Mekanix: A Sketch-based Constructive Learning Tool for Engineering Education

Travis Kosarek, Chris Aikens, Alexis Chuck, Martin Field, Drew Logsdon, Laura Murphy, Patrick Robinson, Alyssa Nabors, Paul Taele, Stephanie Valentine, 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 entry-level engineering courses, students are required to learn and understand basic statics problems. It is common for these courses to have a large number of students, forcing the professor to assign multiple choice problems as opposed to the preferred learning method of hand drawn truss diagrams. These hand drawn diagrams are not always unique solutions and, due to time constraints, this type of homework cannot be graded efficiently.

To alleviate the time constraints faced with grading truss diagrams, we developed and application named Mekanix. Mekanix utilizes sketch recognition to provide the identical functionality of pen and paper, but gives the student real-time feedback when making an error and relieves the professor from grading these hand drawn solutions by automatically comparing them with a stored correct solution. Due to the complexity of truss diagrams and the variability of the drawing order of strokes that form trusses, we developed a specialized algorithm for truss recognition. This paper provides a detailed overview of Mekanix's recognition process. A user study to evaluate Mekanix's educational benefit will be conducted in the Fall 2010 semester at Texas A&M University. By comparing results from similar problems completed with both pen and paper and a similar truss diagram application, WinTruss, we can measure the effectiveness of Mekanix.

INTRODUCTION
In their first semester, mechanical and civil engineering students learn the fundamental concepts of engineering. A large section of the time spent in these introductory classes is devoted to solving statics problems. Statics problems usually require the student to draw free body 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 example of a truss is shown in Figure 1. An excellent foundation of how to construct a truss is critical for a student's future success as an engineer.

Figure 1- A truss used in the construction of a bridge.

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 with pen and paper. This method works best when an active learning approach [2] 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 to their knowledge framework. [7]

CHI 2011, May 7–12, 2011, Vancouver, BC, Canada.

While this method seems ideal, the large size of introductory engineering courses prevents hand-drawn solutions from being used due to time constraints placed on providing feedback to the students. To combat these 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 increase the educational value of these courses, 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 try to recognize shapes by vision- [5], and geometric-based techniques [1, 3]. These techniques 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 an application.

The obvious benefits of free-sketch techniques of recognition led us to choose a geometric-based system for Mekanix.

RELATED WORK

LADDER [3] is a sketch recognition language that is used for the recognition of shapes. LADDER uses geometry-based recognition to define how a shape is formed. Recognizers can be defined by first drawing a shape. LADDER then automatically creates a recognizer for that shape based on constraints like "below", "near", or "coincident". This is 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 [6] 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 the highest confidence value as the recognized shape.

WinTruss [8] is an application used 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. The system then 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. WinTruss can then solve the member force values of the constructed truss diagram only after the external forces have been applied to the truss diagram. The system is designed to allow the user to draw and simulate the forces acting on a truss; it does not, however, provide instruction or feedback on how trusses should be formed.

Newton's Pen [4] 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. The application gives basic feedback, but only to inform the user of the number of forces left to be drawn.

IMPLEMENTATION

Mekanix itself is built from two main prior works. The first, LADDER [3], is a geometry-based sketch recognition language. LADDER provides many useful classes to build complex recognizers specific to our application like multi-headed arrows, axes, and supports. It does this by basing sketches on low level primitives such as lines, arcs, curves, and polyline shapes that can be formed from strokes drawn by the user. By combining low level primitives, it becomes easier to extend and create recognizers that can be perceived as a combination of primitive shapes or other recognized shapes.

The other previous work that Mekanix utilizes is PaleoSketch [6]. PaleoSketch is a sketch recognition library for recognizing shape primitives. By simply sending a stroke to the PaleoSketch recognition system, we are able to recognize general strokes as lines, polylines, arcs, circles, ellipses, arrows, and curves. This allows us to spend less time on the stroke to primitive shape recognition and focus our efforts on the complex shape recognition needed by our application.

We chose to use geometric recognition in Mekanix for three main reasons:

- Both LADDER and PaleoSketch use geometry-based recognition techniques. By building off of pre-existing work, it becomes easier to focus on the purpose of our application.

- We only care about the strokes that are in the sketch, not the order they are drawn in. The stroke order does not change the meaning of a truss diagram; therefore, gesture-based recognition is not necessary.
- We want recognition of the active sketch after a new stroke has been added. Many symbols can be recognized by the subshapes that define them. We want the capability to recognize as much information as we have available in the sketch even if the set of subshapes were previously recognized as a different symbol.

