Robust Recognition and Interpretation of Mechanical Design Sketches

Christine Alvarado

31 October 1999

Abstract

Current computer-based design tools for mechanical engineers are not tailored to early stages of design. Most designers use pencil and paper at first, and only input their design into CAD systems after it is nearly complete. The tradeoff between the ease of drawing and the precision of a CAD tool is too great in for engineers who are just sketching out rough designs. My thesis will present a tool which is specifically geared towards designers in the earlier stages of design. Our system interprets and represents mechanical sketches as the user sketches them, then allows the user to simulate her system using a two-dimensional kinematic simulator. In particular, my thesis will introduce a method to translate a sketch into the designer's intended mechanical devices, despite ambiguities in the sketch.

1 Introduction

Sketching plays a key role in the early stages of mechanical design. Engineers often want to throw down their ideas quickly, as naturally as they can, crossing out and changing pieces as they sketch to get a rough visual representation of how their system will look. Only after their design is relatively stable do they take the time to input their system into computer CAD tools.

The reason for this is simple: computers are too difficult to use for the payoff that they provide at this stage. Consider writing a report using a computer. For the trained typist, typing is no more difficult than writing (some would say that it is simpler and faster), and the computer provides a world of payoffs for the writing of the document: clean representation, ability to edit without having to rewrite the whole report, spell checking, and so on. But while computers are especially good at representing and working with textual input, they are especially bad at working with drawings. There are freehand drawing programs, and with a stylus and electronic pad to simulate a pen and paper, drawing on a computer is not very difficult. But why would an engineer go through the trouble of getting to a computer and launching the drawing program and dealing with the overhead involved in learning how to use the program if he could just pick up a pen and draw on a piece of paper? More than likely, he wouldn't.

Consider, on the other hand, a system that would allow the engineer to draw as freely as she draws with a pen and paper, but that would watch the engineer as she drew, understanding her sketch as she worked. At any time she wanted, the engineer could stop drawing and simulate the system as it was currently drawn to get a feel for how all the parts of the system were working together. Along the way the computer could monitor her design decisions and ask her to explain parts of the system which might be confusing or design decisions that might not be obvious to another engineer. Finally, when she was done with her drawing and satisfied with her design she could just press a button and the computer would automatically input the major pieces of her design into the CAD tool of her choice. She would only have to fix minor details that were not obvious in her drawing. The computer system would have all the knowledge of another designer and not need constant help to work with the engineer's sketch.

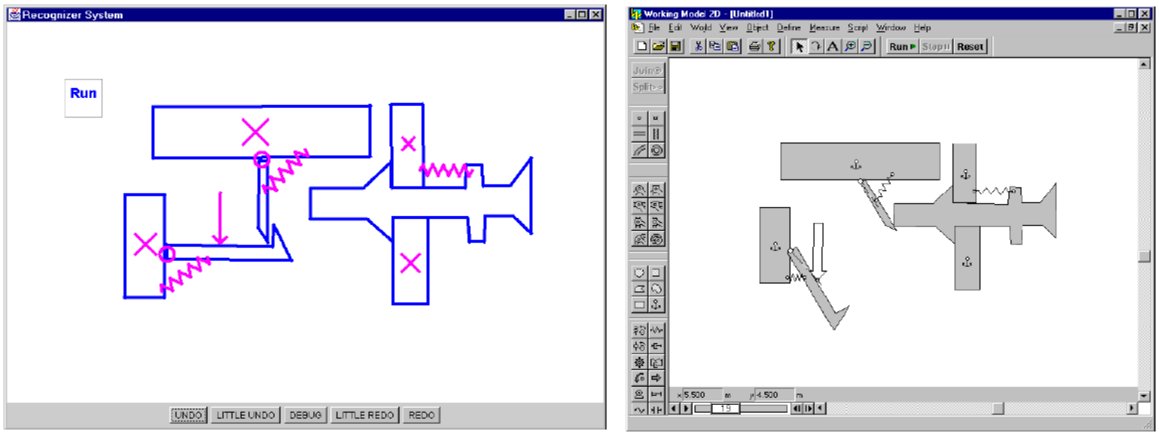

Figure 1: The user draws the sketch of the circuit breaker (left). When she clicks the run button, she sees a simulation of her sketch (right).

