massachusetts institute of technology — artificial intelligence laboratory

Segmentation and Alignment of Speech and Sketching in a Design Environment

Aaron D. Adler

AI Technical Report 2003-004 February 2003

Segmentation and Alignment of Speech and Sketching in a Design Environment

by

Aaron D. Adler

Submitted to the Department of Electrical Engineering and Computer Science in partial fulfillment of the requirements for the degree of

Master of Engineering in Computer Science and Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February 2003

Certified by: Randall Davis
Professor of Computer Science and Engineering
Thesis Supervisor

Certified by: Howard Shrobe
Principle Research Scientist
Thesis Supervisor

Accepted by: Arthur C. Smith
Chairman, Department Committee on Graduate Students

Segmentation and Alignment of Speech and Sketching in a Design Environment

by

Aaron D. Adler

Submitted to the Department of Electrical Engineering and Computer Science on February 3, 2003, in partial fulfillment of the requirements for the degree of Master of Engineering in Computer Science and Engineering

Abstract

Sketches are commonly used in the early stages of design. Our previous system allows users to sketch mechanical systems that the computer interprets. However, some parts of the mechanical system might be too hard or too complicated to express in the sketch. Adding speech recognition to create a multimodal system would move us toward our goal of creating a more natural user interface. This thesis examines the relationship between the verbal and sketch input, particularly how to segment and align the two inputs. Toward this end, subjects were recorded while they sketched and talked. These recordings were transcribed, and a set of rules to perform segmentation and alignment was created. These rules represent the knowledge that the computer needs to perform segmentation and alignment. The rules successfully interpreted the 24 data sets that they were given.

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

Thesis Supervisor: Howard Shrobe
Title: Principle Research Scientist

Acknowledgments

I want to thank my advisors Randall Davis and Howard Shrobe for all the advice and help that they have given me particularly with my thesis, but also in general.

I also would like to thank Jim Glass, Scott Cyphers, and the rest of the SLS group for their help with speech recognition.

I would like to thank Project Oxygen for their financial support.

I would like to thank all the people who sketched and spoke on videotape, this work would not have been possible without you.

I would like to thank all the members of the Design Rationale Group, particularly Christine Alvarado, Mike Oltmans, and Metin Sezgin for developing ASSIST, without which this thesis would not have been possible.

I would like to thank Gary Look, Harold Fox, and Max Van Kleek for being great officemates and for offering helpful advice. I would also like to thank Jacob Eisenstein for a few LaTeX pointers.

I would like to thank my parents and sister Rachel for their support, encouragement and valuable feedback about this thesis.

Finally, I would like to thank Leah Frank for being so wonderful, supportive, and helpful. I would also like to thank my friends for offering encouragement and support.

This thesis is dedicated to the brave crew of STS-107: Rick D. Husband, William C. McCool, Michael P. Anderson, David M. Brown, Kalpana Chawla, Laurel Blair Salton Clark, and Ilan Ramon. They gave their lives returning from a mission that was dedicated to exploring new scientific frontiers.

February 1, 2003.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Sketches are a convenient and expressive way to communicate information. Sketches can be used when trying to explain something to a colleague, or to describe an idea or concept during a meeting. The sketch is often augmented with a verbal explanation that provides more detail and enhances the ideas and concepts in the sketch. The verbal description can be used to clarify parts of the sketch that are hard to explain visually or to express things that are too hard or too complicated to sketch. This thesis lays the groundwork for creating a multimodal system that will allow a user to use sketching and speech simultaneously to create a diagram of a mechanical system.

Figure 1.1 depicts a simple sketch of two blocks that fall, impact a triangular object[1], and continue falling onto a platform which has a pivot at its midpoint. Several small details affect the operation of this system: if the triangular object does not have sides that have the same slope, the blocks will bounce off at different angles and have different impact times with the platform; if the pivot is not at the center of the platform, the platform will not be balanced. These factors affect whether the blocks fall off or stay on the platform. A simple verbal description would allow the user to create this device more easily and accurately.

Mechanical engineers often sketch initial ideas out informally. New ideas may take the form of such a sketch, whether it is on a whiteboard, on a napkin, or on the back of an envelope. However, if they want to enter this information into a computer aided drafting (CAD) program they have to start all over again, drawing it with a mouse

[1]The "X"s in the illustration represent anchors, which make objects stationary even under the influence of forces, such as gravity.

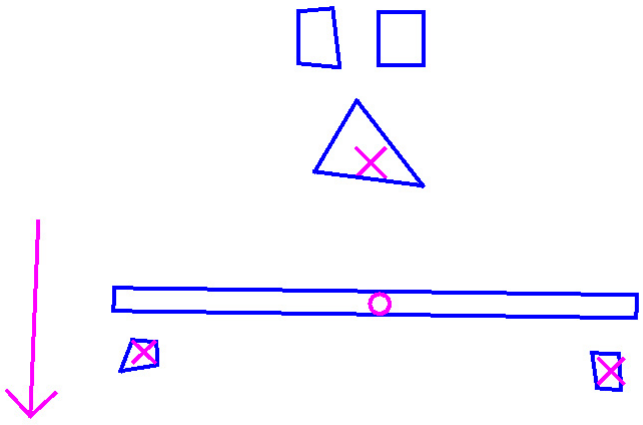


Figure 1.1: A simple mechanical system whose behavior changes dramatically depending on its dimensions.

and a keyboard. Eventually the drawing may be entered into a computer, but frequently the original drawing is discarded, taking with it the information and design rationale it contained. Members of the Design Rationale group previously developed ASSIST (A Shrewd Sketch Interpretation and Simulation Tool) and ASSISTANCE (ASSIST Augmented with Naturally Conveyed Explanations) to help address this problem. ASSIST interprets sketches and can simulate the physics of the resulting system [1]. (See Section 2.2.1 for more details.) ASSISTANCE builds on this by allowing users to describe the behavior of a device after it is drawn [15] (see Section 2.2.3 for more details). These tools assist the users in the early phases of design by allowing them to see how early design concepts would work.

This thesis builds on the previous work by examining how to add speech recognition capabilities to ASSIST so that the system can take advantage of verbal input, in addition to the sketched input that it already uses. We believe the resulting multimodal system will be considerably easier to use. Adding speech recognition to the system will move us toward our goal of providing as natural a user interface as possible.

1.1 Motivation

Let us assume that a teacher is trying to explain the mechanical system in Figure 1.2 to her physics class. This diagram consists of two symmetric and identical ramps each with an identical ball on it. At the bottom of the ramp, there are three equally spaced pendulums. Below the ramps there is a bin to catch the balls. Table 1.1 shows a possible order for the sketching actions and speech phrases that the teacher might use when describing the device.

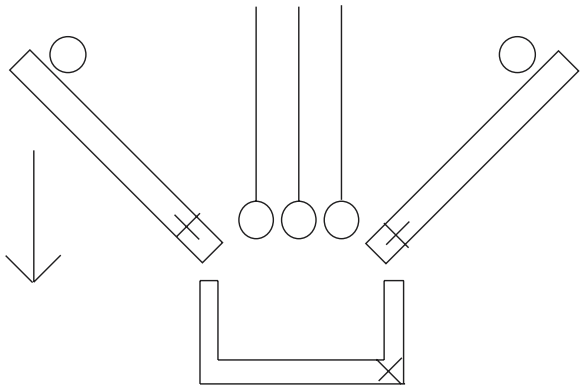


Figure 1.2: A mechanical system that might be sketched.

Although the explanation of the system might be perfectly clear to a class full of students, a computer might have considerable difficulty understanding the drawing. Is the computer sure that the teacher said "we'll" or was that "wheel"? Was that a circle that was just drawn or was it a complicated polygon? Were those just two overlapping lines or was it really an "X" indicating an anchor? With just one modality, the computer may not be able to decipher all of the user's actions. By allowing both input modalities, each input can be used to disambiguate the other in a process called *mutual disambiguation*. Mutual disambiguation allows the resulting system to be more accurate[18] by using the sketching input to help decipher the speech input, and vice versa. It also provides the user with a more natural and powerful way to interact with the computer. For more information about mutual disambiguation see Section 7.1.

The problems of segmenting and aligning the input correctly are important ones. First, to segment the input, we need to group the events that are related in each modality. The grouping is done by analyzing

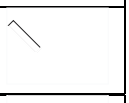
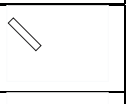
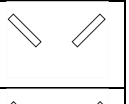
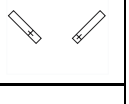
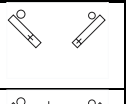
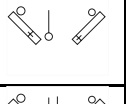
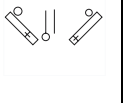
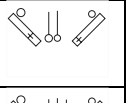
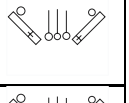
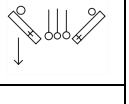
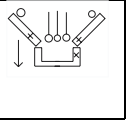
Speech Utterance	**Sketch**
"Next, we'll draw a mechanical system"	
	
"There are two <ummm> symmetric ramps"	
	
"Each of the ramps is anchored to the surface.	
"The two ramps have an identical ball at the top."	
	
	
"At the bottom of the ramps we have three <ahhh> equally spaced pendulums."	
	
	
	
"There is gravity in the system, so the balls will roll down the ramps and hit the pendulums."	
"To catch the balls we have a box at the bottom of the ramps.	

Table 1.1: A possible sequence of events for sketching and talking about Figure 1.2. The sketching actions and speech utterances that occur on the same line happen at the same time. The sketched images show the state of the sketch at that particular time.

the content and timing information of the speech and sketching. This process segments the inputs into groups of sketching events that are related and groups of speech events that are related.

The speech and sketch inputs that refer to the same object do not necessarily occur at the same time [18]. For example, a user might say "we have some balls up here" and sketch five circles. However, the utterance might have started before, after, or during the actual sketching of the objects. The sketching may also have preceded the utterance or exceeded its length to a varying degree. Based on the timing data of the sketching and speech event, we align the two inputs so mutual disambiguation can be performed.

The example above in Table 1.1 exhibits several characteristics that are explored in more detail in this thesis. The relative order of the sketching and verbal commands varied throughout the explanation: sometimes the teacher would sketch first, sometimes she would talk first, and sometimes she would talk and sketch at the same time. A system that handles multimodal interaction needs to be able to *handle any input order*.

The teacher also used several key words that make the multimodal interaction more *powerful* than just sketching. She gave a considerable amount of information by using words such as "symmetric" and "identical".

The multimodal interaction also allows the sketching interaction to be *simplified*. It would be nearly impossible to get the balls, ramps, or pendulums to be truly identical in a sketched environment, but by adding the verbal interaction, it becomes quite easy. In a CAD environment, it is possible to produce identical parts, but it involves copying and manipulating and the use of menus, which is more involved than simply stating verbally that something is "identical." The interaction is also *more accurate* because some of the information is duplicated in the two input modalities – for example, the word "anchor" and the action of drawing an "X".

1.2 Segmentation and Alignment

The first step in the process of mutual disambiguation is figuring out exactly what it is that we are disambiguating. The fact that the physics teacher said "an identical ball at the top" tells us that we should be looking for two balls. But we do not want to compare that verbal utterance to the part of the sketch where the teacher sketched the pendulums.

This thesis looks at this problem of segmentation and alignment. The first step is to segment the input. This means figuring out which parts of the sketch are related, for example, figuring out that the first few stokes are all part of the ramps, and that the next strokes formed the balls. The same sort of grouping has to be done for the speech. Once these groups are established, the next step is alignment. Alignment involves associating the sketching actions and speech utterances that involve the same things. Another way of looking at it is determining a time boundary between topics. Alignment can be done with a simple time boundary because it was observed (see Section 3.2.4) that people do not draw one thing while talking about another.

This thesis does not address the actual disambiguation within the groups of sketching and speech events; instead, it focuses on accurate segmentation and alignment, two critical steps that set a foundation for disambiguation.

1.3 Contributions

This work makes several contributions. It examines a collection of videotaped data of users sketching and speaking at a whiteboard and uses this information to develop a set of rules to aid in aligning and segmenting the sample data. The rules use as little information as possible from the inputs so that the aligned inputs can later be mutually disambiguated. The rules performed well when they were tested on the sample data from the videos and compared to a hand segmentation of the same data.

These rules will subsequently be used to align and segment actual data from ASSIST and from a speech recognizer. The videos show how important disfluencies and pauses are when trying to match the audio input with the user's sketching. Disfluencies are speech utterances such as "ahhh" and "ummm." The disfluencies indicate user thought and are a delaying tactic. This can provide important clues, even though the utterances do not provide any information by themselves.

The goal of this work is to provide the user with a natural user interface for early mechanical design work. By adding multimodal capabilities to the existing sketching program, it should be able to handle any input order from the modalities, and provide the user with a simple, powerful, and more accurate way to communicate with the computer.

1.4 Structure of this Thesis

Chapter 2 of the thesis provides some more background information, discusses the problem in further detail, and provides several motivating examples that lay out why this problem is both interesting and important. The thesis proceeds to describe the steps taken toward achieving the goal. Specifically Chapter 3 describes the gathering of data to examine the problem more closely – first audio data and then video data. The thesis then describes the analysis of the data and the various methods of segmentation – by hand and using a rule-based system in Chapter 4. Chapter 5 discusses modifications to ASSIST based on what was learned from the data gathered. It concludes with a discussion of related work in Chapter 6 and future work in Chapter 7. The conclusion can be found in Chapter 8.

Chapter 2

Understanding the Problem

This chapter provides several examples that illustrate the power of multimodal interaction and explores additional background information about the prior work on the various parts of the sketching system.

2.1 Examples

2.1.1 Newton's Cradle

There is a system of pendulums called Newton's Cradle that consists of a row of metal balls on strings. When you pull back a number of balls on one end, after a nearly elastic collision, the same number of balls will move on the other end of the system. Figure 2.1 illustrates two of the possible sequences of events for Newton's Cradle. This is a relatively simple system that can be used to show Newton's energy conservation principles – conservation of linear momentum and conservation of energy. Although this system seems simple enough to sketch, it is in fact nearly impossible to sketch so that it operates properly. The system works because the metal balls at the end of the pendulums just touch each other, and each pendulum is identical to the others. In the sketching system, you would have to draw all the pendulums exactly the same, and then align them perfectly. This is very difficult to do, but if the user could simply say that there were "five identical, evenly spaced and touching pendulums," the device would be easy to create. This illustrates that speech can be used to clarify things that

cannot be shown by sketching alone.

2.1.2 Car on a Hill

A common example used to demonstrate ASSIST is a sketch of a car on an incline. We can then run a simulation of the car rolling down the hill in "Working Model 2D," a physics simulator by Knowledge Revolution, Inc., as shown in Figure 2.2. This example can also be enhanced with speech. If a user informed the system that the car has "two identical wheels" the system could make the two wheels identical and also center the pin joint in the middle of the wheel. This accuracy is very difficult to achieve by just sketching input. A similar example is illustrated in Figure 1.2, where the user wants to have symmetrical ramps with identical balls.

2.1.3 Other Types of Examples

Other more detailed properties of the devices could be expressed verbally. Working Model allows its users to specify the material of objects (e.g., wood, metal, or ice) and thus change properties such as the coefficients of friction. Currently, there is no way to specify these properties from ASSIST, but with speech it would be easy to verbally articulate such properties.

Although it is not our primary focus, a speech interface would allow a user to give verbal commands to the system along the lines of "make three copies of the pendulum" or "move the pendulums closer together." The system could handle simple drawing commands, but we want to maintain the natural feel of the system. In other words, the goal is not to create a system where the user can just use voice as a substitute for menus – rather our desire is for an intelligent interactive system that tries its best to understand what the user is doing and help them complete their tasks.

Some potential problems require a deep understanding of what the user intends to do. For example, even a simple utterance such as "make three equally spaced pendulums" can be complicated. With no sketching input, it would require knowing how to space the pendulums and how big to make the pendulums. Thus, a combination of sketching and speech has the potential of providing the best of both worlds.

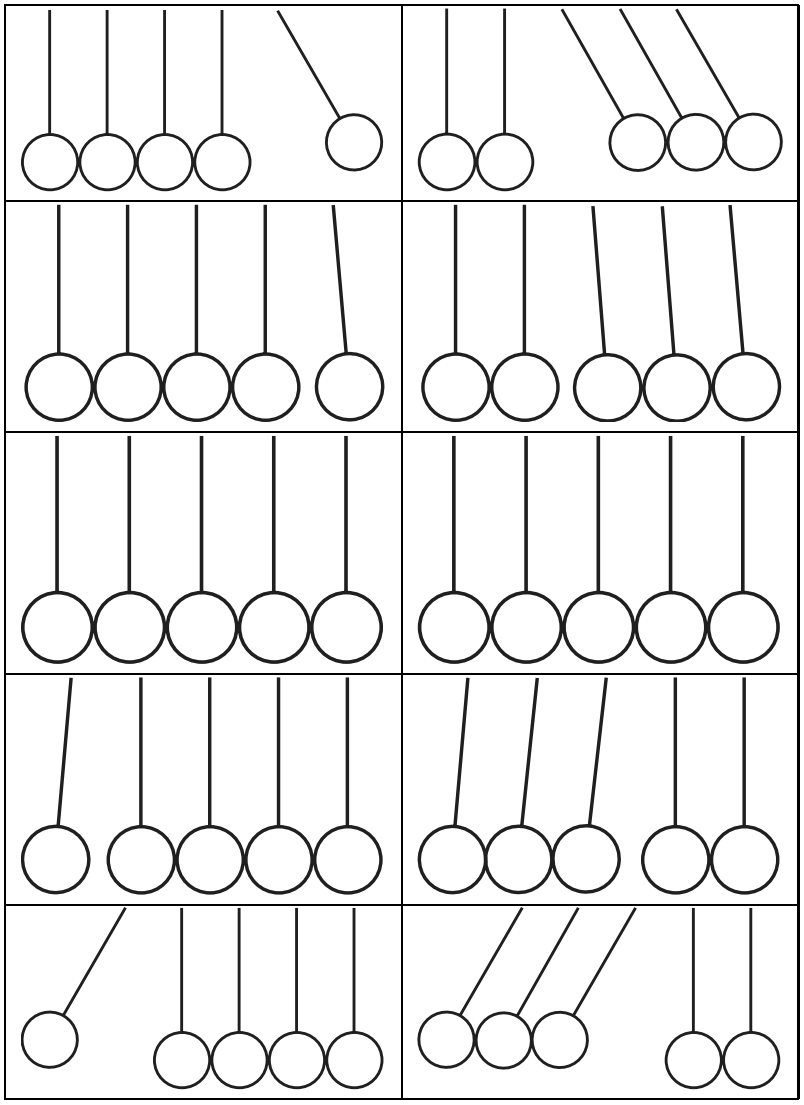


Figure 2.1: The left side of this figure shows Newton's Cradle when one of the pendulums is pulled back and released. The right side shows what happens when three pendulums are pulled back and released. This system depends on the pendulums being identical and just touching each other.

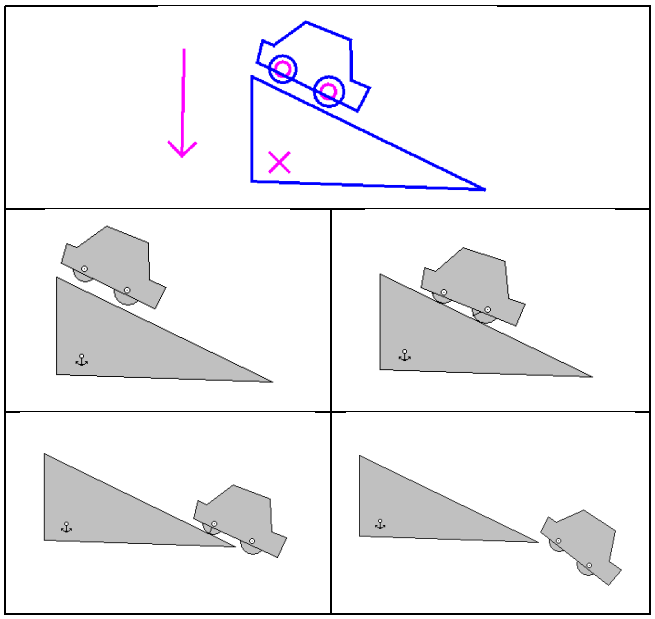


Figure 2.2: The top image shows the sketch in ASSIST. The bottom figures show the simulation sequence in Working Model.

2.2 Prior Work on the Tool

This thesis builds on previous work on the sketching system by members of the Design Rationale group. Specifically, it uses the ASSIST system [1] as well as the recognition toolkit[22]. It also uses some of the ideas developed in ASSISTANCE[15] which also accepts verbal input.

2.2.1 ASSIST

ASSIST[2, 3] allows the user to sketch in a natural fashion employing a variety of stylus-style input devices. Currently, the user can provide input to the computer using a mouse, a Mimio^{TM} marker on a projected display surface, or on a tablet PC; all of these send time-stamped position data points. The user's drawing is displayed on the display surface and when the system recognizes part of it, it cleans up the drawing (see Figure 2.3) and changes the line color to reflect that it understands what the user has drawn. The user can undo interpretations if the system makes a mistake, and can move or delete parts of the sketch. Figure 2.4 shows the ASSIST window that contains the "Try Again" button to choose a different interpretation and the "Run" button to start the simulation.

The system currently recognizes mechanical engineering drawings, and ASSIST can interface with Working Model, which allows the user to view a simulation of the drawing (see Figures 2.2 and 2.5 for an example). The simulation provides valuable feedback to the user because it allows him to see mistakes and adjust the parameters of his model accordingly.

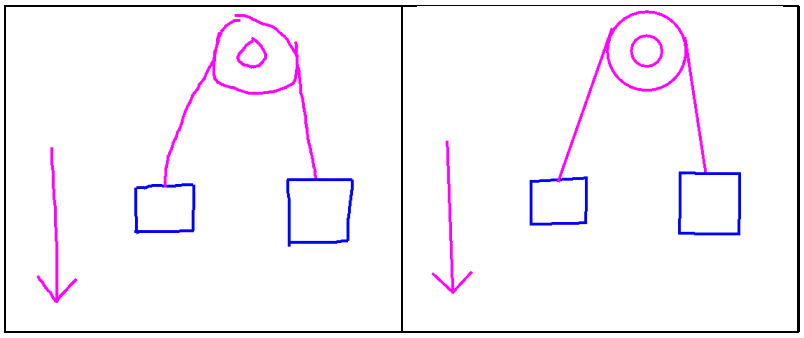


Figure 2.3: The raw (left) and cleaned up (right) version of the user's strokes in ASSIST.

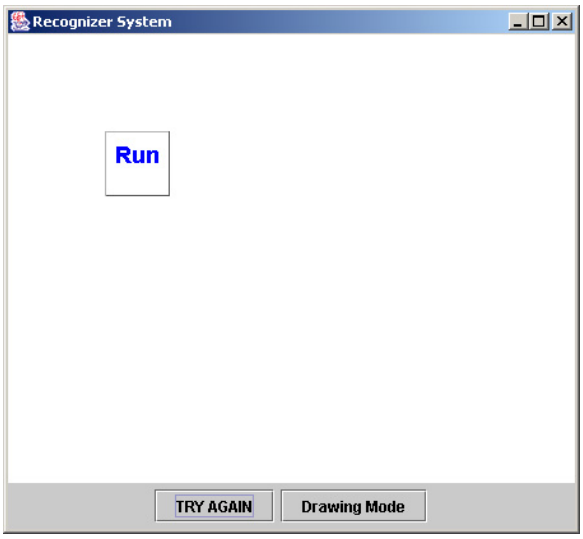


Figure 2.4: The ASSIST window.