Symbol Recognition

Because we chose to base our application on top of pre-existing work, our symbol recognition process is straightforward. The steps are as follows:

- Use LADDER to record a list of points as a stroke as they are entered into the sketch.
- Send each stroke to PaleoSketch to be recognized as a sketch primitive (lines, circles, arcs, etc.).
- Add the shape returned from PaleoSketch into a collection of shapes.
- Send groupings of shapes from the collection to complex recognizers defined by Mekanix. In each recognizer, geometric constraints are applied to the group of inputted shapes. If the constraints hold, the complex shape is returned.
- If a complex shape is formed, add the complex shape to the collection of shapes and remove its subshapes from the collection.

If a recognized shape is added to the collection of shapes, it is possible for the recognized shape, combined with other shapes in the collection, to form a more complex shape. To encapsulate this case, recognition is tried on the collection of shapes again, after any new shape is added.

It is important to our application that all recognition executes in real-time or as close to real-time as possible, so for a large number of recognizers, recognition can become slow.

Truss Recognition

A truss, as defined previously, is a support used in many structures such as bridges, houses, and other buildings. Examples of trusses are shown in Figure 2.

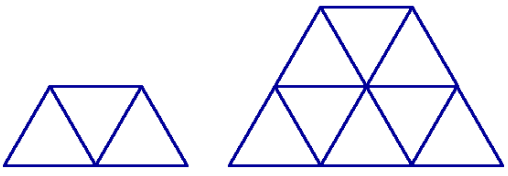

Figure 2 - Example trusses with triangular units only.

Trusses are commonly made from triangular subshapes but may also incorporate other polygons as well. An example of trusses constructed with subshapes other than triangles can be seen in Figure 3.

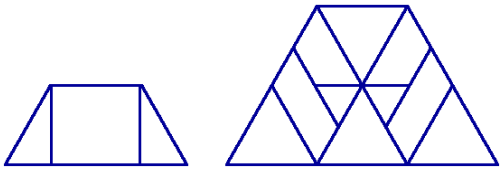

Figure 3 - Example trusses with dissimilar polygon units.

Naïve Truss Recognition

The problem with simply using geometric recognition on truss symbols is that trusses must be defined as either a collection of purely primitive subshapes or a combination of primitive and complex subshapes.

If the first example were to be used, a truss could be made from only primitive subshapes. This means that for every unique truss structure, a recognizer would need to be defined. This would require a large number of recognizers and would cause the overall recognition speed of our application to decline.

If the second example were to be used, a truss could be constructed from a combination of primitive and complex subshapes as shown in Figure 4.

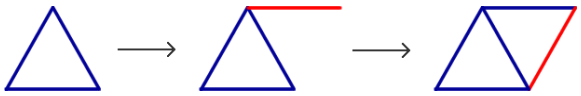

Figure 4 - A truss constructed from a combination of a polygon and two lines.

Using this approach, we attempt to generalize the construction of a truss by defining the order in which these combinations can occur. However, we quickly run into the same issue that the previous example had. For a large variety of trusses to be recognized, every combination of primitives and complex subshapes would need to be defined as individual recognizers.

Intelligent Truss Recognition

In order to avoid the need for a large number of recognizers, a solution that could generally recognize trusses constructed from any combination of polygon-shaped units is necessary.

Our process for recognizing a truss is described as follows:

Begin with the sketch directly after a stroke has been drawn. Divide all of the strokes in the sketch into smaller segments by segmenting at points of intersection. These steps are illustrated in Figure 5.

Figure 5 - Dividing strokes by intersection points.

Construct an adjacency matrix from the intersected strokes. Intersection points become "nodes" and each segmented stroke becomes an "edge".

Assign values to each existing edge. To do this, we take the first endpoint of the edge and find the number of adjacent nodes to that edge. We do the same for the second endpoint. The sum of the adjacent nodes to both endpoints becomes the weight of the edge within the adjacency matrix.

For each edge in the adjacency matrix, we create an additional adjacency matrix. At each compared edge's location in this additional adjacency matrix, we assign the difference between the current edge weight and the compared edge weight. We then take the absolute value of this weight and add a value of one so that every edge will be a positive value greater than zero.

For each additional adjacency matrix created, we run a weighted shortest path algorithm from the current edge's first endpoint to the current edge's second endpoint. This shortest path algorithm has two restrictions:

- The direct path from the first endpoint of the current edge to the second endpoint cannot be taken.
- A path cannot retrace an edge that has already been included in the current path.