This type of system provides minimal overhead to the user because it is as simple and natural to use as a pen and paper. Yet it provides enough advantages that it would be worth the engineer's effort to draw on the computer during the initial stages of design. First, since she can view a simulation of her sketch at any time during the design process, she can more easily visualize her system without having to take the time to work out complex math equations and predict her device's behavior in her head. Also, because the whole design is done at the computer, the software can take the burden of documenting design changes from the engineer, who rarely stops to document these change when designing on paper. Finally, an intelligent sketch-based system provides a natural interface to CAD tools and can save time further along in the design process.

The rest of this proposal will discuss the issues involved in constructing the system described above. I will begin with an overview of the main problems that must be solved, focusing on the specific problem that my work will address. Then I will outline my proposed solution and describe a system, called SimRecSystem (Simulation enabled Recognition System), which will implement the solution described in this proposal.

2 Intelligent Sketch Tools: What's Missing

The ideal sketch based design program described above is far from possible at this point in time. As mentioned earlier, computers are used currently only at the end stages of design because they are good at formal representation of mechanical devices. However, they are not good interpreting informal representations or sketches of such devices and translating them into a format that can be used in fabrication or simulation. Below I describe the major issues which must be addressed before a system such as the one described above can be built.

2.1 Steps to an Intelligent Sketch-Based Design Tool

Our goal is to build an intelligent sketch interpretation system. Clearly there are many levels involved in getting such a system to work. Currently computers are skilled at representing graphics and drawings in terms of pixel representations, but the computer has absolutely no understanding of the drawing beyond how it can be represented graphically.

CAD and simulation tools resolve the representation problem be providing the user with fixed menus of parts from which to choose. In other words, to "draw" a gear, the user must pick the symbol for "gear" from a table or list and then adjust the size and placement of the part within the entire system.

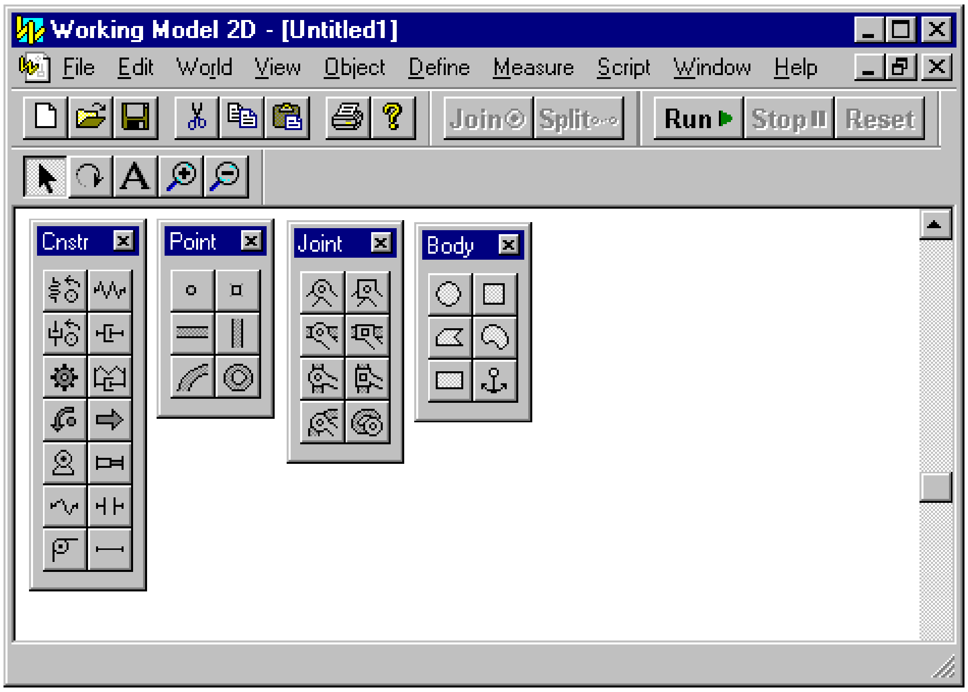

Figure 2: A typical menu-based interface. The user is constrained to pick menu items to place them on the screen.

On the other hand, our system must not confine the user to picking from menus, but must let the user draw freely, as if she were sketching on paper. So, for an intelligent sketch system to achieve the same power of representation as more formal computer design tools, it must recognize what the user is drawing as he draws it and choose an appropriate representation for each piece as it is drawn.

