Just Truss Me: Symbol Recognition

Stephanie Valentine, Chris Aikens, Alexis Chuck, Martin Field, Travis Kosarek, Drew Logsdon, Laura Murphy, Patrick Robinson, Alyssa Nabors, Paul Taele, Erin McTigue, Julie Linsey, Tracy Hammond
Sketch Recognition Lab
Computer Science Department
Texas A&M University
911 Richardson
College Station, TX 77843-3112

ABSTRACT
In their first semester, mechanical and civil engineering students learn the mechanics of trusses, one of the most fundamental concepts of engineering. From houses and bridges to skyscrapers and playground equipment, trusses are the support systems of everything around us, so a clear understanding of them is imperative to success. Professors want to assign homework problems involving hand drawn truss diagrams, but the grading of such diagrams requires extensive time commitment. Especially in large classrooms, this amount of time is difficult to attain. To combat this problem, we introduce Mekanix, a sketch recognition system that can recognize, correct, and provide feedback on a student's hand-drawn truss diagram in real-time. We use geometric constraints to recognize the diagram's components from the primitive shapes (i.e. line, circle, triangle, etc.) they comprise. In order to make our recognizers robust enough for classroom use, we allow for several configurations, variations, and drawing styles for each shape. Designed to enhance learning, Mekanix is an unobtrusive and helpful tool that benefits the professor and teaching assistant as much as the student.

INTRODUCTION
In their first semester, mechanical and civil engineering students learn the fundamental concepts of engineering. A large portion of the time spent in these introductory classes is devoted to solving statics problems. Statics problems usually require the student to draw free body diagrams and planar truss diagrams.

A free body diagram can be used to analyze all of the internal and external forces acting on an object, while a planar truss diagram is simply a two dimensional representation of a structure. This type of structure 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.

In current practice, the most effective method for learning how to construct a truss is to draw the truss along with the forces acting upon it on pen and paper. This method works best when an active learning approach is taken, that is, a learner should be engaged and cognitively active while learning. Timely feedback should be given to the learner when a mistake is made to prevent the learner from adding false information into their knowledge framework.

While this method seems ideal, the large class sizes of introductory engineering courses prevents hand-drawn solutions from being used often because of 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.

To stimulate the educational value of these courses, the need for a better method of grading these hand-drawn truss diagrams is necessary. Hand-drawn homework problems, such as truss diagrams, afford themselves the use of sketch recognition as a solution. Sketch recognition allows a user to freely draw any combination of strokes and attempts to recognize and interpret what the user intended by the sketch.

BACKGROUND
There are two main categories of sketch recognition: gesture-based and free-sketch. Gesture-based recognition systems track the movements of the pen (or mouse) and recognize shapes based on the gestures. Gesture recognition requires that each element of the shape be drawn in succession. For example, this system recognizes a circle drawn in a clockwise direction differently that one drawn counter-clockwise. Because of the specific nature of the gestures, recognition accuracy can be quite high, but

learning the movements can be tedious and time-consuming.

Free-sketch systems focus more on what a shape looks like than how it is drawn. These systems use a combination of techniques to recognize shapes, including vision- [Oltmans 2007], and geometric-based techniques [Alvarado 2004; Hammond 2005]. These methods allow users to sketch naturally, permitting them to begin using the programs with little or no instruction. This is ideal for educational software because teachers and professors want the students to learn the concepts of the course, not specifics of software. The obvious benefits of free-sketch recognition techniques led us to choose a geometric-based system for Mekanix.

RELATED AND PRIOR WORK

LADDER [Hammond 2005] is a sketch recognition language that is used for the recognition of shapes. LADDER uses geometric recognition to define how a shape is formed. Recognizers can be defined by first drawing a shape. LADDER takes the sketched shape and recognizes the primitive shapes such as lines, arcs, circles, etc. LADDER then automatically creates a recognizer for that shape based on constraints like "below", "near", or "coincident". This is first accomplished by recognizing primitive shapes such as lines, arcs, circles, etc. LADDER then uses these constraints to define or describe the higher level shapes.

