2021 ASEE ANNUAL CONFERENCE
Virtual Meeting | July 26–29, 2021 | Pacific Daylight Time
Paper ID #33875

Mechanix: An Intelligent Web Interface for Automatic Grading of Sketched Free-Body Diagrams

Matthew Runyon, Texas A&M University

Matthew Runyon is a PhD student in the Department of Computer Science and Engineering at Texas A&M University. He received his bachelor's degree in Mechanical Engineering at Texas A&M University. He has been working with Dr. Hammond in the Sketch Recognition Lab with research focuses in artificial intelligence, human-computer interaction, and education.

Dr. Vimal Viswanathan, San Jose State University

Dr. Vimal Viswanathan is an associate professor in the Mechanical Engineering Department at San Jose State University. His research interests include design innovation, creativity, design theory, and engineering education.

Dr. Kimberly Grau Talley P.E., Texas State University

Dr. Kimberly G. Talley is an associate professor in the Department of Engineering Technology, Bobcat Made Makerspace Director at Texas State University, and a licensed Professional Engineer. She received her Ph.D. and M.S.E. from the University of Texas at Austin in Structural Engineering. Her undergraduate degrees in History and in Construction Engineering and Management are from North Carolina State University. Dr. Talley teaches courses in the Construction Science and Management and Civil Engineering Technology Programs, and her research focus is in student engagement and retention in engineering and engineering technology education. Contact: talley@txstate.edu

Dr. Tracy Anne Hammond, Texas A&M University

Dr. Hammond is Director of the Texas A&M University Institute for Engineering Education & Innovation and also the chair of the Engineering Education Faculty. She is also Director of the Sketch Recognition Lab and Professor in the Department of Computer Science & Engineering. She is a member of the Center for Population and Aging, the Center for Remote Health Technologies & Systems as well as the Institute for Data Science. Hammond is a PI for over 13 million in funded research, from NSF, DARPA, Google, Microsoft, and others. Hammond holds a Ph.D. in Computer Science and FTO (Finance Technology Option) from the Massachusetts Institute of Technology, and four degrees from Columbia University: an M.S in Anthropology, an M.S. in Computer Science, a B.A. in Mathematics, and a B.S. in Applied Mathematics and Physics. Hammond advised 17 UG theses, 29 MS theses, and 10 Ph.D. dissertations. Hammond is the 2020 recipient of the TEES Faculty Fellows Award and the 2011 recipient of the Charles H. Barclay, Jr. '45 Faculty Fellow Award. Hammond has been featured on the Discovery Channel and other news sources. Hammond is dedicated to diversity and equity, which is reflected in her publications, research, teaching, service, and mentoring. More at http://srl.tamu.edu and http://ieei.tamu.edu.

Dr. Julie S. Linsey, Georgia Institute of Technology

Dr. Julie S. Linsey is an Associate Professor in the George W. Woodruff School of Mechanical Engineering at the Georgia Institute of Technological. Dr. Linsey received her Ph.D. in Mechanical Engineering at The University of Texas. Her research area is design cognition including systematic methods and tools for innovative design with a particular focus on concept generation and design-by-analogy. Her research seeks to understand designers' cognitive processes with the goal of creating better tools and approaches to enhance engineering design. She has authored over 150 technical publications including over forty journal papers, and ten book chapters.

Mechanix: An Intelligent Web Interface for Automatic Grading of Sketched Free Body Diagrams

Abstract

Sketching free body diagrams is an essential skill that students learn in introductory physics and engineering classes; however, university class sizes are growing and often have hundreds of students in a single class. This situation creates a grading challenge for instructors as there is simply not enough time nor resources to provide adequate feedback on every problem. We have developed a web-based application called Mechanix to provide automated real-time feedback on hand-drawn free body diagrams for students. The system is driven by novel sketch recognition algorithms developed for recognizing and comparing trusses, general shapes, and arrows in diagrams. We have discovered students perform as well as paper homework or other online homework systems which only check the final answer through deployment to five universities with 450 students completing homework on the system over the 2018 and 2019 school years. Mechanix has reduced the amount of manual grading required for instructors in those courses while ensuring students can correctly draw the free body diagram.

Keywords: FBD, sketch, statics, dynamics, engineering education.

Introduction

Free body diagrams are an integral part of problem-solving in many fields of engineering. These diagrams are critical in Mechanics and related topics. They are often taught to students in the first physics or statics courses due to the prevalence of free body diagrams in engineering problem-solving. Two popular statics textbooks [1], [2] introduce free body diagrams in early chapters just after explaining vectors and fundamental laws of physics. Rosengrant et al. showed that students will use free-body diagrams even when they do not receive credit for drawing them and that students who can correctly draw diagrams are more successful in solving problems correctly [3]. Furthermore, Sweller showed that using the visual aid of a free body diagram helps reduce the problem's cognitive load and increase learning effectiveness [4].

Understanding and drawing free-body diagrams accurately is an essential aspect of a student's success in engineering. Students should have thorough feedback [5] about free body diagrams in entry classes so that they do not form misconceptions and unnecessarily struggle with future classes; however, many university courses can be quite large, especially in the introductory courses such as statics and dynamics. The increased number of students results in an increased number of homework submissions for an instructor or teaching assistant to grade, which carries a considerable time burden. As a result, students may not receive detailed feedback promptly or receive a minimal amount of feedback. This has led many instructors to use web-based homework systems, allowing students to immediately know if their answers are correct while decreasing the

grading burden for the instructor and assistants. This automated feedback can help reduce students' reliance on the instructor's feedback [6].

Unfortunately, the move towards online homework systems often diminishes instructors' ability to provide meaningful feedback to students on their diagrams since the typical software only checks the final answers and does not require diagrams. Some instructors may require students to turn in their hand-drawn diagrams for feedback. This feedback helps students identify and correct their misconceptions about a topic [7], but often instructors do not have the resources required to provide valuable feedback on time and must resort to checking only for completion.

