Extending the ASSIST Sketch Recognition System

by

Rebecca Anne Hitchcock

Submitted to the Department of Electrical Engineering and Computer Science

in partial fulfillment of the requirements for the degree of

Master of Engineering in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2003

© Massachusetts Institute of Technology 2003. All rights reserved.

Author ..............................................

Department of Electrical Engineering and Computer Science

May 21, 2003

Certified by ...........................................

Randall Davis

Professor of Computer Science and Engineering

Thesis Supervisor

Accepted by ..............

Arthur C. Smith

Chairman, Department Committee on Graduate Theses
Extending the ASSIST Sketch Recognition System

by

Rebecca Anne Hitchcock

Submitted to the Department of Electrical Engineering and Computer Science
on May 21, 2003, in partial fulfillment of the
requirements for the degree of
Master of Engineering in Electrical Engineering and Computer Science

Abstract

Freehand sketching is an important part of the design process for mechanical engineers. Designers will often draw a basic sketch of the system they are designing before the details have been worked out. Later in the design process this sketch will be copied into a CAD tool, resulting in a duplication of effort and a loss of information contained in the sketch. ASSIST is a sketch recognition system that attempts to address this issue by combining the freedom and natural feel of freehand drawing with the power of a CAD tool. The work of this thesis was to extend ASSIST in order to make it more useful in practical settings. This goal was met in a number of ways: The vocabulary of the system was expanded to include a larger number and more complex set of mechanical devices, additional low level recognizers were included to allow more freedom in the types of strokes that could be drawn and recognized, mechanisms for expressing parameter magnitude were added, and the use of context was expanded to allow re-recognition of moved strokes and use of alternative recognitions as subcomponents for higher level devices.

Thesis Supervisor: Randall Davis
Title: Professor of Computer Science and Engineering
Acknowledgments

I would first like to thank my advisor Randy Davis for all his help on this project. His encouragement, guidance, and discussions throughout this work have been invaluable in completing this thesis.

I would also like to thank Christine Alvarado for her help acquainting me with the work in general and the details of ASSIST specifically, Metin Sezgin for his help with the toolkit, and for providing me with an arc recognizer on request, and Mike Oltmans for his help with ASSIST and system setup issues I encountered during this work.

I would also like to thank the MIT Project Oxygen Sponsors for their financial support.

I would also like to thank my friends and colleagues in the AIRE group for sharing their workspace, their company, and their munchies with me throughout my time at the AI Lab.

Finally, I would like to thank my family, without whom none of this would be possible. and my friends, without whom I would never have made it through MIT with my sanity intact. Their support and encouragement are what have allowed me to succeed, and for this I am forever grateful.
# Contents

1 Introduction ................................................. 15
   1.1 The Importance of Sketching ............................ 15
   1.2 ASSIST .................................................. 16
      1.2.1 Overview ........................................... 16
      1.2.2 Working Model ....................................... 17
      1.2.3 Example .............................................. 18
   1.3 Extensions ................................................ 19
      1.3.1 Enlarging the Vocabulary ............................ 19
      1.3.2 Extendible Objects ................................... 20
      1.3.3 Parameter Magnitude ................................. 22
      1.3.4 Re-recognition based on new context ................. 23
   1.4 Outline ................................................... 25

2 Previous Work ............................................... 27
   2.1 Low level Stroke Processing ............................ 27
      2.1.1 Feature Detection .................................... 27
      2.1.2 Curve Fitting ....................................... 29
      2.1.3 Shapes Recognized .................................. 29
   2.2 ASSIST ................................................... 30
      2.2.1 The Structure of ASSIST ............................. 30
      2.2.2 Original vocabulary ................................. 36
3 Motivation

3.1 ASSIST as a Design Tool ........................................... 37
3.2 ASSIST as an Education Tool ....................................... 37
3.3 Simulator Devices ..................................................... 39

4 Added Functionality .................................................. 41

4.1 Allowing Extendible Devices and Building on High Level Recognitions 42
  4.1.1 Spring Damper .................................................. 42
  4.1.2 Pulley ......................................................... 43

4.2 Adding and Using Low Level Recognizers ......................... 45
  4.2.1 Curved Strokes: Ropes ...................................... 45
  4.2.2 Arcs: Torque .................................................. 46
  4.2.3 Spirals: Rotational Springs .................................. 47

4.3 Adding Parameter Magnitude to the Visual Vocabulary .......... 48
  4.3.1 Force ....................................................... 48
  4.3.2 Rotational Spring ......................................... 48
  4.3.3 Torque ..................................................... 49

4.4 Re-recognition and Pruning ........................................ 50
  4.4.1 Motor ....................................................... 51
  4.4.2 Pruning: Ropes and Pulleys ................................. 52
  4.4.3 Re-recognition of Moved objects ........................... 55
  4.4.4 Gear ....................................................... 57

5 Future Work .................................................................. 61

5.1 Additional enhancements to ASSIST ............................... 61
  5.1.1 Curved Bodies .................................................. 61
  5.1.2 Pruning ....................................................... 62
  5.1.3 Further Re-recognition ....................................... 63
  5.1.4 Refining the Temporal Reasoner ............................. 64

5.2 Multi-modal Interfaces ............................................... 65

5.3 Next Generation Recognition System .............................. 66
# List of Figures

1-1  A Sample Interaction With ASSIST: Each stroke is recognized as it is drawn (left). When the user is ready, he presses “Run” to see the simulation (right)........................................... 18

1-2  Additional recognizers allow for more complex interactions. Here the user can draw a powered car with friction modeled by a rotational damper ................................................................. 19

1-3  The two pulley systems shown (top) can be combined to make one longer pulley system(bottom). Many mechanical systems include arbitrarily extendible parts. .................................................. 21

1-4  The force indicated by the longer arrow is stronger than the force represented by the shorter arrow. Allowing the user to express parameters in this way provides a method for expressing relative strength without requiring exact measurements or calculations. ............................... 22

1-5  Objects in the system are re-recognized as their context changes. When the middle gear is removed from the top figure (middle) the outer gears are re-recognized as unrelated bodies. When they are moved together again(bottom) they are re-recognized as interlocking gears. .................. 24

2-1  A stroke representing a square, and its direction, speed, and curvature data. ................................................................. 28
2-2 An example of a sketch in ASSIST. The figure is annotated to indicate the order in which the strokes are drawn. When the sixth stroke is
drawn, strokes three, four, and five are re-interpreted as part of a
damper.

4-1 A Spring Damper is composed of a spring attached to a damper.

4-2 A simple pulley.

4-3 This figure shows the different arrangements that can lead to a new pulley. The top two show situations where a rope is added and the bottom three show situations where a wheel is added.

4-4 A torque acting on a jointed body. A torque is represented by an arc with an arrow head.

4-5 A rotational spring is represented by a spiral.

4-6 A complicated system. Torque in this system would be scaled based on the distance from the center of mass of the body on which the torque acts.

4-7 A motor is indicated by a circle and a trapezoid. Because the motor is a point constraint, the cleaned up version appears at a fixed size.

4-8 The old recognition tree for quadrilaterals. This lead to inconsistent recognitions in the Widget Pool for the quadrilateral body interpretation.

4-9 The recognition tree for a straight line. The rod interpretation will be chosen, and under the old pruning algorithm, only the interpretations indicated by the red dashed line will be stored in the widget pool. This prevents future recognizers from building on the rope interpretation.

4-10 A small circular body(left) is re-recognized as pin joint when its context is changed(right).

4-11 When one gear is moved away from the other, both the moved gear and the remaining gear are re-recognized.
List of Tables

2.1 Recognitions under consideration and the strokes they include for the example of Figure 2-2 ........................................ 33

2.2 An example of a resolution for one possible scoring of the interpretations in Table 2.1. In this case the rod interpretation for stroke 3 is the highest scored interpretation. ................................. 35

2.3 An example of a resolution for another (more realistic) possible scoring of the interpretations in Table 2.1. In this case the Damper recognition is scored highest. ............................................ 36
Chapter 1

Introduction

1.1 The Importance of Sketching

Drawing is an important part of any design process. An engineer often draws a basic sketch of his system long before he has worked out the details. This sketch helps him to think through his ideas, see how pieces of the system might fit together, and show his work to others. This level of sketching is an informal process, and its power lies in the freedom of the designer to make unconstrained use of the simple media of pencil and paper.

In the later stages of design, an engineer will often take his sketch and copy it to a CAD or simulation system. This system gives him new powers. The system he creates in the simulation software can be seen in action, given specific parameters, and saved to disk for future reference. Once this stage is reached, the pencil and paper sketch is left behind, often thrown away or lost.