Luke Weisman and Manoj Muzumdar developed the RecSystem (Recognition System) [23, 11]. Christine Alvarado built on their work to create ASSIST. ASSIST refined the original work by allowing the interpretation of drawn objects to change as the user draws. This permits recognition of more complex domains, such as the mechanical engineering domain. In the mechanical engineering domain, the system can recognize objects like pulleys, springs, pivots, pin joints, dampers, blocks, rods, and strings. For more detailed information about ASSIST, see Section 5.2.

2.2.2 Toolkit

The toolkit[22] was built by Metin Sezgin. His Masters thesis focuses on low level recognition of the user's strokes. His system uses multiple sources of information, such as curvature and pen speed, to detect features and produces an interpretation of the stroke in terms of lines, curves, ovals, and polygons. The toolkit also provides the system with a cleaned up version of the stroke.

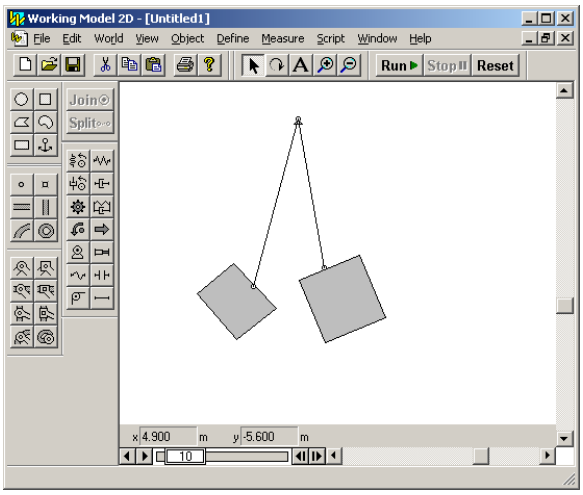


Figure 2.5: A screen shot of Working Model.

2.2.3 ASSISTANCE

ASSISTANCE[16] is a tool built by Mike Oltmans on top of ASSIST. It provides another layer to the program by allowing the user to draw a sketch in ASSIST, and then use additional sketching and voice annotations to describe the behavior of the system. ASSISTANCE forms a behavioral model for the device and can answer questions about its behavior. This provides the user with a more natural and less formal interface for the early stages of design.

2.3 Creating a Natural Design Tool

The motivation behind the work of our group is to create a design tool that is natural for the user. The system should provide the user with feedback about their design, enabling the user to design more efficiently and more easily. The tool must be less trouble than it is worth so that users are willing to use it. The computer should, in effect, be looking over the shoulder of the user trying to understand what the user is doing to the best of the computer's ability.

In the domain of speech and sketching, some ideas are more easily expressed verbally and others are more easily specified visually. In an effort to create a natural interface for the user, we want to provide

both speech and sketching capabilities to the users and allow them to choose the modality they want to use. It is not clear what balance of speech and sketching users of the system will choose; it could vary depending on the user, the task, or the situation (e.g., if the user is the only one in the room). We imagine a natural design tool allowing the user to sketch and talk about their design to the computer as naturally as talking about the problem with a colleague.

CAD drawing tools are good for entering the final design into a computer, but are not good at the initial design stage because they require details of the design to be known. Often the initial sketch of an idea is done on the back of an envelope before the idea is well formed enough to use a CAD tool. The idea behind ASSIST is to capture such early designs and the rationale behind them. The tool must be both natural and easy-to-use so that it is appealing to the user.

The tool also has the potential of being used in educational environments as a collaboration or teaching tool. For example, students could work out physics problems using our program, providing both verbal and speech inputs and then simulate the resulting system to see what happens. A teacher could also use the system in a similar way – explaining a physics or mechanical engineering problem to the class by sketching on the board and explaining the system as she drew it. The system could be listening to the verbal explanation and use this to help disambiguate sketch elements. After the explanation a simulation could be run to show the result.

2.4 Motivation for Adding Speech

The existing ASSIST system does a good job at sketch recognition, but some things are inherently hard to draw and get right. By adding speech recognition to the system, the user could explain what they were drawing at the same time they were sketching it. This would allow the user to have a natural conversation explaining their sketch to the computer just like a conversation they could have with another person. This interaction will hopefully allow the users to provide the system with more information and allow the system to capture the sketch in more detail and the rationale behind the design.

We do not want the speech to be limited to simple, single word commands. In other words, we want to avoid having speech like "block" while pointing at a location. Rather, we want to allow the user to say whatever comes to mind and have the system gather everything it can from the speech input. The user should not think about the computer

on the other side of the system; the user should be able to have a natural interaction with the system while talking and drawing naturally.

Speech will allow the system to do things that are not currently possible with the sketch interface. Working Model is capable of giving objects textures, which can change properties like the coefficients of friction, however, there is not currently a way to specify such things with the sketching interface. Adjusting the sketch would be easier with verbal interactions than with sketched input. For example, if the user doesn't like the spring constant in the simulations, they could ask the system to change it with a combination of voice and gesture.

Chapter 3

Obtaining Sample Data

To investigate how the speech and sketch inputs correlate and to develop a better understanding of how users might intersperse the two input modalities, it was necessary to obtain sample data. Two varieties of data were collected. First, audio data were collected, then video data. This chapter describes the data collection and analysis.

3.1 The First Attempt: Audio Recordings

The initial experiment was designed to gather data to better understand vocabulary usage and speech patterns of users when they spoke and sketched simultaneously. Three test subjects were asked to sketch six sample figures on a whiteboard and concurrently describe the pictures. The subjects were tape recorded as they sketched.

3.1.1 Generating Sample Figures

Ideas for the sample figures were sketched in ASSIST. To create cleaner and more symmetric drawings, the ASSIST sketches were captured and cleaned up in a drawing program. The six devices (shown in Figure 3.1) were chosen to be types of mechanical systems that someone might draw in ASSIST. The devices were loosely based on concepts found in a college physics textbook such as pendulums, ramps, springs, and pulleys. From left to right, top to bottom the sketches are: a car on a hill; a system of ramps, balls, and pendulums; a box with a spring platform and some balls; a block on a ramp connected to another block

with a pulley; two blocks connected by a pulley with no gravity; a ball on a ramp that will hit two pendulums. The same color scheme that ASSIST uses was maintained in the figures.

3.1.2 Results

The test subjects were asked to take small versions of the drawings and enlarge them on the whiteboard. The tape recordings of their speech were then digitized to facilitate the analysis of the results. The speech was transcribed and the results examined for clues indicating how to segment the input into pieces that could be associated with the different parts of the figures. (See Appendix A for the audio transcripts.)

Several observations were made from watching the live sketching and listening to the recorded transcripts. The different subjects had different styles for describing the devices. One of the subjects tended to comment verbally on the drawing after he drew it, rather than before. Other subjects tended to work through the device aloud. Figure 3.2 shows two contrasting transcripts from two subjects who sketched the ramp system that can be seen in Figure 3.1. The first subject had a pretty good idea about what she wanted to say, but the second subject did not think about how to describe the device before he started talking about it.

Pauses in speech are important because they are a good indicator of a change in topic. Since pauses are easy features to detect, they provide a good tool to segment the audio input. It was also observed that subjects tended to say words like "ahhh" and "ummm" when they were stalling for time. Some of the subjects had different interpretations of the figures than the intended interpretations. One subject thought that the device with the spring platform had just been released and had thrown the balls into the air, whereas our original interpretation of the device was that the balls were about to fall onto the platform that started at rest.

3.1.3 Insufficient Information

The data from the transcribed audio recordings and the empirical observations of the test subjects were not detailed enough to determine the nature of the relationship between the sketching and the speech. The empirical observations could only hint at what a more detailed analysis might yield.

Another point of interest was the vocabulary that was used for describing parts of the devices that were hard to draw. We were trying

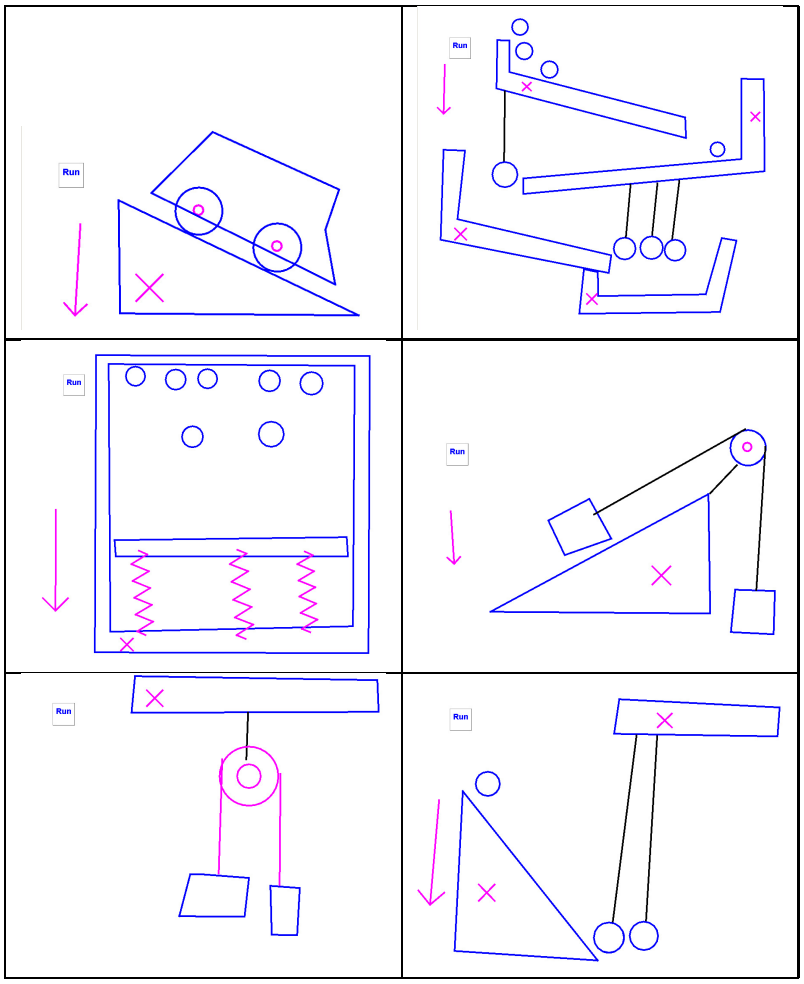


Figure 3.1: Left to right, top to bottom, the six sketches for the audio data.

So there are a set of ramps <pause> <ahh> fixed in free space with a number of <ahh> balls suspended from the bottom of the ramps, and a number of free balls <umm> rolling down <ahh> the collection of ramps under the influence of gravity.

<laughter> Our second example is a Japanese Pachinko machine. And for the Pachinko machine what we're gonna have is a series of slides. One like that. And that's fixed. Another one that looks like <pause> that goes in the other direction. like that. And you have two more <pause> like that. So the balls are gonna fall – they're gonna start up here and they're gonna slide down this way like this. And there's gonna be a pendulum, right there hanging from the first ramp. And a series of three pendula <pause> pendulum? like that. And again we are on a planet where gravity is like that.

Figure 3.2: The first transcript is from a subject who thought about the device before sketching. The second transcript if from a subject who thought about the device aloud.

to discern what vocabulary people would use to describe something like Newton's Cradle (see Section 2.1.1), where there were identical components in the device. We wanted to observe what parts of the diagrams people would naturally talk about and which parts they would naturally sketch.

As previously mentioned, the figures used to gather the data were based on the ASSIST drawings. The figures did not indicate the similarity or congruency of any of the objects in the drawing. For example, although in Figure 3.1, in the upper right device, the pendulums are all identical, this is not indicated in any way. As a result, the test subjects did not use the sort of vocabulary that we hypothesized. For example, no one talked about parts of the drawing being "identical".

Another issue was that the test subjects were all familiar with ASSIST. This resulted in the subjects doing things that they knew ASSIST did because the figures looked like ASSIST drawings. It is possible that users without knowledge of ASSIST would use a different vocabulary.

3.2 The Second Attempt: Video Recordings

As an improvement to the audio recordings, video recordings were made of new subjects sketching and talking about six different figures. These recordings led to more detailed observations about the nature of the multimodal interaction. The observations ultimately led to the development of a set of rules that can be used to segment the speech and sketching inputs (see Section 4.3.4).

3.2.1 Setup

The setup of this experiment was similar to the setup of the audio recording experiments. Six subjects were asked to sketch six figures at a whiteboard. Small versions of the figures were attached to the whiteboard. The subjects were told to enlarge the figures and pretend that they were describing the sketches to a small group of people – like a physics tutorial. A video camera recorded what each of the subjects sketched and spoke (see Figure 3.3). The subjects were chosen to have limited familiarity with ASSIST and the purpose of the experiment.

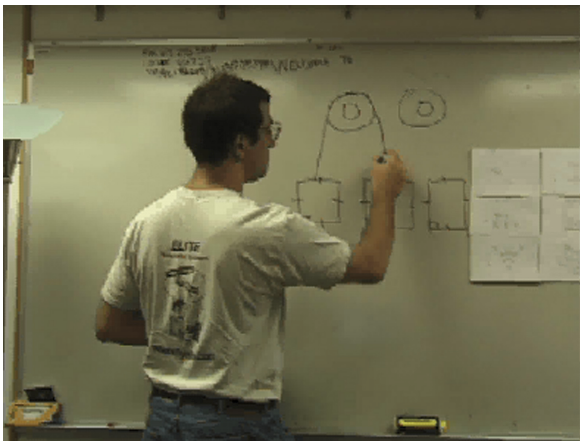


Figure 3.3: One of the subjects sketching at the whiteboard.

The figures being sketched were also revised to have more symmetry and to include grey lines on them indicating the congruency of various parts of the drawings. Figure 3.4 shows these figures. For example,

if there were identical balls on a ramp in the figure, the balls would have grey slash marks on them to indicate that they were identical. Different sets of congruent components of the figures were notated with a different numbers of slash marks. Parts of the figures also showed identical parts with a numbering scheme. Some of the distances in the figures needed to be noted the same as some other distances. To accomplish this, dotted grey lines with slash marks were used. The idea behind these markings was to use standard geometric notations that the users would understand without giving them any particular vocabulary to use. It was feared that providing a set of vocabulary would bias the subjects toward using that particular vocabulary instead of whatever vocabulary came naturally to them. The devices in Figure 3.4 are, from left to right, top to bottom: a set of identical blocks suspended from identical pulleys; a set of identical ramps, pendulums, and balls; a car on a ramp; three identical balls about to fall onto a spring platform; five identical balls about to fall onto a spring table inside a box; a set of ramps, balls, pendulums, and a bucket.

3.2.2 Data Transcription

The videos were transcribed in a multi-phase process. First, the videos were transferred from Mini-DV tapes into QuickTime format on a Apple Power Mac G4 computer. Then the videos were compressed but maintained the 29.97 frames a second that the video camera recorded. The high frame rate was necessary for later determining when the subjects began and ended their sketching and how this corresponded to their speech.

The most accurate start and end times possible for both the speech and the sketching were desired. The easiest way to do this was to play back the videos in Adobe Premiere, which provided a display of each frame of video as well as a visualization of the audio track. Each frame of video was about 1/30 of a second. The audio track visualization was very helpful in determining the location of the beginning and end of the verbal utterances. The utterances were separated into the smallest identifiable parts and associated with starting and ending times. By playing clips of the videos and looking at the audio visualization, the start and end times could be determined. Disfluencies in the speech, such as "ahhh" and "ummm", were also noted. All information about the video was recorded in a spreadsheet.

Once the content of the speech and the start and end frames were noted in the spreadsheet, the sketching was analyzed. The video was played back slowly, down to a frame-by-frame speed, to find out the

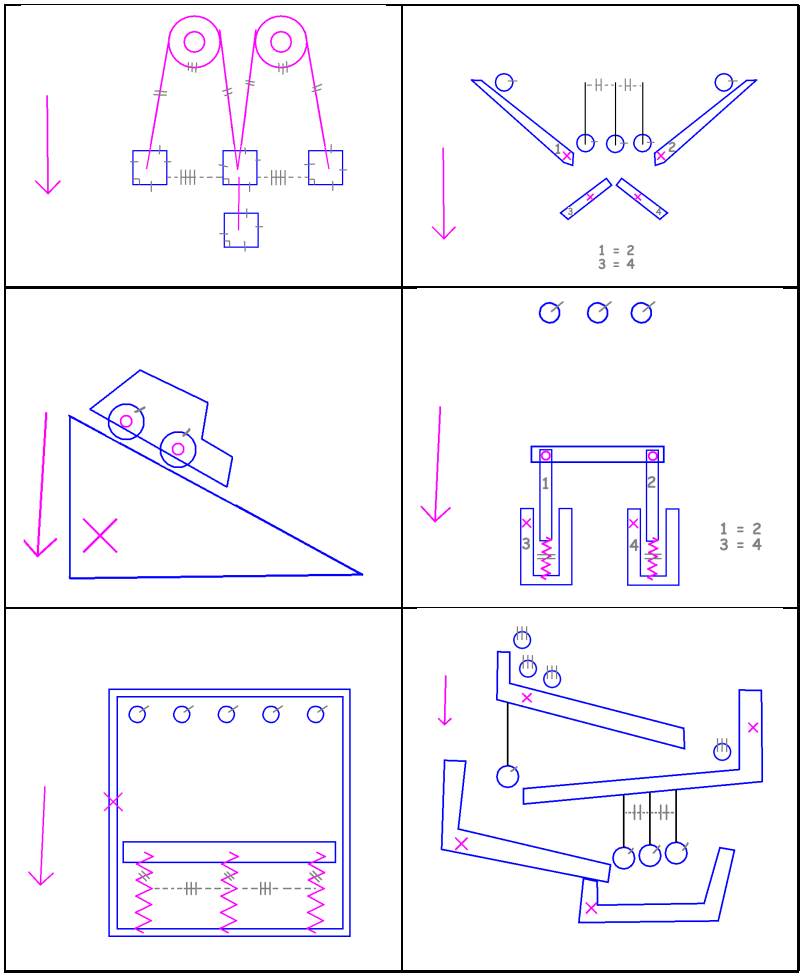


Figure 3.4: Left to right, top to bottom, the six sketches for the video data.

points at which the subjects would start and stop drawing parts of objects. The starting and ending points of a stroke were defined as the times that the subject touched the marker to the whiteboard and the time they lifted it off the whiteboard. Each stroke of a particular object was noted in the spreadsheet. For example, if a subject was drawing a block and drew it with four separate strokes, each of the strokes would be noted in the spreadsheet as a separate drawing action. The drawing actions were denoted in the spreadsheet with brackets "[" and "]" (see the video transcripts in Appendix B). A few lines of a transcript are shown in Table 3.1 as an example. The transcripts were surprisingly accurate, see Section 4.3.1 for details.

Start		End		
sec	frame	sec	frame	Task Name
11	12	11	26	So now
11	29	12	8	we have a
12	9	12	26	box
12	23	13	26	[rectangle:r1:draw draws part of outside box]
14	16	15	16	[rectangle:r1:draw draws part of outside box]
16	28	20	19	[rectangle:r2:draw draws inside box]

Table 3.1: Example of a video transcript.

Several difficulties were encountered in the transcription process. During the audio transcription, there was occasionally considerable background noise that made the transcription difficult. Subjects would also change the pace of their speech dramatically and often. The very nature of the speech also made it difficult to isolate the speech at a word level; it was sometimes hard to tell when one word stopped and the next began. These difficult segments were broken down to a phrase level, which was not a problem because each phrase was about the same topic.

There were also a few difficulties with the video aspect of the recordings. It was occasionally hard to see exactly when a subject started drawing an object because their arm or hand would momentarily obscure what they were drawing. Observing shadows and estimating movements helped to solve this problem.

All of the videos were not transcribed. Only 24 of the 36 videos (six subjects with six videos each) were transcribed. A variety of examples were chosen so that there was some data for each of the six figures. The remaining 12 videos were translated into QuickTime format, but not transcribed so that they could be saved for later use as test data.

3.2.3 Creating Charts and Graphs

It was difficult to make observations from the raw data itself, so several types of charts and graphs were created. The graphs were used to shed more light on specific features that seemed promising. The first set of graphs showed the speech and sketching events and their durations. The goal of this set of graphs was to create a timeline view of the data to view the length of and relationships between the events. The timeline was stretched vertically in the graphs so that all the events are visible. The graphs indicated that the gaps between events might be an important factor in segmenting the input. The gaps seemed to have the potential of helping segment the events into groups about the same topic. An example timeline graph is shown in Figure 3.5 and all the timeline graphs can be found in Appendix B.

A second set of graphs depicted the gaps between events (both speech and sketching). If events overlapped, it appears as a zero on the graph. Otherwise, the number of frames of gap before each event was plotted (see Figure 3.6 for an example and Appendix B for all of the graphs). Then, the particular speech or sketching event that indicated a change in topics was highlighted and compared to the other gap times in the graph. Although most of the events when the topic changed had larger gaps than the other gaps in the graphs, this was not always the case. The length of the gaps also tended to vary within the graph and between graphs. Sometimes there would be large gaps that did not indicate a change in topic, and sometimes the gaps that indicated a change in topic were shorter than gaps that did not indicate a change in topic. For example, in Figure 3.6 one of the longest gaps is for the phrase "we've got a ramp". However, there should not be a start of a new segment with that phase because the drawing of the ramp occurred prior to that utterance. If we were to start a new segment there, we would not group the utterance about the ramp with the sketching of the ramp. The time gap graphs seemed to show that the time gaps alone are not enough to determine when a change in topic occurs.

3.2.4 Interesting Features and Analysis

We had to determine the knowledge that the computer would need to perform segmentation and alignment. We looked at the video transcripts to find features that would be useful in accomplishing this task.

Several key words appear to be a good indicator of a change in topic. Words such as "and" or phrases such as "and then" or "we have" seemed like they might be good clues to a change in topic. See

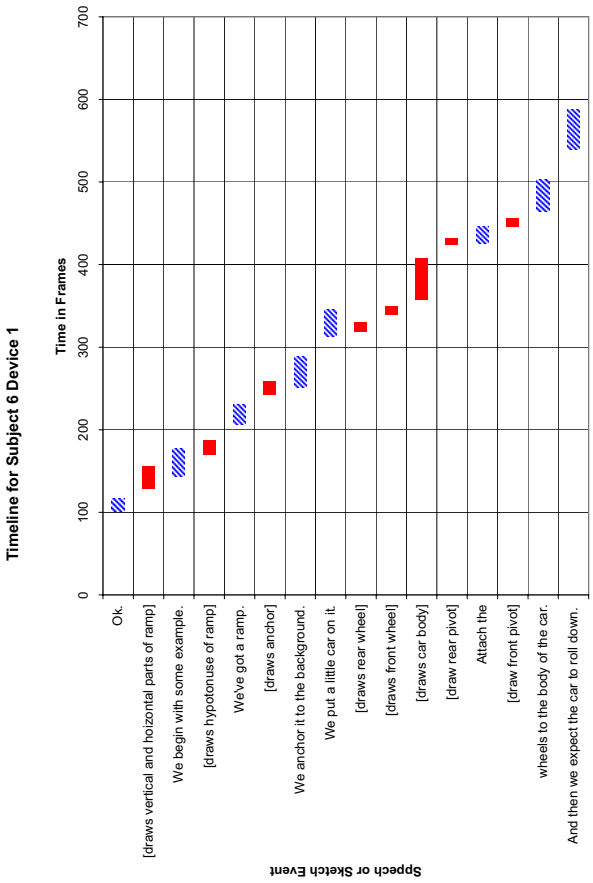


Figure 3.5: An example timeline graph.