We propose a solution to the feedback and time constraint problems in the form of a web application called Mechanix that utilizes novel sketch-recognition algorithms to assess a sketched free body diagram and automatically grade students' responses to homework questions. This is the modern iteration of our previous system with the same name [8]. We recreated the system as a web application to support as many students and devices as possible. The interface has been improved to help guide students while remaining unobtrusive. We designed new algorithms for recognizing trusses and curved arrows. The new truss algorithm gives us more flexibility in what kinds of trusses we can accept while curved arrows expand the number of problems we can support to include those with applied moments. The solver has been improved to allow equilibrium equations in terms of variables and flexible input of units rather than using a dropdown. The improved version provides a closer-to paper and pen experience to the students by reducing input latency. It provides step-by-step guidance to the students to teach them the general solution procedure rather than steps specific to the problem like our previous system provided. The new version of the software also makes the drawing of FBD possible on the system with minimum practice using a stylus, mouse, or finger on a tablet. Mechanix has been implemented for five semesters at five universities, with almost 500 students completing homework on the platform at that time.

Related Work

Existing Statics Tutoring Systems

There are systems that help in physics tutoring, such as the Andes physics tutoring system [9] and the Free-Body Diagram Assistant [10]. The Conceptual Helper [11] was designed for helping students with qualitative physics problems. These types of problems may involve diagrams but do not typically involve numbers. They can be conceptually more difficult, especially if students cannot visualize how the system might act. These systems are alternatives to pen-and-paper homework using interfaces and interactions similar to those found in computer-aided design (CAD) software. These systems allow for easier interpretation of diagrams by the computer, but students might not benefit from drawing their diagrams by hand, such as increased learning retention.

There are also several online homework systems such as Mastering Engineering, WileyPlus, and McGraw-Hill Connect. These systems are commercial products offered by three major textbook publishers. Some problems may require algebraic comparisons, which have been

implemented by some prior systems [12], [13]. These systems typically require the final answer to a problem but do not require any diagrams. This situation can result in students struggling with some concepts if they do not draw their free body diagram on their own since it is not required. Some of these platforms offer free-body diagram interfaces, but the feedback on the students' sketches is minimal.

Sketch-based Systems

Sketch recognition systems have been used to recognize different types of sketched gestures and diagrams in many areas. The seminal work in gesture recognition was produced by Rubine [14], in which he defined 13 crucial features. These have been expanded to primitive shape recognition systems such as PaleoSketch [15], which recognizes low-level shapes such as lines, arcs, and circles. These low-level shapes can be combined using geometric recognition systems such as LADDER [16], making sense of more complex shapes by breaking them down into combinations of primitive shapes. LADDER uses domain-specific knowledge to define how the components of shapes are combined and related to each other. There are also uses of geometric recognition in various domains, such as circuit diagrams [17]. There are other domain-specific systems such as nuSketch [18], which focuses on the geometric relationships between shapes such as "above" or "below." This was extended to COGSketch [19], which has users label their diagrams with domain-specific labels and then uses them combined with geometric placements of the shapes to provide feedback. These geometric building blocks proved useful in the creation of the system in this paper.

There are systems such as Newton's Pen [20] and our previous version of Mechanix [8], which utilize early digital stylus computers to allow students to sketch diagrams and equations in physics problems while providing some feedback about their diagrams; however, these systems rely on older technologies that can limit the systems and make system usage more difficult. Tools like Newton's Pen II have sought to overcome the original Newton's Pen's technical limitations with broader support and capability [21]. Another interesting system for physics-based problems is Physics Book [22] which focuses more on animating physical systems and problems with springs and pulleys. Our system is focused on free body diagrams of general shapes and trusses.

Description of Mechanix

One of the biggest hurdles we faced with our previous implementation was that the application was a Java download. This situation caused many problems with the system administrators at various schools because they were reluctant to install our application on their devices. This reluctance limited the number of classrooms and, therefore instructors, that could adopt the software. Additionally, early tablets were unable to utilize Mechanix since they could not run Java. We decided a web application would be the best platform for our new version because it would be easier for everybody to access and not interfere with any institutional IT policies.

Interface

Figure 1presents the problem solving interface of Mechanix. The design of this interface focuses on the two main areas of the *side panel* and the *sketch surface*.

Side Panel

The side panel contains many interactions for students. We chose a side panel for these actions because it allowed the remaining area, which is used for the sketch surface, to be more square in standard landscape usage. The side panel is highlighted with labels A and B in Figure 1. Students can input symbolic or numeric answers, and generalized feedback if given for certain mistakes. Compared to our previous version, we reduced the detail of the scaffolding to try to focus on the problem solving process rather than the steps to complete a specific problem.