There are two problems with this procedure. First, it requires the engineer to duplicate his effort. He first draws his idea with a pencil, and then draws it again using a software package. In addition, an important part of his effort is discarded. Although the sketch may not contain the measurement and precision of the record stored in the computer, it contains other information that may prove to be important. A freehand sketch may give important insight into the thought process behind the design, an insight that cannot be gained from the sterile CAD drawing.
It is because of these problems, and the benefits of both the freehand sketch and the computer, that work has been done in sketch recognition. This research allows the designer to combine the steps described above, sketching a system, and having the computer automatically understand, simulate, and incorporate the sketch into a more formal design environment. Thus the transition is seamless, and both the formal design file and the informal sketch can be saved as record. A sketch recognition system has added benefits as well: the sketch can be simulated at any phase of the drawing, not only when it is in its completed form, and continuous feedback is gained even during the early stages of design.

1.2 ASSIST

1.2.1 Overview

One such recognition system was developed in [3]. ASSIST (A Shrewd Sketch Interpretation and Simulation Tool) was designed for the mechanical engineering domain and recognizes a basic vocabulary of mechanical objects. The work of this thesis is to expand and generalize this system, both by adding new devices to the vocabulary, and expanding the user interaction and types of recognition that are possible.

The goal of ASSIST is to provide an intuitive interface for designing mechanical systems. The user can draw his system as a freehand sketch, and have the system recognize his drawing not only at the stroke level, but as parts of a mechanical system which can be simulated at any point. This approach combines the freedom of a sketch with the power of a simulation tool. ASSIST understands the drawing without forcing the user to explicitly label each object drawn.

After each stroke, the system cleans up the stroke, and colors it to indicate the mechanical part that has been recognized. In order to achieve this understanding, ASSIST is forced to resolve ambiguities. For example, a circle could be intended as a circular body, a pin joint, or a selection gesture. ASSIST uses a number of principles when resolving ambiguity.
The first of these is the idea that simpler is better. One interpretation that explains many strokes is more likely correct than a number of distinct interpretations for each stroke. For example, if a four lines form a polygon, this is more likely intended as a polygonal body than as four distinct rods.

Another source of disambiguation information is temporal evidence. Strokes that belong to the same object are likely to be drawn sequentially. If a user draws a stroke, moves to another part of the drawing, and then returns to draw more near the first stroke, this is more likely the beginning of a new object rather than the completion of something started with the first stroke.

A third source of disambiguation is user feedback. Although the system will make interpretations of the user's strokes without feedback from the user, it does take this feedback into account when it is received. If the system chooses an incorrect interpretation for the user's stroke, the user may press the "Try Again" button, and be presented with a list of possible interpretations. The user can then select the interpretation they intended. This direct and explicit feedback is one source of evidence, but even if the user gives no explicit feedback, they are providing information. ASSIST assumes that the longer a part of the drawing goes unchanged or unchallenged, the more likely it is that that section of the drawing was recognized correctly.

1.2.2 Working Model

ASSIST uses a commercial simulator called Working Model 2D to simulate the recognized drawings. When the user presses the "Run" button, ASSIST passes a description of the system to Working Model using Working Model Basic (a variant of visual basic)[10]. Working Model is then responsible for the physics behind the sketch, simulating the bodies and enforcing the constraints that ASSIST recognized in the user's sketch. This allows the user to see how the design sketched would behave.
Figure 1-1: A Sample Interaction With ASSIST: Each stroke is recognized as it is drawn (left). When the user is ready, he presses “Run” to see the simulation (right).

1.2.3 Example

This section presents an example of a simple interaction with ASSIST to draw and simulate the car and ramp example of Figure 1-1. The user begins by drawing the polygon that represents the body of the car. When the user lifts his pen the system recognizes his stroke as a body, cleans it up, and colors it blue. The user then draws the circles that represent the wheels, which also turn blue as they are recognized as bodies. When the smaller circles are drawn within the wheels, they are recognized as pin joints and turn pink. The user then continues, drawing the ramp down which the car rolls, the anchor which holds it to the background, and the arrow which represents gravity. As each object is drawn, it is recognized by the system, and colored to indicate understanding, blue for bodies, and pink for constraints. If the user wants to see his system in action, he presses the run button, bringing up a simulation of his system as shown in the right part of Figure 1-1.

The interaction with ASSIST is designed to be as natural as possible. They system does not interrupt the user as he is drawing. The system is able to resolve ambiguities without explicit input from the user (such as distinguishing between the pin joints and the wheels). In addition, the user can call up the simulator at any time to see his drawing in action. Modifications can then be made, and the user can update the
Figure 1-2: Additional recognizers allow for more complex interactions. Here the user can draw a powered car with friction modeled by a rotational damper drawing based on what he sees in the simulation.

1.3 Extensions

The work of this thesis is to expand ASSIST in a number of ways, both in functionality and vocabulary, to make ASSIST a useful tool both for design and education. There are four types of extensions that were made to the system in this work. Two of these types are changes that extend the number of different types of objects and changes that allow more complex extendible objects. These allow more complex devices to be designed. In addition, changes were made to allow the drawing to be more expressive by including means to indicate relative parameter magnitude. Finally, changes were made in the pruning and re-recognition algorithms, allowing the drawing to grow and change fluidly.

1.3.1 Enlarging the Vocabulary

In order for a mechanical design tool to be useful, it must recognize a sufficient vocabulary of objects. If the system does not know the device the user is trying to draw, it has no hope of recognizing it, frustrating the user.

In section 1.2.3 an example was given of a simple interaction with ASSIST. In
this example a car rolled down a hill. This simple toy example showed the power of having a drawing brought to life by recognition and simulation. For real life examples however, the vocabulary of ASSIST was insufficient to describe the possible interactions between objects. Say, for example, a user wished for the car to go up the hill. A force could be attached to the car, pushing it up the hill, but this does not accurately describe the way a car is powered. A car is driven by a motor driving the axle of the wheel. In addition, true axles have friction, and in a design such as this the user may wish to indicate a non ideal pin joint using a rotational damper. If these objects are part of the vocabulary they can be used to create these complex systems, as shown in Figure 1-2.

In this thesis, these and other constructs were added to ASSIST to make it a more realistic tool. By expanding the vocabulary of the system, the tool has become more useful to engineers and teachers in the field who wish to design real world systems with a diverse set of devices.

1.3.2 Extendible Objects

In addition to extending the vocabulary of base mechanical components that can be recognized by ASSIST, this work extended the types of things that can be recognized. In its earlier version, most of the objects ASSIST recognized consisted of a small constant number of strokes, and once they were recognized they could not be changed or extended.

Many mechanical systems, however, depend on a series of linked parts such as gear trains and pulley systems. In order to design these systems, ASSIST must allow for extendible parts. Consider the example of two pulleys shown in the top of Figure 1-3. As shown, these pulleys don’t interact. If the designer wished to change this, there are two possible ways in which those two pulleys might interact: they might both attach to the same object, or they might both attach to a shared pulley wheel (as in the bottom part of the figure) creating the single combined pulley system shown. In order to be truly useful, ASSIST must be able to handle sketches like this.
Figure 1-3: The two pulley systems shown (top) can be combined to make one longer pulley system(bottom). Many mechanical systems include arbitrarily extendible parts.
Figure 1-4: The force indicated by the longer arrow is stronger than the force represented by the shorter arrow. Allowing the user to express parameters in this way provides a method for expressing relative strength without requiring exact measurements or calculations.

1.3.3 Parameter Magnitude

In her thesis, Alvarado discusses the drawbacks of the interface she makes with Working Model [3]. One such drawback is the absence of a method to specify device parameters. Values such as the strength of a force or the constant of a spring cannot be specified in a system that takes as input only drawings of mechanical parts. Thus, the values are set to the Working Model defaults. In order to specify these parameters, the user would have to simulate the drawing in Working Model, and change the parameters there.

It may seem that in the early stages of design for which this tool is designed it is unlikely that specific parameters are relevant. In fact it is likely that the calculations have not been done which would provide them. Requiring the user to enter them in ASSIST would disrupt the flow of thought that occurs during a freehand drawing. There is, however, an important exception to this. Even in this early stage it is possible that relative strengths are important. It may be crucial to the design to be able to indicate that one body has a greater force applied to it than another, or that one spring has a higher constant than another.

In this case, forcing the user to enter the parameters explicitly in Working Model

22
detracts significantly from the usability of our tool. As the designer thinks of the need for relative strengths he has no place to enter them. This may cause him to stop his drawing, and his flow of thought, to simulate and enter the values. In addition, if he only knows the relative and not absolute relationship between the two values, he finds himself artificially creating numbers that may have no meaning later.

An additional drawback to entering the values only in Working Model is that no record of this is maintained in the sketch. Thus the goal of capturing the design rational is lost. Later, in the more formal sketch, values will be assigned to these parts, but the sketch will not contain the key design decision that lead to these assignments, namely “force A is much larger than force B”. In addition, the interface between ASSIST and Working Model is such that values must be reentered each time the sketch is simulated.

In order to address these problems, ASSIST has been extended to set the parameter values based on visual cues in the drawing, rather than based on a single arbitrary default. For example, a force drawn with a longer arrow is simulated as a stronger force than a force drawn with a shorter arrow. Thus, the user has a means to indicate relative strength in his drawing in a way that affects the simulation. In addition, if the user finds the value in the simulation to be too large or too small, he can replace the object in his drawing with a more appropriate version, which will later remind him of the need for a large or small value for that parameter.