PaleoSketch [Paulson 2008] is a sketch recognition library used to recognize hand-drawn primitives like lines, ellipses, arcs, curves, etc. To do this, PaleoSketch creates confidence values on what shape a stroke could potentially be. PaleoSketch then chooses the shape with highest confidence value as the recognized shape.

WinTruss [Sutton 2000] is an application to design and solve truss diagrams. Before the user can begin drawing trusses, the application's environment must be set up with specific information about units, grid spacing, and the materials being used to build the structure. After this is done, the system allows the user to use tools such as the "beam tool" to draw a beam on the screen, define the actual length of the beam, and label it as needed. After the external forces have been applied to the truss diagram, WinTruss can solve the member force values of the constructed truss diagram. The system is designed to allow the user to draw and simulate the forces acting on the truss, however, it does not provide instruction or feedback on how trusses should be formed.

Newton's Pen [Stahovich 2007] is a "pentop computer" application, meaning that it runs on a processor inside the pen itself. The application uses vision-based sketch recognition to accept or reject very simple free body diagrams. To recognize shapes, the pen digitizes the ink that it inscribes on paper and compares the digitized strokes to a bitmap of the "perfect" configuration for that shape. The program runs as a finite state machine, so each piece of the diagram must be drawn in a specific order and configuration. Newton's Pen gives basic feedback to the user, but only to inform of the number of forces left to be drawn.

METHOD

Building Blocks of Geometric Recognition

The hierarchical building-blocks of our recognition are points, strokes, and shapes.

Points are the simplest of these. The program generates a point for each movement of the mouse. It records the x and y coordinates the current time, and sometimes the pressure and tilt of the pen.

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. This compares to pen strokes on paper. 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, but a single stroke can create multiple primitive shapes. Strokes are segmented using a cusp detector **[**Wolin 2010] and the primitive shapes are recognized by PaleoSketch [Paulson 2008]. Examples of primitive shapes are line segments, circles, arcs, curves, polylines (several line segments drawn in a single stroke), triangles, etc.

Complex shapes are built first of primitives and composed hierarchically to allow for more and more complex shapes. A real-world example of a complex shape is a house. People generally draw a house as a rectangle with a triangle above of it. Thus, the members of our house shape are a triangle (made up of three lines) and a rectangle (made up of four lines). Mekanix creates complex shapes only after the member shapes pass our geometric-constraint-based recognizers.

Geometric Constraints

The human brain has the ability to see a nearly horizontal line and still perceive it to be horizontal, and our recognition needs to work similarly. Mekanix needs to perceive shapes the same way a human brain would, so we need a way to quantify "reasonably." We do this through geometric constraints. Example constraints are belowness, leftness, horizontalness, verticalness, slantedness, and nearness [Johnston 2009]. Constraints return confidence values between 0 (not confident at all) and approximately 1 (entirely confident). In our program, we take a confidence value of 0.55 to mean that the "reasonable" condition is met.

Figure 1. Is the red triangle below the orange 'y'?

Perhaps the constraint that sounds the simplest is belowness. From a mathematical perspective, a simple comparison of the y-coordinates of the shapes would determine a logical and exact definition of belowness. If we took the mathematical approach to the shapes in Figure 1, the red triangle would be confirmed to be below the orange 'y' shape, when human perception says otherwise. If asked what is below the 'y', human eyes would simply start at the 'y' shape and look down. By this method, the red triangle is certainly not below the orange 'y'. Constraints need to be as perceptual as they are geometric.

Steps to Recognition

Our recognition happens in five simple steps:

1. Record points as pen traces on screen and add points to new stroke.
2. Send each new stroke to PaleoSketch (a primitive shape recognition system) [Paulson 2008] to find primitive shapes (line, circle, arc, polyline, etc.).
3. Add new shape to collection of all shapes.
4. Send groupings of shapes to complex shape recognizers that apply geometric constraints.
5. Combine into complex shape, return to step 3 and cycle until no more complex shapes can be found.