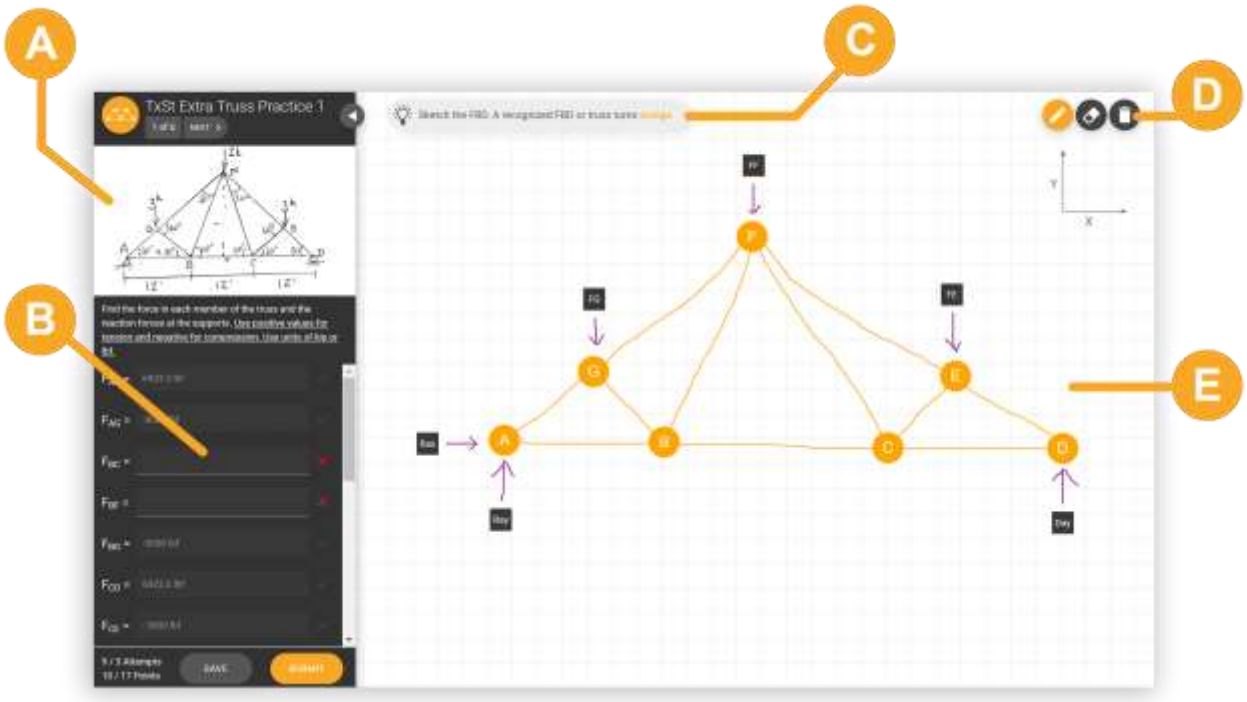

Figure 1The interface for solving free body diagrams. (A) Problem image and description. (B) Equations that must be solved and solutions entered. (C) Instructions on diagram sketching. (D) Sketch, Erase and Clear tools. (E) The sketching canvas.

Sketch Surface

The remainder of the screen after the side panel is dedicated to the sketching surface. Students are provided a grid background to aid with sketching their diagrams as well as the coordinate axis for the problem. The sketching tools are relatively basic to simplify using Mechanix and focus on drawing the correct diagrams. The tools are draw, erase, and trash, as shown by label D in Figure 1.

The draw tool tracks the student's pen or mouse and places digital ink on the page where it travels. It can also capture pressure and tilt if the input device supports it via the browser Pointer Events. Students can use a mouse or trackpad if necessary by holding the primary button while moving the mouse. Once specific shapes are recognized, the strokes turn from gray to orange or purple to indicate recognition. Also, the instructions shown with label C in Figure 1are updated.

Erasing provides the students with a new cursor to help indicate they are in erase mode. The act of erasing is performed on the entire stroke rather than individual pixels. This makes it quicker for students to erase erroneous or unrecognized strokes. When the student hovers over a stroke in erase mode, it is highlighted red to indicate which stroke will be erased. This interaction is less clear with a pen-enabled device as most pens do not support hover interactions very well. Since the free-body rarely needs to be redrawn, it is not erasable after it has been recognized. This feature is to help prevent students from accidentally erasing their entire body when trying to erase an arrow that may intersect the body.

The current iteration of Mechanix was initially designed to facilitate truss analysis problems. Because the sketch surface recognizes only one free body diagram, the global system (full truss) is sketched in Mechanix and any local systems (joint or section free body diagrams) are sketched on a student's own paper. As such, the truss problems can be specified to be solved with the Method of Joints or the Method of Sections in the instructor directions.

Mechanix also facilitates solving free body diagram, or general body, problems. The most testing has occurred with free body diagrams problems that only have point loads acting upon the body. These point loads can be along the x- and y-axes as well as along any diagonal. We have recently developed the feature to recognize distributed loads and applied moments to increase the types of problems that can be assigned in Mechanix. Figure 2 shows an example of a general body problem which involves diagonal forces, applied moments, variable answers, and numeric answers.

Due to the flexibility of the general body recognition, we have adapted Mechanix to automatically grade some rigid body statics problems. The current requirements are that the bodies must be drawn relative to their position in the image, but Mechanix is able to facilitate simpler problems such as 2 bodies linked together on an inclined plane.