These restrictions are in place so that the shortest path, if found, will most likely be the smallest possible polygon branched from the current edge. If the algorithm returns a possible path, the strokes that make up the path are recognized as a polygon. With this system, multiple possible shortest paths can be found; therefore, it is possible to have multiple polygons that can be branched from one edge. An example of polygons that have been found is shown in Figure 6.

Figure 6 - Polygons recognized by a shortest path.

The next step is to add each polygon that is returned to a collection of polygons. It is common for duplicate polygons to be found, therefore duplicates can be discarded.

We can now easily step through this collection and combine any polygons that share an identical edge into a truss. An example of this is shown in Figure 7.

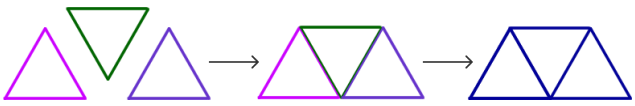

Figure 7 - Combining polygons that share edges into a truss.

Although this method is complex, it has been proven to be efficient and accurate for truss recognition.

EVALUATION

Throughout the development of Mekanix, we have conducted numerous user studies with users possessing various levels of experience in mechanical engineering. These levels of experience range from undergraduate mechanical engineering students to experts in the field. In each user study, we tested for both recognition performance and application usability.

During the Fall 2010 semester at Texas A&M University an extended user study will be performed by placing Mekanix in the curriculum and comparing it to other methods of solving truss diagrams. These other methods include truss diagrams drawn with pen and paper and diagrams constructed with the WinTruss application.

Traditional pen and paper has been proven to be an effective method of learning how to draw truss diagrams. The main drawback is that feedback on the truss diagram's correctness is not timely. It is difficult for graders to return a large amount of corrected hand drawn diagrams and to provide the necessary feedback to help students learn from their mistakes.

WinTruss is a computer application that allows the user to construct truss diagrams by using tools to add each technical aspect of the truss, such as beams, nodes, and forces. After entering many specifics about the truss diagram, WinTruss can automatically solve for internal member forces and other stresses on the truss. While this is useful, the steps required to construct a truss can often times be confusing for first time users and requires a lot of specific training on both truss diagrams and how to use WinTruss itself to be an effective learning tool.

The user study will be performed by volunteers from an introductory engineering course. The students in the course will be divided into four sections. Three sections will be required to use one of three processes of constructing trusses, Mekanix, WinTruss, and pen and paper. The fourth will receive no tutoring sessions. Similar homework problems will be given to each section to solve and each will be graded. The grades of students in each section will act as indicators to the effectiveness of each process of drawing trusses. Based solely on the grades of students in each section, we hope to show that Mekanix is a better learning tool for the student while also making grading easier for the instructor.

CONCLUSION

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.

FUTURE WORK

After the extended user study has taken place, we will incorporate the data collected into our application by refining recognizers to allow for more accurate recognition. The truss recognition itself will also greatly benefit from this data. Future work will include a more encompassing weighting algorithm to allow for larger non-specific polygons to be recognized.

We plan to create a web service to process the correctness of sketches, thus removing the correct sketches from the student's local machine. This will prevent the student from accessing the solution from the application and presenting it as their answer.

Our current system fully recognizes truss diagrams and can compare them to stored solutions given by the instructor. Our goal is to allow the instructor to assign open-ended problems where an exact answer is not necessarily the only correct solution. In doing this, Mekanix will need to be able to solve statics related equations and apply them correctly to the truss diagram. The next step is to expand our system to allow general free-body diagrams to be entered and recognized as well.

ACKNOWLEDGMENTS

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, C. (2004) Sketch Recognition User Interfaces: Guidelines for Design and Development. In *Proceedings of AAAI Fall Symposium on Intelligent Pen-based Interfaces*, 2004.
2. Bonwell, C.; Eison, J. (1991). *Active Learning: Creating Excitement in the Classroom AEHE-ERIC Higher Education Report No.1*. Washington, D.C.: Jossey-Bass.
3. Hammond, T., and Davis, R., 2005, LADDER: A Sketching Language for User Interface Developers, Computer and Graphics, Elsevier, pp. 518-532.
4. 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, 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, B., and Hammond, T., 2008, PaleoSketch: Accurate Primitive Sketch Recognition and Beautification, 13th International Conference on Intelligent User Interfaces, pp. 1-10.
7. Piaget, Jean. (1950). *The Psychology of Intelligence*. New York: Routledge.
8. Sutton, M., and Jong, I., 2000, A Truss Analyzer for Enriching the Learning Experience of Students. In *2000 ASEE Annual Conference Proceeding*