1.3.4 Re-recognition based on new context

As discussed above, Alvarado’s ASSIST made use of context to disambiguate multiple interpretations for a stroke. However, once a decision was made, there was only limited re-recognition if the context in which the decision had been made changed. A circle inside a body would be recognized as a pin joint based on this context, but if the circle were moved to the outside of a body, it was not given a chance to be re-recognized. It would still be considered a pin joint, and to have it change meaning, the user would have to delete and redraw the circle.

The work of this thesis extends the idea of context based recognition. Consider
Figure 1-5: Objects in the system are re-recognized as their context changes. When the middle gear is removed from the top figure (middle) the outer gears are re-recognized as unrelated bodies. When they are moved together again (bottom) they are re-recognized as interlocking gears.
the example of a user designing the gear train pictured in Figure 1-5. Two tangent circles are considered to be gears in ASSIST. Thus, if the user were to delete the center circle, there would no longer be a gear train, as shown in the center figure. The two outer circles would be only circular bodies, and there would be no constraint governing the interaction between them. They in fact would no longer be gears. Thus these two circles need to be re-recognized based on their new context.

If the user were then to move the circles next to each other, as in the bottom of the figure, they could again be interpreted as gears. It is interactions such as this one that require the new functionality presented in this work. If we accept that context is an important tool in disambiguation, then we must build a system which remains sensitive to changes in this context.

1.4 Outline

Chapter two of this work discusses the previous work on this system. It discusses both the low level stroke processing system developed by Tevük Metin Sezgin in [7] and Alvarado's original design of ASSIST as described in [3].

Chapter three discusses motivations for this work, and for the set of devices and extensions chosen.

Chapter four describes in detail the design and implementation of the extensions made to the system.

Chapter five discusses future work. Additional extensions and enhancements that were not completed in this work are discussed.

Chapter six presents a summary and conclusion for this work.
Chapter 2

Previous Work

2.1 Low level Stroke Processing

In order to build a system like ASSIST, some basic stroke parsing technology is required. The current version of ASSIST accomplishes this low level stroke parsing using a toolkit designed by Tevfik Metin Sezgin in his Master’s thesis [7]. This system takes raw data and returns low level shapes such as lines, curves, polygons, etc. This section describes that work as it pertains to ASSIST.

2.1.1 Feature Detection

The first step in parsing raw stroke data is to locate the feature points. Feature points are defined as points that mark the beginning and ending of lines or curved segments. The goal is to find these points without false positives such as finding a vertex on what is in fact a curved portion of the stroke. Sezgin uses a combination of curvature and speed data in order to find the feature points.

The first step in this process is to find the points in both sets of data that might be feature points. For the curvature data, these are clearly the points of highest curvature. For the speed data, because users often slow down as they draw a corner of a shape, the minimums often correspond to feature points. Blindly selecting all extrema, however, will lead to many false positives. This is due to the noise inherent
in sketching, as seen in Figure 2-1. Thus Sezgin uses two filtering methods to choose those that are most likely to correspond to actual vertices.

The first of these is average based filtering. This method chooses a threshold based on the average value and uses it to divide the graph into regions. The global extremum in each region is then considered to be a corner. The second method is scale space filtering. This method looks at the data through multiple scales using the framework developed in [9]. In using this method, Sezgin attempts to remove the dependence on constant thresholds.

Once the potential corners for both the speed and curvature data have been found, they are combined into a hybrid fit for the stroke. The first approximation is to use the intersection of vertex sets of the curvature and the speed. In many cases, however, there are vertices that one or the other of the metrics cannot find.

To find these vertices, Sezgin includes points found by only one metric if they significantly increase the accuracy of the fit. The vertices are first ranked within each set using an appropriate metric for each type of data. Then the highest ranked vertices from each set are compared, and whichever decreases the least squared error
by the most is added. This procedure is repeated until the overall least squared error drops below a set threshold.

2.1.2 Curve Fitting

The procedure discussed in section 2.1.1 produces piecewise linear interpretations, but many strokes encountered in practice contain curved portions. In order to determine which sections of the strokes are curved, the system compares the arc length of the stroke segment to the Euclidean distance between each pair of consecutive vertices. For curved regions the arc length will be significantly higher than the straight line distance.

Once the curved portions are detected, each curve is fit to a Bézier curve defined by two end points and two control points. The control points are calculated for each curve as functions of the tangent vectors using empirically determined constants.

Next this approximation is checked for accuracy. Because finding the least squared error to the Bézier curve is computationally intensive, this error is approximated as the least squared error to a piecewise linear approximation to the curve. If the error is higher than the empirically determined error threshold, the curve is broken into two sections and each one is recursively re-fit.

2.1.3 Shapes Recognized

The interface to the low level toolkit is via a managing object known as a Classifier. Each stroke has a Classifier, which can be queried to find information about its shape. The first time it is queried it performs the processing described in sections 2.1.1 and 2.1.2 above. The result is then cached for future queries. When a system such as ASSIST queries the classifier, it can return the most likely fit for the stroke, and can also provide least squared errors for both the chosen fit and all rejected fits.

The original toolkit provides four types of fits: line, circle, polyline, and complex. A complex fit is any non circular fit made up of multiple segments where not all segments are linear. The first three of these are included in Alvarado’s ASSIST using
recognizers that query the Classifier to determine an interpretation.

2.2 ASSIST

2.2.1 The Structure of ASSIST

In order to understand the enhancements made in this work, it is important to understand the overall structure of ASSIST [3]. ASSIST processes user strokes by passing them through three main steps: recognition, reasoning, and resolution. The interpretations that are found and chosen are organized in structures called widget pools. This section discusses these four aspects of ASSIST in greater detail.

Widget Pools

The strokes and interpretations in ASSIST are organized in a series of data structures called widget pools. ASSIST has three different widget pools, the main pool, the surface pool, and the recognize pool. The main widget pool contains the current set of interpretations, and all those interpretations that they derive from. For example, if a stroke is recognized as a pin joint, the main widget pool will contain a pin joint interpretation, a circle interpretation, and a raw stroke (known as a “squiggle”) interpretation. Those interpretations which are not selected are pruned from the main widget pool in Alvarado’s ASSIST.

The surface pool contains the interpretations that the user sees. In our example above there would be only the pin joint interpretation for the user’s stroke. In the surface pool, there can only be one interpretation for each stroke, so the circle and squiggle interpretations would not appear here.

The recognition pool is a widget pool that is built for each new stroke added to the surface. It contains the interpretations that are spatially and temporally close to the current stroke. These are the strokes that are available for recognition of the new stroke. The purpose of this pool is to present a subset of the widgets to the recognizers which are most likely to be combined with the current stroke.
In order to fill these pools, and provide the functionality described, strokes are passed through a three stage process: recognition, reasoning, and resolution.

Recognition

The first step in choosing an interpretation for a stroke is recognition. This stage generates all possible interpretations for the current stroke. In this stage, no judgments are made about whether or not an interpretation is likely, just whether it is possible. Thus the output of this stage is a set of possible explanations for a stroke.

In order to generate this set of choices, ASSIST uses a set of components called recognizers. Each recognizer is responsible for recognizing one type of object. Recognizers have three main features, a list of input types, an output type, and a recognition function that determines if the input stroke could in fact be a part of an instance of its object. Take for example, a quadrilateral recognizer. The sole input type for a quadrilateral is a straight line, and the output type is a quadrilateral. The recognize function would take as input a single stroke which had been recognized as a line, and attempt to combine it with other strokes that have been recognized as lines on the surface to form a quadrilateral. The recognizer would be responsible for checking that
the lines meet appropriately, and that they do in fact form a closed shape. Once this had been determined, the recognizer would return “true” and inform the surface that the input stroke, combined with the three strokes it had found, could be replaced by a quadrilateral.

In order to find all possible interpretations, these recognizers must work together. In order to see how this works, consider Figure 2-2 in which the last stroke of a damper is drawn. Before this last stroke is added, all the lines that have been drawn have been recognized as independent rods.

Once it is added, the stroke is entered into the system as a Squiggle, or uninterpreted stroke. The system then finds all recognizers that use Squiggles as an input type. This would include all recognizers for low level objects such as circles, polylines and lines. Many of the recognizers that take squiggles as input use the toolkit described in section 2.1 above. Each recognizer that returned true would have a chance to contribute its recognitions to the surface. In this case, the line recognizer would fire, and a line interpretation would be added.

Once the line recognizer had fired, all recognizers that take lines as input types would be given a chance at the stroke. In Alvarado’s ASSIST this included rods, pulleys, dampers, polygons, etc. In our case the rod recognizer would succeed, as would the damper recognizer. In this second case, the recognizer would make use of nearby strokes on the surface to complete the object.