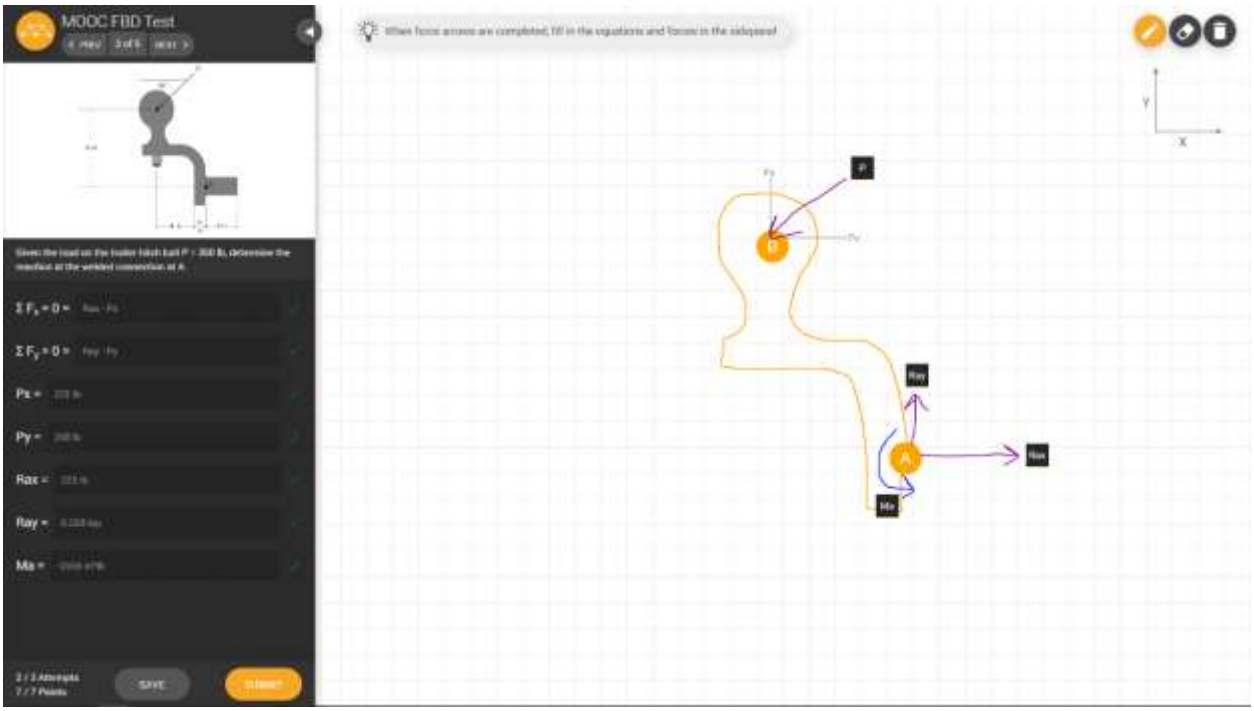

Figure 2An example of a general free body submission showing the flexibility of answer and force inputs.

Problem Solving Experience

Students must draw their free body diagram in order to receive credit in the system. A free body diagram is an illustration used to visualize how forces and reactions occur on a given body. A free body diagram should consist of the following:

1. A simplified version of the important portion of the system (the body)
2. Forces drawn as arrows pointing in the direction of the force on the body
3. A coordinate system

Since the system provides a coordinate system, the students only need to draw the body and forces. The students start by sketching the body in their diagram, as shown by label E in Figure 1. In general shape problems, where the point of the problem is to have student practice accurately sketching free body diagrams, the students have a faded underlay of the problem image on the sketch surface to trace. This image helps the students who want to make sure their diagram looks similar to the image while also helping the template matching recognition be a little more robust. After the student sketch matches the instructor's version, their sketch turns orange with nodes matching the instructor's diagram, the background image is removed, and the students can proceed with sketching forces.

In problems that involve a truss, the unlabeled nodes of the truss are shown after each pen-up interaction (whenever the stylus leaves the surface or the mouse/track pad button is released). The appearance of these unlabeled nodes let the students know how the system is perceiving their sketch. Once the student's diagram is matched to the instructor's diagram, it turns orange, and the nodes are labeled according to the instructor's diagram.

Students then draw any number of required forces on the diagram connected to nodes. Once an arrow is recognized, it changes colors to purple and provides a text box near the portion of the arrow furthest from the node so that the student may name the force. An answer box in the side panel is created for each uniquely named force. Students are allowed to draw the forces in any direction. If they draw a diagonal force, the force's x and y components are added to the diagram with gray dashed lines to simplify the student interaction of needing to specify the angle of their force, what axis the angle is based on, and if the angle is degrees or radians. Two answer boxes also appear in the side panel, requesting the x- and y-components of the diagonal force rather than the diagonal force itself. The appearance of the component arrows and the two answer boxes also reinforces to the student that any diagonal force can be broken down into its components.

After sketching their free body diagram, the students move on to solving the problem. This is the final step and consists of students providing answers in text boxes as shown by label B in Figure 1. One unique feature of the system is that we can ask students for the equations of static equilibrium in terms of their sketched diagram. Since the system understands where and in what direction forces were drawn by the student, these equations can be compared with the instructor's answer. Students must type their units in the answer box and freely convert between equivalent units such as newtons and kilonewtons. Providing a separate box for units was avoided explicitly as the box would remind students to add units rather than the student needing to remember the importance of units on their own.

The students can then submit their answers for grading and receive feedback about their answers. There is additional feedback for several types of errors, including missing forces, extraneous forces, missing units, the wrong type of units, and flipped positive/negative signs. More detailed feedback can be added as the system matures; however, the instructors agreed that the feedback must not be too specific. As learned from the previous version of Mechanix, students given too much information (e.g., telling them they are missing a force in the x-direction at node A) will not learn to solve the problems but will just follow the prompts. This level of feedback will lead to high homework grades, but the student will still struggle on exams because the precise feedback is not provided by a test.

Creative Design Mode

There is one additional mode of the problem available in Mechanix called creative design. Students are given constraints, such as bridge length and maximum capacity of a truss member, in this problem type and asked to build a truss bridge that satisfies the constraints and supports the largest load possible. The interface is similar to the other problem types with the addition of students

labeling the angles between their beams and their beams' lengths. If enough values are provided to distinctly determine all lengths and angles within a triangle, the remaining boxes are automatically filled to help the student. A more detailed explanation of this mode as well as an analysis of student feedback can be found in [23].

Use of AI Technology

Mechanix must be able to recognize the sketched free body diagrams of students as well as determine if their answers are correct or incorrect. These tasks are non-trivial given the messy nature of sketched data and the fact that there are multiple acceptable answers for a single problem. The sketch recognition occurs in two main parts, which are described in the following sections.

Sketch Segmentation

The first step of recognition is to segment the stroke created between a single pen-down and subsequent pen-up action. This step involves breaking the stroke into its separate substroke components. In this application, we split strokes based on corners using the Shortstraw corner finding method [24]. This method provides the locations of the corners within the stroke. Including the first and last points as the first and last corner, the points between each pair of corners is then converted into a substroke.

For each substroke created, we check if it intersects any other substrokes within a threshold of 30 pixels (the size of the displayed nodes) Euclidean distance. If two endpoints intersect, such as in Figure 3a, then no additional steps are taken. If one substroke intersects the middle section of another, such as in Figure 3b, then the intersected substroke is split in two, resulting in three substrokes. If the substrokes cross each other, such as in Figure 3c, each is split, resulting in four substrokes.