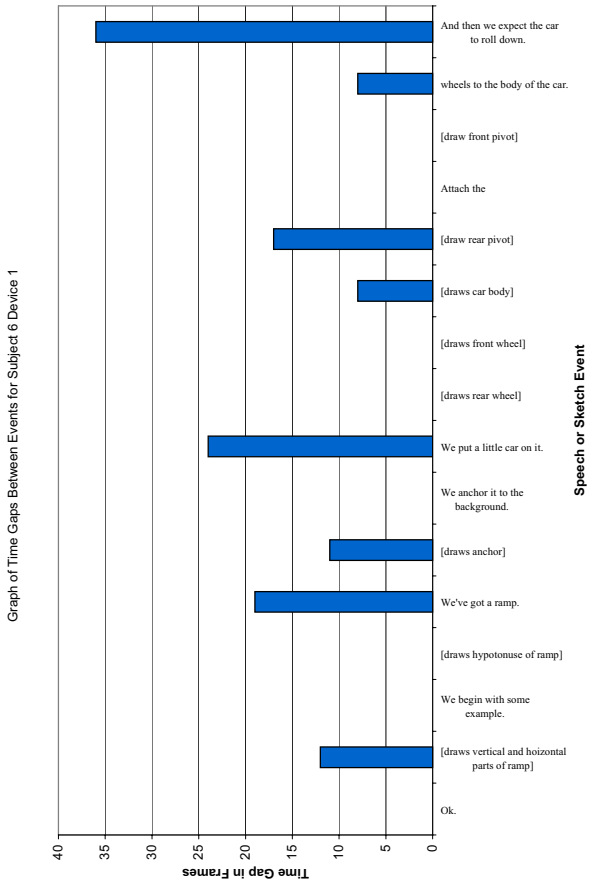


Figure 3.6: An example gap graph.

Table 3.2 for a list of these phrases. Some of these words can be tricky because "and" can be used at the start of a sentence or it can be used as a conjunction in a sentence. The two types of occurrences of the same word imply different things as far as separating the topics of the sentence is concerned. "And" at the start of a sentence usually indicates a new topic, but when "and" is used within a sentence, it usually indicates the opposite – the same topic.

Key Words and Phrases

and	and then	then
so	we have	there is
there are	next	we've got
possibly	ok	it's
with	there's	which
plus	i'll	let's
that's	this is	there are
the		

Table 3.2: Key words that were identified from the video transcripts

In this set of experiments, people did not draw one thing while talking about some other part of the device. This is an important observation because it means that when speech and sketching overlap, the person is talking about related parts of the device, not two distinct parts. Along the same line, a lack of a pause usually indicates that the user is staying on the same topic, while a long pause is most likely an indication of a new topic. Disfluencies in the speech seemed to occur when the user was thinking about what to say or draw next, and could also prove to be a useful feature to apply in the segmentation process.

Subjects also had a tendency to change how quickly they were speaking. When they had a lot to say they spoke quickly, and when they didn't have much to say they tended to stretch out their words. If the rate of speech could be identified, this might help to segment the speech.

The subjects tended to point at different parts of the sketch or gesture at different parts to indicate motion. Other subjects started with a general description and then moved on to discuss the device and draw it in detail. The subjects also tended to draw similar objects together, such as drawing a sequence of balls on a ramp at the same time.

3.2.5 Difficulty Analyzing Key Features

The interesting features mentioned in Section 3.2.4 are all clues that should help figure out how to segment the sketching and speech events. However, it is not easy to segment the data by looking at it. The trends found in the data need to be formalized so that the analysis is more concrete and unbiased by the knowledge of where a change in topics would reasonably occur. The next chapter examines how this can be done.

Chapter 4

Segmenting Data

This thesis focuses on the segmentation and alignment of the sketching and speech inputs. This is a necessary step in mutual disambiguation because in order to use the two inputs to help disambiguate each other, we need to know which speech utterance relates to which sketch components. For example, Figure 4.1 illustrates one way that the speech and sketching inputs might occur. By segmenting the two inputs we can then align them and perform mutual disambiguation on the aligned segments. This chapter describes the different ways segmentation was performed on the data from the videos.

4.1 Segmentation by Hand

To provide a calibration point, the data from the videos were segmented by hand, by examining the Excel graphs as shown in Figure 4.2. Even so, it was not always clear where the different segments should be. Some segments fit into multiple groups. This manual segmentation also helped to determine the knowledge that the computer would need. The knowledge that was used to determine where to segment the data was primarily the observations made in Section 3.2.4.

4.2 Knowledge-Based Segmentation System

The observations mentioned in Section 3.2.4 were used develop a set of rules designed to encapsulate the knowledge that was required to separate the speech events and to separate the sketching events. The

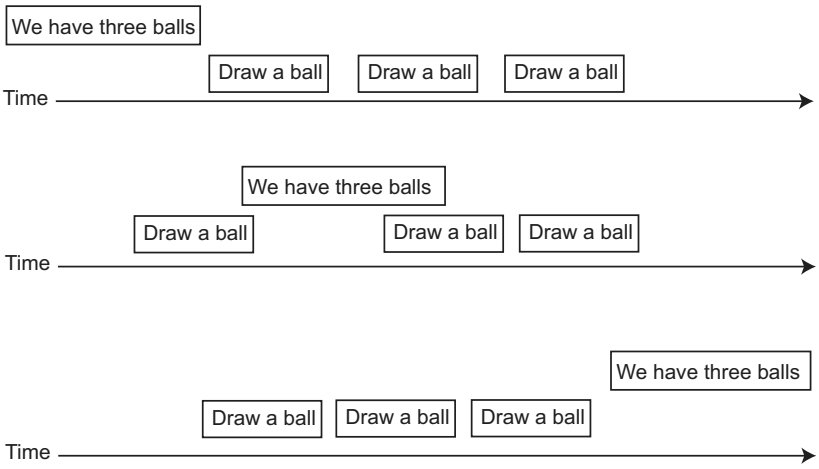


Figure 4.1: Possible orders speech utterances and sketching actions could occur.

rules focused on syntactic knowledge so that additional knowledge of semantics could be used in the disambiguation phase. With this knowledge, the rules segment and align the speech and sketching data that they are given.

These rules were written in Java Expert System Shell (JESS), a rule-based system developed at Sandia National Laboratories. JESS has the advantage that it can interface smoothly with ASSIST, which is also written in Java. The rules were divided into groups corresponding to input files that dealt with the same kinds of knowledge about the events. In addition to the rules that were based on the observations from the videos, which were used to segment and align the data, there are a set of rules that function as a very basic truth maintenance system to remove incorrect assertions.

JESS can be extended with simple Java functions that perform more complex operations. This functionality was useful in creating rules for specialized operations, such as examining the strings that contained the spoken text from the transcripts. For example, a function was created to look for the word "and" in the speech strings.

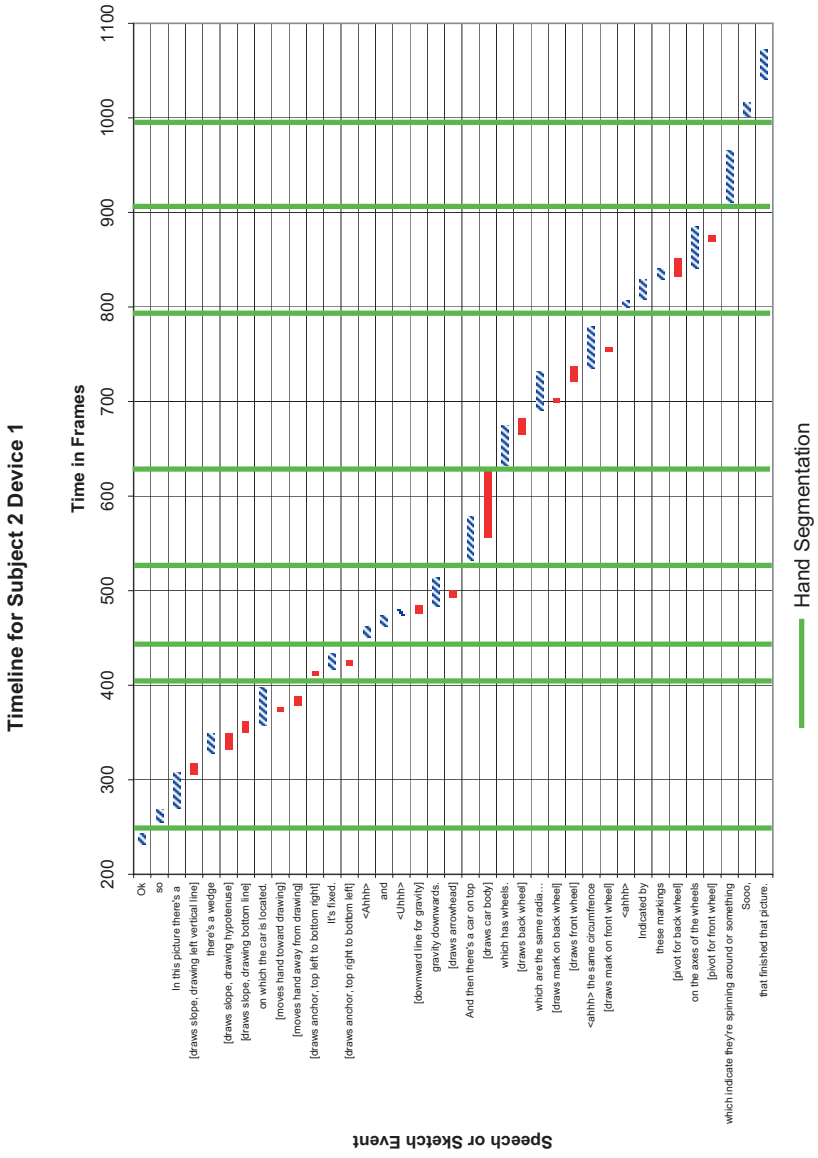


Figure 4.2: An example of the hand segmentation of the data.

4.2.1 Input Files

The Excel files that generated the timeline graphs were converted into comma separated value (csv) format containing the timing information and the text describing the sketch or speech utterance. The text that described the sketch was prefixed by a tag string that described which object the sketched segment was a part of, in the format: [objType]:[objID]:[actionType]. For example, the tag "rectangle:r2:draw" would indicate that the particular sketch segment was part of a rectangle named "r2" and was a drawing action (as opposed to an erasure). The system has no understanding of the objects, in the sense that it does not know the difference between a rectangle and a wheel. Similarly, object IDs (i.e., "r2") are a way to name an object and have no underlying meaning. The tags are meant to simulate simple knowledge that ASSIST would know. ASSIST knows that certain lines make up a rectangle but does not know that the rectangle might be the body of a car.

4.2.2 Problems Sifting Through Data

JESS generates considerable output when executing the rules. The rules have varying degrees of complexity. For example, some of the simple rules create assertions about the time gaps in the input. There are higher level rules that build on the output of simple rules, e.g., a rule that determines if a time gap is large. It became difficult to determine the behavior of the higher level rules. There was a need for a tool to help sift through the data that JESS generated.

4.3 WATCH

To solve the problem of sifting through all the data, WATCH was developed. WATCH stands for "Working Analysis Tool for Chronological Highlights". This tool is used to view the massive output of JESS in a more organized format. WATCH takes the output of the JESS rules as input and displays a colorized timeline of the output with bars indicating the type and length of each JESS rule (see Figure 4.3). Each row in the timeline has a unique color and represents a certain type of assertion. The name of the assertion type appears on the left. More information about a particular instance of an assertion can be obtained by clicking on the corresponding bar as shown in Figure 4.4. Here we have clicked on a bar in the middle of the top row.

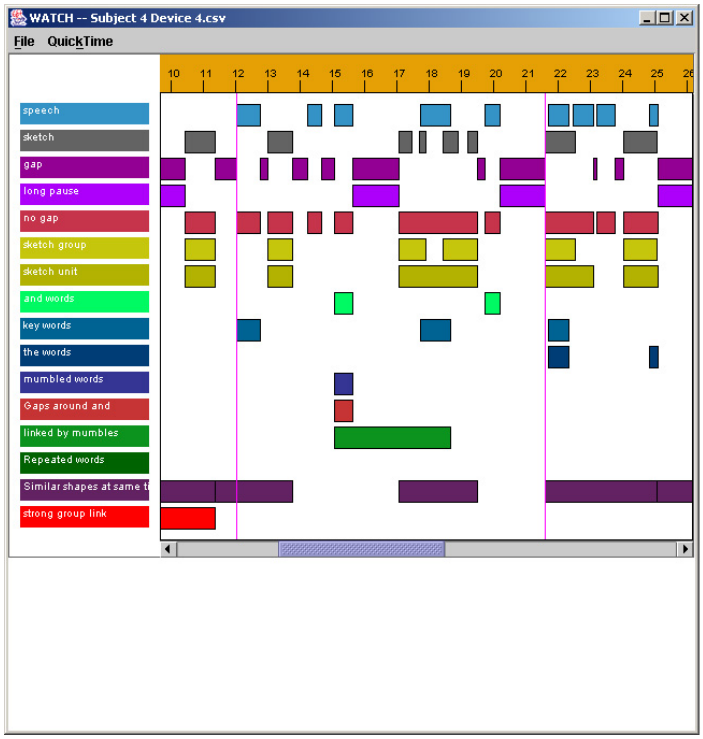


Figure 4.3: A screen shot of WATCH.

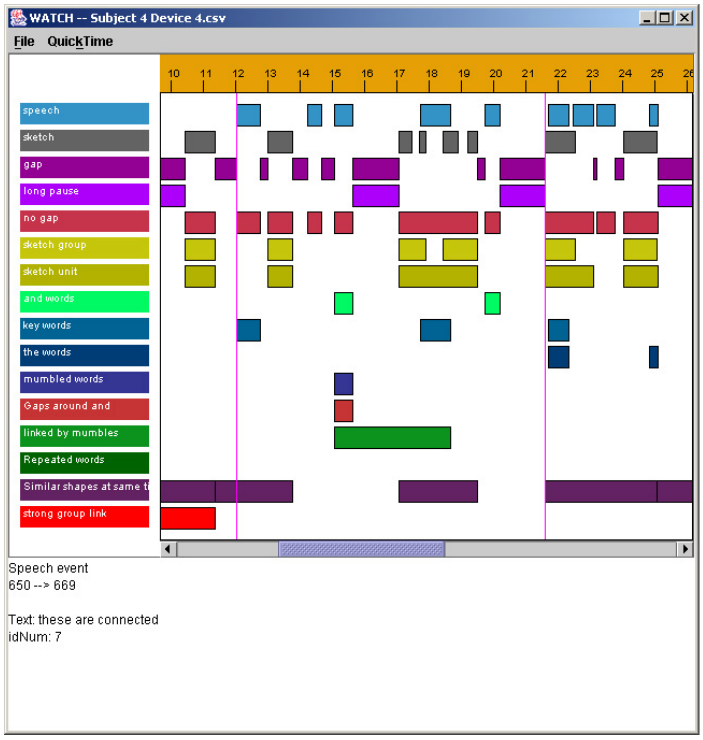


Figure 4.4: A screen shot of WATCH showing the information that is displayed at the bottom of the window when the user clicks on a timeline bar.

4.3.1 Data Verification

A key feature of WATCH is that is that it allows the user to associate a particular QuickTime movie with a data file. When a movie is associated with a data file, the user can view the QuickTime movie file in a separate window inside WATCH, as shown in Figure 4.5. Multiple methods of playback are provided. By clicking on a bar in the timeline, a user can:

- play the movie segment corresponding to the time range of the bar.
- play the whole movie beginning at the start time of the timeline bar.

The movie association proved to be extremely helpful because it could be used to test whether the start and end frame data in the data file were accurate. By clicking on the speech and sketch components, the video segment for only that time range could be played back. This allowed the accuracy of the timing data to be tested because it was easy to verify what was being said and sketched in the short video clip. By viewing all the clips from the videos for the sketch and speech events, the accuracy of the timing data was verified. The largest error in any of the clips was a few frames. The accuracy of the transcription process is more than sufficient to create and test rules on. The actual live data will most likely not be perfect either.

4.3.2 Other Features

WATCH has some other useful features. It keeps track of the most recently opened data files for quick access, and it stores the association between the data files and a corresponding video file so that a user doesn't have to find the movie file more than once. WATCH has a few settings files that allow the user to identify the assertion names and define parameters such as the colors of the bars in the timeline. There is also a settings file that defines what fields the assertions have. These files are used to parse the JESS output and to create the graphical view of the data. Having the settings in files allows the users to adjust parameters of WATCH or add new types of assertions without having to recompile WATCH.

4.3.3 How WATCH Works

WATCH works by parsing the input files (in csv format), creating assertions for each of the sketching and speech events in the input file,

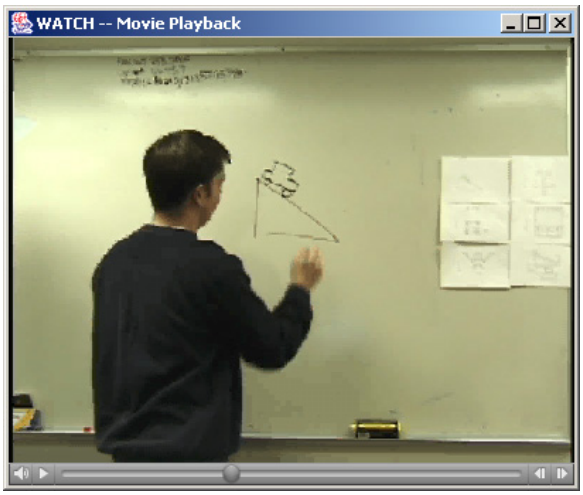


Figure 4.5: A screen shot of the QuickTime playback window in WATCH.

and running rules on the resulting assertion database. The rules are divided into files based on function. These files are grouped by what assertions they depend on. Each rule group is run on the assertion database separately. This is done so that it can be insured that some of the rules run before other rules. By partitioning the rules, the more basic rules can be run before the more complex rules, which depend on the assertions created by the basic rules. When all the rules are run, settings files tell WATCH how to parse all the assertions. Based on the data from the rules, a graphical, interactive, timeline is created. Each type of assertion is translated into a row with its own color. The break assertions (see Section 4.3.4) are treated differently and show up as vertical lines in the timeline instead of bars. These vertical lines are pink and can be seen in Figures 4.3 and 4.4.

4.3.4 Rules in WATCH

The following sections describe some of the rules that are used in the knowledge-based system. The full set of rules can be found in Appendix C. There is an example of the rules running on a small set of data in Section 4.3.5.

The rules determine where the break points occur. These break points serve to segment the input streams, and should separate differ-

ent topics. Figure 4.6 shows the general layout of the types of rules and what is taken into account for break/segmentation calculation. The gaps and no-gap assertions are very important. No breaks can occur that overlap a no-gap assertion. No-gap assertions indicate places where the change in topic cannot occur. For example, we prohibit a change in topic when the user is talking or sketching. The rules are described in more detail in the sections that follow.

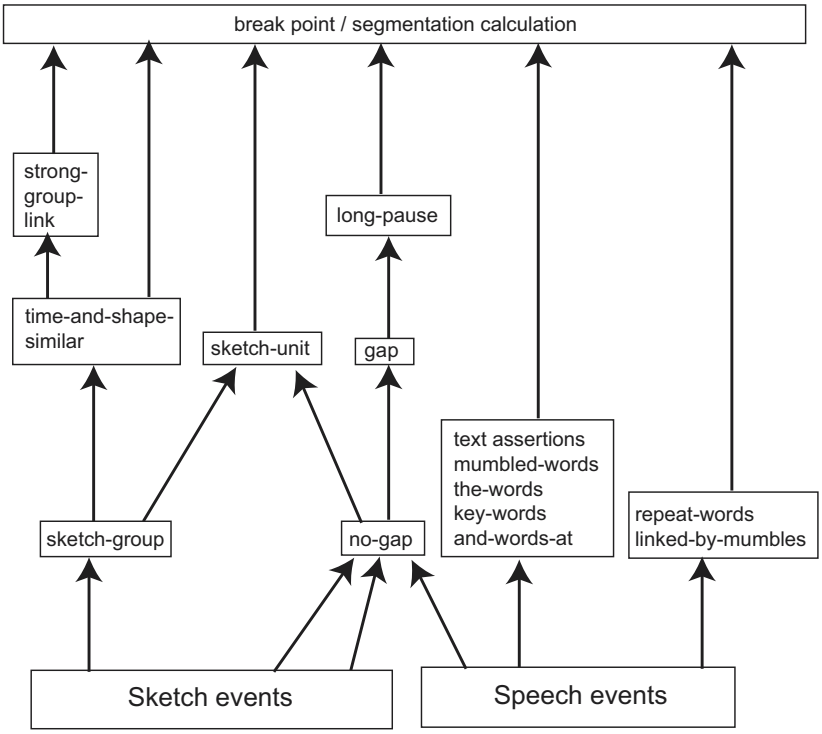


Figure 4.6: This illustration shows the relationship between the various types, the layers of assertions, and the types of assertions that influence the segmentation calculation.

Templates

Templates in JESS allow you to create a structure for assertions. There are three templates that are used by the WATCH system.

sketch template Created from the sketch events in the data file

text The text describing the sketch

objType The type of the object, e.g., Rectangle

objID A string to uniquely identify the object

startTime The starting time (in frames) of the sketch

endTime The ending time (in frames) of the sketch

actionType Indicates if it is a draw event or an erasure

idNum This number indicates the order in which objects were sketched

speech template Created from the speech events in the data file

text The text of the speech utterance

startTime The starting time (in frames) of the speech utterance

endTime The ending time (in frames) of the speech utterance

idNum This number indicates the order in which utterances were spoken

sketch-group template This template represents a group of sketched components that are part of the same object. For example, a sketch-group might be made of the four lines that form a rectangle.

startTime The starting time (in frames) of the sketch

endTime The ending time (in frames) of the sketch

objType The type of the object, e.g., Rectangle

objID A string to uniquely identify the object

groupID A string to uniquely identify the group

idNums A list of all the idNum for the sketched components of this group

The idNum field in the speech and sketch templates is filled as the events are read in from the data file. This is just a counter so that the system can more easily find the next or previous speech utterance or sketch event.

A sketch-group is a group of sketched components that is used to group pieces of the same object. For example, a rectangle that was drawn with four different strokes would have four sketch template-based assertions, but the four sketch assertions would be grouped into one sketch-group template assertion.

Text Rules

The text rules look at the speech utterances and make some assertions about them. There are several groups of words that the text rules search for. The groups are:

- and words: just the word "and"
- the words: just the word "the"
- key words: all the other words from Table 3.2.4
- mumbled words: disfluencies,
  e.g., <ahhh>, <ummm>, <hmmm>

Various search functions were written in Java to implement the search of the speech utterances. The searches look for the words at various locations in the utterances – some rules look at the beginning of the utterance, some look at the end, and some look anywhere. An example rule can be found in Figure 4.7. This rule looks for a speech assertion and then tests to see if the word "and" is contained in the speech utterance. If it occurs at least once, then it adds a new "and-words-at" assertion that contains the start and end times of the speech utterance and the number of times "and" occurred.

```
;;; This rule adds statements for speech text containing and-words
(defrule speech-and-words
 (speech (startTime ?a) (endTime ?e) (text ?b))
 (test (> (num-contains-strings ?b $?*and-words*) 0))
 =>
 (assert (and-words-at ?a ?e num (num-contains-strings ?b $?*and-words*)))
)
```

Figure 4.7: Speech rule for "and".

The link-mumbled-words rule (see Figure 4.8) looks for a "mumbled-words-at-end" assertion. It then looks for a speech event in which the mumbled words occurred and then finds the subsequent speech event. The rule creates a "linked-by-mumbles" assertion with the start time of the first speech event and the end time of the second speech event. It was observed that mumbled words tended to link two speech events together, meaning the user was still talking about the same subject.