Next the new interpretations would be passed on to any recognizers that used them as input. In Alvarado’s ASSIST neither rods nor dampers are input types to larger objects. When all recognizers have had a chance at the generated interpretations, all possible recognitions have been found, and the recognition is complete.
Interpretation | Stroke(s)
--- | ---
Squiggle | 3
Line | 3
Rod | 3
Squiggle | 4
Line | 4
Rod | 4
Squiggle | 5
Line | 5
Rod | 5
Squiggle | 6
Line | 6
Rod | 6
Damper | 3,4,5,6

Table 2.1: Recognitions under consideration and the strokes they include for the example of Figure 2-2

Reasoning

At the end of the recognition phase, we are left with a collection of possible recognitions. The next step is to determine which of these is correct. Continuing with the example of the previous section, we have the recognitions shown in Table 2.1. Notice that this list includes possible interpretations for previous strokes 3, 4, and 5, and not just the interpretations for stroke 6. This is because the damper recognizer found a new possible interpretation for these previous strokes (i.e., as part of a damper), so the system must reconsider their interpretations. Strokes uninvolved in the current recognition, such as strokes 1 and 2, are not reasoned about on this new stroke.

The strokes in this list are next passed through a series of reasoners. Each reasoner encapsulates a small piece of knowledge about the world, such as "circles are more likely to be pin joints if they appear on bodies." A reasoner is called by passing two interpretations for the same stroke, and it returns a boolean response indicating which interpretation it feels is most likely. Although each reasoner contains a specific piece of knowledge, there are a number of general heuristics that govern the way

\footnote{As part of this work spring dampers and rotational dampers were added to ASSIST. In this case, these recognizers would have a chance to see if the damper was part of a spring damper or rotational damper before the recognition finished. Spring dampers are discussed in greater detail in section 4.1.1.}
they function, as discussed above in section 1.2.1. The include the idea that simpler representations using more strokes are more likely correct, that the user is likely to draw strokes that make up the same object sequentially, and the idea that mechanical objects that have meaning in the domain, such as a rod, are more likely correct than abstract interpretations such as a line.

In order to make use of the information from the reasoners, we need a way to organize their results. This is done using graph algorithms on a graph built by the recognizers and reasoners. Each recognition found by the recognizers is added as a node in the graph. Each time a reasoner says “Interpretation A is more likely than interpretation B” an edge is added from node A to node B. In this way we build a graph of the relationships between the various interpretations.

The next step, of course, is to determine from this graph which node is most likely. For this ASSIST uses a topological sort on the generated graph. Because of the nature of topological sorts, there are two problems that can occur. The first of these is that cycles, which could result if two reasoners have different opinions on which interpretation is more likely. In this case, there is evidence for both sides, and the drawing is ambiguous. ASSIST avoids passing such graphs to the topological sort by collapsing all cycles to a single node.

The second difficulty that can arise is that the problem is under-constrained. In this case, we again have an ambiguity, this time resulting from the fact that no reasoner has an opinion on the relative likelihood of the interpretations. Again the problem is solved by collapsing the nodes. Nodes which are ambiguously ranked are placed in one node. In either case of the two cases above, if a node is selected that contains more than one interpretation, the choice is made arbitrarily as to which object to display.

Once the topological sort is complete, the nodes in the graph are scored. Scores are evenly distributed between 0 and 10 over the interpretations. If there are more than eleven nodes, all nodes after the first ten are scored as zero. In addition to the scores given by the reasoners, interpretations scores can be changed by user feedback. As an interpretation stays on the surface, its score is increased on the grounds that
the longer the user allows an interpretation to be displayed, the more likely it is to be correct. If the user rejects the interpretation, it is set to a minimum value.

### Resolution

Once the interpretations have been scored, the final step is to find a consistent set of recognitions. The algorithm to do this first picks the interpretations with the highest score, then eliminates recognitions that contradict this interpretation. This is repeated until all interpretations have either been chosen or eliminated. Consider the two cases in Tables 2.2 and 2.3. In the first case, we see that the rod interpretation has the highest score. We chose this, and then the damper interpretation is no longer possible, as it would require two interpretations for stroke 3, both a rod and a part of a damper. Then the other strokes would be sequentially chosen as rods. In the second (more realistic) case, the damper interpretation would be chosen first, eliminating all other interpretations. In both cases, the recognition or recognitions chosen would be placed on the surface. In Figure 2-2 the second case is shown.

<table>
<thead>
<tr>
<th>Interp.</th>
<th>Stroke(s)</th>
<th>Score</th>
<th>Step: 1</th>
<th>Step: 2</th>
<th>Step: 3</th>
<th>Step: 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Squiggle</td>
<td>3</td>
<td>0</td>
<td>eliminated</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Line</td>
<td>3</td>
<td>3</td>
<td>eliminated</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rod</td>
<td>3</td>
<td>10</td>
<td>chosen</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Squiggle</td>
<td>4</td>
<td>0</td>
<td>eliminated</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Line</td>
<td>4</td>
<td>3</td>
<td>eliminated</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rod</td>
<td>4</td>
<td>7</td>
<td>chosen</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Squiggle</td>
<td>5</td>
<td>0</td>
<td></td>
<td>eliminated</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Line</td>
<td>5</td>
<td>3</td>
<td>eliminated</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rod</td>
<td>5</td>
<td>7</td>
<td>chosen</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Squiggle</td>
<td>6</td>
<td>0</td>
<td></td>
<td></td>
<td>eliminated</td>
<td></td>
</tr>
<tr>
<td>Line</td>
<td>6</td>
<td>3</td>
<td>eliminated</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rod</td>
<td>6</td>
<td>7</td>
<td>chosen</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Damper</td>
<td>3,4,5,6</td>
<td>8</td>
<td>eliminated</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 2.2: An example of a resolution for one possible scoring of the interpretations in Table 2.1. In this case the rod interpretation for stroke 3 is the highest scored interpretation.
<table>
<thead>
<tr>
<th>Interpretation</th>
<th>Stroke(s)</th>
<th>Score</th>
<th>Step: 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Squiggle</td>
<td>3</td>
<td>0</td>
<td>eliminated</td>
</tr>
<tr>
<td>Line</td>
<td>3</td>
<td>3</td>
<td>eliminated</td>
</tr>
<tr>
<td>Rod</td>
<td>3</td>
<td>7</td>
<td>eliminated</td>
</tr>
<tr>
<td>Squiggle</td>
<td>4</td>
<td>0</td>
<td>eliminated</td>
</tr>
<tr>
<td>Line</td>
<td>4</td>
<td>3</td>
<td>eliminated</td>
</tr>
<tr>
<td>Rod</td>
<td>4</td>
<td>7</td>
<td>eliminated</td>
</tr>
<tr>
<td>Squiggle</td>
<td>5</td>
<td>0</td>
<td>eliminated</td>
</tr>
<tr>
<td>Line</td>
<td>5</td>
<td>3</td>
<td>eliminated</td>
</tr>
<tr>
<td>Rod</td>
<td>5</td>
<td>7</td>
<td>eliminated</td>
</tr>
<tr>
<td>Squiggle</td>
<td>6</td>
<td>0</td>
<td>eliminated</td>
</tr>
<tr>
<td>Line</td>
<td>6</td>
<td>3</td>
<td>eliminated</td>
</tr>
<tr>
<td>Rod</td>
<td>6</td>
<td>7</td>
<td>eliminated</td>
</tr>
<tr>
<td>Damper</td>
<td>3,4,5,6</td>
<td>10</td>
<td>chosen</td>
</tr>
</tbody>
</table>

Table 2.3: An example of a resolution for another (more realistic) possible scoring of the interpretations in Table 2.1. In this case the Damper recognition is scored highest.

2.2.2 Original vocabulary

When creating ASSIST, Alvarado chose a set of parts based on the work of her predecessor [6]. This list was chosen to include the basic icons used regularly by mechanical engineers in the field. These included circular and polygonal bodies, forces, pin joints, springs, dampers, anchors, and simple pulleys.
Chapter 3

Motivation

3.1 ASSIST as a Design Tool

One important motivation behind extending ASSIST was to create a tool that would be useful to mechanical engineers in the field. Real world drawings and systems are very complex, and contain a wide variety of mechanical parts. Thus one motivation for this work is to extend the vocabulary to a set that will allow construction of most mechanical devices. Ideally, we hope to include all devices that a mechanical engineer would expect to be able to use as subcomponents for his design. This work takes a large step in that direction.

In addition, real world systems are more complex than those the original ASSIST could handle. They often include arbitrarily extensible components, such as pulley systems and gear trains. Engineer’s drawings include these things, and without this functionality ASSIST could not truly be a useful design tool. Thus, another motivation of this work is to provide engineers with the complexity they need to design useful systems.

3.2 ASSIST as an Education Tool

Another important motivation for this work is as an educational tool. Students learn more effectively when they are given an opportunity to interact with the concepts
they are learning. It is because of this that lab work is so often included in engineering curriculums. This gives the students a chance to get hands on experience with exploring how systems work.