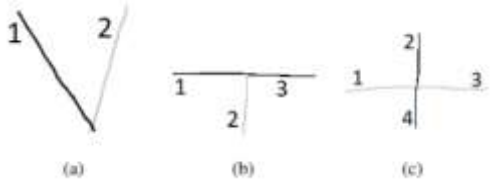

Figure 3Three cases of substrokes intersecting

Shape Recognition

After the stroke has been segmented, the unrecognized substrokes are checked to see if they are any of the expected shapes. Currently, the system recognizes trusses, general free body shapes, and arrows. A problem consists of either a truss with arrows or a general shape with arrows. Since there is a rigidly defined order of sketching the diagrams, recognition is checked in this order. This

helps prevent some false recognition, such as a subset of a truss being seen as a large arrow before the truss is finished.

After the truss or general shape has been recognized, arrow recognition is preformed using an arrow recognition algorithm based on Hammond's Tahuti recognizer [16]. If a shape is recognized, the substrokes are grouped and removed from the unrecognized set. If no shape is recognized, the unrecognized substrokes are maintained and added to the substrokes generated from the next stroke, at which point recognition occurs for the larger set of unrecognized substrokes. By focusing on only the shapes that can be drawn at each stage of the problem, the recognition becomes more robust as there is no need to check both types of shapes and risk false recognition. The two-step recognition solution is simpler than a solution where every stroke causes recognition to run over all existing strokes and shapes with a hierarchy of shape precedence. This situation also prevents any possibility of the system recognizing some portion as a more complex shape and confusing the student as to why their recognized shapes suddenly changed.

Truss Recognition

The definition of a truss is extremely broad and more complicated than just a structure made of only triangles; any structure which has only two-force members (i.e., along the axis of the member) is considered a truss. This allows for trusses containing arcs or squares (e.g., queen post truss). Mechanix focuses on a specific subset of trusses called planar trusses, which occur in a two-dimensional space. Although trusses can contain non-triangles, the problems explored in Mechanix and many introductory courses all contain only triangles, so we focus on recognizing trusses containing only triangles.

For the purposes of this project, we define a truss to have the following properties:

1. A connected graph
2. Made up of only triangles
3. Each triangle shares at least one side with another

Trusses are recognized in the following four steps:

Graph conversion: The first step in recognizing a truss is to convert the substrokes into a graph. Each substroke end is a possible node, and the substroke itself is an edge in the graph. If the ends are within a threshold Euclidean distance of 30 pixels from an existing node in the graph, then the substroke is considered to have one end connected to the existing node. This threshold works well with modern devices but can easily be adapted to be resolution-independent.

Connected graph check: After the substrokes have been converted to a graph, the next step is to check that the graph is connected. This check is achieved through a standard depth-first search starting from any node while keeping track of the nodes visited. Once the search completes, if the visited node set does not contain all of the nodes in the graph, then there is some disjoint portion

of the graph since a connected graph will have a path between any pair of nodes. If there are any disjointed sections, then the sketch cannot be a truss because trusses are connected by definition.

Triangle check: The third step is to check that a path of length 3 exists for traveling from each node to itself. This check ensures that each node is a component of a triangle. The search is done via a breadth-first search that tracks the number of steps taken from the beginning node to the current node. Whenever this value is three, if the node matches the starting node, then the path is a triangle. We keep track of distinct triangles based on their nodes. For example, in Figure 5, the triangle containing nodes A, B, and D would be stored as triangle ABD with edges AB, AD, and BD.

Adjacent triangle check: The final step is to check that all triangles share an edge with at least one other triangle. This check prevents sketches that may be composed entirely of triangles but are not trusses, such as in Figure 4. This determination is achieved by counting the occurrence of edges in each distinct triangle. After counting the number of times each edge occurs, each distinct triangle is checked to confirm at least one of its edges occurs in two triangles.

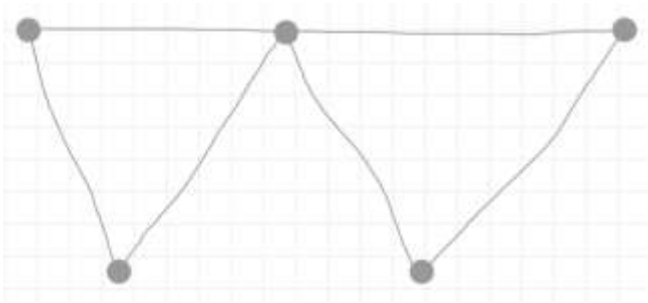

Figure 4Example of a sketch for which all nodes have a triangle path leading back to them, but the sketch is not a truss since there are triangles that share no sides with another triangle.

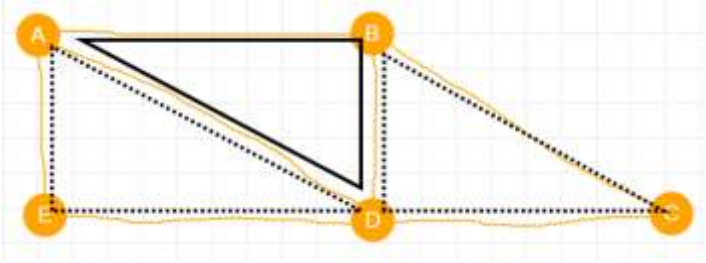

Figure 5A truss showing how all triangles must share a side with at least one other triangle. Both dotted triangles share a side with the solid line triangle.

Truss Comparison

In order to check if a truss matches the expected truss for a problem, we must compare it to the solution drawn by the instructor. In order for the trusses to match, the graph representations must have all of the same nodes and edges, which means the adjacency lists must match. First, we

convert the solution sketch and submitted the sketch to two separate graphs. We can then immediately say the graphs do not match if they have a different number of nodes. After this simple check, we need a way to label the nodes to identify if the adjacency lists match. Graphs could have the same adjacency lists but not be the same graph depending on how the nodes are labeled, so the labeling must be consistent based on the rotation of the truss. For example, the trusses in Figure 6 have the same adjacency lists but are not the same truss since one is rotated 180 degrees.