Next, the computer must have some notion of the behavior of mechanical systems. Even if we assume that the computer can recognize and represent each of the pieces in the engineer's sketch perfectly, there may be several possible interpretations for the system's behavior. For example, consider the sketch in Figure 3. The sketch does not completely constrain the

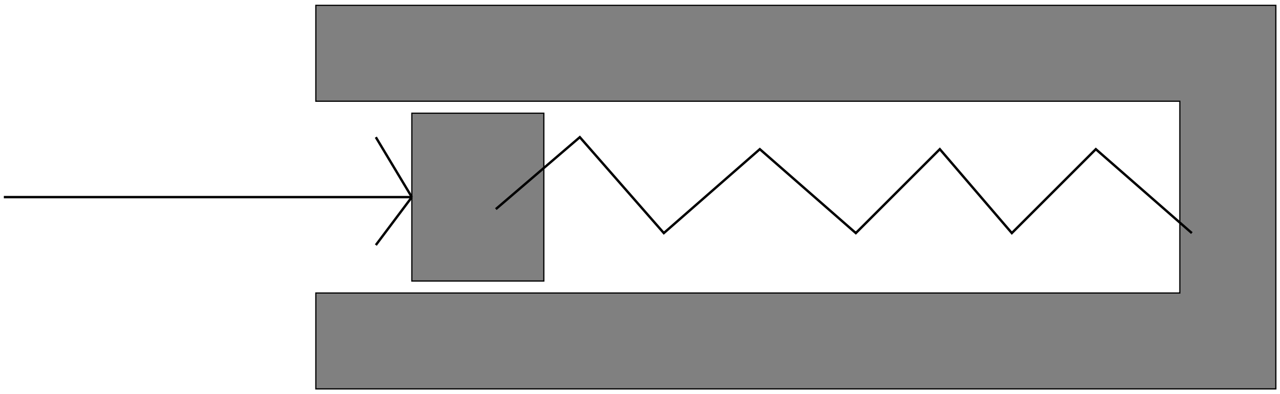

Figure 3: An ambiguous system. Depending on the length of the spring, the spring constant, and the strength of the force, the system will behave differently

behavior of the device: depending on the strength of the force and the length and stiffness of the spring, the block may move left, move right, or stay still. Without additional information the computer would have to be able to present the user with any of these three possibilities for the block's motion.

More importantly, the computer should be able to rule out silly or nonsensical behaviors. In the example above, if the block does not exactly fit against the sides of the tube, or the force is slightly off-center, the block could twist and get jammed inside the tube. The computer should be able to rule-out this non-intuitive, but literal, interpretation of the sketch.

The final area of problems that needs to be addressed involves design techniques. If the sketch system is going to be able to intelligently record design changes and prompt the user for the rationale behind certain design changes or decisions, the computer needs to have some sense of which design decisions would be obvious to another designer and which would be confusing. It should only interrupt the user if he does something which would be confusing to others.

2.2 Recognition and Representation

The proposed sketch system needs recognize the language of a mechanical sketch. Standard sets of parts combine in standard ways to convey specific meaning. This thesis will provide a method for assigning the correct interpretation to the mechanical parts in the designer's sketch despite ambiguities in the drawing. It will concentrate on the issues involved with recognizing and representing the pieces of an engineer's sketch as the he sketches.

2.2.1 Basic RecSystem

My thesis work is based upon the work of Luke Weisman and Manoj Muzumdar. They began work on the recognition and representation problem by first building a recognition and representation method that could be used across multiple domains. [9, 4] They constructed a system, RecSystem, which could recognize and represent a variety of low-level geometric shapes. It also provided the user with a method of interacting with these shapes through gesture commands.

On top of this basic recognition capability, they then build several context-sensitive recognition applications. For example, one extension of the base system was a mode for the user to design finite state machines, in which circles were interpreted as nodes, and lines were edges between the nodes. When nodes were moved around the page, the edges would then follow. Another mode allowed the user to draw floor-plans for a house. For more information on the basic system see [9].

2.2.2 Recognition of Mechanical Parts

One very important part of the recognition applications built into the basic RecSystem was the universal, natural icons used in the recognition process. When drawing finite state machines almost everyone uses circles to represent the states and arrows to represent transitions. Thus, a person using RecSystem's finite state machine mode could draw FSMs naturally, and the machine would have no trouble recognizing them.

In order to create a useful mechanical design sketching tool, we need to come up with a set of icons that are typically used in mechanical design sketches. A system that required users to change their notion of how to sketch mechanical parts would not be natural to use, and again, designers would resort to sketching with pen and paper. Fortunately the set of icons used by mechanical engineers in their sketching seem to be fairly consistent. Muzumdar added a basic set of mechanical icons to the RecSystem. [4] I will extend this basic set in my work.