The step that we will explain in most detail is step four: applying geometric constraints to find complex shapes. Take, for example, the roller support given in Figure 2. This shape requires a triangle and two circles as members. The recognizer for this shape checks the following conditions:

- The triangle must contain a horizontal line, and that line must be below the other two lines.
- The intersection point of the other two triangle lines and the midpoint of the horizontal line must create a vertical line.
- The circles must be smaller than the triangle.
- The circles must be similar in size to one another.

Figure 2. A sketched example of a simple roller support

- The circles must be below and near the triangle.
- The midpoints of the circles must form a horizontal line.

A graphical representation of these constraints is given in Figure 3.

If the component shapes meet all of the constraints for a specific configuration, the recognizer combines them into a new shape. The collection removes each of the component shapes and then adds the new shape. 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 to make strokes). All recognition executes in real-time.

Upon positive recognition, the recognizer assigns the new shape a label that basically acts as a name tag to the other shapes. The recognizers use obvious labels for the shapes they represent. An arrow gets the label "arrow", a triangle gets "triangle", etc. These labels allow multiple recognizers, and thus multiple configurations, of each shape. The roller support mentioned above is only one of many roller support configurations that exist in the domain of truss diagrams. All of the configurations have the same meaning, so the program simply labels all successful configurations "roller support".

More About Our Recognizers

It takes finesse and refinement to create geometric constraints that are neither too loose nor too tight. Even shapes that sound incredibly simple can take hundreds of lines of code to define. Arrows for example, are a very

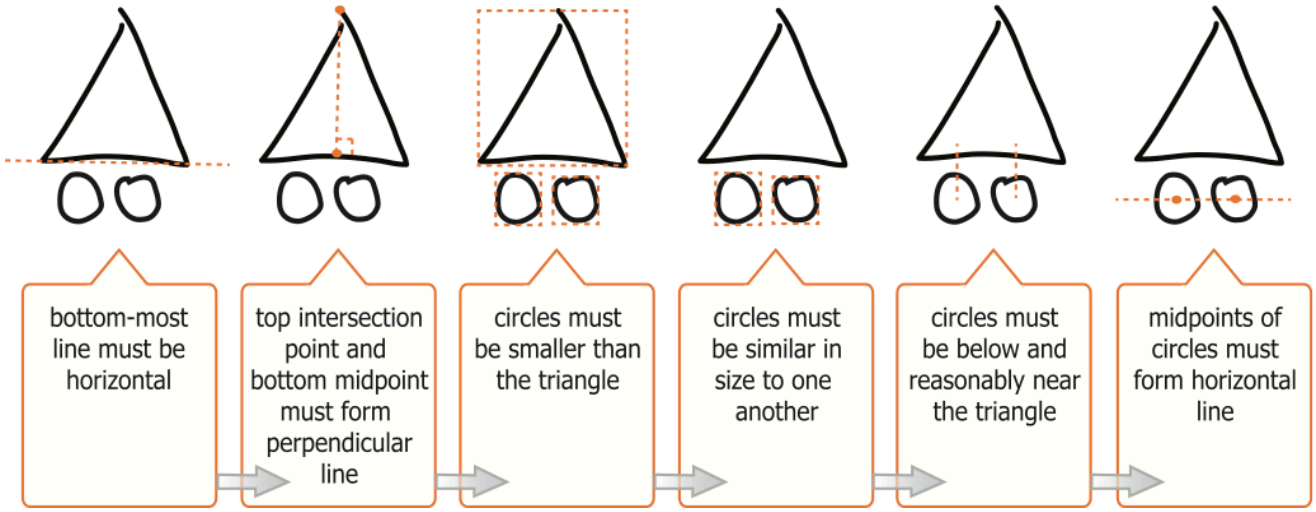

Figure 3. Example geometric constraints for a roller support

common shape and have a fairly concise set of geometric constraints. Arrows are very difficult to define, though, because there are many, many ways of drawing them. A user may choose to draw an arrow in a single stroke, or two strokes, or three, or four, or five strokes. The shaft of an arrow may be very close to the point of an arrow head, but sometimes it is far away. An arrow can point in any direction and be any size. So many variations of this shape exist that Mekanix needs several recognizers to cover all possibilities.