The repeat-words rule (see Figure 4.9) checks two successive speech utterances for any overlap in words. If there is any overlap in vocabulary, the rule creates a "repeat-words" assertion that contains the start

```
;;; Rule for linking mumbled words to the next speech utterance
(defrule link-mumbled-words
 (mumbled-words-at-end ?start-1 ?e)
 (speech (startTime ?start-1) (endTime ?end-1) (idNum ?id))
 (speech (startTime ?start-2) (endTime ?end-2) (idNum =(+ 1 ?id)))
 =>
 (assert (linked-by-mumbles ?start-1 ?end-2))
)
```

Figure 4.8: link-mumbled-words rule.

time of the first speech event, the end time of the second, and the number of words that overlapped. When this rule is implemented in the live system, it may need to be adjusted so that common noise words such as "the" aren't considered overlapping words.

```
;;; Rule for indicating the number of words in one speech
;;; utterance that also occur in the following speech utterance
(defrule repeat-words
 (speech (text ?t1) (startTime ?s) (idNum ?i))
 (speech (text ?t2) (endTime ?e) (idNum =(+ ?i 1)))

 ; test the number of overlap in words from the first speech to the
 ; second and fire the rule if there is an overlap
 (test (> (num-words-overlap ?t1 ?t2) 0))
 =>
 (assert (repeat-words ?s ?e num (num-words-overlap ?t1 ?t2) ))
)
```

Figure 4.9: repeat-words rule.

Gap Rules

The next set of rules deals with the gaps between events. The gaps were observed to be a very useful indication of where the segments should be (see Section 3.2.4). Based on empirical evidence from the sample data, a long gap between events was defined as 25 frames or about 0.83 seconds. An "and-gap" used in some of the rules is defined as 7 frames or about 0.23 seconds.

The "no-gap" assertions are created for each of the sketch and

speech events. These assertions start at the beginning of the event and end at the end of the event. The no-gap assertions are also created that span the duration of an overlapping sketch and speech event. The no-gaps assertions preclude break points from occurring during them, and as was observed (see Section 3.2.4), there should not be breaks within speech or sketching events. There is also a rule that adds a no-gap assertion that spans the time that different parts of the same object are being sketched. In other words, if the user is drawing a rectangle, there should not be a gap (leading to a possible break) in the middle of the rectangle even if the user took multiple strokes to draw it. The different parts of the same object are found by looking for sketch assertions that have the same objType, the same objID, but a different idNum (see Figure 4.10 for this rule). The no-gap assertions are combined using another rule to create the longest possible "no-gap" assertions.

```
;;; This rule adds a no-gap assertion between pieces of the same sketched item
(defrule same-object
    ?o1 <- (sketch (objType ?a) (objID ?b) (idNum ?c) (startTime ?s1)
           (endTime ?e1))
    ?o2 <- (sketch (objType ?a) (objID ?b) (idNum ?d) (startTime ?s2)
           (endTime ?e2))
    (test (neq ?o1 ?o2))
    (test (< ?c ?d)) ;;insures rule runs once, not twice
    =>
    (assert (no-gap (min ?s1 ?s2) (max ?e1 ?e2)))
)
```

Figure 4.10: same-object rule.

Gap assertions are filled in between no-gap assertions. Particularly long gaps cause the creation of a "long-pause" assertion. The long pause is useful for the break rules because particularly long pauses between any events usually indicate a change of topic (see Section 3.2.4). There is an exception to this that was observed in the data, namely that if the user is erasing something, the user may not be pausing because he is changing topics, but rather because he is thinking about the erasure. There is a rule that will remove the long-pause assertion if this is the case.

Sketch Group Rules

A sketch group is a group of sketch components that together form a whole object. For example, if a user sketches a pulley, there are many sketch components that make up the pulley. All of the components would be in the same sketch group. A sketch group is created when a sketch event is discovered that does not have a sketch group yet. A rule then creates a new sketch group assertion and increments the unique id counter. Another rule will add other connected sketch events to the sketch group. The sketch group keeps track of the overall start and end times of the sketch events it contains. It also has a list of all the idNums of the sketch components in the sketch group.

The time-and-shape-similar-objects rule looks for two sketch groups that have the same object type. It will then verify that the two shapes were drawn consecutively by making sure that an id number from the second shape is consecutive with respect to the first shape (this is done by the consecutive-ids function). The idea here is that if a person draws two of the same shape in a row it is likely that the shapes are related. The rule creates a "time-and-shape-similar" assertion.

The strong-group-link rule builds on the time-and-shape-similar-objects rule. This rule creates a "strong-group-link" assertion when the time-and-shape-similar assertion does not overlap with any speech event. The theory behind this is if a user draws similar shapes without talking about them, then they are more likely related shapes.

Sketch Unit Rules

A "sketch-unit" assertion is created when a "sketch-group" assertion and a "no-gap" assertion overlap. The "sketch-unit" assertion spans the entire time frame of the sketch-group and the no-gap assertions.

Break Rules

Break assertions are one of the key parts of the system. As Figure 4.6 shows, many of the assertions that were previously created influence the break assertions in one way or another. The break assertions are added in a special way with functions so that a global list of the break points can be kept. This list is used to determine the relationship among breaks. Some of the rules will eliminate breaks that are too close together. Many of the rules are straightforward. The breaks are also typed so that later rules know why a particular break was created. For example, the "long-pause-break" rule triggers on a long-pause assertion

and asserts a break. There are also rules that will create breaks near key words and gaps.

One particularly complicated rule is shown in Figure 4.11. The "sketch-unit-break" rule will assert a break if there is a gap followed by a sketch-unit and if this is not the first sketch-unit since the last break. This involves keeping track of where the sketch units are in addition to where the breaks are. The reason for checking if this is the first sketch-unit since the break is because this rule divides the different sketch-units. If this is the first sketch unit since the break, another break is not necessary to separate this sketch-unit from the previous one. See Figure 4.12 for an illustration.

```
(defrule sketch-unit-break
 (gap ?gs ?start)
 (sketch-unit ?start ?end)
 (test (eq FALSE (first-sketch-unit-since-break ?start
                                                (create$ (length$ $?*breaks*)
                                                $?*breaks* $?*sketch-units*))))
 =>
 (insert-break sketchUnit ?start)
 (assert (break sketchUnit ?start))
)
```

Figure 4.11: sketch-unit-break rule.

Other rules will retract previously asserted breaks. For example, the "retract-sketch-unit-break-similar-shapes" rule will retract a break of type sketchUnit if it overlaps with a time-and-shape-similar assertion. Other rules retract break assertions if they overlap with a link-by-mumbles or strong-group-link assertion.

4.3.5 Rules Running on an Example Transcript

Table 4.1 has part of a transcript in it. Using this transcript we will describe which rules are fired and what assertions are made. For each of the speech events a speech assertion is created, and for each sketching event a sketch assertion is created. The next step that happens is no-gap assertions are created for every time period where there is a sketch or speech assertion. For example, a sketch assertion would be created from frame 487 to frame 506. Then a no-gap assertion is created for the same time range. Once the no-gap assertions are created, gap assertions fill in the rest of the time on the timeline. There is a large

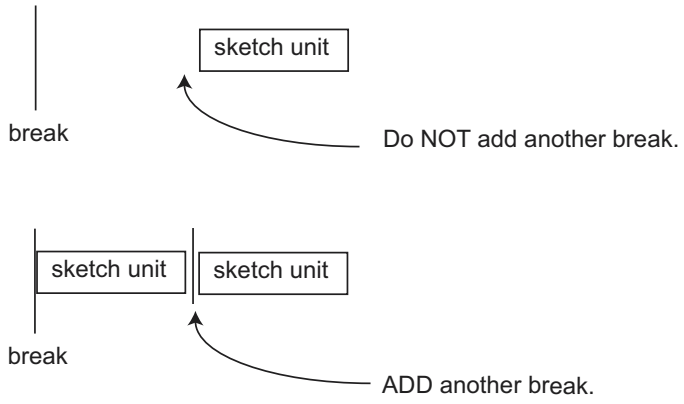


Figure 4.12: An illustration of the when a break is added because of a sketch unit and when it is not.

Start Frame	End Frame	Task Name
487	506	"[triangle:t1:draw draws slope, drawing left vertical line]"
498	520	"So, we have a ramp."
520	534	"[triangle:t1:draw draws slope, drawing bottom line]"
542	568	It's a fixed object.
553	569	"[triangle:t1:draw draws slope, drawing hyptenuse]"
585	595	"[triangle:t1:draw draws slope, extending bottom line]"
601	619	and <ahhh>
602	609	"[triangle:t1:draw draws slope, extending hypotenuse]"
677	702	We have gravity
685	698	[gravity:g:draw draws line for gravity]
705	723	[gravity:g:draw draws arrowhead]

Table 4.1: A sample transcript to show how the rules work.

gap assertion from frame 0 to frame 487. There is also a gap between frames 619 and 677. The long gaps leads to the creation of long-pause assertions because any gap longer than 25 frames causes the long-pause rule to fire.

There are two things being sketched in this transcript: a triangle, and the gravity arrow. The sketch-group rule groups the sketch components of the objects together. In this case, the triangle sketch group spans frames 487 to 609, and the gravity arrow sketch group spans frames 685 to 723. In each case, a sketch-unit is also created. The criteria for a sketch-unit is a sketch-group, which we have, and a no-gap assertion. The no-gap assertion comes from the speech utterances that slightly exceed and slightly preceded the sketch-groups. The utterance "and <ahhh>" exceeds the triangle sketch-group, and the utterance "We have gravity" precedes the gravity arrow sketch-group.

The text of the speech is also important. In this example, the "and <ahhh>" utterance leads to the creation of an and-words assertion and a mumbled-words-at-end assertion both between frames 601 and 619. There are also several key-words assertions that are created due to the phrases "we have", "so", and "it's". The mumbled-words-at-end assertion is interesting because it links the speech event after it to the speech event containing the mumbled word. In this case it links "and <ahhh>" and "we have gravity" with a linked-by-mumbles assertion that spans frames 601 to 702.

Two breaks are asserted because of the long-pause assertions. Whenever there is a long-pause assertion, a break is asserted at the end of the pause. This results in breaks at frames 487 and 677. However, the break at frame 677 is retracted by the rule system because there is a rule that removes breaks when they overlap with a linked-by-mumbles assertion, which is the case here.

4.3.6 Analysis of Rules

The rules seem to do a fairly good job of ascertaining the break assertions that segment the data. The results were evaluated in two ways. First, the output of the rules was compared to the hand segmentation. To do this, both segmentations were drawn on the same graph as shown in Figure 4.13 and in Appendix D. The computer-generated results corresponded well with the hand-generated segmentation. For the four transcripts that we hand segmented, there were 29 break points. The computer had the same break point for 24 of these. The computer found an additional 19 break points, 18 of which were reasonable break points. Even though the computer and hand interpretations differed

in 24 places there was enough ambiguity to make many of those valid break points. Since the hand-segmentation was done understanding all the speech and the spatial relationship between different sketched components, it is reasonable if the computer does not exactly match. If it did not place breaks, and thus the segments, in unreasonable places, then it did a good job. It is important to keep in mind that the output of the computer rules does not need to be perfect because it is just a first cut. The rules use only some of the information that is encoded in the data. There is much more information that could be utilized using ASSIST, more detailed speech recognition, and mutual disambiguation.

One concern was over-fitting of the data. Although there were 24 data samples, the data that was gathered would probably differ from the data produced by ASSIST and the speech recognition in real time. The rules were kept general and meaningful to preserve the logic behind the rules.

The rules also need to be able to look at only the data that has been seen so far and make a reasonable decision about what to do. When the system is running with real time speech input and sketching input, it will need to function in real time or as close to real time as possible. This means that the rules cannot look very far ahead, if at all. This was kept in mind when creating the rules.

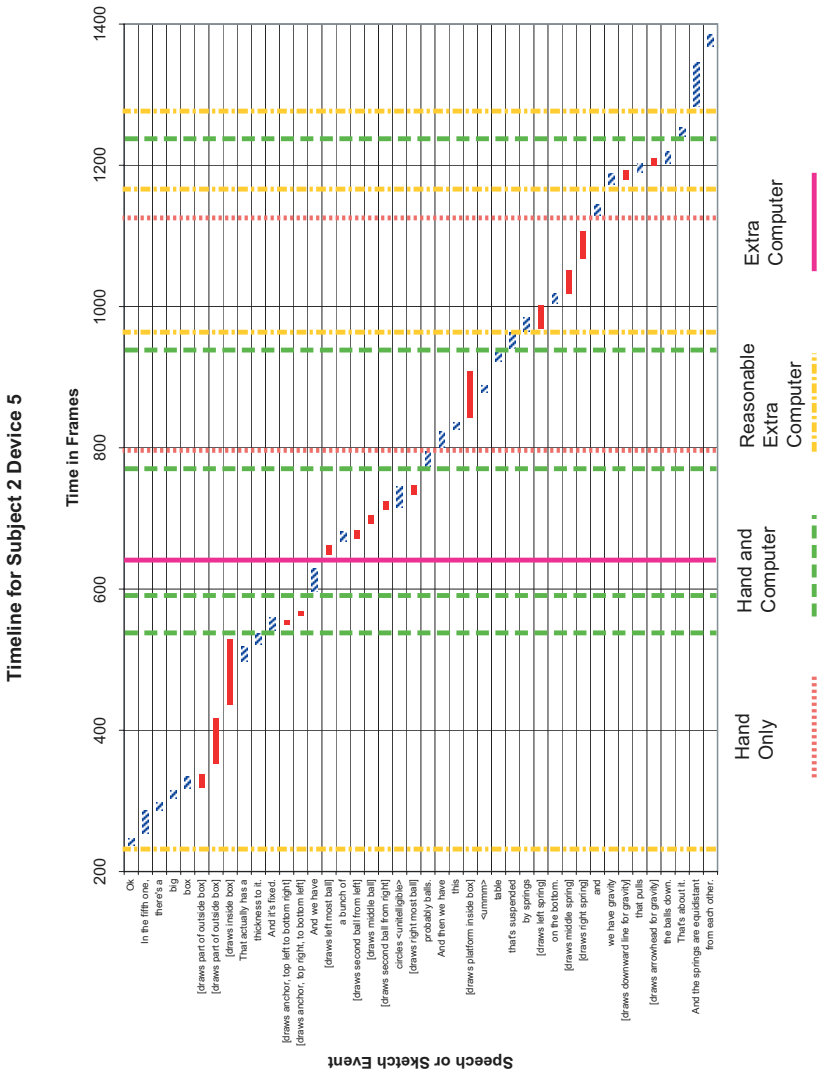


Figure 4.13: Subject 2 Device 5 with lines indicating the segmentation points of both hand segmentation and computer segmentation.

Chapter 5

Modifying ASSIST

The motivation for collecting the sample data, creating WATCH, and developing the set of rules for WATCH, was to derive the knowledge necessary to meld ASSIST and a speech recognition system to produce the live and interactive version of the multimodal system. There are two parts that needed to be able to communicate with each other – ASSIST and the speech system. This chapter will describe both parts and how they need to be modified to communicate, the issues involved, and the progress toward this goal.

5.1 Galaxy Speech System

Several different speech recognition systems were considered for the speech part of the multimodal system. Among the choices were the Galaxy Speech System developed by the Spoken Language Systems (SLS) group in the MIT Lab for Computer Science, and the ViaVoice system by IBM. For us, the Galaxy system had several advantages. Galaxy is a speaker independent system, which means that it can recognize a wide variety of accents without any training. The SLS group is in the same building, which allows extensive communication and collaboration with the developers. This gives us the advantage of being able to modify the speech system when we need to. The Galaxy system was designed for telephone access to information. For example, the group currently has systems for weather, restaurant information, and flight scheduling that can all be accessed by phone from around the world. The underlying architecture is the Galaxy Communicator which is now maintained by MITRE. The systems built with Galaxy are interactive. The computer and human have a conversation to provide the human

with the desired information. The conversational interface can handle anaphora, deixis, and ellipsis.

The Galaxy system consists of a hub with many servers that communicate with it. For example, there is the dialogue management server, the language generation server, the text-to-speech conversion server, the audio input server, the speech recognition server, a back end database server, and a context tracking server. The hub acts like a router and sends "galaxy frames" between the different servers. The frames get filled in with information as they pass to different servers. All the state is kept in the hub. Each of the servers has rules that act like a program and control the flow of the data.

SLS has even been working on a multimodal server. This server can integrate speech with mouse movements. They have a working demo of a planets simulation domain where a user can point at planets and change their color or direction and speed of motion. This system does not fit our requirements, however, as the system is driven from the speech perspective and is a fairly command based interface for multimodal interaction.

For our system, we were able to use just the recognizer part of Galaxy. The recognizer is trained from the phone data, but still works well with microphone input. For our domain, we were able to train a recognizer with the sentences that were gathered from the videos. This was done by generating a vocabulary from the words that occurred in the videos and using the sentences as a basis for what the system should expect to see. The recognizer is capable of generating a list with certainty values for the best interpretations (called an n-best list) for the speech phrase that it heard. This is needed for our system so that it can use this list along with the sketch input to come up with the best possible interpretation. The recognizer also comes up with a confidence score at the word level. The scores are zero-centered log-likelihood scores where positive scores indicate that a word is probably correct and a negative score indicates that the word is probably incorrect. This information could prove very useful in the multimodal system.

There are several issues in getting the recognizer integrated with the sketching system. SLS develops in C and on Linux, the sketch system is written in Java, and Working Model requires Microsoft Windows. Although these are not insurmountable difficulties, they do require some integration effort. Another issue is improving the vocabulary in the recognizer. The current vocabulary is missing some words that it should contain. For example, "wheels" is in the vocabulary but "wheel" isn't. This leads to recognition of "wheel" as "we'll". Once the system is up and running, training data can be used to fill in the vocabulary,

and recordings of the speech can be used to improve the recognizer. Because the training data for the system was recorded on phone lines, it is only recorded at 8kHz, but the data from a microphone is capable of a much higher recording rates of at least 16kHz.

The speech system needs the speech to be in phrases to process it. We would like to have the microphone on all the time and process as much the data as possible. The recognizer requires a section of speech to work with because it makes two passes on the incoming speech. The first pass uses a Viterbi search, which runs in real-time using a bigram model. The second pass uses an A* search and a trigram model to produce the n-best list. Because the system runs two passes, it makes a streaming system harder to implement. The system either needs to use an end-point detector or it needs to be given the sections of speech.

5.2 How ASSIST Works

ASSIST keeps track of many possible interpretations for each of the user's strokes. This allows the system to do things like create a motor from what was a circle and a trapezoid. ASSIST stores the interpretations of the stokes as "widgets." The collection of interpretations are stored in widget pools. There are three varieties of widget pools:

main widget pool contains all the interpretations.

surface widget pool contains all the widgets visible to the user.

recognition widget pool contains all the widgets that the recognizers can use. The main pool gets too large to use to trigger recognizers, so the widgets that are temporally or spatially close to the last stroke are kept in the recognition pool.

There are three major steps that ASSIST performs: recognition, reasoning, and resolution. Each will be described in more detail in the following sections.

5.2.1 Recognition

Recognition uses a toolkit of low-level recognizers that were developed by Metin Sezgin[22]. This step produces possible widgets for the part of the sketch that was just drawn. Widgets are also produced from a combination of previous widgets. For example, a rectangle widget might be formed from four line widgets.

5.2.2 Reasoning

The next step in the process is the reasoning step. In this step all of the interpretations for the last stroke are scored. The higher the score for an interpretation, the more likely it is to be true. There are two factors that make up the score: the categorical rules and the situational rules. The categorical rules are general rules for ranking the interpretations, whereas rules about specific instances of objects are situational rules. The contextual reasoners rank the interpretations relative to each other; in order to create numeric values from the relative rankings, several graph algorithms are used. For more information on this and the factors that go into the rankings, see [1].

5.2.3 Resolution

The last step is resolution. In this step the best interpretations for the strokes are chosen. The best interpretation is the one with the highest score that does not conflict with the previously chosen interpretations. This is accomplished using a greedy algorithm. The interpretation with the highest score is chosen and any conflicting interpretations are removed. The process is repeated until no interpretations remain. The result is then displayed to the user by straightening the strokes and objects that the user drew.

5.3 Modifications to ASSIST

As the first step on the path to creating a multimodal system, the speech recognizer and ASSIST must be able to talk to each other. The first goal is to modify WATCH so that it can receive live data from both ASSIST and the speech recognizer. ASSIST will eventually have to take into account the speech recognizer's input. The most appropriate place would be in ASSIST's reasoning component. However, to keep things simple to begin with, a step was added after resolution to send WATCH the current interpretation. Although only the surface level interpretations are sent to WATCH, it can access the lower level interpretations from the surface interpretations. WATCH displays the Squiggle components (the original stroke) in the timeline, but uses the higher level data to decide what object the Squiggle belongs to, e.g., rectangle or triangle.

WATCH was modified to take this live input instead of input from a file. The situation grows more complicated because the user can erase strokes in ASSIST after they draw them. This is not an issue

in the data file because all the strokes were identified as part of an object. This raises problems in a few places – it requires that the system compensate when a user erases strokes. To do this, the system needs to somehow keep track of more information about what was drawn and what was erased or otherwise compensate for the time that the user drew and erased an object. The other issue is maintaining the correctness of the assertions in the system. Because many assertions build on lower assertions, if a part of the sketch is removed, it can affect many assertions that should be changed or deleted as a result. A sufficient architecture for this has not yet been created although there has been progress toward creating a better truth maintenance system. It was decided to create the more functional truth maintenance system outside of JESS, using JESS's ability to call Java functions. The number of rules to undo assertions that were incorrectly made was beginning to grow too large. A dependency tree structure was implemented in a Java class that could be called from JESS. Thus an instance of this Java class keeps track of the necessary information to retract some of the assertions when necessary.

So far the modifications to ASSIST have been very minimal – in fact the modifications have been in only one direction – from ASSIST to WATCH. WATCH does not communicate back to ASSIST. Following are several screen shots showing the ASSIST window and the WATCH window, which were running at the same time. Figure 5.1 shows ASSIST communicating with WATCH.

In addition to integrating ASSIST and WATCH, progress has been made on getting the speech recognizer to be able to talk to WATCH. Currently, the speech recognizer is triggered by pressing the shift key to talk, and releasing the key at the end of the speech. More work is needed to integrate the end point detector that was mentioned previously in Section 5.1 to allow for continuous speech recognition. The recognizer now sends its output to port 3000, and the program that triggers the recognizer can receive the information and display it to the screen. Figures 5.2 and 5.3 show some examples of the output of the recognizer. The numbers indicate the log likelihood values for the certainty of the words. The recognizer is only trained on a small set of data and needs additional data to recognize the sentences that we need it to.