This hands on experience, however, is often disjoint from classroom learning. A concept is explained in class, and only much later is it explored in lab. This can lead to the lectures being confusing and useless to the students who see only the abstract concept.

Demonstrations can help alleviate this problem to some extent. Professors in mechanical engineering classes often create animations to show to their classes to demonstrate how certain mechanical parts interact. This provides some of the needed context to a lecture, but still does not provide the hands on experience of a lab. Clearly an extensive lab set up is not plausible in a classroom setting. This is where ASSIST becomes useful.

With ASSIST, the professor could draw a system and explain the parts as he added them. Once the sketch was complete, he could simulate the system to show the students its behavior. The sketch could then be transferred to personal tablet computers at each student’s desk, where a small section of class time could be dedicated to personal exploration. The student could add to the sketch, move the parts, and change the parameters to see how his changes affected the system. The class could then be brought back together to discuss what they had found.

It was with this idea in mind that many of the changes to ASSIST were created. These changes allow more flexibility in the drawings. For example, allowing scalable parameters will give students an opportunity to experiment with how changing strengths affects the behavior of a system. In addition, the flexibility in the gear representation is designed to allow students the ability to explore different configurations of gears easily. This is a concept that is often non-intuitive to students, especially in the introductory level courses.

Plans have been made to include a version of this research in future sections of introductory mechanical engineering classes at MIT. It is hoped that the advances made in this thesis will help enhance the educational experience of the students
involved.

3.3 Simulator Devices

A final source of inspiration when determining which devices to include came from Working Model, the system used to by ASSIST to simulate the user’s sketches. The goal was to take advantage of a greater part of Working Model’s power, and to allow all of the aspects of the simulator to be accessed through the sketch interface. By including recognizers for most of Working Model’s key constraints, we eliminate the need to add aspects to the system using the menu based commands of Working Model. Most types of systems can be designed using the more natural and flexible sketch interface.
Chapter 4

Added Functionality

The purpose of this work was to extend ASSIST in a number of ways. In doing this, we found that there were four main technical categories of interesting enhancements to make to the system. The first of these was to create devices that can build upon other high level components. This allowed both more interesting and complex devices, and devices that are arbitrarily extendible. Next, we wished to allow more freedom in the shapes of strokes that ASSIST can understand and interpret. This resulted in adding many low level recognizers to the system, allowing a variety of types of curved strokes. Thirdly, we wished to add the idea of relative parameter magnitude to the visual vocabulary. This allows users to add constraints of different values, freeing them from the restriction of a single default for each type of constraint. Finally, we wished to expand the idea of recognition based on context by allowing recognitions of objects to change as their context does. This includes both making objects re-recognize when they are moved, and maintaining alternate high level recognitions which can be used in the case of new context. In this chapter we explore each of these categories in greater detail.
Figure 4-1: A Spring Damper is composed of a spring attached to a damper.

4.1 Allowing Extendible Devices and Building on High Level Recognitions

In order to increase the usefulness of the system, many new devices were added to the recognition vocabulary. Recognizing these complex objects was done in many cases by building upon other high level objects, or recursively extending objects by looking for smaller instances to build upon.

4.1.1 Spring Damper

One of the new devices that was added to ASSIST as a result of this work is the spring damper. The common representation for a spring damper in the mechanical engineering domain is a spring and a damper attached to each other (Figure 4-1). Recognizing this device in ASSIST was done by building on the existing spring and damper recognitions.

In addition to the spring and the damper, the spring damper has two lines at each end attaching them and giving them a place to attach to the bodies they constrain. The points that the spring damper act on are given by these end points.

Thus, in order to recognize a spring damper, there needs to be a spring, a damper, and two sets of connections, one for each end. These connections, however, have no mechanical meaning independent of the spring damper. It would not make sense for the system to recognize “connectors” or to inform the user when they were recognized. Therefore, these constructs are internalized as subclasses in the recognizer. When the recognizer is called on a line, it first attempts to create a connector out of the line. If this fails, the recognizer returns false without looking for springs or dampers. If it succeeds, then the connector is used by the recognition as a possible component of a
This is a simple example of an idea that becomes more important as the devices grow. As recognitions become increasingly complex, it makes sense to modularize the subcomponents of the recognition. However, not all of these subcomponents have meaning, and those that do not should not be displayed to the user. We see this again in the next section, where pulley wheels are another example of such internal subcomponents.

4.1.2 Pulley

Pulleys were included in Alvarado's original ASSIST as simple constraints indicated by two ropes coming from one wheel as in Figure 4-2. Real pulley systems are often more complex than this simple structure, allowing the rope to be guided through several wheels, or allowing slack in the ropes that make up the pulley system.

In order to achieve this, the pulleys in ASSIST were extended in many ways. The pulleys are made up of wheels and ropes. Like the connections of spring dampers, pulley wheels are subcomponents of pulleys that are not meaningful to the user. They are stored internally to the pulley recognizers. Ropes are also taken as input to the pulley recognizer. Complications resulting from using this particular higher level widget as a sub component are discussed in section 4.4.2.

In addition to pulley wheels and ropes, pulley systems can be made up of other smaller pulley systems. The recursive nature of the problem can be seen in figure
Figure 4-3: This figure shows the different arrangements that can lead to a new pulley. The top two show situations where a rope is added and the bottom three show situations where a wheel is added.

4-3. When a line is added, it can be combined with either a wheel and a rope, or a wheel and another pulley. Likewise, a wheel can combine with either two ropes, a rope and a pulley, or two pulleys to make a new pulley system. Notice how pulley wheels are not recognized independently of pulleys.

In order to address this the recognizer for pulleys has to take as input other pulleys. The free ropes can be tested for possible attachment to new pulleys. In addition, we need a way to create the new pulleys. One option would be to extend one of the already existing pulleys. This has a number of drawbacks. The first is simply that it doesn't treat all cases symmetrically. In the case where two existing pulley systems are combined by a new wheel, one of the smaller pulleys would have to be extended and the other replaced.

Another, more important drawback is that extending the pulleys directly does not present all possible alternatives to ASSIST. If the smaller pulleys are extended rather than replaced, the user does not have the option of choosing to revert to the smaller pulley plus some combination of other interpretations for the new strokes. Although the interpretation including the smaller pulleys is not what is chosen by default, the
user should have the option to disagree with the reasoning of the system. Thus, the original pulleys must remain unchanged in the widget pool.

4.2 Adding and Using Low Level Recognizers

In order to represent the larger set of mechanical devices, it was necessary to include additional low level recognizers. Circles and straight lines are insufficient for describing devices such as ropes (curves), torques (arcs), and rotational springs (spirals). In some cases this required simply using a larger set of the low level recognizers implemented in the toolkit described in section 2.1, and in some case this required adding new shapes to it.

4.2.1 Curved Strokes: Ropes

The toolkit contains an option for biasing its recognitions towards either curves or polylines. As none of the devices in Alvarado’s ASSIST contained non-circular curves, this bias was set to exclude these curves from being considered. Every stroke that was drawn by the user was approximated either as linear, piecewise linear or as circular.

In adding to the system, however, we wished be able to recognize ropes. Thus the bias was adjusted to allow complex strokes made of both piecewise linear sections and sections made up of curved portions modeled by Bézier curves (discussed in section 2.1.2). Once this interpretation is allowed, the appropriate structure must be added to ASSIST to support it. This includes adding a recognizer that queries the toolkit, and a widget class to represent the new complex shape.

Once complex interpretations are added to the system, we have the tools needed to support ropes. Ropes are recognized from any open stroke. This means that both curved strokes and straight lines can be recognized as ropes. A straight line therefore now has two possible interpretations, a rod or a rope, even without considering other strokes. In order to disambiguate, we use Alvarado’s principle that more constrained interpretations are more likely to be correct. Because a rod can only be recognized from a straight line, and a rope could come from any open stroke, we include a
Figure 4-4: A torque acting on a jointed body. A torque is represented by an arc with an arrow head.

reasoner that prefers rods over ropes. If the user intended his straight line to be a rope, he may choose this interpretation from the “try again” menu.

Including ropes in the system increases the variety of pulley systems that can be drawn: users can now draw pulleys with curved ropes of the sort in Figure 1-3, allowing a user to express slack in his pulley system. This functionality, in turn introduces some interesting problems in the recognition tree, as will be discussed in section 4.4.2.

4.2.2 Arcs: Torque

Some additional stroke classifications were required. For example, one of the desired recognitions was torque. A curved arrow as in Figure 4-4 is the typical representation for this idea.

While the toolkit supported curves and circles, there was no specific representation for curves that form only part of a circle. Curved sections that were not recognized as complete circles were described by Bézier curves, and information such as center and radius could not be obtained. To aid this work, Sezgin added an arc recognizer to his toolkit. This recognition could then be used along with perpendicular lines to form the torque symbol shown.
4.2.3 Spirals: Rotational Springs

The standard representation for a rotational spring is a spiral as seen in Figure 4-5. Because the toolkit contained no representation for a spiral, adding rotational springs to ASSIST required adding an additional low level recognizer for this shape.

