Sketch Recognition Algorithms for Comparing Complex and Unpredictable Shapes

Martin Field
Texas A&M University
mfield@cse.tamu.edu

Stephanie Valentine
Saint Mary's
University of Minnesota
slavle07@cse.tamu.edu

Julie Linsey
Texas A&M University
jlinsey@tamu.edu

Tracy Hammond
Texas A&M University
hammond@cse.tamu.edu

Abstract

In an introductory Engineering course with an annual enrollment of over 1000 students, a professor has little option but to rely on multiple choice exams for midterms and finals. Furthermore, the teaching assistants are too overloaded to give detailed feedback on submitted homework assignments. We introduce Mechanix, a computer-assisted tutoring system for engineering students. Mechanix uses recognition of freehand sketches to provide instant, detailed, and formative feedback as the student progresses through each homework assignment, quiz, or exam. Free sketch recognition techniques allow students to solve free-body diagram and static truss problems as if they were using a pen and paper. The same recognition algorithms enable professors to add new unique problems simply by sketching out the correct answer. Mechanix is able to ease the burden of grading so that instructors can assign more free response questions, which provide a better measure of student progress than multiple choice questions do.

1 Introduction

In high-enrollment introductory courses, the professors and teaching assistants (TAs) often do not have the time or resources to give individual attention to every single student. ENGR 111 at Texas A&M University is one such course, with an annual enrollment well over 1000 students. In ENGR 111, students learn introductory statics including free-body diagrams (FBD) and static truss analysis. ENGR 111 is students' opportunity to gain the fundamental concepts and skills that are the foundation for more advanced work in engineering. We developed a software platform called Mechanix that students can use to complete homework assignments, quizzes, and exams. Mechanix applies sketch recognition techniques to interpret student diagrams, compare them to reference solutions provided by the professor or TA, and provide immediate feedback about any errors present in the solution.

In addition to the learning benefits, the cognitive overhead of using Mechanix is minimal. Mechanix allows students to complete problems in almost the exact same way as they would using pen and paper. Students draw the diagram naturally, with very few constraints on their drawing style. As students draw, Mechanix is able to determine which reaction forces or other answers the students will be computing, and provides them with spaces to type in those answers (eg. the magnitude of a reaction force or the factor of safety). Mechanix provides feedback both on the drawn diagram and the computed answers by comparing the student's submission to a reference solution provided by the instructor or TA. Mechanix also saves student submissions and displays them to the instructor or TA along with the same feedback, thus facilitating the grading process.

One of the first steps of any problem is to draw the focus of the diagram: the 'body' in a free-body diagram or the truss in a truss analysis problem. This central component of the diagram is extremely important; almost every other component in the diagram is attached to it or related to it in some way. This paper presents new algorithms for reliably recognizing arbitrarily shaped 'bodies' and complex trusses. The roles of our new algorithms are twofold: identify 'bodies' and trusses, and compare them to the reference solution.

2 Background

Sketch recognition systems typically fall into one of three categories: gesture recognition [Rubine, 1991; Wobbrock *et al.*, 2007], vision-based recognition [Kara and Stahovich, 2005; Miller *et al.*, 2000], and geometric recognition. For non-truss shapes like forces and supports, Mechanix uses a geometric approach very similar to LADDER [Hammond and Davis, 2005], where complex shapes are described as combinations of simpler shapes that abide by geometric constraints. LADDER implements a domain-specific language for specifying the relationships between subcomponents in a shape. We define primitive shapes as shapes that cannot be represented as a combination of simpler shapes. The PaleoSketch recognizer [Paulson and Hammond, 2008] is a purpose-built recognizer for identifying primitive shapes. PaleoSketch supports a very broad range of primitive shapes including lines, arcs, ellipses, spirals, helices, and filled dots. The algorithms we will present use PaleoSketch as a low-level stroke processing step before high-level recognition takes place.

Similar to LADDER, nuSketch [Forbus *et al.*, 2004] is a sketch understanding system focused on the geometric relationships between shapes. The COGSketch [Forbus *et al.*, 2008] system was built on top of nuSketch, and provides some educational facilities for tutoring a user based on the content of their sketch. COGSketch is not a recognition system, because

users are required to label the shapes they draw. However, COGSketch is able to apply spatial reasoning to understand the relationships between shapes and determine if something might be out of place. In order to adequately describe the "correct answer," instructors must first learn a complicated process to manually specify the geometric relationships of each diagram component. Mechanix improves on the COGSketch model by allowing professors and TAs to add new content simply by drawing the correct solution once.