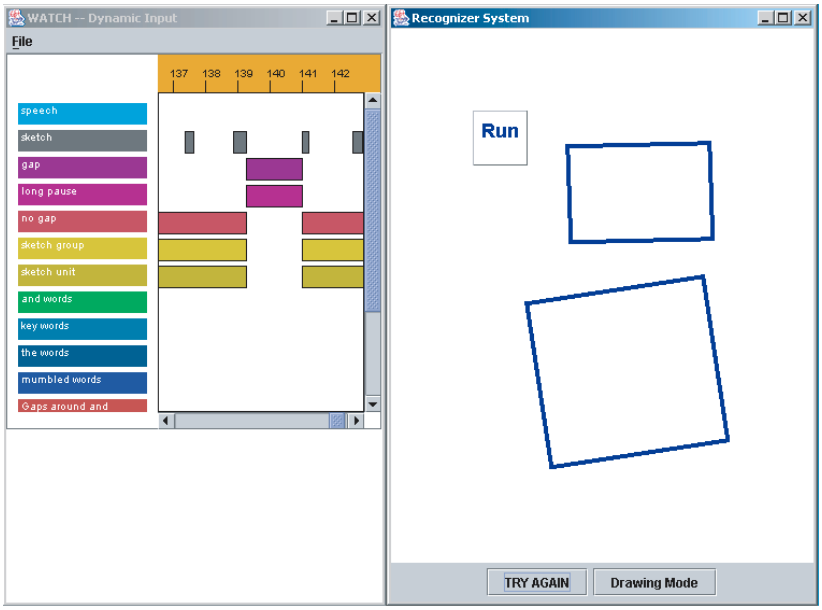


Figure 5.1: ASSIST communicating with WATCH.

```
<pause1> 3.12 in -3.69 this 0.16 picture 4.15 there's
     4.61 a 5.21 wedge 5.25 on 3.65 which 4.85 the 4.23
     car 5.00 is 4.90 located 4.19
<pause1> 3.10 in -3.71 this 0.14 picture 4.14 there's
     4.60 a 5.20 wedge 5.24 on 3.64 which 4.84 the 4.22
     car 4.99 is 4.89 located 4.34 <pause2> -4.07
<pause1> 2.45 this -0.92 picture 4.13 there's 4.59 a
     5.20 wedge 5.23 on 3.62 which 4.83 the 4.21 car
     4.98 is 4.88 located 4.17
<pause1> 3.08 is -7.47 picture 3.85 there's 4.58 a 5.19
     wedge 5.22 on 3.61 which 4.82 the 4.20 car 4.97 is
     4.87 located 4.16
<pause1> 3.07 in -3.78 this 0.09 picture 4.11 there's
     4.57 a 5.18 wedge 5.21 on 3.60 which 4.13 car 4.77
     is 4.87 located 4.15
<pause1> 3.03 and -9.12 this 0.07 picture 4.10 there's
     4.56 a 5.17 wedge 5.20 on 3.59 which 4.80 the 4.18
     car 4.95 is 4.86 located 4.14
<pause1> 3.03 these -9.03 picture 3.82 there's 4.55 a
     5.16 wedge 5.19 on 3.58 which 4.79 the 4.17 car
     4.94 is 4.85 located 4.13
<pause1> 3.03 it's -10.43 picture 4.08 there's 4.54 a
     5.15 wedge 5.19 on 3.56 which 4.78 the 4.16 car
     4.93 is 4.84 located 4.11
<pause1> 2.37 this -1.02 picture 4.06 there's 4.53 a
     5.14 wedge 5.18 on 3.55 which 4.77 the 4.15 car
     4.92 is 4.83 located 4.27 <pause2> -4.22
<pause1> 3.00 is -7.62 picture 3.78 there's 4.52 a 5.13
     wedge 5.17 on 3.54 which 4.76 the 4.13 car 4.91 is
     4.82 located 4.26 <pause2> -4.24
```

Figure 5.2: Output from the SLS speech recognizer trained from our data, part 1.

```
<pause1> 5.14 gravity 3.81 downwards 4.41
<pause1> 5.13 gravity 3.79 downwards 4.45 <pause2> -4.57
<pause1> 5.13 gravity 3.78 downwards 4.49 <pause1> -4.74
<pause1> -5.14 <pause1> 4.71 gravity 3.77 downwards 4.38
<pause1> 5.05 and -5.97 gravity 3.75 downwards 4.37
<pause1> 5.10 gravity's -7.67 downwards 4.36
<pause1> 5.09 gravity 3.74 downwards 4.47 is -6.40
<pause1> 5.08 gravity 3.72 downwards 4.44 <pause1> -4.82
     <pause2> -4.70
<pause1> 5.07 gravity 3.71 downwards 4.45 to -6.02
<pause1> 5.06 gravity 3.70 downwards 4.44 it's -6.35

<pause1> 4.43 indicated -4.74 by 4.84 these 4.40 markings
     3.32 on 3.64 the 0.31 wheels 3.43 <pause2> 4.21
<pause1> 4.90 to -5.91 indicate -4.25 by 4.30 these 4.39
     markings 3.30 on 3.63 the 0.30 wheels 3.42
     <pause2> 4.20
<pause1> 4.41 indicate -4.46 by 4.29 these 4.38 markings
     3.29 on 3.62 the 0.28 wheels 3.40 <pause2> 4.19
<pause1> 4.40 indicated -4.80 by 4.81 these 4.37 markings
     3.28 on 2.68 wheels 3.39 <pause2> 4.18
<pause1> 4.39 indicated -4.83 by 4.80 these 4.36 markings
     3.27 on 3.59 the 0.25 wheels 3.31
<pause1> 4.86 to -6.00 indicate -4.34 by 4.26 these 4.35
     markings 3.25 on 2.65 wheels 3.36 <pause2> 4.16
```

Figure 5.3: Output from the SLS speech recognizer trained from our data, part 2.

Chapter 6

Related Work

One of the earliest papers on multimodal interaction is Bolt's "Put-That-There" paper[4]. This system used basic speech recognition and equipment that could track where the user was pointing. This simple system allowed users to move colored shapes around on a screen. It was able to allow the free use of pronouns and figure out which object the user was talking about. The system also allowed users to give objects names, and then use those names to refer to the objects. The system was a first attempt at creating a more natural user interface. Since then there have been many systems that have improved on it.

Sharon Oviatt points out some of the advantages of multimodal interfaces in [17]. The paper discusses how multimodal input simplifies the users' vocabulary and improves accuracy with accented speakers with the additional input. She also discusses how users are less frustrated by errors and seem to move between input modalities as they see errors.

6.1 QuickSet

There has been considerable work on multimodal interfaces at the Oregon Graduate Institute. Much of this work has involved the QuickSet system. The QuickSet system is a collaborative multimodal system that is built on an agent-based architecture. The user can use voice and pen-based gestures to create and position items on a map. Primarily it is used for military simulation work. For example, a user could say "medical company facing this way <draws arrow>". Although QuickSet is more command based and does not have the sketching capabilities that our system does, it does use continuous speaker-independent

speech recognition with a vocabulary of about 900 words[19].

The QuickSet system was designed for use in a military environment. As such, some of the focus was on efficiency. For example, one of the studies using QuickSet notes that it is nine times more efficient (faster) to use verbal and sketched input rather than just a graphical user interface[20]. The study had users setting up a military simulation with users saying phrases like "Open space".

The goal of their system was to create an efficient system for the military, which is different than our goal of creating the most natural user interface possible. In a situation where speed matters, it makes sense for users to use abbreviated phrases. In our system, the user might explain actions more verbosely since speed is not such a large influence in the early stages of design. Similarly, with a map-based system, the user starts with something to refer to – the map. In our system, the user starts with a blank screen.

The group at OGI has also looked into the patterns of multimodal integration in a simulated dynamic map system[21]. Their study found that multimodal communication was most likely to occur in spatial location commands. The speech system that was used in the study was one where the user had to hold down the pen stylus to talk.

This type of communication differs from the descriptive style of speech that we saw in our sample data. This difference could influence the nature of the integration patterns of the speech and sketching that they observed. Where the commands in the map system replace a menu type interaction, we are attempting to provide an interface in a design environment that is as natural as talking to another human. In a system like ours where a user is trying to explain the system as a whole, the multimodal communication would seem to be much richer than simply trying to give a computer commands. Along the same lines, instead of a push-to-talk microphone as in their study, it is our desire to have an open microphone and do the best we can with the speech input.

6.2 Other Multimodal Systems

There are other systems that have multimodal capabilities. IBM has a Human-Centric Word Processor[19] that lets the user dictate text and use multimodal input to perform editing tasks. The system has to be trained for each user and has a manual switch that changes the speech interface from dictation mode (large vocabulary) to editing mode (small vocabulary). The only pen input that it currently accepts is pointing

with the pen.

There are other systems that use a variety of different input modalities. Boeing's Virtual Reality Aircraft Maintenance Training Prototype [19] combines a virtual reality interface with speech recognition to allow users to see how easily an aircraft can be repaired and to train people to repair it once it is built. NCR has built a Field Medic Information System[19] that allows medical personnel to document a patient's condition and other data relating to the patient while in the field. The system allows users to enter information with a pen interface or by voice. BBN developed a Portable Voice Assistant[19] that uses pen and voice input to enter and retrieve information on the World Wide Web. Their system integrates simultaneous speech and gesture inputs using a frame-based system.

6.3 Other Architectures

An alternative approach to the n-best list method described in this thesis (in Section 7.1) is a weighted finite-state device described in [8]. By using this method, speech and gesture are more tightly coupled because the two inputs combine to form one output. More recently, AT&T Labs has developed MATCH[9]. MATCH provides a speech and pen interface to restaurant and subway information for New York City. This program uses the finite-state device and lets users make simple queries. This tool provides some multimodal dialogue capabilities, but it is not a sketching system and has only text recognition and basic circling and pointing gestures for the graphical input modality.

Another system that relates to what we are trying to do is BEATRIX [14]. This program is capable of understanding a physics problem with a combination of a diagram and some written English text. BEATRIX uses a blackboard to help establish the reference relationships between the graphical and text references. For example, if a coefficient of friction is given in the text, the appropriate objects in the diagram must be found. BEATRIX can solve the physics problems and derive the answers. Although BEATRIX dealt with a similar domain, the objective was somewhat different. Our system attempts to capture the design rationale behind a particular design in the early stages of design. In the early stages of design, the calculation aspect of BEATRIX is not so applicable, nor do we have the certainty of text input, instead we have natural speech. However, the architecture and ideas the are outlined in the BEATRIX work may be applicable to our system. In fact, the next generation of our system is blackboard based

like BEATRIX (see Section 7.3).

6.4 Other Sketching Systems

The other related field is sketching. There are several groups working on sketching e.g. [10, 6]. Each system is a little bit different, but Forbus, et. al. note that there are not any sketching systems that use mixed-initiative dialogues[6]. This requires a greater level of conceptual understanding, which is a long-term goal we have (see Section 7.5).

Another sketching systems is sKEA or sketching Knowledge Entry Associate developed at Northwestern University[7]. The sKEA work focuses on conceptual and visual understanding and can operate in arbitrary domains. This wider domain coverage somewhat reduces the interaction flexibility because sometimes the users have to provide more information than they would in a more specialized system. This system gives up some naturalness to provide greater domain coverage. In the longer term, they plan to add a dialogue capability to sKEA[7].

The same group at Northwestern (the Qualitative Reasoning Group) has also developed nuSketch COA Creator[6]. This program is used to create military course of action (COA) diagrams. nuSketch itself is designed as a general-purpose multimodal architecture. It allows the users to sketch and talk. Users use commands such as "Add severely restricted terrain" to add symbols to the military map. This system again uses command based speech which is not what we are striving toward.

Chapter 7

Future Work

The work described in this thesis is a start in creating a natural multimodal system that combines sketch understanding with a natural voice interface. The work can proceed in many possible directions, including, among other things, implementing the mutual disambiguation aspect of the system and providing more advanced speech interaction.

7.1 Mutual Disambiguation

The mutual disambiguation aspect of the system has not been fully implemented. Although both parts, the speech and the sketching, are ready for integration, the last few steps remain. The sketching system scores the possible interpretations and the scoring needs to be modified to take into account the speech input. The speech system currently provides an n-best list for the recognized speech. This list will be used to help recognize the sketch. Eventually, the system will also be able to use the speech input to do more advanced things and interact with the user in a conversation. The disambiguation will increase the accuracy of the sketching system and make the user interface more natural. It will also allow the system to understand more of what the user is doing.

7.2 Replication and Modification of Widgets

Currently everything that the user sketches is backed by actual stokes that the user has drawn. However, sometimes the user may wish to

make copies of objects and manipulate the objects on the screen. Modifying the system so that everything will still work might be a challenge. Acquiring the stroke data for these new objects or modifying the existing stroke data is difficult because exactly what the user wants to do or where they want to move an object might not be clear.

7.3 Next Generation System

An ongoing project of our research group is to create a next generation version of ASSIST. The next system will be more powerful in many ways. It is designed around a language to provide domain specific knowledge to a domain independent sketching system. It will allow the user to teach the system new shapes with only a few examples. More importantly for this work, it will be designed around a blackboard system[5, 12, 13] and has been designed with multimodal input in mind. A future version of this system could more easily interact with the sketch knowledge as a knowledge source on the blackboard.

7.4 Extracting More Information from Speech

The first version of the multimodal interaction will only look for clues to help the sketch recognition. However, the speech is a rich input and there is much more information that can be extracted. The references to previous objects, for example "that's", can be used to further disambiguate the sketching input. Numerical references can be used in the disambiguation as well as numerous other speech clues such as information about the design rationale behind the user's design decisions.

7.4.1 Properties of Objects

Working Model allows objects to have textures. Speech could be used to allow the user to specify the texture of objects. This would be a primarily verbal interaction since there is no way to specify the texture currently in ASSIST. The addition of textures, such as ice or metal, changes the frictional coefficients in Working Model and thus would allow the simulations to be more accurate. It would allow the user to simply make statements such as "there are two metal balls on the slope" and have the simulation be different than if they were wooden balls.

7.4.2 Adjustments to the Sketch

The speech could also allow the user to make adjustments to the sketch. For example, the user could say "the pendulums are equally spaced" and the system could make the appropriate adjustments. This is tricky because the system has to know which pendulums the user is talking about, as well as what it means to make them equally spaced. The system would have to know how to make them equally spaced, and be responsive to user feedback about what it does. For example, if the user says "make them closer together", the system should be able to make the appropriate adjustments. The idea here is to make the system as natural as possible and create a natural interaction for the user.

7.5 Intelligent Feedback – Prompting the User When Confused

The system may not always understand what is going on. However, if it understands enough of what the user was drawing, it could ask the user an appropriate question. As previously mentioned, the system might not know how close to make the pendulums that the user said were equally spaced. The system could ask how close the user wanted the pendulums, or could describe the different choices to identify the pendulums the user wanted to move. It might ask if it should make the left pendulums or the right pendulums equally spaced. Other possible interactions could include clarifying the drawing, asking questions about parts of devices that the computer does not understand, or identifying possible errors. This level of understanding would allow the system to capture the rationale behind design decisions and allow future reference to this information.

The key to prompting the user is to ask intelligent questions at an appropriate time. The timing is very important – if the system asks the user a question at the wrong time, when the user is in the middle of drawing something else, the user will become annoyed. If the user is annoyed, then the system has failed to be less trouble than it is worth.

7.6 Incorporating Gesture Recognition

In the previous example, the system could ask the user how close to make the pendulums, but the user might not be able to respond naturally. A member of our research group is currently working on gesture recognition. With gesture recognition, the system could recognize how

far apart the user's hands were when he responded to the question by saying "this far apart." This interaction with the system would be completely natural and make the user feel very much like they were interacting with another human and not a machine.

Gesture recognition can be used as another input to the multimodal system. The gestures could help to disambiguate the other forms of input. For example, the user might point to certain parts of the system and motion in certain directions while talking. This input could be used to correctly simulate the user's ideas or understand what they were drawing. It incorporates some of the ideas of ASSISTANCE[15] and some of the other ideas that have developed elsewhere.

Chapter 8

Conclusion

A multimodal interface to our sketching system would create a more natural user interface that will provide the user with a simple, powerful, and more accurate way to communicate with the computer. To reach this goal, we must be able to segment and align the speech and sketching inputs as we receive them.

Data from six subjects who sketched and spoke about six different devices was captured. Important features, such as pauses and certain vocabulary, were identified. A set of rules that use this knowledge were created to perform the segmentation and alignment. In order to better understand the more complex rules, WATCH was created. WATCH displays the rules in a timeline layout to help visualize the interaction of the rules.

The rules were tested on the transcripts from the videos and did accurate job segmenting and aligning the input. Segmentation and alignment of the speech and sketching inputs is a crucial step that will allow future work on mutual disambiguation. Future work will involve integrating speech recognition and ASSIST to create the multimodal system.

Appendix A

Audio Recording Transcripts

Each Section in this Appendix contains the three audio transcripts for one of the devices.

A.1 Device 1

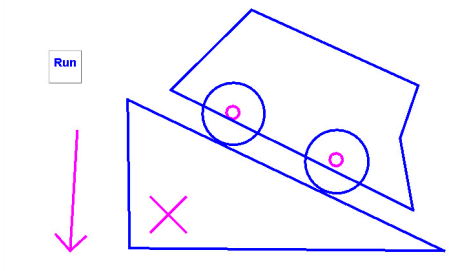


Figure A.1: This is device 1.

A.1.1 Subject 1

So <ahh> we have this ramp here <umm>, the x is for the thing it is tied down to - to the ground. And then <umm> there is this little cart <pause> type car with wheels, and the wheels are attached to the car,

like that. And then this downward arrow is for specifying grav- grav- gravity.

A.1.2 Subject 2

So, here's an inclined <pause> uhhh ah here's a ramp, umm it's fixed <pause> and <pause> and a car on the ramp, <pause> and there is gravity in the system.

A.1.3 Subject 3

So our first example is going to be <pause> a aaah cart, on a very short incline. This is gonna be an enormous cart like this. And it's gonna have <pause> four wheels but you only see two wheels. And here are the pivots. <unintelligible> And the <pause> ahh this is going to be the ground - this is going to be the earth. And we have aaah and so we're on a planet where gravity goes in that direction.

A.2 Device 2

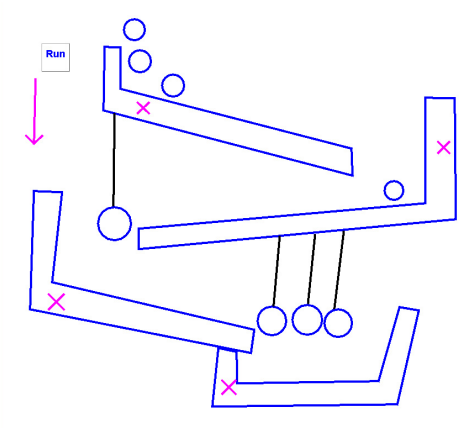


Figure A.2: This is device 2.

A.2.1 Subject 1

Okay, so we have this little <mmm> gadget <umm> there like three – four ramps that sort of go down so that you have little balls <umm> that would sort of climb down this way - or not climb down, but like fall down <pause> to the next one - like so. And then - and then there is another one <pause> here. So this ball would roll down to here - oh except there is a little <pause> pendulum thing, which swings sort of this way I guess. So what would happen is this ball would roll down and then it would hit this thing and then it would fall down here or something like that. And then <umm> the last one is this <pause> basket kind of a thing which would catch the ball falling down. And here again we have 3 pendulums thingies. These are all <umm> stationary. There is also gravity.

A.2.2 Subject 2

So there are a set of ramps <pause> ahh fixed in free space with a number of <ahh> balls suspended from the bottom of the ramps, and a number of free balls <umm> rolling down <ahh> the collection of ramps under the influence of gravity.

A.2.3 Subject 3

<laughter> Our second example is a Japanese Pachinko machine. And for the Pachinko machine what we're gonna have is a series of slides. One like that. And that's fixed. Another one that looks like <pause> that goes in the other direction. like that. And you have two more <pause> like that. So the balls are gonna fall – they're gonna start up here and they're gonna slide down this way like this. And there's gonna be a pendulum, right there hanging from the first ramp. And a series of three pendula <pause> pendulum? like that. And again we are on a planet where gravity is like that.

A.3 Device 3

A.3.1 Subject 1

Okay, <umm> so this next one <pause> is a <pause> this is like a trampoline kind of device. So inside this box there is a <pause> a little <ahh> flat surface and <umm> which is attached to the ground by three springs. So that you know, it is like a trampoline so you bounce

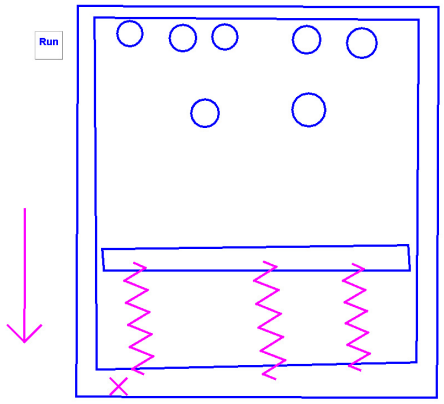


Figure A.3: This is device 3.

up and down or something. So - so you have <umm> balls that are hanging in the air. <laughter> And then, I guess the idea is that the balls are going to bounce up and down, in the box. So this box is <umm> tied down to the ground.

A.3.2 Subject 2

So in a in a closed container <pause> under the influence of gravity <pause> umm we have a set of balls that were resting on a umm spring loaded platform <pause> and here the ahh springs went from <pause> ahh <pause> compression to <pause> ahh <pause> the springs that were in compression were just released and now the balls that were resting on the platform were sprung into the air.

A.3.3 Subject 3

My third example is going to be a container <pause> like this. Filled with very little gas. And here are the gas particles <pause> and the bottom of the container is gonna have a platform that is connected with <pause> that is ahh held up by a series of compressed springs. And <pause> here's the wall of the container. It's going to<pause> be fixed. And again we are on a planet – once again we are on a planet where gravity goes down.

A.4 Device 4

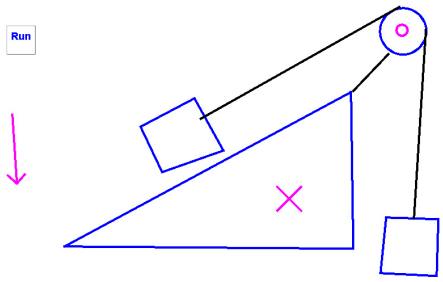


Figure A.4: This is device 4.

A.4.1 Subject 1

The next one is <pause> a pulley. So you have a ramp <pause> and then <umm> a little pulley here and so you... <pause> Let's see you have two objects, <pause> that are attached by a string like this. So that either one, which ever is heavier or something, <umm> pulls down the other. Or I - I guess what would happen is this thing would go down the ramp and - and then pull this thing up.

A.4.2 Subject 2

Ahh we have a fixed ramp <pause> ahh a pulley at a pulley system at the top of the ramp and we have one block <pause> umm with a with a piece of rope a piece of rope attached to that block umm the rope goes over <pause> the pulley umm <pause> and on the other end of the rope there is another block.

A.4.3 Subject 3

Our fourth example <pause> is an incline similar an inclined plane to <ahh> the first one we did. A wedge <pause> is going to be fixed. And it's gonna have two masses <pause> connected to each other through a pulley - like that. And gravity is going to go down.

A.5 Device 5

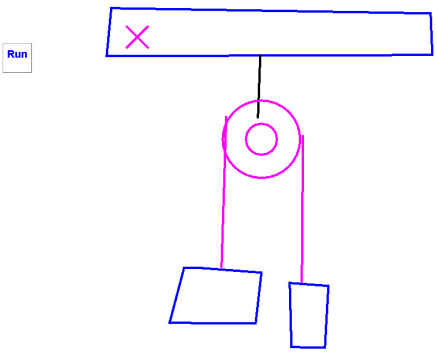


Figure A.5: This is device 5.

A.5.1 Subject 1

And here we have another <pause> pulley which is hanging from <umm> this thing, which is tied down to the ground or the board whichever. So a pulley here and you have two objects that are <pause> just hanging down. I'm not really sure what would happen.

A.5.2 Subject 2

We have aaa <pause> a plat... a fixed platform and hanging from that platform is a pulley. And <pause> over the pulley is a rope with two blocks on either end.

A.5.3 Subject 3

Our sixth example - fifth - sixth - fifth example is going to be a <pasuse> ahhh ummm a pulley hanging from the ceiling. Here we have the ceiling. And here we have the pulley. Ummm that has two weights, two masses suspended. And again we are on a planet, no we're not on a planet. Ok. We are on a planet without gravity. Alright.