Because the spiral is a low level shape, it was added at the level of the toolkit. The goal of the spiral recognizer is to find the spiral of the form $r = r_0 + a(\theta - \theta_0)$ that best fits the given data points. In order to make this fit, we begin by finding the center of the bounding box for the set of points in the stroke. We then compare both end points to this center point and determine which is closer. This allows us to order the points so that spirals drawn starting at the center can be handled equivalently to spirals drawn ending at the center.

Once the order of the points has been determined, we wish to find a better estimate of the center of the spiral than the center of the bounding box. In order to do this, we use the arc fit. We pass a small portion of the spiral to the arc recognizer, and take as the center the center of the arc that best fits the subsection. This gives us a more accurate estimation of the true center of the spiral.

Once the center has been found, we can calculate values for $r_0$ and $\theta_0$ directly. Next, all points must be converted to polar form, with monotonically increasing angles starting at the center of the spiral. We subtract the initial values, and then can perform a simple linear best fit to the equation $r' = a\theta'$ where $r' = r - r_0$ and $\theta' = \theta - \theta_0$. This gives us the value of $a$, and allows us to calculate the set of ideal points that best match this spiral.

Once the spiral fit has been made, it needs to be compared to other fits generated by the toolkit. The spiral fit is only considered in the case in which the stroke would be considered a complex fit by the original recognizers. If this is the case, and the
least squared error to the linear fit falls below a empirically determined threshold, then the spiral fit is preferred.

4.3 Adding Parameter Magnitude to the Visual Vocabulary

As discussed in section 1.3.3, including ways to express parameter magnitude allows important information about relative strengths to be included in a freehand sketch. This information can be expressed simply by size, as shown in Figure 1-4, but there are other ways to express this information, depending on the individual constraints and their representations.

4.3.1 Force

As discussed above, the value of a force is expressed by drawing a longer arrow to represent the larger force. This is an intuitive expression of the magnitude, as one would expect a larger object (the arrow) to have a larger amount of strength. The base value of a force is still chosen as a reasonable empirically determined constant, but it is now scaled by the length of arrow representing the force. In this way we combine the simplicity and natural feel that come with not requiring a numerical value with the ability to control relative strengths. This allows the user more freedom of expression in his drawing.

4.3.2 Rotational Spring

As discussed in section 4.2.3, rotational springs are represented by spirals centered at the point the constraint is desired. They include both the constraint and the implicit pin joint at that location.

Rotational springs, like linear springs, are parameterized by their spring constant. In the same way that larger arrows are considered to be stronger forces, we use size of the spiral - its maximum radius - as an indication of intended strength.
In the case of rotational spring, however, size is not the only visual cue we are given by the user. The spring constant is an indication of how tight the spring is, and as such, it makes sense to use the tightness of the spiral drawn as an indicator of intended strength. If the user draws two spirals with equal radii but different numbers of winds, the one that is wound more tightly is likely intended to offer more resistance. Using this principle, ASSIST assigns higher spring constants to more tightly wound spirals.

4.3.3 Torque

Torque is another example of a constraint with a strength parameter; again we use the size to scale the strength. Torques drawn with large radii arcs are made stronger. The position of the arrowhead (clockwise vs. counterclockwise) is used to indicate the direction of the torque.

Size and other features of the drawing of the constraint itself are not the only information we can use to set parameter size, however. In different contexts, an object drawn the same way can take different meaning. Similarly, constraints drawn in different places may be intended to carry different strengths.

Torques are an example of such a constraint: the magnitude of a torque is dependent on the distance of the force from the center of rotation. Consequently, it
is logical that our representation of a torque should also be dependent on this dis-
tance. To achieve this goal, torques with a larger such distance carry larger strength
in ASSIST. In order to accomplish this, however, we needed to decide from where to
measure this distance.

The naive approach is to choose the center of rotation to be the center of mass of
the body on which the torque acts. Because our bodies all have uniform density, this
is easily calculated. If the body is jointed, however, such as the example in Figure
4-4, the center of rotation is no longer the center of mass, but the location of the
joint. In ASSIST, pin joints, motors, rotational springs, and rotational dampers can
all create such a joint. If there is a single joint attaching the body to the background,
the distance used to scale torque magnitude is the distance to that joint, not the
center of mass.

The situation becomes more complicated if there are multiple joints, or if the
body is jointed to other bodies. Consider the example in Figure 4-6. Which point
should be used as the center of rotation for estimating torque strength? The center of
rotation of the body or of the entire device? What point will this device rotate around
in steady state? It will depend on the locations of the connections, and the collisions
that take place once the rotation begins. Thus determining the center of rotation
of this group of bodies is tantamount to calculating the path of the simulation. It
is not the intent of ASSIST to be a mechanical simulation program. For all more
complicated cases than the single joint case, the center of mass of the body is used.
This is an acceptable choice, as this value is only meant as an intuitive indication of
the lever arm of the torque, not as a precise value.

4.4 Re-recognition and Pruning

As can be seen by the previous section, this work is concerned with more than simply
adding new devices to ASSIST. It is also important to add functionality that enhances
the user experience, either by making the representations more intuitive, or by giving
the user more power. This section describes changes in the way objects are re-
Figure 4-7: A motor is indicated by a circle and a trapezoid. Because the motor is a point constraint, the cleaned up version appears at a fixed size.

recognized by ASSIST that allow the drawing to grow and evolve. Changes are made to allow all higher level recognitions to be used as subcomponents of larger devices, to allow strokes to be reinterpreted if they are moved to a new location and thus a new context, and to allow relationships between objects to change as they move apart or together.

4.4.1 Motor

One device which led to some of these issues is the motor. The symbol chosen for the motor is a circle above a trapezoid (Figure 4-7). This symbol was inspired by the symbol used by the simulator to represent it. The pivot point of the motor is the center, and as with the pin joint, the user’s strokes are replaced by an icon of fixed size.

The motor is recognized from the lower level objects circle and quadrilateral. This led to problems with the original recognition tree in ASSIST. In the absence of a circle to form a motor, a quadrilateral will most likely be recognized as a quadrilateral body. Quadrilateral bodies were recognizable either from quadrilateral objects, or from polygon objects which have four sides. Thus a quadrilateral stroke has the recognition tree shown in Figure 4-8. This tree contains two distinct quadrilateral bodies. When the reasoning determines that a quadrilateral body is the most specific interpretation, it sees these two interpretations as equal, and chooses one of these arbitrarily.

Because of the way that ASSIST stored recognitions, only those recognitions from which the chosen recognition descend directly were kept in the widget pool. This
meant that if the quadrilateral body on the right in Figure 4-8 was chosen, there would be no quadrilateral in the widget pool for the motor recognizer to recognize from in later strokes.

One change that was made to address this problem was to change the recognition to be more consistent. Quadrilateral bodies should descend only from quadrilaterals, and not the less specific polygons. This allows only one recognizer to be concerned about verifying the properties of a quadrilateral, and prevents the inconsistencies in the recognition tree.

Although this change makes sense for this specific case, this difficulty is indicative of a larger problem. As will be seen in section 4.4.2 there are cases in which the alternative recognitions are needed for future stroke interpretations.

4.4.2 Pruning: Ropes and Pulleys

As seen in the discussion above, Alvarado's ASSIST pruned recognitions that were not selected by the reasoners. Only those recognitions from which the chosen interpretation descended directly were kept in the widget pool for future recognitions. For example, if a circle was recognized as a pin joint based on its context, a geometric
Figure 4-9: The recognition tree for a straight line. The rod interpretation will be chosen, and under the old pruning algorithm, only the interpretations indicated by the red dashed line will be stored in the widget pool. This prevents future recognizers from building on the rope interpretation.
circle recognition would be kept, but alternate recognitions such as a circular body would be pruned. Keeping the geometric circle recognition available allowed future recognizers that needed low level shapes to re-recognize the strokes based on further context.

As discussed in section 4.2.1, the ropes added to ASSIST can be recognized from any open stroke. This means that both straight lines and curved lines can be considered ropes. In the case of a straight line, although the rope recognition is found, the rod recognition is preferred. As illustrated in Figure 4-9, if the user does not immediately correct the system using Try Again, the rope recognition would be pruned from the widget pool, leaving only the rod recognition and the straight line recognition to be used by future recognizers.

As ASSIST grows more complicated, however, we cannot assume that future recognizers only build upon low level recognitions. More sophisticated recognizers will also make use of alternate high level recognitions. The pulley, for example, is intuitively made up of ropes, not lines or curves, and ideally we would like to build on the rope recognitions. Under the previous method of pruning, however, once a straight line was recognized as a rod, the rope recognition would no longer be available. The only option would be to design the pulley recognizer to recognize from the lower geometric primitives, which obscures the hierarchy of the device.

In order to allow for more hierarchical recognitions, we cannot remove alternate recognitions from the widget pool. If we disable this pruning, then the rope recognitions will remain, and they will be available for use by the pulley. This has a number of benefits. First, it is a more intuitive representation. Pulleys are made up of ropes, not of lines and curves, and this hierarchical representation is clearer. In addition, it allows slack ropes and taut ropes to be handled identically. In both cases, there will be a rope recognition in the widget pool that can be found and used by the pulley recognizer.