Kara and Stahovich 2005 introduced a vision-based recognizer that combined multiple distance metrics to increase recognition accuracy. The Kara and Stahovich recognizer uses instance learning: incoming sketches are preprocessed, rasterized, and compared to a bank of similarly processed templates. We are especially interested in the comparison step, where the authors use a combination of four distance metrics: Hausdorff distance, modified Hausdorff distance, Tanimoto coefficient, and Yule coefficient. Our 'body' recognition algorithm uses the first three metrics in the same way, though we skip the rasterization step and apply the metrics directly to the stroke data.

Newton's Pen [Lee *et al.*, 2008] is a pen-based tutoring system for statics. Newton's Pen runs on the FLY pentop computer based on the Anoto digital pen and paper technology. Newton's Pen applies vision-based sketch recognition to recognize a simple free-body diagram (such as one focusing on a single node of a truss) and provide constructive feedback about the diagram and the governing equations. The recognition capability of Newton's Pen is limited by the hardware in the FLY pentop computer. Newton's Pen constrains the user to draw free-body diagram components in a very specific order. For example, to specify a force, the user must first draw the force arrow, then label it, then draw a leader line, then draw an arc to denote the internal angle of the force and the leader line, and finally label the angle. If users deviate from the prescribed order, recognition fails. Newton's Pen understands simple, one-node free body diagrams and governing equations, but is not intended to recognize a full truss problem or free-body diagram problem. Our new recognition algorithms allow Mechanix to support more complicated problems that help students learn larger concepts.

The Andes physics tutoring system [Vanlehn *et al.*, 2005] and the Free-Body Diagram Assistant [Roselli *et al.*, 2003] are two existing systems that allow students to create electronic solutions to homework assignments. Both systems were designed as alternatives to pen-and-paper homework assignments to make classroom adoption easy for professors. Andes and the FBD Assistant were both designed to help guide students through the process of solving free-body diagram problems in the domain of physics. Both systems have an interface similar to common diagramming software: students select graphical objects from a tool palette and place them on the screen using the mouse. For example, to place a force, the student would select the force tool from the palette, and click once to place the end of the arrow. Then, the student could move the mouse to position the head of the arrow at the proper angle and distance from the opposite end. A second click locks the head of the arrow in place. Mechanix provides the same feedback and guidance over a very similar subject, but improves on Andes and the FBD Assistant by providing a natural sketch-based interface that allows students to more easily express their thoughts. The algorithms presented in this paper are integral to Mechanix's ability to recognize the sketches drawn by students while they solve FBD or truss analysis problems.

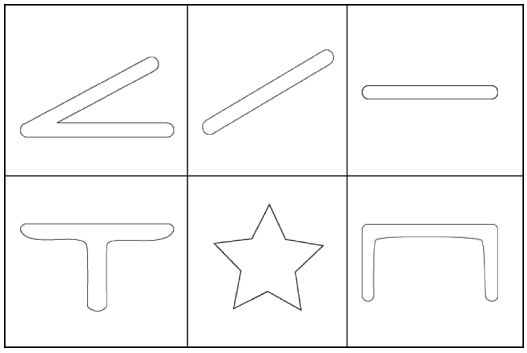


Figure 1: Six 'body' shapes taken from problem statements and used in our user study. Our algorithm can identify any arbitrary 'body' the instructor draws and compare that to the student's drawing.

3 Overview

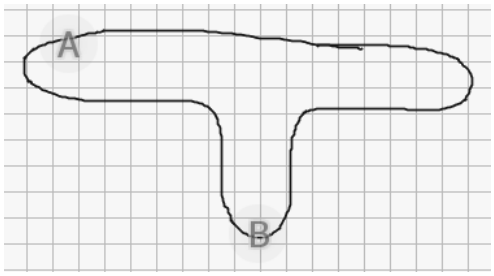
In order for Mechanix to automatically correct a student's solution and provide immediate feedback, the instructor or TA must first provide a reference solution for each problem. The instructor provides the reference solution by drawing it; the same way the students will provide their solutions. Mechanix recognizes each component of the reference solution – the truss or 'body', the input forces, the output forces – and builds an understanding of the solution. Then, when students submit their work, Mechanix uses the same process to recognize each component and build an understanding of the student's submission so that it can be compared directly to the reference solution. Recognizing and comparing trusses and freeform 'bodies' is an integral step toward recognizing and comparing entire diagrams.

