our purpose was mainly debugging and refinement; we found it promising that 22 students chose to use the software again.

In the second semester of deployment, 20 of 64 student volunteers from a regular section of ENG 111 used Mechanix, and the remaining 44 used traditional pencil and paper for comparison. Mechanix was used for three homework assignments. The grades for the first assignment were similar between the experimental and control groups. On the second and third assignments, however, the experimental Mechanix group scored an average of 25% higher than the control group.

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 64 used pencil and paper. Again, three homework assignments were given. Due to some unforeseen server problems, many students chose not to continue using Mechanix after the first or second assignment.

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

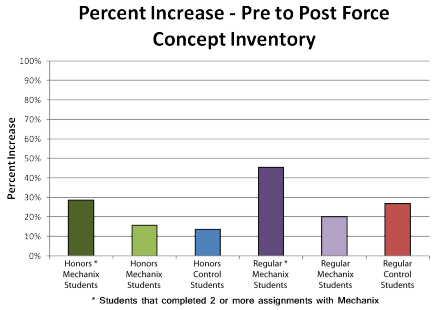

Figure 9: Assignment Results

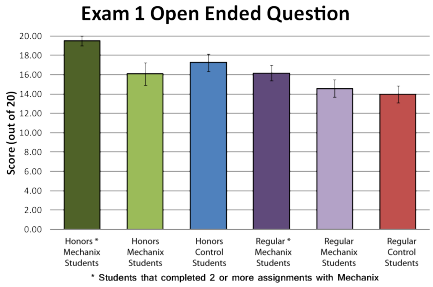

Figure 10: Exam Results

showed substantially more improvement than the other students, as seen in Figure 9. Students that used Mechanix for at least two of the three assignments scored higher on the open-ended question on the exam, as seen in Figure 10. These findings indicate that Mechanix fosters learning of statics concepts better than traditional pen and paper.

In post-experiment 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 doing the problem the 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.

Students also offered suggestions on how to improve Mechanix. Many students expressed the need to create an eraser that can erase certain parts of the drawing. This has been created and added to Mechanix, but has not yet been tested in the classroom. Per the students' suggestions, we are working to solidify the client-server interaction so as to lessen the number of interruptions to the user.

[1]HAProxy http://haproxy.1wt.eu/

[2]Jetty HTTP Server http://jetty.codehaus.org/

Future Work

Although Mechanix is today a robust system with real classroom usage, there are several ways of expanding our work.

One goal that we have is to reach a wider audience. The use of Mechanix in other schools and universities is currently in the works by our team and at least one other school has agreed to collaborate in order to get access to this educational tool. Because of this, numerous enhancements in terms of security, portability, and scalability are currently being developed so we can deploy our system in a wider set of schools and environments.

Similarly, Mechanix has the potential to be used outside of the classroom, to do homework assignments or practice problems. As a web-based tool this is already possible, but we have the challenge of making the software portable enough so it can run smoothly across different hardware platforms and operating systems. This also includes the possibility of running Mechanix on portable devices such as phones or tablets.

The software itself has several improvements that can be made. Many improvements were suggested by the students in the post-experiment focus groups. These include interaction preferences, a wider set of recognized shapes, more intelligent and explicit feedback, powertools to edit the sketch, and ways of self-assessing assignment progress.

Another side of Mechanix that has been developed but has a great potential of expansion is the instructor mode Instructor mode can be expanded by giving the instructor a wide range of statistics so they can immediately asses the progress of their students without the need of external tools. Also, our work in equation generation and solving should provide them with faster ways to create problems in the future using a similar feedback interface to let them know potential inconsistencies with their proposed problems. Another powerful tool we plan to provide instructors is the ability to integrate Mechanix with standard student platforms such as BlackBoard, or eLearning.

Conclusion

In this paper we describe 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 can provide instant feedback while still allowing students to hand draw their solutions.

References

Alvarado, C., and Davis, R. 2004. Sketchread: a multi-domain sketch recognition engine. In *UIST '04: Proceedings of the 17th annual ACM symposium on User interface software and technology*, 23–32.

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 vs. linear menus. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, 95–100. New York, NY, USA: ACM.

Field, M.; Valentine, S.; Linsey, J.; and Hammond, T. 2011. Sketch recognition algorithms for comparing complex and unpredictable shapes. In *Twenty-Third International Joint Conference on Artificial Intelligence*.

Fissi, L. 2011. Bridge architecht website.

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.

Hill, M. 2011. Mc graw hill connect website.

Johnston, J., and Hammond, T. 2010. Computing confidence values for geometric constraints for use in sketch recognition. In *Proceedings of the Seventh Sketch-Based Interfaces and Modeling Symposium*, SBIM '10, 71–78. Aire-la-Ville, Switzerland, 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. Kozma - 1994 - will media influence learning reframing the debate.pdf. *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.

Paulson, B., and Hammond, T. 2008. Paleosketch: accurate primitive sketch recognition and beautification. In *Proceedings of the 13th international conference on Intelligent user interfaces*, IUI '08, 1–10. New York, NY, USA: ACM.

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*.

Rubine, D. 1991. Specifying gestures by example. In *SIGGRAPH '91: Proceedings of the 18th annual conference on Computer graphics and interactive techniques*, 329–337.

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. *Int. J. Artif. Intell. Ed.* 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 *UIST '07: Proceedings of the 20th annual ACM symposium on User Interface Software and Technology*, 159–168. New York, NY, USA: ACM.

Wolin, A. 2010. Segmenting hand-drawn strokes. Master's thesis, Texas A&M University.