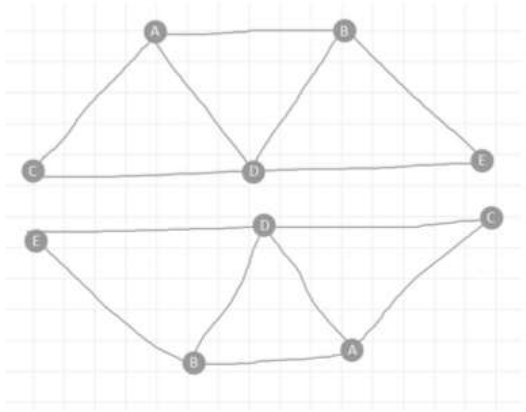

Figure 6Two trusses with the same adjacency lists that are not the same truss.

In order to confirm the trusses are the same and account for rotation, we sort each graph's nodes by their x-y coordinates. The sort orders the nodes such that those with higher y-coordinates come first (towards the top of the page). If the value difference between two nodes is within a threshold, then the node with the lower x-coordinate (towards the left of the page) comes first. We use a threshold of 20 pixels for the sort, which makes the maximum distance between two nodes 28 pixels to be considered the same node. No two nodes should be within the threshold in both directions since the maximum distance for them to be considered the same is lower than the distance used to combine nodes when segmenting. This sort ensures that the truss is a similar rotation to the answer truss because the nodes will be sorted in the same order as long as the truss is not too skewed. A graphical representation of the sort is shown in Figure 7, where the node labels correspond to their position after sorting.

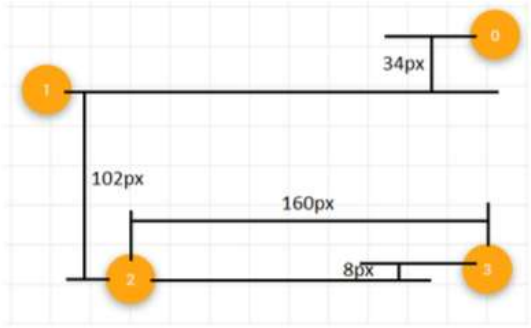

Figure 7Example of how nodes are sorted by y-value and then x-value with an overlap threshold of 20 pixels. Node labels correspond to the position after sorting.

Once both trusses are converted to graphs with sorted nodes, the nodes are indexed starting from zero to provide a consistent representation between the two graphs. Now, each node is compared to check that the adjacency lists match exactly between the two graphs. In the truss in Figure 5, node A would be called 0, and it would be adjacent to nodes B, E, and D , or 1, 2, and 3,, respectively. If the adjacency list matches for each node in the graph, then the trusses must match. Note that there may be some cases where trusses may be slightly different such as if node A in Figure 5was shifted to be directly over node B. This is a future work to check the angles of the edges in the graph to have a stronger confirmation that the trusses match.

General Shapes

General shape free-body diagrams can be any shape that the instructor wants. As a result, there is not really any recognition that is done to confirm the shape is a general shape. While we could detect closed shapes as possible free-body diagrams, the free-body shape does not have to be a closed shape, so this could limit the types of free-body diagrams supported by Mechanix. Instead, we simply compare the shape drawn by the student to that of the instructor.

General Shape Comparison

In order to compare the shapes drawn by students and the instructor, we utilized the $P gesture recognizer [25] with only the instructor answer as the template for matching. This situation posed some problems for students drawing with vastly different aspect ratios or proportions than the instructor's diagram. In order to help the students, we added an underlay of the image from which the free-body diagram was drawn. This background image allows the students to sketch a free body with less error since the aspect ratio of the diagram is important to accurately visualize the problem. We set a confidence threshold of 0.7 based on testing for the diagrams to be considered matching.

Once the diagrams are considered a match, the nodes are placed on the diagram from what the instructor drew in their solution. There can be extra nodes in order to make the students think about where they need to place forces instead of students simply adding forces to all nodes that

appear. The nodes are placed based on their position within the bounding box for the answer sketch.

Answer Comparison

The answers for the given and drawn answers can pose many problems. Since each arrow can be drawn in either direction, the number of answer permutations is 2n for n arrows. Rather than have an instructor input all permutations, the instructor must only input one correct permutation, and all others can be checked against via the answer checking algorithm. Another problem is that students may label their arrows anything they want (within our restriction of requiring a letter as the first character). This labeling affects checking static equilibrium equations as these equations are meant to be answered in terms of the variable names of the arrows.

The first step to checking if the answer is correct is to determine what forces are equivalent on the student's submission and the instructor's answer. Any arrows that are on the same axis and intersect the same node must be equivalent arrows in the submission and answer. In the event of multiple arrows in the same axis at a node, both combinations of potential name swaps are considered. Now we know what arrows are equivalent to each other in the submission and answer, as well as if there are any missing or extra arrows at nodes from the submission.

Once the equivalent arrows are found, their answer values can be compared. Since we know the arrows are on the same axis and intersect the same node, the only difference between them could be their direction. If the student draws the arrow in the opposite direction of the answer diagram but submits the negative value of the answer diagram, then the student's answer is also correct. We simply multiply the submission value by -1 if the arrows point in opposite directions. The values are then checked for equivalence. We allow a 3% margin of error for the answers to account for rounding differences. Units are checked using the MathJS library unless the answer is zero, in which case the units are optional. We also check and attach applicable feedback at this point if the student has the correct answer with the wrong sign by comparing the negative of their submission to the answer.

After the arrow values are checked, the given equation values are checked. For answers which contain variable names, such as the equations of static equilibrium, the variable names must be converted based on the equivalent arrow names in the submission and answer. A regular expression is used to replace instances of the submission arrow names in the answer equation and vice-versa. If the arrows were drawn in different directions in the submission and answer diagram, then a negative sign is added during the replacement. We check both the submission against the answer and the answer against the submission due to the possible algebraic combination of terms. For example, a problem may have two applied forces, both called Q, but the instructor may decide to label them Q1, Q2. The student may submit both forces labeled Q, and neither the instructor answer nor student submission are wrong. If just the submission is converted to the instructor variable space, then the resulting equation is as follows.

$$\Sigma F_y = 0 = 2Q \Rightarrow \Sigma F_y = 0 = 2Q1$$

This conversion does not yield the answer in the instructor's variable space due to ambiguity in the equivalent name across the instructor and student variable spaces. The instructor answer converted to the student's variable space is as follows

$$\Sigma Fy = 0 = Q1 + Q2 \Rightarrow \Sigma Fy = 0 = Q + Q$$

This conversion yields the same equation as the student's answer and thus should be counted correct.