Mechanix's recognition process is online: after each stroke is drawn, it attempts to combine that stroke with others into a more complex shape or recognize a new shape from that stroke alone. This process is very similar to [Hammond and Davis, 2005]; it is unnecessary to check any combinations that do not include the new stroke, because those would have already been checked when the last of those strokes was added. For each new stroke, Mechanix considers four possibilities: it could be part of a truss or 'body', a markup stroke, part of a more complex shape, or it may not be able to be recognized.

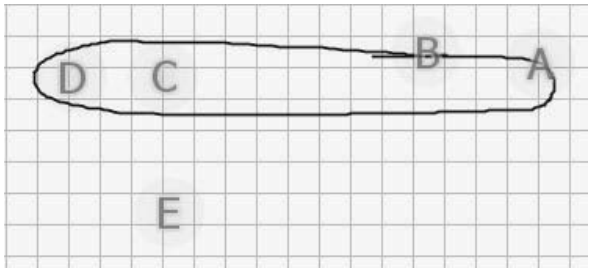
Once a truss or 'body' has been identified, Mechanix skips the first possibility, since it is no longer relevant. The singular nature of this optimization is not required, if there was a problem with multiple trusses or 'bodies', then Mechanix could wait until all of the requirements were satisfied before skipping the first possibility. The following sections focus on truss and 'body' identification and comparison.

4 Free-Body Diagrams

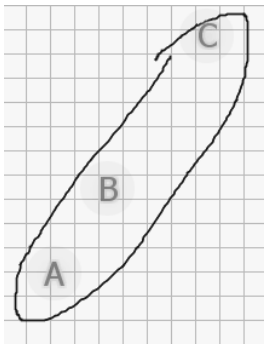
The first subject taught in ENGR 111 is the analysis of input forces and reactions with free-body diagrams. Such analysis focuses on an arbitrary object (eg. an escalator, chair, or



(a) a simple 'body'



(b) note: node E is in the correct location



(c) although it is a simple rotation, this 'body' should not match (b)

Figure 2: Three correctly recognized 'bodies'. A student knows the 'body' was recognized correctly because Mechanix adds instructor-defined nodes to the shape. Nodes are used later in the problem-solving process as attachment points for forces.

box). Recognizing 'bodies' is interesting because there is a large variety of possible shapes that could constitute a 'body', and matching two shapes should be fairly lenient, so that the student's sketch does not need to look exactly like the reference solution (which the student cannot see). We will describe the steps we have taken to allow the student to draw freely but ensure that the general structure of the 'body' is the same as the reference solution.

4.1 Body Identification

We consider any stroke or set of strokes that forms a closed shape as a possible 'body'. To identify closed shapes, we first use PaleoSketch [Paulson and Hammond, 2008] to segment the raw stroke data. To check if a collection of segments forms a closed shape, we pick a segment and pick one of its endpoints to start from. Considering all of the other segments in the collection, we find the closest endpoint. If that endpoint is close enough (within 9% of the total path length of the strokes), then we move on to that segment and consider its other endpoint. 9% is an emiprically-determined threshold that allows students to draw 'bodies' in multiple strokes withouthaving to precisely connect every stroke at the endpoints.We repeat this process until we cannot find an endpoint that is close enough or if we run out of segments. If the last segment's second endpoint is close enough to the first segment's endpoint, then the collection of segments forms a closed shape.

If some subset of the segment collection forms a closed shape, we still reject the original collection. All possible collections of segments will be considered from the largest collection to the smallest, so that subset will eventually be tested. We have tested this brute-force approach extensively with users and have found that the time requirements are reasonable. Since only unrecognized segments are considered, the user must have about 12 (depending on hardware) unrecognized segments before recognition time exceeds the two second timeout. In the few cases where users experienced a recognition timeout, the notification provided a timely warning to those users that their diagrams were not being correctly recognized, which they were not aware of.