RESULTS

It will take at least a semester of classroom use to formulate quantifiable data on the educational benefit of Mekanix, so currently, we can make few meaningful conclusions. We have, however, performed several user studies with civil and mechanical engineering graduate students and professors to ensure that the software we send into the classroom is robust and intuitive. Each of the five studies helped us to iteratively assess the accuracy of our recognizers and the usability and flow of our interface.

We have a formal classroom study planned to take place during the fall 2010 semester. In our study, we will gather about 75 participants from a single undergraduate engineering course usually taken by freshmen. Each student will receive extra credit for his/her participation in the study. We will split participants into four groups. Three of the groups will attend five two-hour sessions (ten hours total) during which they will receive several problems to solve. One group will solve the problems using our Mekanix software, one will use WinTruss (a competitor software), and one will use simple pencil and paper. The fourth group will attend no extra tutoring sessions.

We will track the test scores of the participants before, during, and after the tutoring sessions to determine the impact of Mekanix's instant feedback on overall learning. We will also invite a sampling of the participants in the first three groups to a focus group discussion, where we will ask for feedback on the effectiveness, intuitiveness, and helpfulness of each method (Mekanix, WinTruss, and pen/paper).

CONCLUSIONS AND FUTURE WORK

Mekanix recognizes, corrects, and provides feedback on a student's hand-drawn truss diagram in real-time. We use geometric constraints to recognize the diagram's components from the primitive shapes they comprise. In order to make our recognizers robust enough for classroom use, we allow for several configurations, variations, and drawing styles for each shape. Designed to enhance learning, Mekanix is an unobtrusive and helpful recognition tool that benefits the professor and the teaching assistant as much as the student.

In the future, we hope to create a web system to process the correctness of sketches, thus removing the "key" sketches from the student's hands. The "keys" currently take little investigation to discover if a student is knowledgeable about the system.

Also, we would like to expand our system to allow the full spectrum of free-body diagrams, not just trusses.

ACKNOWLEDGMENTS

We thank the CRA-W/CDC Distributed Research Experience for supporting this opportunity.

This work funded in part by NSF IIS grants: NSF 0935219: Civil Engineering Sketch Workbook, NSF 0942400: Sketched-Truss Recognition Tutoring System, and NSF 0943999: REU Supplement for 0757557.

REFERENCES

1. [Alvarado 2004] Alvarado, C. (2004) Sketch Recognition User Interfaces: Guidelines for Design and Development. In *Proceedings of AAAI Fall Symposium on Intelligent Pen-based Interfaces*, 2004.
2. [Hammond 2005] Hammond, T., and Davis, R., 2005, LADDER: A Sketching Language for User Interface Developers, Computer and Graphics, Elsevier, pp. 518-532.
3. [Johnston 2009] Johnston, J., and Hammond, T., 2009, Assigning Confidence Values to Geometric Constraints, 2009 Intelligent User Interfaces Workshop on Sketch Recognition, pp. 1-4.
4. [Lee 2007] Lee W., Silva, R., Peterson, E., Calfee, R, Stahovich, T., Newton's Pen – A Pen-based Tutoring System for Statics., 2007 EUROGRAPHICS Workshop on Sketch-Based Interfaces and Modeling, pp 59-66.
5. [Oltmans, 2007] Oltmans, M., Envisioning Sketch Recognition: A Local Feature Based Approach to Recognizing Informal Sketches. PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, May 2007.
6. [Paulson 2008] Paulson, B., and Hammond, T., 2008, PaleoSketch: Accurate Primitive Sketch Recognition and Beautification, 13th International Conference on Intelligent User Interfaces, pp. 1-10.
7. [Sutton 2000] Sutton, M., and Jong, I., 2000, A Truss Analyzer for Enriching the Learning Experience of Students. In *2000 ASEE Annual Conference Proceeding.*
8. Wolin, A. *Segmenting Hand-Drawn Strokes.* Texas A&M University Masters Thesis, May 14, 2010.