Application Use and Payoff

Mechanix has been deployed for the past five semesters in 5 different universities in the United States. Just over 450 students have used Mechanix to complete at least one homework assignment. The assignments consist of general shape free-body diagram problems, various types of truss analysis, and creative design problems.

The students completed these homework assignments with relatively few issues, and the issues that were identified were able to be fixed quickly. Since Mechanix is a web application, any changes or fixes could be pushed in an instant instead of the student needing to download an update. After the first few semesters, there have been very few bug reports that did not turn out to be user errors.

We have conducted focus groups with students at some of the universities using Mechanix, and overall their thoughts on using Mechanix were positive, with many students noting that they liked the simplicity of our interface compared to their other online homework systems. The instructors also noted they like how Mechanix provides feedback about the student diagram and requires students to add units on their own. One instructor said it seemed like the students who used Mechanix were less likely to forget their units on an exam taken soon after the homework completed on Mechanix.

From analyzing the homework scores and other metrics, we found that students performed just as well on Mechanix as they did with other homework systems [26]. Figure 8 shows the comparison of the performance of students in two groups at one of the participating schools. The Experimental group used Mechanix to solve their problems in a Statics course while the Control group used their regular online homework system. Both systems allowed three attempts to get the correct answer for the problems. It is observed that both groups perform at the same level for the FBD problem while the Mechanix group performs better on the problem on solving a truss. The previous iteration of the software also shows similar results [8]; however, the previous version allowed for unlimited graded attempts and had detailed help about every component that should exist in the diagram. In the new version, the step-by-step instructions were removed to prevent the students from merely following an in-depth guide on solving the problem on the screen instead of figuring it out on their own. Also, the number of graded submissions was reduced to three since all the instructors allowed

three attempts on their other online homework systems. Due to these changes, it is a good sign that students perform as well as other online homework systems. None of the other systems require or grade the free body diagram, but it is part of the assignment for Mechanix, so the experience is closer to a traditional paper and pencil assignment than other online systems.

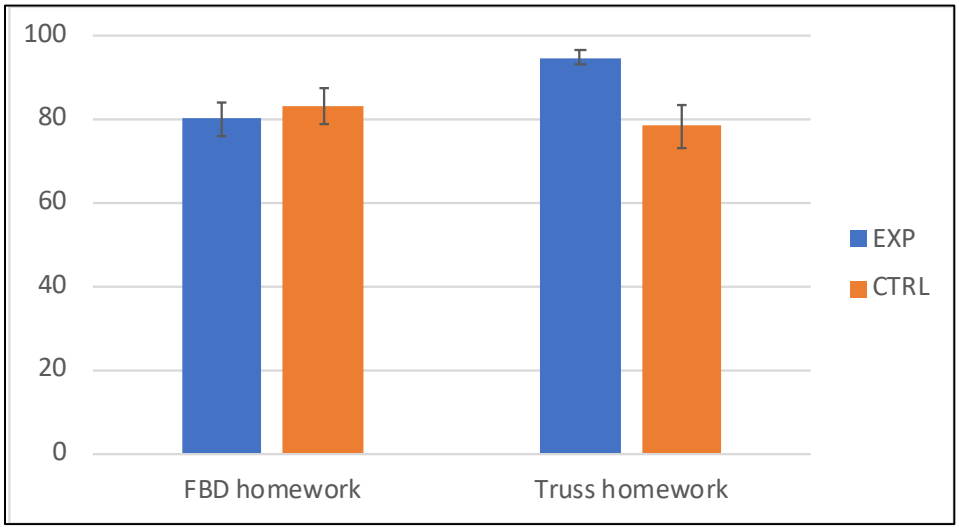

Figure 8Comparison of average homework scores in two homework assignments in one of the participating schools. The Experimental group used Mechanix and the Control group used the online homework system that they regularly use.

One additional metric we noticed was that 38.6% of students who did not get full credit by the third attempt continued trying the problem. Of those who submitted more attempts, 39.2% eventually got the problem completely correct (not for credit). This is just one way that an online system with automatic feedback can provide more help than an instructor would be able to on homework since the students were able to continue checking their answer immediately instead of just looking at the solution after getting it wrong. We used these measurements instead of a time-based measurement because it is difficult to accurately track the actual time spent on a problem by each student. We have no way of reliably differentiating between a student taking a 10 minute break and a student working out the math of a problem for 10 minutes. Further, some students simply submitted 1 incorrect answer and never attempted to correct their mistakes.

Overall, Mechanix has graded over 3000 attempted problems by students with over 10000 total attempts. The amount of time for an instructor or teaching assistant to grade 10000 attempts while providing feedback that Mechanix can is around 100 hours if adequate feedback can be generated and portrayed to the student in just 30 seconds (which is a low estimate for the feedback given). In addition to the time savings for instructors, students receive their feedback immediately, which helps their learning.

Conclusion and Future Work

In this paper, we presented Mechanix, an intelligent web application that is capable of automatically grading hand-drawn free body diagrams for an introductory statics course using a combination of existing and novel sketch recognition algorithms. Mechanix is currently deployed to 5 universities and has been in use to complete homework assignments for five semesters by 450 students. Over 10000 problem submissions have been graded by Mechanix in this deployment. Future work for the project includes determining recognition thresholds based on device type and screen size, further improving the types of feedback we can give, and creating new instructor tools to better analyze their students' performance. Mechanix can be used to reduce the grading burden on instructors while requiring students to draw free body diagrams. Students perform just as well as in other homework systems where they are not required to draw the diagram. Additionally, 38.6% of students continued trying after exhausting their graded attempts, with 39.2% of those eventually getting the problem correct. Our deployment has been successful so far with few hiccups, and we hope to be able to grade more problem types while providing good feedback to students going forward. Currently, Mechanix can also accommodate dynamics problems and problems involving multiple rigid bodies. These problems are being tested and the results will be published in our future papers.