4.2 Body Comparison

'Body' comparison is similar in nature to instance-based sketch classification: how do we best compare the user's sketch to the template shape? However, most instance-based sketch classifiers compare the user's sketch to a number of templates to determine a classification, but we only have one template and have to produce a binary *match/no match* classification.

To compare two 'bodies', we apply a combination of the Hausdorff distance, the modified Hausdorff distance, and the Tanimoto coefficient to determine the similarity between the student's sketch and the reference solution. These distance metrics are typically applied to images, so we apply some preprocessing steps to regularize the sketches, though we do not rasterize them. First, we resample the shapes to have 64 evenly-spaced points each using the method from [Wobbrock *et al.*, 2007]. This makes the recognizer robust to drawing speed variations, since digitizers sample at a constant rate slowly-drawn strokes will have more points than quickly-drawn strokes. Next, the shapes are scaled uniformly so that the largest dimension spans 40 pixels and the shape is centered on the origin. Although 40 pixels was chosen arbitrarily, with double-precision subpixel coordinates, the size of the bounding box only serves to constrain the two shapes to the same space and would not represent a significant loss of precision for any bounding box size within a reasonable range.

To calculate the Hausdorff distances between two shapes A and B, we create two distance vectors D_A and D_B such that

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

Where P_A is a vector of the points in A and P_B is a vector of the points in B, and D_B is similarly defined. The Hausdorff distance is the maximum value from D_A and D_B. The modified Hausdorff distance is the average of the averages of D_A

and D_B. More formally:

$$H_d(A,B) = \max(\max(D_A), \max(D_B))$$
$$H_{mod}(A,B) = \frac{\bar{D}_A + \bar{D}_B}{2}$$

Note that our definition of H_{mod} differs from [Kara and Stahovich, 2005].

Because we are not ranking the student's sketch against a collection of templates, but instead comparing it to one template to make a binary *match/no match* decision, we convert the Hausdorff distances into a "match probability" using

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

The value 20 was chosen as a threshold representing half of the bounding box size, since the Hausdorff distances are dependent on the scaling factor. This is not a real probability, as it is possible (and in fact likely) for two vastly different shapes to have a negative "match probability" by either or both of the Hausdorff metrics.

The third metric we apply is a variant of the Tanimoto coefficient, which measures the overlap between two images. Traditionally, the Tanimoto coefficient is defined as

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

where n_{ab} is the number of overlapping pixels, n_A and n_B are the total number of pixels in A and B, respectively. Since we are dealing with points in a sketch (double precision coordinates) instead of pixels in an image (integer coordinates), the concept of overlapping is not as clearly defined. We use a lax definition of overlapping, such that a point from A overlaps with a point from B if the distance between the two points is less than or equal to 4. Hence n_{AB} may be different from n_{BA}. We choose 4 as 10% of the bounding box size. We define our Tanimoto variant as

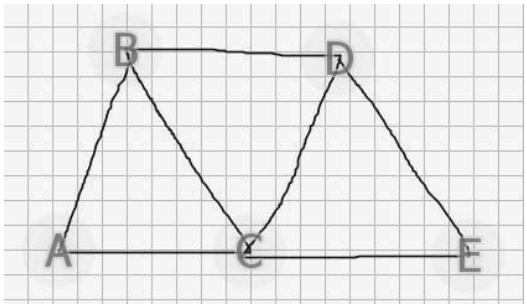
$$T(A,B) = \frac{n_{AB} + n_{BA}}{n_A + n_B}$$

which represents the overall percentage of overlap between two shapes.

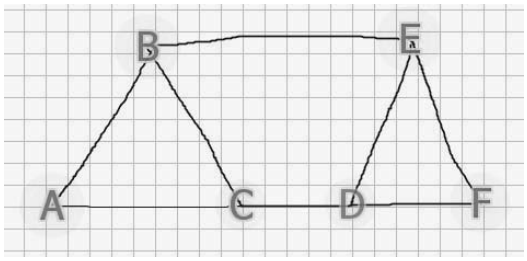
We use the same combination approach as [Kara and Stahovich, 2005]: a simple average of all three metrics. We use 0.65 as our acceptance threshold to determine if the student's shape matches the reference solution.