Obviously there is a downside to disabling pruning. Pruning was implemented as a performance enhancement, and disabling it slows down the system. Each recognizer has a larger set of recognitions to look through when trying to find possible
subcomponents. Although this performance decrease is not prohibitive in practice, it is perceptible. One possible solution to this is to implement a more sophisticated pruning algorithm. A possible approach to this is discussed in section 5.1.2.

4.4.3 Re-recognition of Moved objects

The modifications discussed above in section 4.4.2 allow high level recognitions to be re-recognized as their context evolves with the addition of new strokes to the surface. In this section, we describe work that allows strokes to be re-recognized as a result of moving them to a different location on the surface. Here too the context changes and the strokes should be considered in their new context.

In order to achieve this we need to systematically re-recognize any moved stroke. For a single moved stroke, the solution is straightforward. Complications arise as more strokes are included by using a selection gesture or selecting a complex object made up of more than one stroke.

To implement this goal, we first create a collection of all selected widgets. Once these widgets are stored, we leave selection mode. This is important to insure that the strokes are not re-recognized in the context of a selection. If we were to allow this, then the strokes being moved could be interpreted as editing gestures. For example, a straight line that had been a rod could be recognized as a deletion gesture, deleting itself and all other moved strokes. This change leads to a small change in functionality. Previously, moved widgets would remain selected at their destination. With this change the widgets will no longer be selected after their move. This is, however, a sensible approach. The moved widgets may combine with strokes at their location to build higher level widgets. It would be confusing to the user to have parts of these new widgets selected and other parts not selected.

The next decision that needed to be made was at what level we wished to re-recognize the strokes. As a first attempt, we implemented a re-recognition algorithm which re-recognized strokes as if they were newly drawn, that is, that ignored all current recognitions. In order to prevent confusingly different interpretations, the original order of stroke drawing was maintained, so that any order dependencies in
the original recognition would be reproduced in the re-recognition.

In order to accomplish a complete re-recognition of all strokes the user moves, all possible interpretations on the surface are examined. Those that were squiggles (uninterpreted strokes), were stored in a list. Then all widgets in the selection were removed, so that they could be re-interpreted independently of their current recognitions.

Before passing the selected widgets to the recognition system, we assign them new time stamps. This is done in order to maintain order information for the temporal reasoner. The temporal reasoner applies the reasoning that strokes drawn in the same time period as each other are more likely to belong to the same object. This reasoning is implemented based on the assumption that the user’s focus of attention is on the strokes he has drawn most recently. However, in the case of moving objects, it is clear that the user has returned his attention to the strokes being moved. Thus if these strokes are to be re-recognized it is appropriate to treat them temporally as newly drawn strokes.

Once the new time stamps are applied, the strokes are passed in their original order through the main recognition system. This system treats them as newly drawn strokes, and applies all the recognition and reasoning that would be given to them as such. The new interpretations for the strokes are placed on the surface at the new location. An example of this for a circle re-recognized as a pin joint is shown in Figure 4-10.

This approach works well for cases where all moved objects are made of single strokes. The stroke is given a chance to re-recognize based on the context in which it is placed. For multi-stroke devices, however, re-recognizing at the stroke level is not necessarily the correct approach. Consider the example of a motor. Because the motor is a point constraint, the user’s strokes are resized to a constant size on re-recognition. Imagine the user’s surprise if upon moving the small motor into a small body, he finds the circle of his motor re-recognized into a new shape which redraws his circle larger than the original body.

In order to solve this, we re-recognize at a higher level than the unrecognized
Figure 4-10: A small circular body (left) is re-recognized as pin joint when its context is changed (right)

stroke level. Instead of re-recognizing each stroke, we re-recognize the lowest level recognition that groups strokes in the same way. For example, if we move a pulley, we want to re-recognize the pulley itself for possible extensions to make larger pulleys, but not the individual circles and lines that make it up. In the case of a square body made up of four lines, we wish to re-recognize at the level of the square, not assuming it is a body, nor breaking it down into individual lines. This approach allows the maximum freedom in re-interpretation that won’t break groups that the user perceives as already decided.

4.4.4 Gear

A device that takes advantage of and expands upon this idea of moved objects being re-recognized is the gear. Gears are represented as tangent circles in ASSIST. Drawing one or more tangent circles creates a gear train of circular bodies whose rotations are related by their gear ratios. Gear trains are formed very similarly to pulleys; when two or more gear trains are found to combine to build a larger gear train, a new widget is constructed to represent the larger collection. There is additional complication from the fact that each gear can connect to an arbitrary number of other gears, but careful
An important difference from pulleys, however, is the fact that gear trains are made up of subcomponents that function independently of the constraint. What we consider to be a gear is simply a circular body that happens to interact with another circular body. Ideally, the user should be able to seamlessly separate these circles and have them be recognized as independent bodies, or place them tangent to one another and have them be recognized as related gears.

In order to achieve this, gear trains must be handled differently from other devices in ASSIST. ASSIST does not, in general, allow parts of devices to be separated once they are recognized and accepted as a single item. In order to allow individual gears in a gear train to be moved, the circles themselves are not recognized as part of the gear train. Each circle is recognized as an independent gear widget, and a gear train widget, containing no strokes, is constructed to handle and record the interactions between gears. This allows the user to select and move individual gears without requiring them to move the whole train.

As gears are moved, however, their meanings may change. Consider the two gears in figure 4-11. As one is moved away, it is re-recognized as discussed in section 4.4.3. Because it is no longer tangent to another circular body, it is no longer recognized as a
gear, and instead is recognized as an independent circular body. This still, however, leaves the problem of the gear that is left behind. As it is not being moved, the mechanisms in section 4.4.3 do not apply. Thus the gear train must be aware of which gears are leaving its train, and initiate re-recognition where needed.

To aid the gear train in managing the gears that come and go, the gear train has pointers to all gears that are a part of it, and conversely, each gear has a pointer to the one and only gear train of which it is a part. Given this structure, the naive approach is straightforward. When one gear is moved or deleted from the train, it notifies the train of which it is a part. The train then calls the re-recognition engine on all of its gears. Care must be taken here. As each gear is re-recognized, it will be deleted and replaced by whatever new recognition is made. It is important that this second deletion from the train does not cause a recursive notification to the train to re-recognize the gears again. The train itself also dies, to be replaced by the new train that will be generated during the recognition of the gears.

There is a subtlety in this approach that needs to be handled, however. Consider a train with three gears. If the user deletes two gears simultaneously, the above algorithm will fail. The first gear will be deleted, triggering a re-recognition of gears two and three. New interpretations for these circles, likely as gears that make up a gear train of size two, will be added to the widget pool. Next the system will attempt to delete the second gear. However, this gear will already have been replaced by new interpretations, and its links to those interpretations removed. Thus on its deletion, these new interpretations will remain, and from the user’s perspective the second gear will remain.

In order to solve this problem, we need to be able to roll back the re-recognition of a gear if we discover that it is also being removed in the same user action. In order to accomplish this, we temporarily store re-recognitions of the gears that we re-recognize in the gear train. By the next stroke, this gear train will no longer be active, because if there is more than one gear left, they will be re-recognized into a new and distinct gear train. Thus, if another gear from this train is removed, we know that it is part of the same user action, and we can safely roll back its re-recognition by deleting the
new interpretations from the surface.

Using this improved method, gears can be added, rearranged, moved away, or deleted, and both the gears and gear trains will be consistent with what remains on the surface. This allows users a considerable amount of flexibility in designing and redesigning their gear trains.
Chapter 5

Future Work

This thesis had a number of goals. The desire was to enhance ASSIST by adding functionality both in the form of new devices, and in exploring new approaches to different aspects of the user interface. More than this, however, it is the hope that this work will serve as a stepping stone to future advancements, both in expanding and enhancing ASSIST, and in providing insight and experience to future sketch recognition systems.

5.1 Additional enhancements to ASSIST

This work described a number of enhancements made to ASSIST. Although many avenues were explored, there are still a number of changes that could increase the usefulness and intuitive feel of this system.

5.1.1 Curved Bodies

Currently, ASSIST supports two kinds of bodies: polygonal and circular. These bodies are based on the original low level recognizers used in the system before complex recognitions were accepted from the toolkit. With the addition of complex recognitions, it is possible to include non-circular bodies which contain a curved portion. Currently there are no recognitions for these closed complex strokes.
By including complex bodies, ASSIST would be improved in two ways. The first is that a recognition would be provided for closed complex strokes. Currently, there is no mechanical part associated with these strokes, which means when the user draws them, they are given no meaningful recognition. This can be frustrating to a user who feels the system can't interpret their strokes.

More importantly than this, however, is the ability to include curved bodies in a design. Currently all bodies must be approximated as piecewise linear. By allowing bodies with curved edges, the user can more accurately and expressively describe a variety of designs.