A.6 Device 6

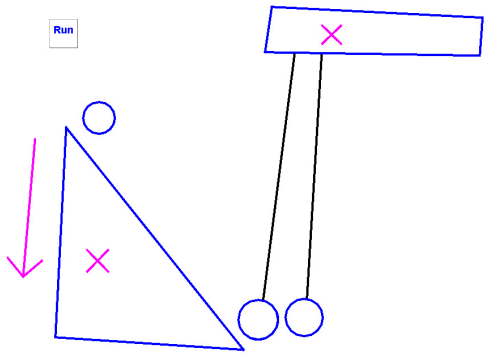


Figure A.6: This is device 6.

A.6.1 Subject 1

<umm> okay, so this is the last one. <umm> <hmm> here we have a little <ahh> ramp like this with the ball - so the ball should roll down the hill because of gravity. But <umm> there are <pause> hanging from this thing there are two <umm> <hmm> pendulums or something. So that when this ball rolls down <umm> it would hit this thing and this would hit that thing.

A.6.2 Subject 2

So in this system with gravity there's a fixed ramp, <pause> a ball at the top of the ramp, <pause> aaah a fixed platform above and to the <pause> right of the ramp and from this platform we have two balls suspended.

A.6.3 Subject 3

Alright. For our sixth and final example. Or seventh or whatever it was - sixth. We're going to have <pause> aaa – another inclined plane or wedge here. And we'll have a ball that starts at the top. Umm and two pendulums hanging from <pause> a elevated surface, right here. And we're on a planet where gravity goes down.

Appendix B

Video Transcripts and Graphs

Each Section in this Appendix corresponds to one device. Each Subsection contains the transcript, timeline graph, and gap graph for one subject.

B.1 Device 1

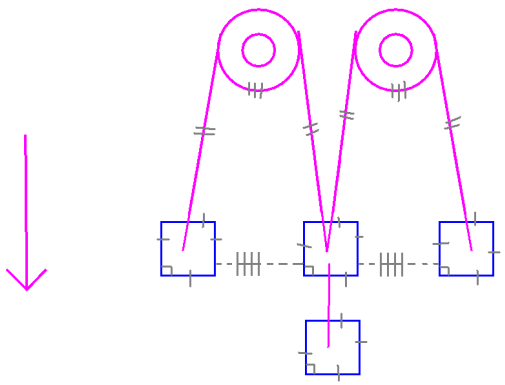


Figure B.1: This is Device 1.

B.1.1 Subject 1

Start		End		
sec	frame	sec	frame	Task Name
11	8	11	21	It's –
11	26	14	9	I'm supposed to draw a car rolling down an incline.
14	27	16	20	So first I'll draw the incline.
15	24	16	3	"[triangle:t1:draw pen down to draw slope, drawing left vertical line]"
18	2	19	0	[triangle:t1:draw drawing bottom of slope]
18	27	19	26	"It's basically a triangle,"
19	26	23	17	and I weight the triangle to the surface so it doesn't <ahh> fall.
20	10	20	17	"[anchor:a1:draw drawing anchor, top left to bottom right line]"
20	22	20	28	[anchor:a1:draw drawing top right to lower left anchor line]
23	24	24	15	[gravity:g:draw downward line for gravity]
24	2	25	29	We have the force of gravity pointing downward.
24	23	25	10	[gravity:g:draw draw arrowhead]
26	27	28	24	"Ok, now I'll start by drawing the car."
28	26	33	14	[polygon:p1:draw draws car body]
28	27	30	12	I'll draw the body of the car.
34	22	35	15	And then I'll draw
35	12	36	2	[wheel:w1:draw draws back wheel]
36	9	36	23	wheels
36	16	37	8	[wheel:w2:draw draws front wheel]
37	20	38	21	and attach the wheels to the body of the car.
37	24	38	4	[pivot:v1:draw pivot for front wheel]
38	19	39	2	[pivot:v2:draw pivot for back wheel]
40	24	42	3	And then I click run.

Table B.1: Transcript of Video for Subject 1 Device 1.

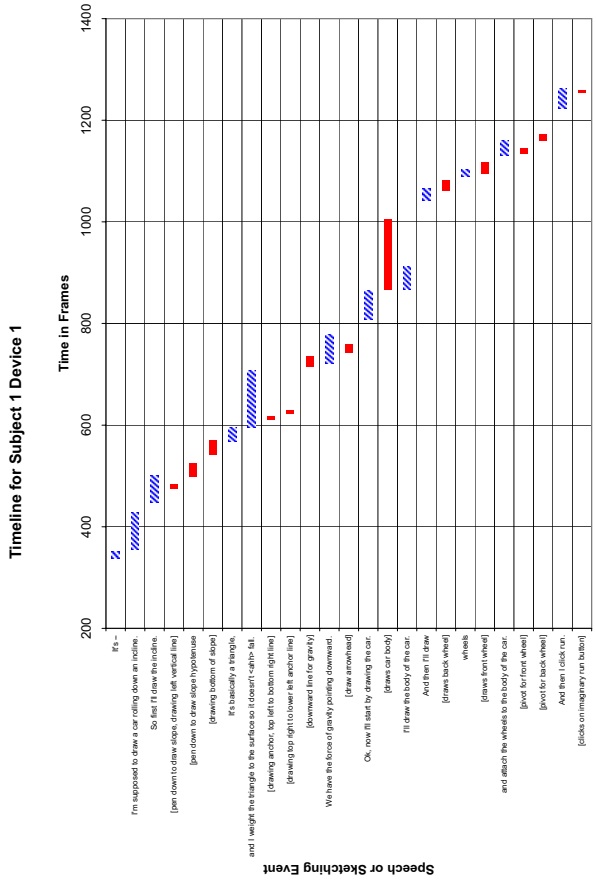


Figure B.2: The timeline graph for Subject 1 Device 1.

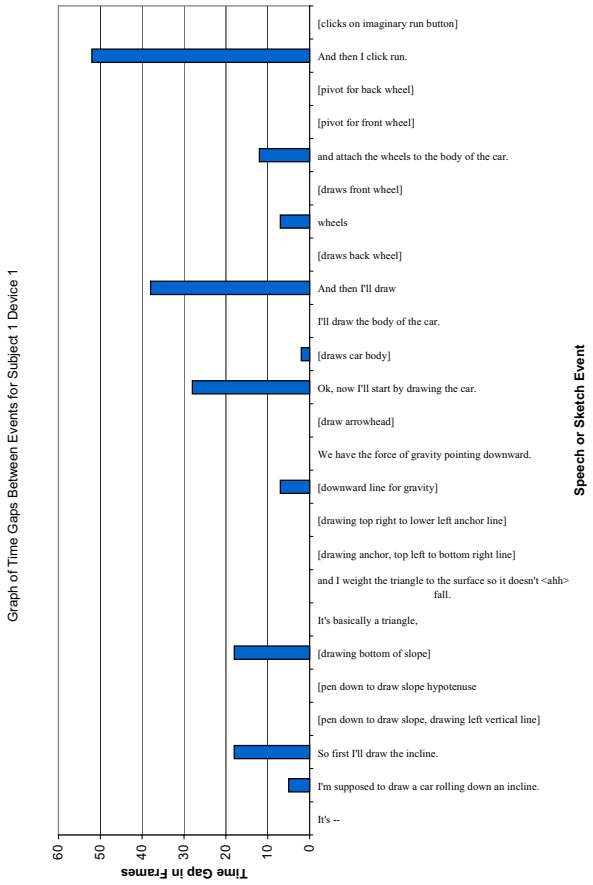


Figure B.3: The gap graph for Subject 1 Device 1.

B.1.2 Subject 2

Start		End		
sec	frame	sec	frame	Task Name
7	22	8	3	Ok
8	15	8	29	so
9	0	10	8	In this picture there's a
10	5	10	17	"[triangle:t1:draw draws slope, drawing left vertical line]"
10	28	11	19	there's a wedge
11	2	11	19	"[triangle:t1:draw draws slope, drawing hypotenuse]"
11	20	12	2	"[triangle:t1:draw draws slope, drawing bottom line]"
11	27	13	8	on which the car is located.
13	20	13	25	"[anchor:a1:draw draws anchor, top left to bottom right]"
13	27	14	14	It's fixed.
14	1	14	7	"[anchor:a1:draw draws anchor, top right to bottom left]"
15	1	15	12	<ahhh>
15	12	15	24	and
15	24	16	0	<uhhh>
15	26	16	5	[gravity:g:draw downward line for gravity]
16	3	17	4	gravity downwards.
16	12	16	20	[gravity:g:draw draws arrowhead]
17	22	19	9	And then there's a car on top
18	16	20	26	[polygon:p:draw draws car body]
21	2	22	15	which has wheels.
22	5	22	22	[wheel:w1:draw draws back wheel]
23	1	24	12	which are the same radia
23	9	23	14	[wheel:w1:draw draws mark on back wheel]
24	1	24	17	[wheel:w2:draw draws front wheel]
24	15	25	29	<ahhh> the same circumfrence
25	2	25	7	[wheel:w2:draw draws mark on front wheel]
26	19	26	27	<ahhh>
26	28	27	19	Indicated by
27	19	28	1	these markings
27	22	28	12	[pivot:p1:draw pivot for back wheel]
28	1	29	15	on the axes of the wheels
28	29	29	6	[pivot:p2:draw pivot for front wheel]
30	11	32	6	which indicate they're spinning around or something
33	11	33	26	"Sooo,"
34	21	35	22	that finished that picture.

Table B.2: Transcript of Video for Subject 2 Device 1.

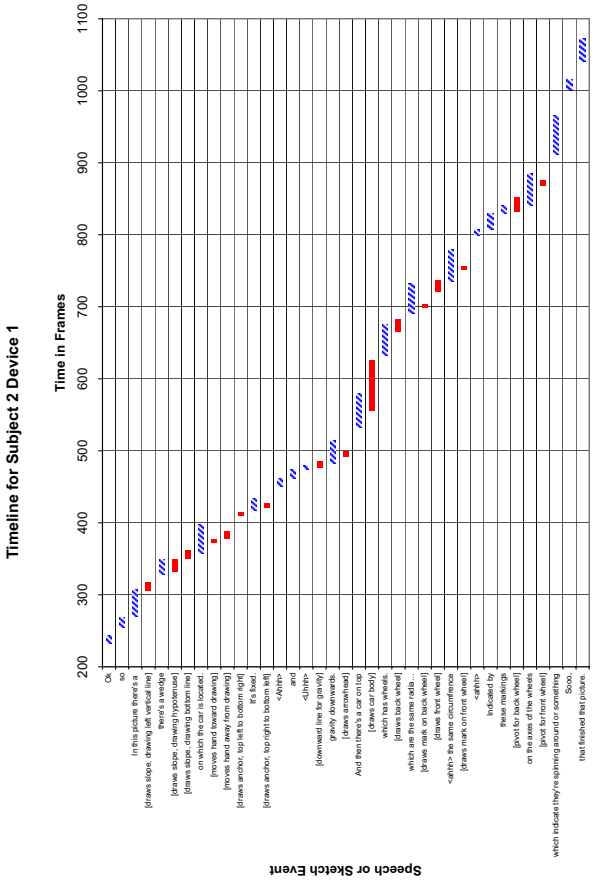


Figure B.4: The timeline graph for Subject 2 Device 1.

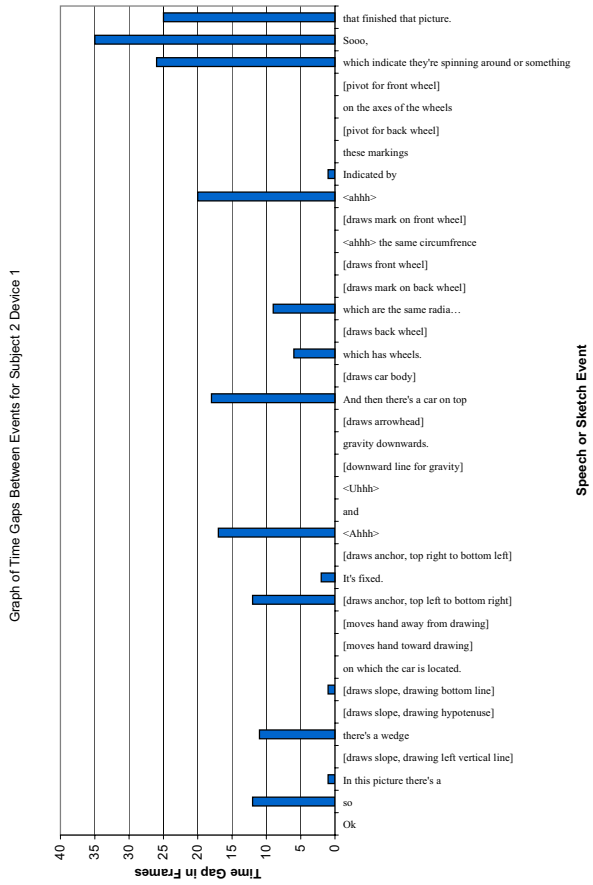


Figure B.5: The gap graph for Subject 2 Device 1.

B.1.3 Subject 3

Start		End		
sec	frame	sec	frame	Task Name
16	7	16	26	"[triangle:t1:draw draws slope, drawing left vertical line]"
16	18	17	10	"So, we have a ramp."
17	10	17	24	"[triangle:t1:draw draws slope, drawing bottom line]"
18	2	18	28	It's a fixed object.
18	13	18	29	"[triangle:t1:draw draws slope, drawing hyptenuse]"
19	15	19	25	"[triangle:t1:draw draws slope, extending bottom line]"
20	1	20	19	and <ahhh>
20	2	20	9	"[triangle:t1:draw draws slope, extending hypotenuse]"
22	17	23	12	We have gravity
22	25	23	8	[gravity:g:draw draws line for gravity]
23	15	24	3	[gravity:g:draw draws arrowhead]
25	23	26	7	and
26	18	27	4	we wanna put a
27	6	27	23	car on top
27	23	28	8	of this ramp.
27	28	28	11	[wheel:w1:draw draws back wheel]
29	17	30	4	[wheel:w2:draw draws front wheel]
31	14	35	18	[polygon:p1:draw draws car body]
36	29	37	7	"Now,"
37	13	37	23	these
37	25	38	13	[pivot:p1:draw pivot for back wheel]
38	27	39	10	[pivot:p2:draw pivot for front wheel]
39	13	40	8	you know these are wheels
40	8	40	26	so of course it
41	8	42	3	rolls down the ramp.
43	21	44	5	and
46	12	46	22	that's
46	28	47	15	that picture.

Table B.3: Transcript of Video for Subject 3 Device 1.

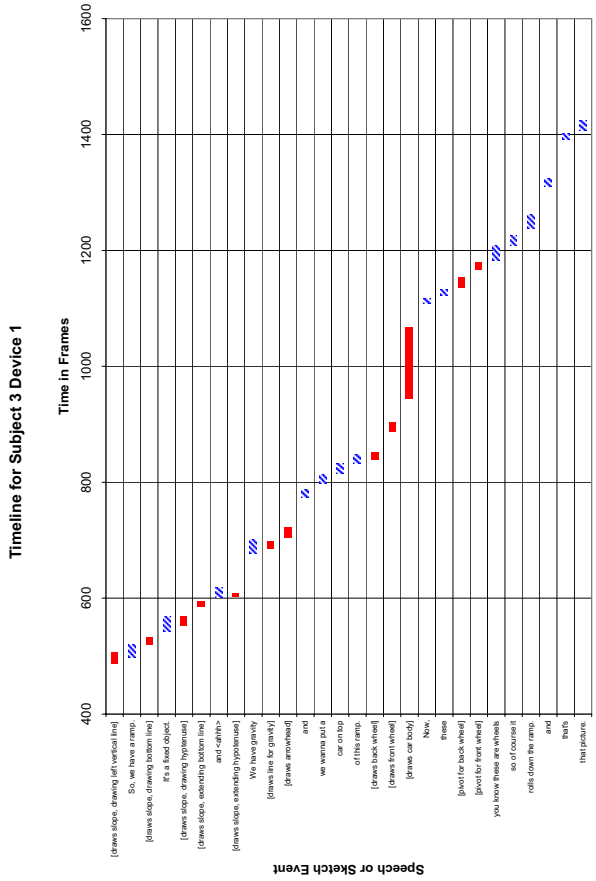


Figure B.6: The timeline graph for Subject 3 Device 1.

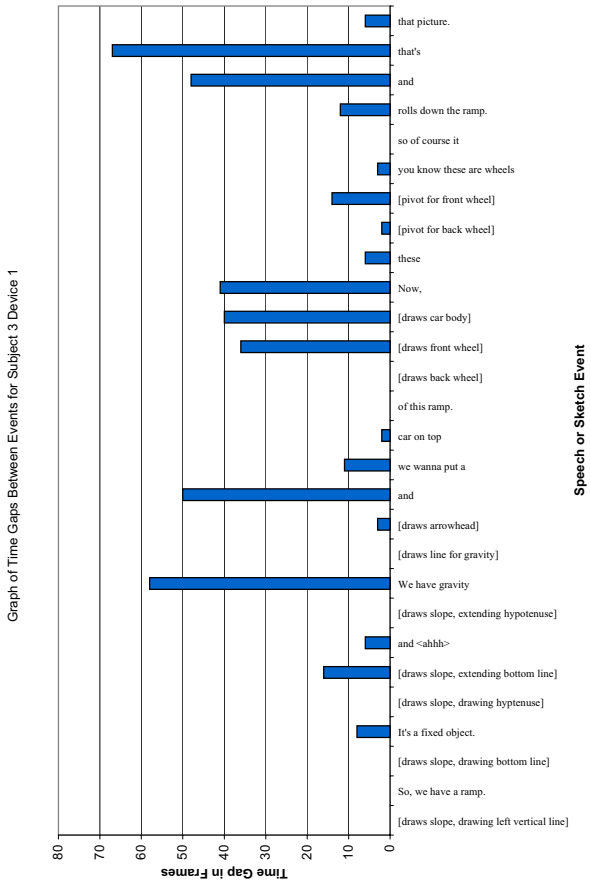


Figure B.7: The gap graph for Subject 3 Device 1.

B.1.4 Subject 4

Start		End		
sec	frame	sec	frame	Task Name
4	29	6	5	"Alright, so first we have a <ahhh>"
6	20	7	10	car on a ramp.
7	29	8	13	"[triangle:t1:draw draws slope, drawing left vertical line]"
8	14	9	4	"[triangle:t1:draw draws slope, drawing bottom line]"
8	15	8	26	"So,"
9	6	9	26	"[triangle:t1:draw draws slope, drawing hypotenuse]"
9	21	10	12	this will be our ramp.
11	18	12	11	[wheel:w1:draw draws back wheel]
12	19	12	27	[pivot:v1:draw pivot for back wheel]
13	12	13	27	[wheel:w2:draw draws front wheel]
14	4	14	13	[pivot:v2:draw pivot for front wheel]
15	9	17	10	[polygon:p:draw draws car body]
17	20	17	29	The car.
19	4	19	24	<aaah>
20	15	20	19	"[anchor:a1:draw draws anchor, top right to bottom left]"
20	21	21	14	Fix this here.
20	21	20	24	"[anchor:a1:draw draws anchor, top left to bottom right]"
21	19	21	28	[gravity:g:draw downward line for gravity]
22	0	22	23	Gravity points down.
22	5	22	18	[gravity:g:draw draws arrowhead]
23	7	23	15	"So,"
23	29	24	17	the car just
24	25	25	14	rolls down the ramp.
26	9	26	15	Gravity.

Table B.4: Transcript of Video for Subject 4 Device 1.

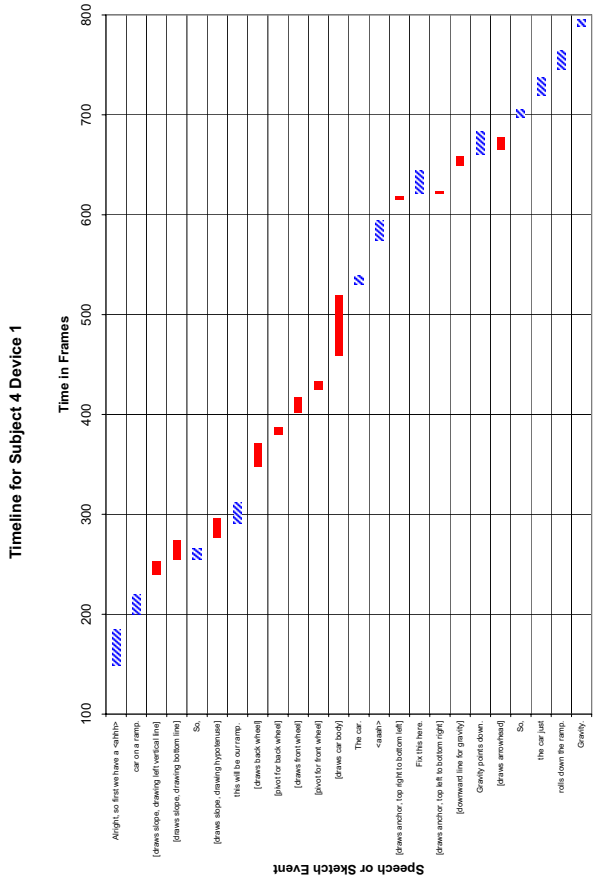


Figure B.8: The timeline graph for Subject 4 Device 1.

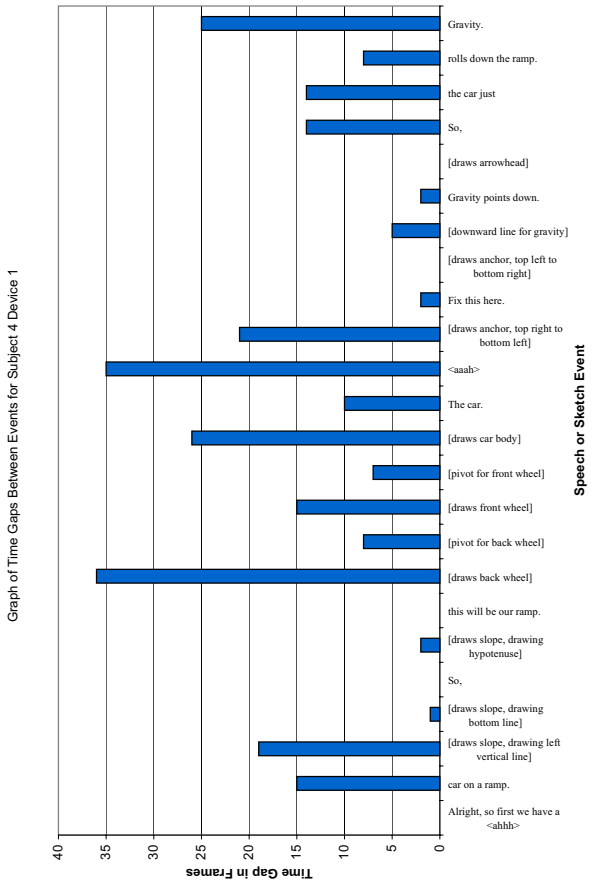


Figure B.9: The gap graph for Subject 4 Device 1.

B.1.5 Subject 5

Start		End		
sec	frame	sec	frame	Task Name
5	25	6	18	"Ok, so"
6	25	7	14	first problem
8	4	8	28	involves
8	22	8	29	"[triangle:t1:draw draws slope, drawing left vertical line]"
9	11	9	23	"[triangle:t1:draw draws slope, drawing hypotenuse]"
9	23	10	4	"[triangle:t1:draw draws slope, bottom line]"
9	26	10	25	an inclined plane
11	21	12	2	<ummm>
12	5	12	18	with a
12	28	13	11	car
13	18	15	29	[polygon:p1:draw draws car body]
14	19	14	29	here
16	2	17	18	up at the top of the inclined plane with
17	28	18	9	[wheel:w1:draw draws back wheel]
18	8	18	22	wheels.
18	25	19	4	[wheel:w2:draw draws front wheel]
20	6	22	5	And these wheels are going to have a certain diameter
22	9	23	25	and we have a certain force of gravity
23	0	23	21	[gravity:g:draw draws gravity arrow]
24	0	24	23	working downward.
25	15	26	4	and <aaah>
26	5	26	29	The wheels are going to
27	0	27	18	spin
28	8	28	17	<aaah>
28	22	31	1	as they spin they are gonna each move a length
31	14	32	22	that's equal to
32	23	34	7	the length of this inclined plane.
34	18	35	22	or at least the first one is
35	23	36	22	and the second one is gonna move
37	15	38	3	some other distance
38	3	39	14	some distance that is slightly less.
40	5	40	16	<aaah>
40	22	41	20	It's less by an amount
41	23	42	17	It's actually less by

Table B.5: Transcript of Video for Subject 5 Device 1.