5 Trusses

Trusses are an important structure for engineers to learn because they have so many applications such as bridges, airplane wings, and buildings (especially supporting roofs). For the purposes of recognition, a truss can be considered as a collection of convex polygons, each of which shares at least one side with another polygon in the truss. Usually, the simplest polygons comprising a truss are all triangles, though squares (without chords) occur in some of the trusses studied by ENGR 111 students. It is also possible for ENGR 111 students to analyze trusses with higher-sided polygons, as long as they are convex, though we have not yet seen any homework assignments that ask them to.



(a)



(b)

Figure 3: Two different trusses that have been correctly recognized by the algorithm presented in this paper, but may confuse other recognizers.The two trusses are composed of the same lines at the same angles – the only difference is whether or not the two internal beams meet in the middle. Our recognizer is sensitive to such details.

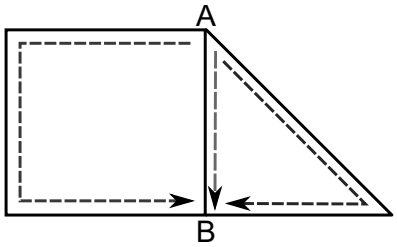
5.1 Truss Identification

In order to identify a truss out of a complete diagram, we utilize the shared-edge property of trusses and the additional property of our domain that no other shapes have such shared edges. The forces, supports, and other shapes that are drawn attached to a truss all connect at a single point, instead of sharing an edge. Thus, if we can find two polygons that share an edge, then we have found a truss. Additionally, any polygon that shares an edge with a truss must also be part of that truss.

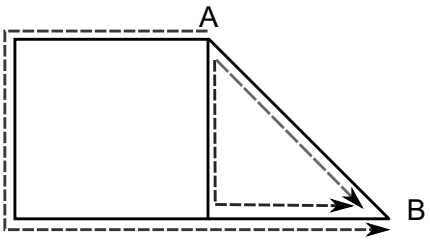
We first use the PaleoSketch algorithm [Paulson and Hammond, 2008] to preprocess each stroke and identify all of the line segments. PaleoSketch will segment a 'polyline' stroke into each individual line segment, and we additionally segment line segments whenever two of them meet, hence all line segments intersect at their endpoints. We use all of the line segments in a diagram to build a graph with each segment as a node, and edges between segments that intersect.

Once we have built the connection graph, we consider each edge AB as a potentially shared edge. We remove the AB edge from the graph and run a breadth-first search (BFS) to search for another path from A to B. If we do not find a path, then AB is not even part of a polygon, and so can't be a shared edge. If we do find a path then AB is part of a polygon, and we remove all of the edges in that path from the graph and use BFS to search for a third path from A to B (AB being the first path). If we do not find a path, then AB is not a shared edge. If we do find a path, then the two paths we found with BFS each represent a polygon sharing the AB edge.

Figure 4a shows a shared edge that will be correctly detected



(a) a shared edge with three distinct paths



(b) a shared edge with only two distinct paths

Figure 4: Two examples of shared edges. Only (a) will be detected by our algorithm.

by our algorithm. If we consider a different vertex as B, as in Figure 4b, then our algorithm will not detect it as a shared edge. The BFS will find the blue path first and remove all of its edges from the graph. At that point, the red path will no longer reach from A to B. We could count the number of simple paths from A to B in one pass with a depth-first search, which would also detect the AB edge in Figure 4b. However, to characterize the structure of a truss, we are more interested in the square and the triangle, and less interested in the right trapezoid. The BFS approach helps us to identify and separate the basic building blocks from which the truss is built.

If we identify one edge as shared, we can use a simple variant of the mark-sweep algorithm. We first mark all of the edges in the two paths we found as 'truss'. Then, any polygons that have an edge inside the marked set can be added to the marked set. Any extraneous symbols, like forces or supports attached the the truss, will have edges adjacent to the marked set but never in the marked set. Once we have considered all other polygons (cycles in the graph), the marked set represents a truss.