Acknowledgments

We would like to acknowledge the NSF for their support via grants 1726306, 1725423, 1725659, 1726047, and 1725785 as well as our other collaborators Dr. Ben Caldwell and Dr. Kristi Shryock for their help with this project.

References

[1] F. P. Beer, E. R. Johnston Jr, D. F. Mazurek, P. J. Cornwell, E. R. Eisenberg, and S. Sanghi, *Vector mechanics for engineers* vol. 1: Tata McGraw-Hill Education, 1977.

[2] R. C. Hibbeler and R. C. Hibbeler, *Engineering mechanics: statics & dynamics*: Pearson Education India, 2007.

[3] D. Rosengrant, A. Van Heuvelen, and E. Etkina, "Do students use and understand free-body diagrams?," *Physical Review Special Topics-Physics Education Research,* vol. 5, p. 010108, 2009.

[4] J. Sweller, "Cognitive load theory, learning difficulty, and instructional design," *Learning and instruction,* vol. 4, pp. 295-312, 1994.

[5] A. A. Lipnevich and J. K. Smith, "“I really need feedback to learn:” students' perspectives on the effectiveness of the differential feedback messages," *Educational Assessment, Evaluation and Accountability,* vol. 21, p. 347, 2009.

[6] E. Odekirk-Hash and J. L. Zachary, "Automated feedback on programs means students need less help from teachers," in *Proceedings of the thirty-second SIGCSE technical symposium on Computer Science Education*, 2001, pp. 55-59.

[7] R. L. Bangert-Drowns, C.-L. C. Kulik, J. A. Kulik, and M. Morgan, "The instructional effect of feedback in test-like events," *Review of educational research,* vol. 61, pp. 213-238, 1991.

[8] S. Valentine, F. Vides, G. Lucchese, D. Turner, H.-h. Kim, W. Li*, et al.*, "Mechanix: A sketch-based tutoring system for statics courses," in *Twenty-Fourth IAAI Conference*, 2012.

[9] K. VanLehn, C. Lynch, K. Schulze, J. A. Shapiro, R. Shelby, L. Taylor*, et al.*, "The Andes physics tutoring system: Lessons learned," *International Journal of Artificial Intelligence in Education,* vol. 15, pp. 147-204, 2005.

[10] R. J. Roselli, L. Howard, and S. Brophy, "A computer-based free body diagram assistant," *Computer Applications in Engineering Education,* vol. 14, pp. 281-290, 2006.

[11] P. L. Albacete and K. VanLehn, "The Conceptual Helper: An intelligent tutoring system for teaching fundamental physics concepts," in *International Conference on Intelligent Tutoring Systems*, 2000, pp. 564-573.

[12] A. S. Gertner, "Providing feedback to equation entries in an intelligent tutoring system for Physics," in *International Conference on Intelligent Tutoring Systems*, 1998, pp. 254-263.

[13] J. A. Shapiro, "An algebra subsystem for diagnosing students' input in a physics tutoring system," *International Journal of Artificial Intelligence in Education,* vol. 15, pp. 205-228, 2005.

[14] D. Rubine, "Specifying gestures by example," *ACM SIGGRAPH computer graphics,* vol. 25, pp. 329-337, 1991.

[15] B. Paulson and T. Hammond, "Paleosketch: accurate primitive sketch recognition and beautification," in *Proceedings of the 13th international conference on Intelligent user interfaces*, 2008, pp. 1-10.

[16] T. Hammond and R. Davis, "LADDER, a sketching language for user interface developers," in *ACM SIGGRAPH 2007 courses*, ed, 2007, pp. 35-es.

[17] C. Alvarado and R. Davis, "SketchREAD: a multi-domain sketch recognition engine," in *ACM SIGGRAPH 2007 courses*, ed, 2007, pp. 34-es.

[18] 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*, 2004, pp. 58-63.

[19] K. Forbus, J. Usher, A. Lovett, K. Lockwood, and J. Wetzel, "CogSketch: Sketch understanding for cognitive science research and for education," *Topics in Cognitive Science,* vol. 3, pp. 648-666, 2011.

[20] W. Lee, R. de Silva, E. J. Peterson, R. C. Calfee, and T. F. Stahovich, "Newton's Pen: A pen-based tutoring system for statics," *Computers & Graphics,* vol. 32, pp. 511-524, 2008.

[21] C. Lee, J. Jordan, T. F. Stahovich, and J. Herold, "Newtons pen ii: an intelligent, sketch-based tutoring system and its sketch processing techniques," in *Proceedings of the International Symposium on Sketch-Based Interfaces and Modeling*, 2012, pp. 57-65.

[22] S. Cheema and J. LaViola, "PhysicsBook: a sketch-based interface for animating physics diagrams," in *Proceedings of the 2012 ACM international conference on Intelligent User Interfaces*, 2012, pp. 51-60.

[23] M. Runyon, S. Polsley, B. Williford, S.-N. C. Liu, J. Hurt, J. Linsey*, et al.*, "An Intelligent System to Analyze Sketched Solutions to Open-Ended Truss Problems," in *26th International Conference on Intelligent User Interfaces*, 2021, pp. 224-233.

[24] A. Wolin, B. Eoff, and T. Hammond, "ShortStraw: A Simple and Effective Corner Finder for Polylines," in *SBM*, 2008, pp. 33-40.

[25] R.-D. Vatavu, L. Anthony, and J. O. Wobbrock, "Gestures as point clouds: a $ P recognizer for user interface prototypes," in *Proceedings of the 14th ACM international conference on Multimodal interaction*, 2012, pp. 273-280.

[26] V. Viswanathan, J. Hurt, T. Hammond, B. Caldwell, K. Talley, and J. Linsey, "A Study on the Impact of a Sketch-based Tutoring System in Statics Instruction," in *ASEE annual conference*, 2020.