2.2.3 Ambiguous Sketches

Assuming we have a set of icons typically used by engineers in their sketches, we still have the problem of recognizing these parts. In the original RecSystem, recognition is immediate and permanent. After the user draws a stroke, each of a series of recognizers examines the stroke to see it it fits the pattern it is trained to look for. The first recognizer that matches its pattern to the users stroke snatches up the stroke and puts down a new object. The recognition process then repeats until no recognizer triggers on the object in question. In this system each stroke the user makes can have only one interpretation and which interpretation the stroke ends up with depends heavily on the ordering of the recognizers. Furthermore, the stroke is always recognized to the deepest level possible. For example, if we have a recognizer that recognizes lines as rods, it would interpret all lines as rods the minute they were drawn, and every line drawn would be represented as a rod.

But clearly not all lines should be rods. Furthermore, given two lines that appear identical when they are first drawn, one may end up being a rod when the sketch is complete while the other may be the edge of a solid body, or the rope in a pulley. The basic problem is that we often cannot disambiguate sketches on a line by line basis and must instead wait until more of the context of the sketch is completed, suspending interpretation until the ambiguities have disappeared.

As an example, consider Figure 4. While they both start with identical strokes, one image becomes a ball and socket construction while the other represents two interlocking gears. If our program uses the aggressive style of recognition used in the initial version of RecSystem, we will not have the freedom to represent both of the images in Figure 4. In the current system, as the user draws, the system must either commit to making the circle and line into the ball of a ball and socket device, in which case it will never be recognized as a gear, or leave its representation as a line and circle, in which case it can be recognized as a gear but will never be recognized as a ball and socket.

Any system that could distinguish between initially ambiguous parts of a drawing would need to have a delayed recognition process. The system would need to maintain an internal representation of multiple interpretations for each pen stroke the user draws until there is

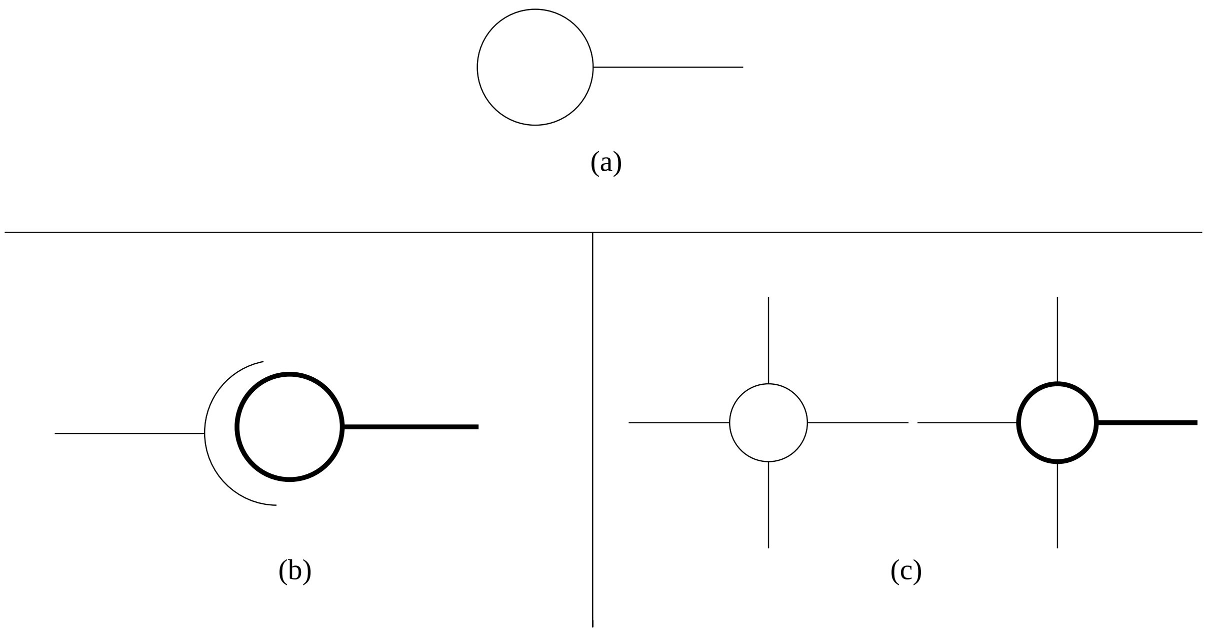

Figure 4: An example of ambiguity in the drawing. The bold strokes in (b) and (c) are identical to the strokes in (a). When (a) is drawn, an observer cannot know whether the device will become a ball and socket as in (b), or a set of gears, as in (c).

enough context to recognize this stroke unambiguously. However, deciding when there is enough recognition context is difficult in itself, especially in the case of low-level or default structures. As noted earlier, not all lines should be interpreted as rods when they are first drawn because it is likely that they will soon be used as lines to construct other mechanical parts. But after some period of time if they have not been used, a system (or human) watching the sketching process can assume that the line is not going to be used in another piece of the drawing. It then makes sense to make the default interpretation and call that line a rod.