5.2 Truss Comparison

When comparing a student's truss to the reference solution, we consider both the sketch and the adjacency graph. It is important that the student's submission has the exact same graphical structure as the reference solution. For example, the trusses in Figure 3 look similar and may be considered the same by many sketch recognizers. However, it is very significant that Figure 3a has 5 nodes and Figure 3b has 6.

In other aspects, it is very important to look at the sketch itself in addition to the adjacency graph. For example, our recognizer is intentionally rotation dependent, since the orientation of the truss and attached forces is important to the solution of the problem. Also, for an asymmetric truss, reflection over any axis will create an isomorphic graph, but the solution to the problem will change because of this new orientation. While it is possible for a student to arrive at the correct solution despite rotating or reflecting the truss, such transformations represent a deviation from the problem statement and are not desirable from an educational perspective (although the algorithms we have developed could easily be relaxed to allow any set of affine transforms while disallowing others).

6 Experiment and Results

Using SOUSA [Paulson *et al.*, 2008], we collected sketches of the shapes in Figure 1 from 17 different users, 180 examples in total. Each user was asked to draw two examples of each shape in a random order, as prompted by SOUSA. Not all participants finished the whole data collection process, but the data is not biased toward any single shape. The six shapes we selected were taken from FBD problem statements and thus represent the kinds of shapes that students will be drawing in Mechanix.

We compared each shape to every other shape and recorded the *match/no match* decision for each comparison. In total, there were 16110 comparisons performed. Out of the 2627 comparisons that should have been matched, we correctly matched 2602 instances, for a recall of 0.991. Out of the 13483 comparisons that should not have matched, we correctly rejected 12321 instances, for a precision of 0.691. This gives us an F-measure of 0.814.

We collected some sketches to test our original (unsuccessful) truss recognition algorithm. We made extensive use of those sketches for the development and tuning of the truss recognition algorithm described in this paper. Although we have not collected any new data since the development of this algorithm, extensive user testing for the general usability of the Mechanix platform indicates that the truss recognition algorithm generalizes well beyond the training data. The same user testing has shown that the precision of the 'body' matching algorithm is sufficient in cases where users intentionally draw a different shape than presented in the problem statement.

We also tested Mechanix in one honors section of ENGR 111. In one class period, the professor gave a demonstration of Mechanix and how to use it to solve truss analysis problems. After the demonstration, students completed a short in-class assignment consisting of two problems. Later that week, students were assigned a homework assignment with six problems and were given the option to complete the same assignment using Mechanix or using pen-and-paper. A majority of the students chose to use Mechanix. We held two focus groups with some of the students from the class to assess their experiences with Mechanix. Every student we interviewed qualified their experience as positive. Although many identified areas needing improvement, recognition was not one of those areas.

7 Discussion

Our 'body' comparison algorithm is heavily biased toward recall. While we might be able to adjust the thresholds to trade a lower recall for increased precision and find a higher F-measure, our application requires an especially high recall, but is not as sensitive to precision. When completing a free-body diagram homework assignment, students will be transcribing the 'body' shape from the problem statement; the goal of our recognizer is to make sure the student has drawn a shape that

is close enough to the reference solution for the student to be able to solve the problem successfully. Based on usability testing, the 'body' comparison algorithm achieves that goal, and very few users are able to intentionally 'fool' the comparison algorithm into accepting a significantly different shape.

The foundation of our 'body' comparison algorithm is a modified version of the template comparison algorithm in [Kara and Stahovich, 2005], which is generally regarded as a highly accurate symbol recognizer. Hence, it may be possible to use our modified comparison algorithm to create a new general-purpose symbol recognizer. Since our comparison algorithm uses stroke points instead of rasterized image pixels, our constants and thresholds could be tuned differently than those used by the Kara and Stahovich recognizer.

Usability testing indicates that both the 'body' and truss algorithms are very robust to different drawing styles and stroke orders. Our algorithms identify complicated shapes and allow students and professors to draw them without any constraints on drawing style. Students have successfully used Mechanix with a Wacom Cintiq monitor, a tablet PC, and a standard optical mouse. Sketching with a mouse still represents an improvement over palette-based systems like Andes and the FBD Assistant, although it is less fun than sketching directly on the screen with a stylus.