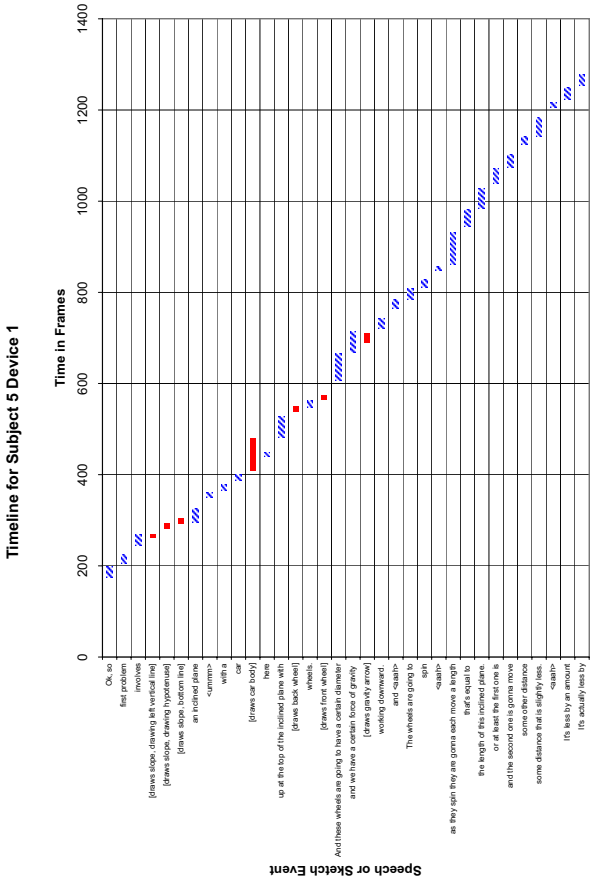


Figure B.10: The timeline graph for Subject 5 Device 1.

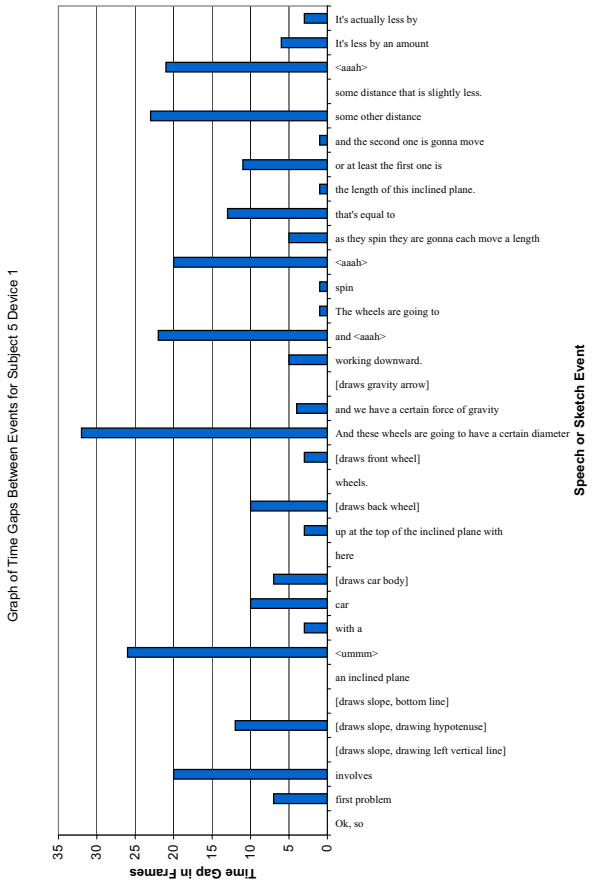


Figure B.11: The gap graph for Subject 5 Device 1.

B.1.6 Subject 6

Start		End		
sec	frame	sec	frame	Task Name
3	11	3	27	Ok.
4	9	5	6	[triangle:t1:draw draws vertical and hoizontal parts of ramp]
4	24	5	27	We begin with some example.
5	20	6	8	[triangle:t1:draw draws hypotonuse of ramp]
6	27	7	21	We've got a ramp.
8	2	8	18	[anchor:a1:draw draws anchor]
8	11	9	19	We anchor it to the background.
10	13	11	16	We put a little car on it.
10	19	11	1	[wheel:w1:draw draws rear wheel]
11	9	11	19	[wheel:w2:draw draws front wheel]
11	27	13	17	[polygon:p1:draw draws car body]
14	4	14	12	[pivot:v1:draw draw rear pivot]
14	5	14	26	Attach the
14	26	15	6	[pivot:v2:draw draw front pivot]
15	14	16	23	wheels to the body of the car.
17	29	19	18	And then we expect the car to roll down.

Table B.6: Transcript of Video for Subject 6 Device 1.

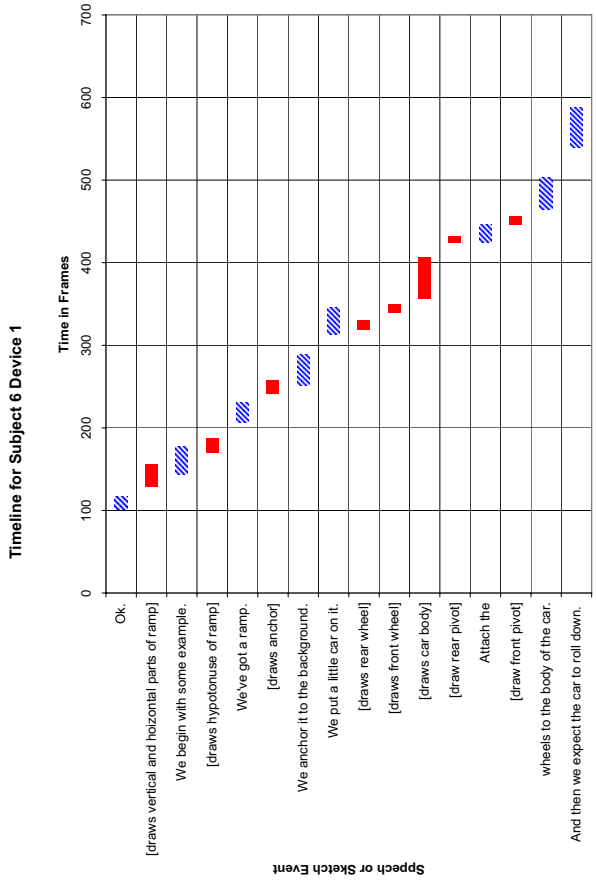


Figure B.12: The timeline graph for Subject 6 Device 1.

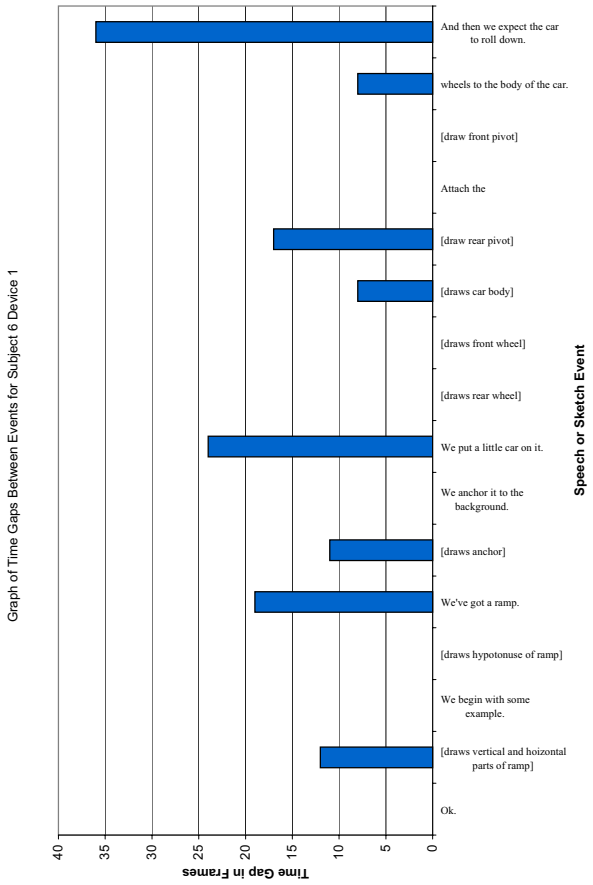


Figure B.13: The gap graph for Subject 6 Device 1.

B.2 Device 2

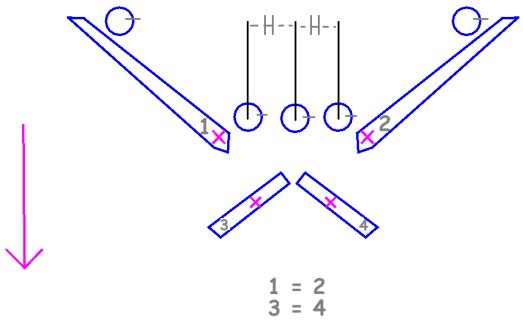


Figure B.14: This is Device 2.

B.2.1 Subject 1

Start		End		
sec	frame	sec	frame	Task Name
3	13	5	3	"Ok, the third one seems like"
6	16	7	11	"I don't know, it's"
7	12	9	8	It's a spring mounted
10	6	10	19	table?
13	16	13	24	Ok.
14	23	15	8	<ummm>
17	14	18	18	"First, I'll draw gravity again."
17	29	18	14	[gravity:g1:draw downward line for gravity]
18	19	19	2	[gravity:g1:draw draw arrowhead]
21	7	22	14	[rectangle:r1:draw draw top block sides and top lines]
22	28	24	12	Draw the horizontal bar.
23	0	23	19	[rectangle:r1:draw draw top block bottom line]
25	12	26	22	[rectangle:r2:draw draw left vertical block]
26	18	27	19	Two vertical bars.
27	8	28	20	[rectangle:r3:draw draw right vertical block]
29	8	29	17	[pivot:v1:draw draw left pivot joint]
29	18	30	4	Join them.
30	5	30	17	[pivot:v1:draw draw right pivot joint]
32	23	33	20	Then I'll draw
33	7	36	20	[polygon:p1:draw draw left support]
35	18	36	8	This thing
36	17	37	15	whatever it is
37	15	37	21	and
38	6	41	29	[polygon:p1:draw draw right support]
38	13	39	25	a similar thing on the other side
44	5	45	27	And then I'll connect them with springs
46	1	47	6	[spring:s1:draw draw left spring]
47	29	49	12	[spring:s2:draw draw right spring]
52	7	53	6	And finally
52	20	53	7	[anchor:a1:draw draw left anchor]
53	22	54	7	[anchor:a2:draw draw right anchor]
53	22	54	26	anchor these two
56	2	56	16	Let's see
56	25	57	29	I think that's – nope
58	12	58	16	there's
59	12	60	9	three <aaah>
61	22	62	7	[circle:c1:draw draw left circle]
62	12	63	6	I don't know what they are
62	24	63	12	[circle:c2:draw draw middle circle]
64	0	64	13	[circle:c3:draw draw right circle]
65	3	65	9	[line:l1:draw draws line on left circle]
65	25	66	1	[line:l2:draw draws line on middle circle]
66	17	66	28	[line:l3:draw draws line on right circle]
67	17	68	22	looks like three <ahhh>
69	18	70	3	<ahhh>
70	13	71	23	pendulums upside-down
88	8	89	24	"Oh, I get it, so the balls drop"
89	25	90	15	and this thing
90	17	91	21	"you know, yeah."

Table B.7: Transcript of Video for Subject 1 Device 2.

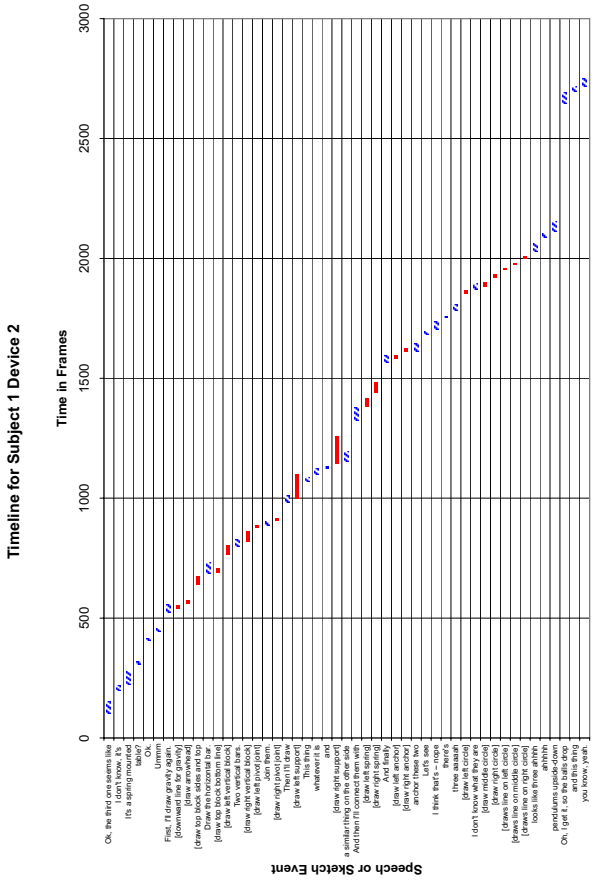


Figure B.15: The timeline graph for Subject 1 Device 2.

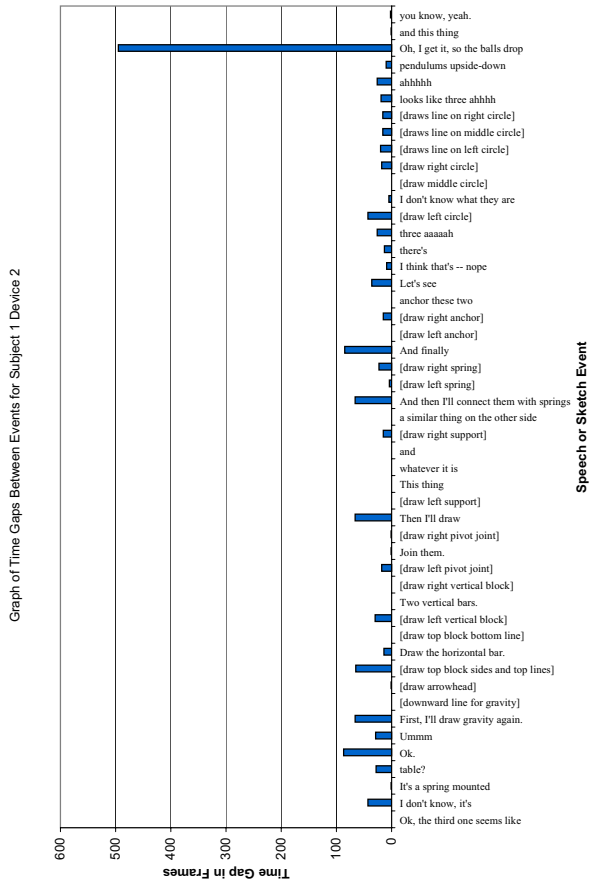


Figure B.16: The gap graph for Subject 1 Device 2.

B.2.2 Subject 3

Start		End		
sec	frame	sec	frame	Task Name
8	5	8	18	“So,”
9	27	10	25	at the top we have
11	16	11	28	[circle:c1:draw draws left circle]
12	17	12	29	[circle:c2:draw draws middle circle]
13	3	13	28	three circles
14	4	14	17	[circle:c3:draw draws right circle]
15	22	16	9	“And then,”
18	19	19	11	Down here we have
19	17	20	6	“two, like”
20	10	23	24	[polygon:p1draw draws left support]
20	23	21	10	buckets
25	12	28	29	[polygon:p2:draw draws right support]
29	14	29	26	And in these
29	27	30	8	buckets
30	8	30	12	are
30	12	31	1	springs
31	9	32	7	[spring:s1:draw draws right spring]
32	26	33	21	[spring:s2:draw draws left spring]
35	2	35	15	And attached
35	18	35	24	to these
35	26	36	7	springs
36	0	36	6	[rectangle:r1:draw draws part of right vertical bar]
36	7	36	17	are these
36	18	38	10	[rectangle:r1:draw draws part of right vertical bar]
36	20	37	9	rods
39	1	41	7	[rectangle:r2:draw draws left vertical support]
41	27	42	5	Which is
42	11	43	2	Which are then like
43	5	43	14	[pivot:p1:draw draws left pivot]
43	6	43	12	attached
43	15	43	23	via a
43	24	44	1	pivot
44	0	44	17	[pivot:p2:draw draws right pivot]
44	3	44	10	point
44	27	45	11	to a rod
45	14	46	2	connecting both
48	13	49	25	[rectangle:r3:draw draws top block]
49	7	49	19	<unintelligible>
52	4	52	17	And
52	17	52	29	of course
52	28	53	10	[gravity:g:draw draws downward line of gravity arrow]
52	29	53	9	there’s
53	12	54	3	and there’s gravity
53	16	54	4	[gravity:g:draw draws arrowhead of gravity arrow]
57	0	57	18	So
59	0	59	15	let’s see
59	19	60	3	moving on to the next picture.
60	3	60	22	to the next picture

Table B.8: Transcript of Video for Subject 3 Device 2.

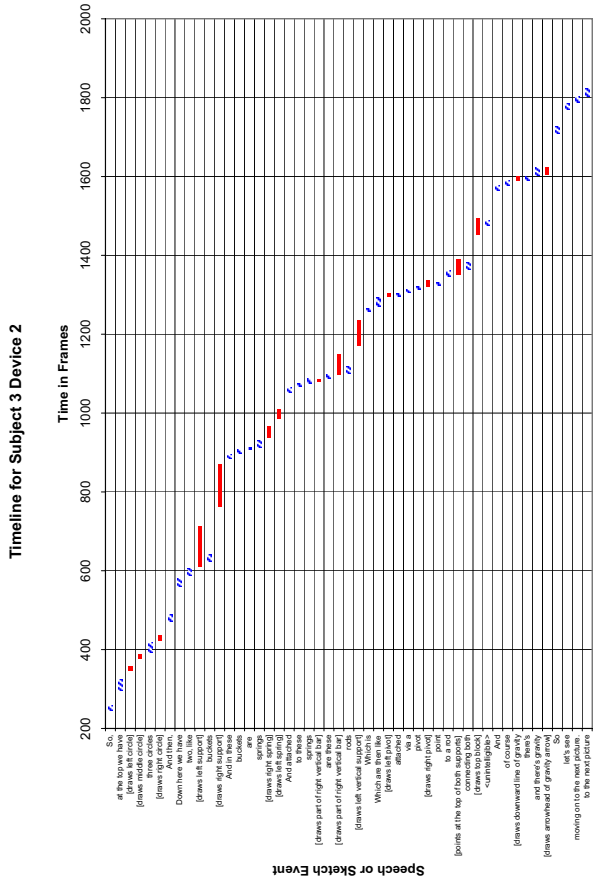


Figure B.17: The timeline graph for Subject 3 Device 2.

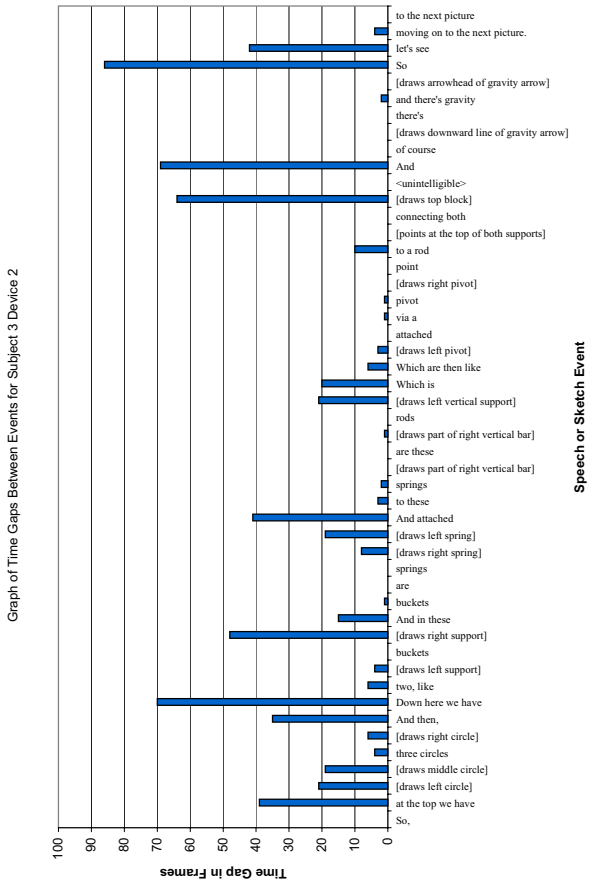


Figure B.18: The gap graph for Subject 3 Device 2.

B.2.3 Subject 4

Start		End		
sec	frame	sec	frame	Task Name
1	13	2	0	"Alright, next"
2	2	2	21	<aaah>
3	28	4	7	[circle:c1:draw draws left circle]
4	3	5	12	got some balls up here
4	22	5	2	[circle:c2:draw draws middle circle]
5	20	6	4	[circle:c3:draw draws right circle]
6	22	7	24	"And, <ahhh>"
11	7	12	14	[rectangle:r1:draw draws top block]
12	2	12	17	platform
13	17	14	0	and
14	8	15	4	[rectangle:r2:draw draws left vertical block]
15	2	16	2	connected to
15	24	16	16	[rectangle:r3:draw draws right vertical block]
17	7	18	0	[spring:s1:draw draws left spring]
17	10	18	3	a springs
18	22	19	16	[spring:s2:draw draws right spring]
21	4	23	12	[polygon:p1:draw draws left support]
22	3	22	19	here
24	1	26	9	[polygon:p2:draw draws right support]
28	9	28	22	And <ahhh>
28	28	29	13	these are fixed
29	3	29	6	"[anchor:a1:draw draws left anchor, from top right to bottom left]"
29	8	29	10	"[anchor:a1:draw draws left anchor, from top left to bottom right]"
29	27	29	29	"[anchor:a2:draw draws right anchor, from top right to bottom left]"
30	1	30	4	"[anchor:a2:draw draws right anchor, from top left to bottom right]"
30	24	31	24	and these are connected
30	27	31	5	[pivot:v1:draw draws left pivot]
31	18	31	26	[pivot:v2:draw draws right pivot]
33	27	34	5	So
34	24	35	15	and gravity goes down
34	28	35	13	[gravity:g:draw draws gravity arrow]
35	15	36	17	So when the thing starts
37	7	37	18	<ahhh>
38	10	38	29	balls 'll fall down
39	2	39	27	springs will bounce

Table B.9: Transcript of Video for Subject 4 Device 2.