3 Flexible Interpretation and Representation

At any point in time the system should have a best interpretation of the pieces in the engineer's sketch. That is, as the engineer draws each stroke, the system should keep track of the possible interpretations for the newest stroke and the impact that stroke has on what has already been drawn. The system should also continuously display its best interpretation of the sketch.

To ensure a clear but flexible representation of the engineer's sketch at all times, I propose the following solution. As the user draws, keep track of all the possible interpretations that stroke could be, or be part of. As the parts are recognized, examine the possible recognitions for the stroke, and give each interpretation a score based on the likelihood of this interpretation of the stroke. Then consider all the possible combinations of interpretations of the strokes which exist so far and choose the best combination of objects.

Consider the recognition of a gear, as examined in Figure 4. The user draws the circle and it is recognized as a circle, and as a circular body. The system displays a circle, (since a circular body looks the same as a circle). Next the user draws a line off the side of the circle. The system recognizes the new configuration as the ball part of a ball and socket. It also recognizes the line as a rod connected to the circular body. It does not replace the displayed circle with the ball and socket representation, because at this point, without the socket, it is still more likely that it is a circular body connected to a rod. As the final two lines are added to the circle, the system recognizes the whole grouping as a gear, and decides to display it as a gear since it is the most likely interpretation. But note that the system still maintains the other interpretations for the objects internally; if at any point in the future it makes more sense to consider the lines to be rods (say if the engineer then drew bodies on the other end of all of them), the system will change its favored interpretation.

To summarize, the three steps system will go through to incorporate a new stroke into the rest of the drawing are:

1. Allow all recognizers to examine the current stroke. Record all possible interpretations returned.

2. Examine the some or all of the strokes in the sketch. For each stroke examined, consider all existing interpretations of that stroke. Rate the likelihood that each interpretaion is correct in light of the newest stroke drawn.

3. Choose the best representation for the strokes in the sketch by choosing the combination of objects that have the highest overall likelihood score.

3.1 Recognition

In the original version of RecSystem, recognizers aggressively grabbed and replaced the new stroke as it was drawn, so if a line got recognized as a rod it could not also get recognized as part of a square. In the new proposed system, SimRecSystem, all possible recognitions of a stroke or group of strokes are represented by a relationship graph.

Whenever a stroke is recognized as an object, that object is created and placed in the recognition queue and linked to the object from which it was created. The newly recognized object is called the child object while the object from which it was recognized is called the parent. Figure 5 shows the recognition graph for a square.

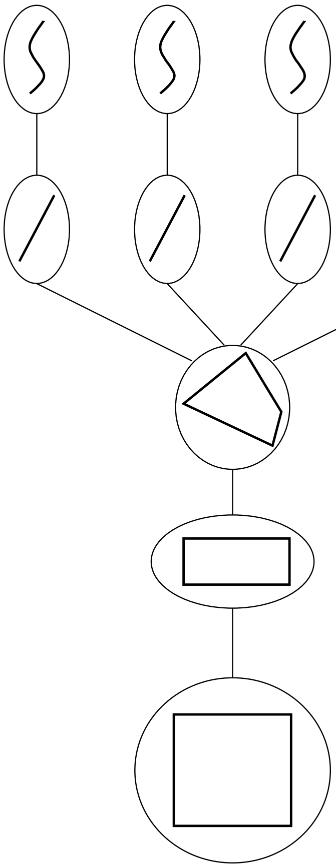

Figure 5: A simple recognition graph for a square. The parents are higher in the figure, and the lines connect parents and children. Each of the user's raw strokes are recognized as lines. Then the lines together are recognized as a quadrilateral, the quadrilateral is recognized as a rectangle and the rectangle is recognized as a square.

3.2 Reasoning

Each object which has been recognized has a score associated with it. This score represents how likely it is that the object is the correct interpretation of the strokes the user drew. Scores are discreet values between 1 and 10.