5.1.2 Pruning

As discussed in section 4.4.2, the original pruning algorithm implemented in ASSIST was disabled as part of this work. While this allowed a wider variety of hierarchical recognitions, it also created a performance cost. The recognizers have a larger number of widgets to sort through, and this results in a small but perceptible delay in the recognition.

Although pruning alternate recognitions prevents hierarchical recognitions of the kind described in section 4.4.2, interpretations that will not be used in future recognitions can safely be removed. In order to determine which recognitions are available for pruning, one could survey the input types of the current recognition module. Each recognizer has a function to return its input types. From these input types, a list could be formed containing those widgets that are leaves in the recognition tree. Building the list from the recognizers assures that it is accurate based on the current system without requiring it to be reprogrammed explicitly when the set of recognizers changes. In addition, this list would need be built only once, at start up time, and the recognizers would not need to be queried for each pruning step.

Once this list is formed, the recognitions in the widget pools can then be filtered based on it. Any recognition not on the list of possible input types can be discarded, as it is not a possible subcomponent, and as a result it cannot be part of future devices built with the involved strokes. This pruning keeps useless previous interpretations
from slowing down the system, while leaving useful interpretations available for future recognition.

An example of the usefulness of this pruning can be seen in the case of rods. In the current system, rods are not sub-components of any devices, so this pruning system would reduce the widget pool significantly, as any straight line could be recognized as a rod and the widget pool therefore contains a rod recognition for any single straight-line stroke.

This is just one example of a possible approach to pruning the widget pools. Other methods for pruning or otherwise optimizing the performance of the recognition would be an interesting area for future work. This would increase the natural feel of the system, and make it feel more like paper, as the user would always be able to draw and be understood without delay.

5.1.3 Further Re-recognition

Section 4.4.3 describes the re-recognition of moved strokes. This re-recognition allows strokes to be considered in whatever context they are placed. This functionality makes the interpretation more adaptable and more responsive to the user’s input.

Even with this added behavior however, strokes are still not always interpreted in their most up-to-date context. When a stroke is moved, it is re-recognized in its new context. The strokes that it is moved near, however, are not in all cases re-recognized in the new context of the strokes moved towards them. If they combine with the strokes moved, they will be interpreted, as usual, as part of higher level devices. If they would become different but independent recognitions as a result of their new context, however, they are not given a chance to discover this.

To understand this case, consider the example from Figure 4-10. In this case the small circle is moved into the body, and re-recognized as a pin joint based on its new context. Consider instead what would happen if the rectangular body was moved to a position surrounding the circle. The body would be re-recognized starting from the level of a rectangle. The context of the pin joint does not affect its recognition, so it remains a rectangular body. The circle however, is not given a chance to re-recognize,
and also keeps its original interpretation as a body. In its current context however, it is more likely intended as a pin joint.

This idea is seen already in the handling of gears. When one circle is moved near another both the moved circle and the stationary circle are re-recognized. In this specific case, this change is handled as part of the gear recognizer. When the gear recognizer finds two tangent circles, it replaces both the one it was recognizing, and the one it used to determine that it was likely a gear. In this case, it is sensible to handle this recognition in the gear recognizer. The second circle is recognized as being part of the same gear train, so its change due to context is more direct than the pin joint case.

In order to resolve this problem in the more general case, we would need to pass the affected strokes through the recognition system independently. It would not, however, make sense to re-recognize the entire drawing each time a stroke was moved. It would be necessary instead to determine a sensible sphere of influence near the stroke's old and new positions. Perhaps strokes that intersected, contained, or were contained by the moved stroke, either in its new or old position, are the strokes that should be re-interpreted. One thing that could be a part of future developments to ASSIST would be further investigation into this problem and an implementation of a sensible choice.

5.1.4 Refining the Temporal Reasoner

The temporal reasoner is responsible for the knowledge that strokes drawn during the same time period are more likely to belong to the same device than those that are temporally far from each other. In its current form, if the strokes are from different time periods it ranks any recognition that maintains the strokes as separate above any recognition that combines them into a higher level shape.

Occasionally this choice leads ASSIST to chose recognitions that surprise the user. This is usually caused by a situation discussed in the resolution portion of section 2.2.1. When contradictory information is received from the reasoners, the disputed recognitions are combined into a single node in the reasoning graph. If this node is
chosen as most likely, a recognition is chosen from it arbitrarily.

Higher level mechanical parts are always considered preferable to geometric interpretations, and geometric interpretations are preferable to uninterpreted strokes. If the strokes are from different time periods the temporal reasoner will prefer both the uninterpreted strokes and the geometric interpretations to the mechanical part that combines them. This leads to a cycle, and as discussed above, all these interpretations are compressed into a single node.

Once these interpretations have been compressed, any selection of interpretations from them is possible. This means that a square could have each of its four lines interpreted differently, one as a rope, one as a rod, one as a geometric straight line, and one as an uninterpreted squiggle. Alternatively they could all be chosen as rods, or ropes, or they could be combined into a square body. These inconsistent recognitions can be frustrating for the user.

Although there is contradictory information, it should still be possible to make a consistent and sensible choice regarding the strokes in question. Ideally the most highly ranked choice that did not combine the time-separated strokes should be consistently chosen. In order to accomplish this, either a more sophisticated algorithm would need to be created for handling cycles in the reasoning, or the temporal reasoner would need to be applied independently, only after the reasoning described in section 2.2.1 had been applied. Investigating this problem would lead to more intuitive recognitions in these cases.

5.2 Multi-modal Interfaces

Sketching is an important tool in describing mechanical systems, but it is not the only means by which designers convey information about the systems they are designing. Multi-modal interfaces allow the engineer to convey information about his sketch in a number of intuitive ways.

An engineer describing a system to a colleague will likely sketch the system, but he will also likely describe it verbally. In some cases the speech will simply support the
sketch, "And here there will be a pendulum" and in some cases it will provide information that is not apparent in the sketch "Here we see three evenly spaced pendulums." In both cases, the information in the speech can be used to help disambiguate the users strokes. This work is being pursued in [1].

In addition to speech, people often use gestures when describing sketches. Consider a user describing the function of a car. He may say "The motor drives the wheel which spins like this." This would presumably be accompanied by a gesture representing rotation. This gesture may contribute additional knowledge about the movement of the system described by the sketch, for example the direction of the rotation of the wheel. This work is also currently being pursued by a member of our group.

5.3 Next Generation Recognition System

ASSIST is designed as a sketch recognition system for the mechanical engineering domain. This and other work has been done on a number of systems that recognize objects within a given domain, for example the domain of software design approached in [4]. Each one of these systems, however, needs to be programmed from the ground up, coding in all the geometric and domain specific knowledge for a given task. The next step in sketch recognition research is to build a generic system that can be used in multiple domains, and it is hoped that the lessons learned in this thesis can provide insight to the next recognition system.

One such system is being built by other members of our group based on a blackboard architecture.[2] This architecture is based on the idea of many knowledge sources interacting with a central data store (the blackboard). An architecture such as this has many benefits. The modular nature of the knowledge sources allows them to be easily swapped in and out, changed, and updated. As a result, it is easy to change to a different domain by simply replacing the knowledge sources. The base recognition engine can be reused and remains unchanged.

Another goal of this system is to make the knowledge sources easy to construct.
Rather than programming each source individually, the goal is to let experts from each target domain describe the constructs they need in an intuitive language.[5] These descriptions could then be passed through code generation to produce knowledge sources that understand the components and constraints required for each device. In addition, work is being done to allow these descriptions to be learned by simply drawing a few test examples.[8]
Chapter 6

Conclusion

Sketching is an important medium for designing and describing systems in many domains. ASSIST is a system that combines freehand sketching with the power of a simulation system to allow engineers a natural means by which to interact with a computer.

The work of this thesis was to extend ASSIST in several meaningful ways. The first of these was to expand the number of low level shapes used by ASSIST. This allowed more freedom in the types of strokes that could be drawn by the user, and consequently, a greater ability to include recognizers for a variety of devices. In addition, the vocabulary of high level devices was greatly expanded. Many of these built on high level recognitions, or were arbitrarily expandable in a way ASSIST had not previously allowed. This allowed a significantly more extensive set of designs to be drawn and simulated.

In addition to adding more recognizers for low and high level shapes, more substantial changes in the functionality of ASSIST were made as well. The concept of parameter magnitude was added to several constraints to allow the user to specify relative magnitudes in his sketch. In addition, the concept of re-recognition was explored extensively. One important result of this is that strokes moved to new contexts have the opportunity to be re-recognized in these new contexts. This change makes the system much more flexible. Strokes are not forever set once they have been recognized and accepted, they can be moved and changed as the sketch evolves.
Future work stemming from this research comes in several forms. More extensions are possible to ASSIST, and possible directions stemming from the work in this thesis are discussed in Chapter 5. In addition, the ideas implemented here and the lessons learned are valuable to future sketch recognition systems.
Bibliography