Our algorithms have been incorporated into the larger Mechanix system, where they successfully serve two important purposes. First, when an instructor or TA is entering a new problem, identifying a 'body' or a truss indicates whether the problem will be a free-body diagram or a truss analysis problem. Because the algorithms are so successful, the instructor or TA does not have to manually specify what type of problem it will be, they can simply start drawing the reference solution. The second purpose is to identify when the student has drawn the 'body' or truss and to compare that shape to the reference solution. Being able to freely draw the diagram during the problem-solving process lets students focus on learning the engineering concepts instead of software details and builds transferable skills that will benefit students even when they are not using Mechanix.

8 Conclusion

We have presented successful algorithms to identify and compare the 'body' shapes of a free-body diagram or the trusses from static truss analysis problems. It is especially important to recognize 'bodies' and trusses accurately, because they are the focus of every diagram that students draw for ENGR 111; students cannot solve the problem until that shape has been correctly recognized. In our tests, the 'body' comparison algorithm achieved a recall of 0.991 with a precision of 0.691. In user testing, both the 'body' and truss algorithms were found to be successful, and did not hinder students using Mechanix or increase the difficulty of learning the software.

9 Acknowledgments

We would like to thank the other members of the Sketch Recognition Lab and the I-DREEM Lab. This project is supported in part through NSF grants 0935219 and 0942400.

References

[Forbus *et al.*, 2004] K. Forbus, K. Lockwood, M. Klenk, E. Tomai, and J. Usher. Open-domain sketch understanding: The nuSketch approach. In *AAAI Fall Symposium on Making Pen-based Interaction Intelligent and Natural*, pages 58–63. AAAI Press, 2004.

[Forbus *et al.*, 2008] K. Forbus, J. Usher, A. Lovett, K. Lockwood, and J. Wetzel. CogSketch: Open-domain sketch understanding for cognitive science research and for education. In *SBIM '08: Proceedings of the 5th Eurographics workshop on Sketch-based interfaces and modeling*, 2008.

[Hammond and Davis, 2005] T. Hammond and R. Davis. LADDER, a sketching language for user interface developers. *Computers & Graphics*, 29(4):518–532, 2005.

[Kara and Stahovich, 2005] Levent Burak Kara and Thomas F. Stahovich. An image-based, trainable symbol recognizer for hand-drawn sketches. *Computers & Graphics*, 29(4):501–517, 2005.

[Lee *et al.*, 2008] WeeSan Lee, Ruwanee de Silva, Eric J. Peterson, Robert C. Calfee, and Thomas F. Stahovich. Newton's pen: A pen-based tutoring system for statics. *Computers & Graphics*, 32(5):511–524, 2008.

[Miller *et al.*, 2000] E. G. Miller, N. E. Matsakis, and P. A. Viola. Learning from one example through shared densities on transforms. In *Proceedings of the 2000 IEEE Conference on Computer Vision and Pattern Recognition*, pages 464–471, 2000.

[Paulson and Hammond, 2008] B. Paulson and T. Hammond. PaleoSketch: Accurate primitive sketch recognition and beautification. In *Proceedings of the 13th international conference on Intelligent user interfaces*, pages 1–10, 2008.

[Paulson *et al.*, 2008] Brandon Paulson, Aaron Wolin, Joshua Johnston, and Tracy Hammond. SOUSA: Sketch-based Online User Study Applet. In *SBIM '08: Proceedings of the 5th Eurographics workshop on Sketch-based interfaces and modeling*, pages 81–88, 2008.

[Roselli *et al.*, 2003] R. J. Roselli, L. Howard, B. Cinnamon, S. Brophy, P. Norris, M. Rothney, and D. Eggers. 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*, 2003.

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

[Vanlehn *et al.*, 2005] Kurt Vanlehn, Collin Lynch, Kay Schulze, Joel A. Shapiro, Robert Shelby, Linwood Taylor, Don Treacy, Anders Weinstein, and Mary Wintersgill. The Andes physics tutoring system: Lessons learned. *Int. J. Artif. Intell. Ed.*, 15(3):147–204, 2005.

[Wobbrock *et al.*, 2007] Jacob O. Wobbrock, Andrew D. Wilson, and Yang Li. 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*, pages 159–168, New York, NY, USA, 2007. ACM.