In the reasoning step, several modules, called reasoners, observe the objects in the graph that have been recognized so far. Reasoners are allowed to examine any the objects that have been recognized to get an idea of the context in which the stroke was drawn, in order to rank the different interpretations of the stroke, and change the weights of the interpretations of the strokes already drawn.

Each time an object's score is changed, the object records which reasoner changed its score. This way after all the reasoners have been run, the system not only knows which objects have which score, but also why their scores were changed.

Reasoners will be built from a series of rules about how likely different interpretations are, based on rules between possible interpretations, and on implicit user feedback. Here are some examples of rules which will drive reasoners:

- If a stroke is recognized as a square and a circle, then it is probably a square, because objects which are recognized as squares are a subset of the objects recognized as circles.
- If a stroke is recognized as a gesture command (a move, a selection, a delete) then it is probably a gesture, and not a physical object.
- The longer the user spends working away from an existing area of the drawing, the more likely it is that all the context needed to interpret that area of the sketch is in place.

These are just a small set of the rules needed to combine the multiple recognitions of objects cleanly and intuitively in a sketching system. I will come up with a more complete of rules by testing the system performance.

3.3 Resolution

After all the interpretations of the objects have been scored, the system must decide on the best overall representation for the pieces in the sketch. To do this, the system will use a greedy algorithm to choose the best piece in the interpretation graph, and then rule out any other parts which this interpretation makes impossible. This way, each stroke that the user has drawn gets exactly one representation. As the user adds and deletes strokes, the resolver continues to update the best interpretations for the parts in the drawing.

The resolver also provides a simple method for secondary interpretations. Say the system misinterprets something that the user draws. When the user informs the system that it got the wrong interpretation, the resolver throws out its interpretation of the user's last stroke and chooses the next best combination of interpretations.

4 Related Work

My work has been influenced both by work in sketch interpretation and in mechanical system simulation.

Similar work in sketch interpretation is perhaps best embodied by the Electronic Cocktail Napkin project by Mark Gross and Ellen Yi-Luen Do. [2, 3, 1] They built a system that employs the same bottom-up recognition method as ours. Their system also has a method for representing ambiguity in the user's sketches, and is capable of refining its interpretations of the user's stokes after the appropriate context has been set into place.

Eric Saund and Tomas Moran also outline several goals in interpreting ambiguous sketches. [6, 7] My work will outline a practical method for implementing many of the multiple representation and disambiguation techniques suggested in their work.

Elisha Sacks presents a method for qualitatively determining the behavior of a mechanical system. [5] Tom Stahovich developed a system which could qualitatively determine the behavior of a system and generate new physical representations of the design. [8] We hope to combine our sketch interpretation system with qualitative behavioral reasoning to generate a natural simulation based on the users sketch.

5 Conclusion

Ambiguity is only natural in sketched design. The very nature of a sketch requires that it not be a formal description of a system. However, we know that there exists some set of knowledge that makes a sketch clear to another (trained) human who examines the sketch. Through intelligent representation and handling of ambiguities in a sketch, I will provide a way to bring computer based sketching systems up to the level of the user. Once the computer can understand an engineer's sketch at the level as another engineer, the computer will become a truly useful tool in design providing a rich and interactive environment that a pen and paper cannot.

References

[1] Ellen Yi-Luen Do and Mark D. Gross. Drawing as a means to design reasoning. *AI and Design*, 1996.

[2] Mark D. Gross. Recognizing and interpreting diagrams in design. In *2nd Annual International Conference on Image Processing*, pages 308–311, 1995.

[3] Mark D. Gross. The electronic cocktail napkin - a computational environment for working with design diagrams. *Design Studies*, (17):53–69, 1996.

[4] Manoj Muzumdar. Icemendr: Intelligent capture environment for mechanical engineering drawing. Master's thesis, Massachusetts Institute of Technology, 1999.

[5] Elisha Sacks. Automated modeling and kinematic simulation of mechanisms. *Computer-Aided Design*, 25(2):107–118, 1993.

[6] Eric Saund and Thomas P. Moran. A perceptually-supported sketch editor. In *Proceedings of ACM Symposium on User Interface Software and Technology*, November 1994.

[7] Eric Saund and Thomas P. Moran. Perceptual organization in an interactive sketch editing application. In *ICCV 1995*, 1995.

[8] Tom Stahovich. Sketchit: a sketch interpretation tool for conceptual mechanism design. Technical report, MIT AI Laboratory, 1996.

[9] Luke Weisman. A foundation for intelligent multimodal drawing and sketching programs. Master's thesis, Massachusetts Institute of Technology, 1999.