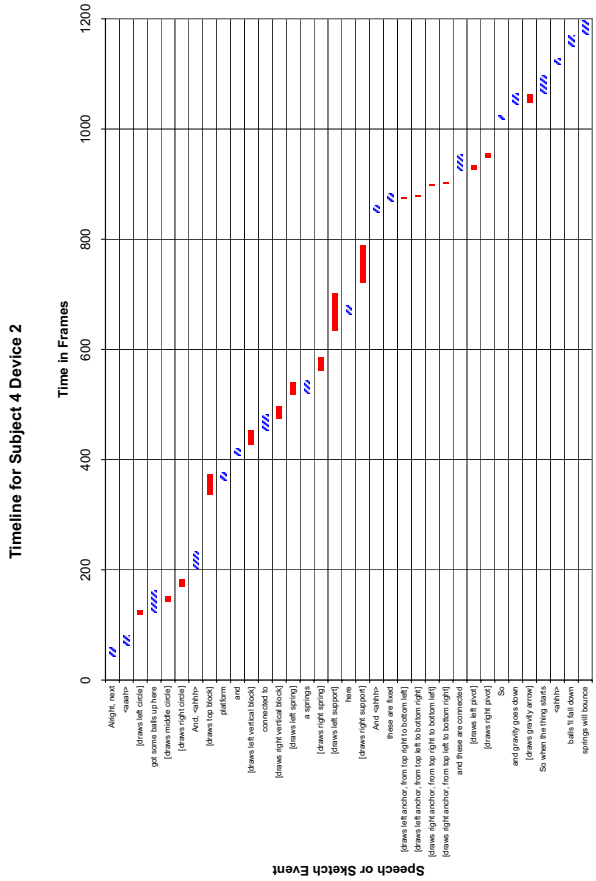


Figure B.19: The timeline graph for Subject 4 Device 2.

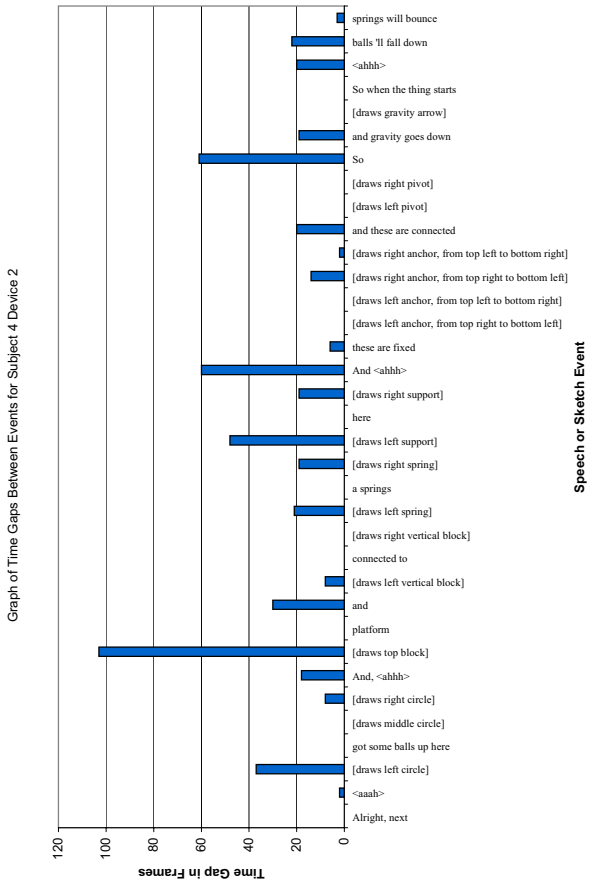


Figure B.20: The gap graph for Subject 4 Device 2.

B.2.4 Subject 6

Start		End		
sec	frame	sec	frame	Task Name
1	9	2	5	The second system
8	12	8	21	Are
8	27	9	18	Are these balls
9	18	10	14	attached to the background?
14	23	15	9	Ohhh!
15	13	15	29	“Oh, I see”
15	29	16	25	“I see, I see, I see, I see.”
17	28	18	11	“Ok, we’ve got”
18	12	18	26	a system
18	27	21	26	[polygon:p1:draw draws left support]
22	29	26	5	[polygon:p2:draw draws right support]
23	11	23	17	with
23	24	24	23	two identical
27	25	27	29	[anchor:a1:draw draws part of right anchor]
28	0	29	7	support structures
28	1	28	7	[anchor:a1:draw draws part of right anchor]
28	26	29	0	[anchor:a2:draw draws part of left anchor]
29	3	29	8	[anchor:a2:draw draws part of left anchor]
31	0	31	15	We’ve got a
31	4	33	6	[rectangle:r1:draw draws top block]
31	16	32	10	horizontal bar
33	27	34	11	And we’ve got
34	15	35	29	[rectangle:r2:draw draws left vertical block]
34	19	35	15	two identical
36	21	38	6	[rectangle:r3:draw draws right vertical block]
36	24	37	14	pistons
38	1	38	13	If you will
38	18	38	27	[pivot:v1:draw draws right pivot]
38	20	39	18	attached to the bar
39	13	39	23	[pivot:v2:draw draws left pivot]
40	17	41	0	And
40	18	41	23	[spring:s1:draw draws left spring]
41	23	42	2	Locked
42	5	42	22	in the support
42	24	43	14	structures
43	3	44	2	[spring:s2:draw draws right spring]
44	3	44	24	is springs
45	17	46	2	And we’ve got
45	22	46	4	[circle:c1:draw draws left ball]
46	21	47	5	[circle:c2:draw draws middle ball]
46	27	47	8	three
47	13	48	16	identical balls
47	13	47	25	[circle:c3:draw draws right ball]
48	26	50	9	that will drop and such

Table B.10: Transcript of Video for Subject 6 Device 2.

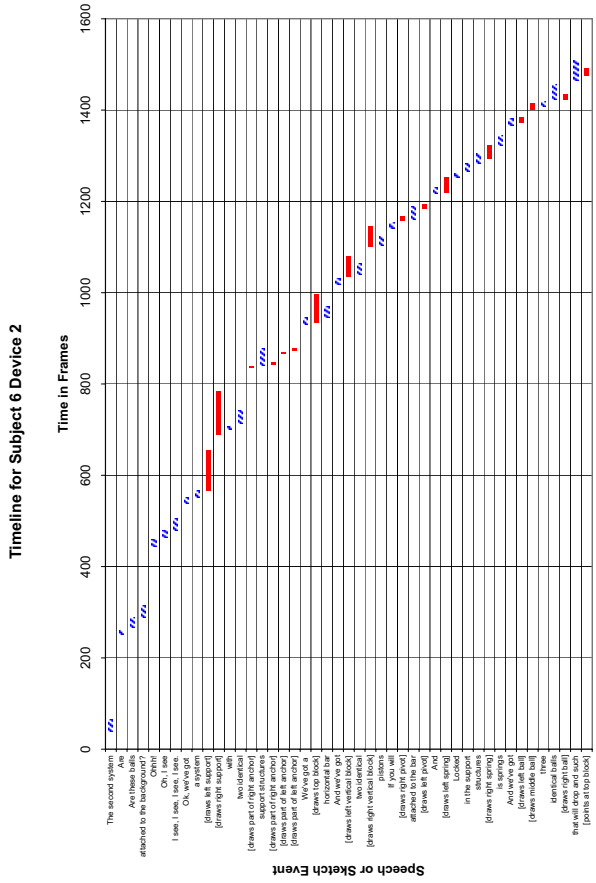


Figure B.21: The timeline graph for Subject 6 Device 2.

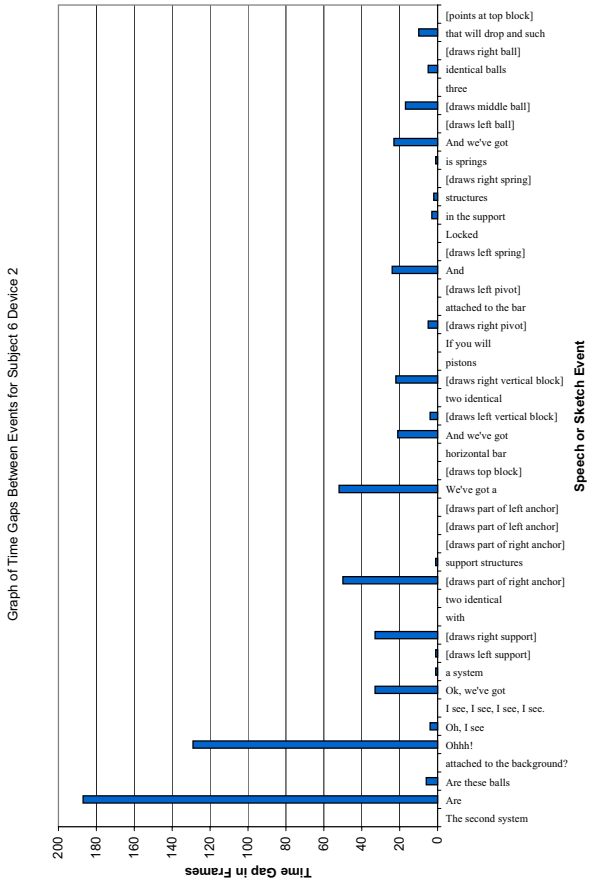


Figure B.22: The gap graph for Subject 6 Device 2.

B.3 Device 3

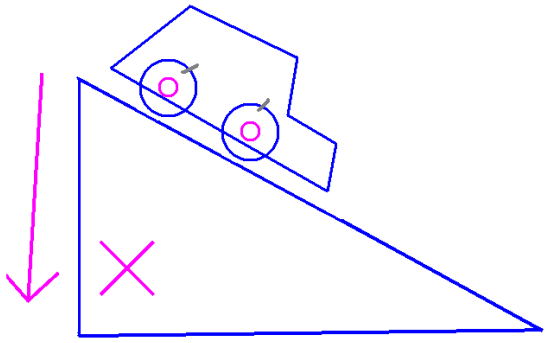


Figure B.23: This is Device 3.

B.3.1 Subject 1

Start		End		
sec	frame	sec	frame	Task Name
1	1	1	4	Ok.
2	4	3	2	Let's move on to the fifth one.
13	26	14	6	Ok.
15	1	15	29	Looks like some
18	16	20	12	"Looks like the bottom of a pinball game, but"
22	24	23	4	We'll start
23	5	23	23	drawing gravity
23	10	23	20	[gravity:g:draw draws downward line for gravity]
23	28	24	9	[gravity:g:draw draws arrowhead for gravity]
25	8	25	24	and
25	29	28	26	there are two identical inclines with balls on them.
28	28	30	5	So we'll start with this
30	5	30	12	[polygon:p1:draw draws part of top left ramp]
31	9	32	6	[polygon:p1:draw draws part of top left ramp]
32	25	33	3	[polygon:p1:draw draws part of top left ramp]
33	13	33	21	[polygon:p1:draw draws part of top left ramp]
34	19	35	17	[polygon:p1:draw draws part of top left ramp]
39	23	40	13	[polygon:p2:draw draws part of top right ramp]
41	0	41	7	[polygon:p2:draw draws part of top right ramp]
41	21	42	15	[polygon:p2:draw draws part of top right ramp]
43	13	44	0	[polygon:p2:draw draws part of top right ramp]
46	3	47	3	we'll weight the inclines
46	10	46	22	[anchor:a1:draw anchors top left ramp]
47	8	47	19	[anchor:a2:draw anchor top right ramp]
50	7	50	16	And
50	18	50	28	[circle:c1:draw draws left ball on ramp]
51	24	52	16	couple balls
51	28	52	10	[circle:c2:draw draws right ball on ramp]
53	23	54	14	We have three
54	15	54	28	[pendulum:d1:draw draws ball of middle pendulum]
55	6	55	24	pendulums
55	8	55	21	[pendulum:d1:draw draws string of middle pendulum]
56	8	56	21	[pendulum:d2:draw draws ball of right pendulum]
57	0	57	12	[pendulum:d2:draw draws string of right pendulum]
57	26	58	9	[pendulum:d3:draw draws ball of left pendulum]
58	16	58	29	[pendulum:d3:draw draws string of left pendulum]
60	15	61	4	And two
61	6	61	16	more
61	13	61	24	[polygon:p3:draw draws part of lower left ramp]
62	8	62	19	fixed.
62	8	63	7	[polygon:p3:draw draws part of lower left ramp]
64	29	65	16	[polygon:p4:draw draws part of lower right ramp]
65	27	66	15	[polygon:p4:draw draws part of lower right ramp]
68	14	68	23	[anchor:a3:draw anchors lower left ramp]
69	2	69	14	[anchor:a4:draw anchors lower right ramp]
71	8	71	16	ok.

Table B.11: Transcript of Video for Subject 1 Device 3.

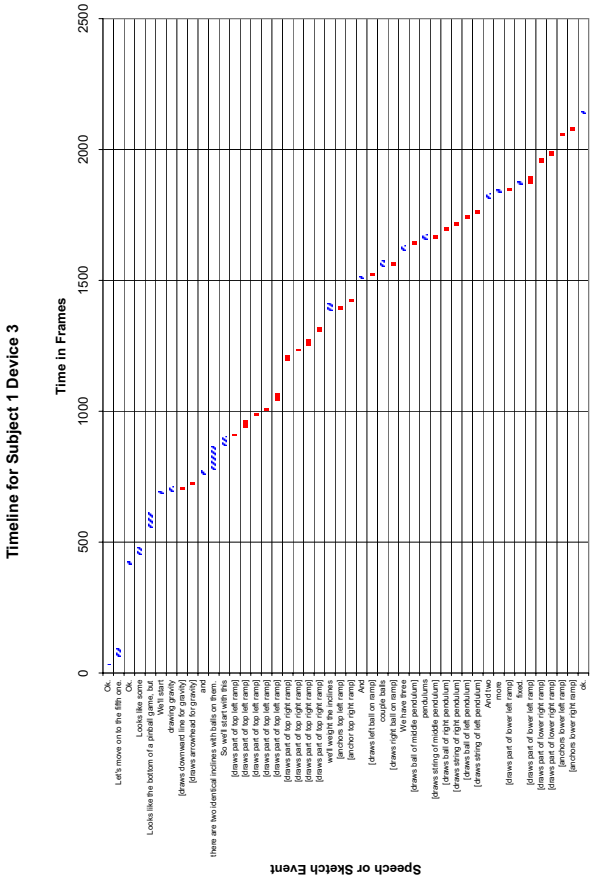


Figure B.24: The timeline graph for Subject 1 Device 3.

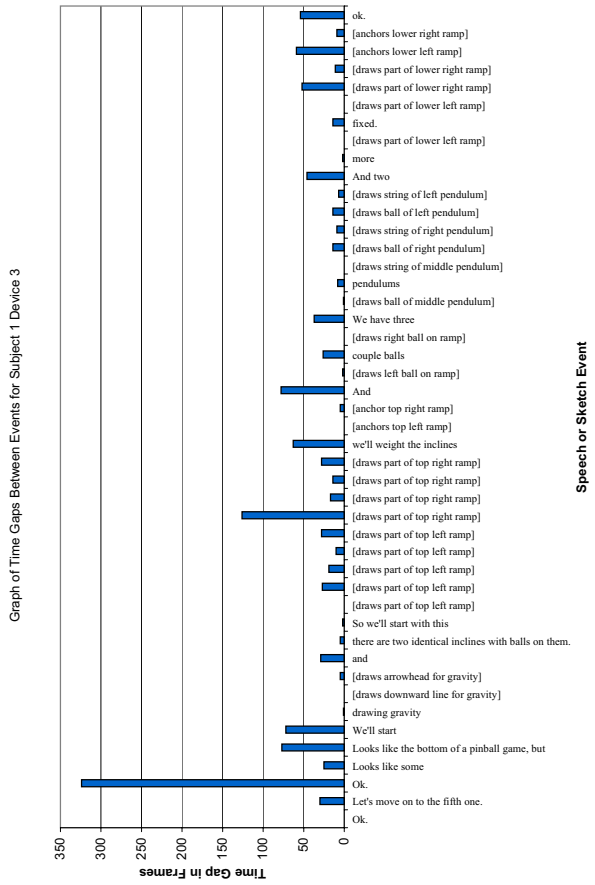


Figure B.25: The gap graph for Subject 1 Device 3.

B.3.2 Subject 4

Start		End		
sec	frame	sec	frame	Task Name
1	23	2	20	"Next, <ahhh>"
4	16	5	19	we should have a couple of ramps
6	8	7	20	[polygon:p1:draw draws part of top left ramp]
7	24	8	11	[polygon:p1:draw draws part of top left ramp]
8	16	8	20	[polygon:p1:draw draws part of top left ramp]
10	4	10	9	[polygon:p1:erase erases part of top left ramp]
10	25	11	8	[polygon:p1:draw draws part of top left ramp]
13	10	14	16	[polygon:p2:draw draws part of top right ramp]
14	22	16	11	[polygon:p2:draw draws part of top right ramp]
17	17	18	7	"And, <ahhh>"
20	4	20	18	down here
20	9	20	20	[polygon:p3:draw draws part of lower left ramp]
21	11	22	6	[polygon:p3:draw draws part of lower left ramp]
22	29	24	6	[polygon:p4:draw draws lower right ramp]
25	5	26	4	And these are all fixed.
25	19	25	26	[anchor:a1:draw anchors top left ramp]
27	8	27	15	[anchor:a2:draw anchors top right ramp]
28	8	28	15	[anchor:a3:draw anchors part of lower right ramp]
29	0	29	9	[anchor:a4:draw anchors lower left ramp]
29	28	30	1	[anchor:a3:draw anchors part of lower right ramp]
30	29	31	10	<ummm>
32	15	32	24	[pendulum:d1:draw draws string of left pendulum]
33	4	33	13	[pendulum:d2:draw draws string of middle pendulum]
33	23	34	5	[pendulum:d3:draw draws string of right pendulum]
34	23	35	29	Three balls hanging here
35	8	35	19	[pendulum:d1:draw draws ball of left pendulum]
35	26	36	6	[pendulum:d2:draw draws ball of middle pendulum]
36	14	36	23	[pendulum:d3:draw draws ball of right pendulum]
37	26	38	11	And
38	16	39	9	two of them start up here.
39	0	39	11	[circle:c1:draw draws left ball on ramp]
40	2	40	10	[circle:c2:draw draws right ball on ramp]
40	16	41	19	And then again gravity's down.
41	7	41	26	[gravity:g:draw draws gravity arrow]
42	6	43	2	"So when it starts, I guess"
43	3	44	5	these balls will roll down the ramps
44	9	44	23	'n hit these
44	27	45	11	'n fall off

Table B.12: Transcript of Video for Subject 4 Device 3.

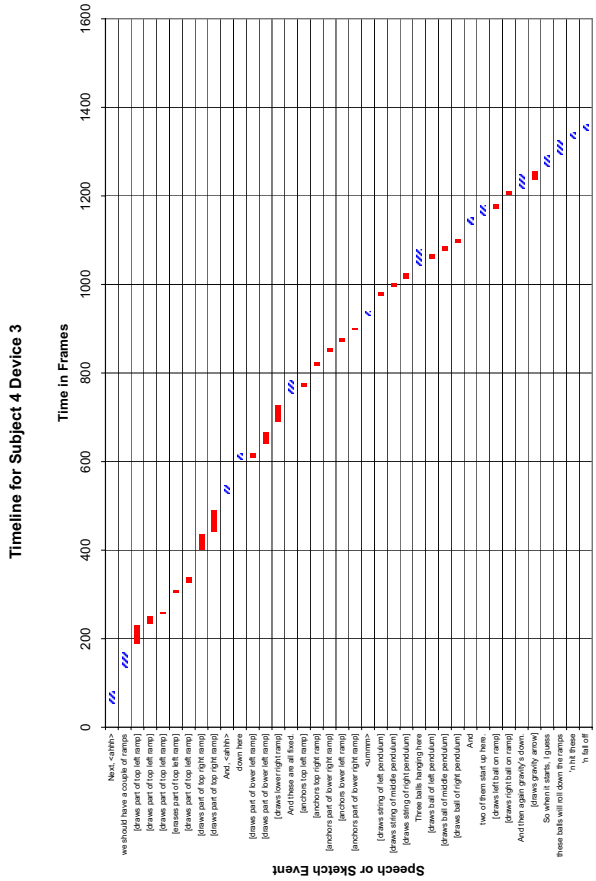


Figure B.26: The timeline graph for Subject 4 Device 3.

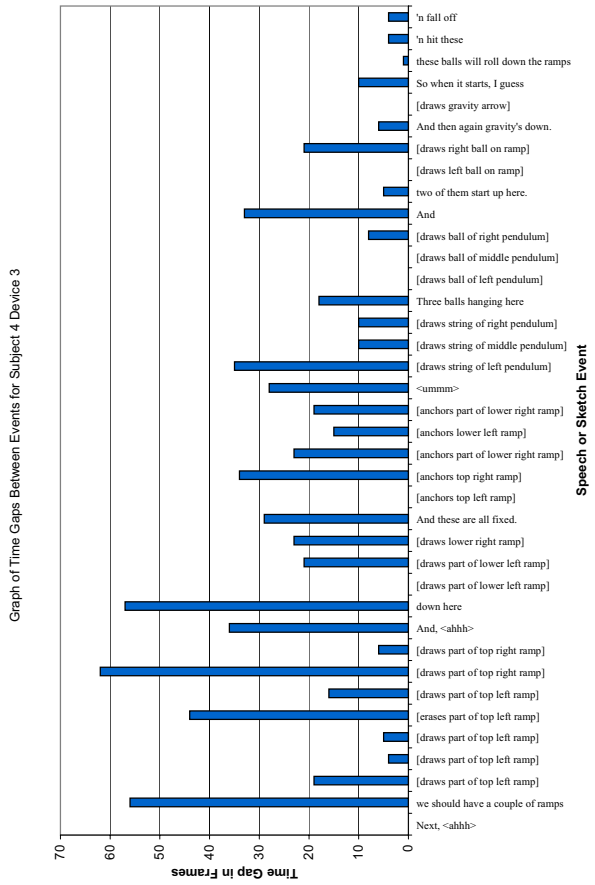


Figure B.27: The gap graph for Subject 4 Device 3.

B.3.3 Subject 6

Start		End		
sec	frame	sec	frame	Task Name
3	4	3	10	<oooh>
4	23	4	29	<hmmm>
8	12	10	13	<oooh> and a little funny ball machine.
10	20	11	0	We've got
10	26	11	22	[polygon:p1:draw draws part of lower left ramp]
12	3	12	16	[polygon:p1:draw draws part of lower left ramp]
12	11	12	19	two
13	6	13	23	[polygon:p2:draw draws part of lower right ramp]
13	13	14	14	identical ramps
14	1	14	14	[polygon:p2:draw draws part of lower right ramp]
16	6	17	18	[polygon:p3:draw draws top right ramp]
16	9	17	28	two large identical ramps
18	9	20	15	[polygon:p4:draw draws top left ramp]
21	17	21	26	[anchor:a1:draw anchors top right ramp]
21	29	22	6	both
22	10	22	17	[anchor:a2:draw anchors top left ramp]
22	24	23	6	anchored
23	0	23	8	[anchor:a3:draw anchors lower left ramp]
23	10	23	15	to the
23	16	23	27	background
23	23	24	3	[anchor:a4:draw anchors lower right ramp]
24	20	25	2	We've got
24	28	25	11	[pendulum:d1:draw draws string of left pendulum]
25	14	25	27	[pendulum:d1:draw draws ball of left pendulum]
26	6	26	16	[pendulum:d2:draw draws string of middle pendulum]
26	12	26	22	three
26	18	27	3	[pendulum:d2:draw draws ball of middle pendulum]
26	28	27	16	identical
27	10	27	19	[pendulum:d3:draw draws string of right pendulum]
27	17	28	10	equally spaced
27	21	28	3	[pendulum:d3:draw draws ball of right pendulum]
28	13	28	27	pendulums
28	28	30	7	placed in the center of the
30	27	31	20	of the structure
31	25	32	7	[circle:c1:draw draws right ball on ramp]
32	9	32	17	and
32	28	33	22	two identical
32	29	33	11	[circle:c2:draw draws left ball on ramp]
33	23	34	3	balls
35	0	35	24	And we let them
35	24	36	28	roll and see what happens

Table B.13: Transcript of Video for Subject 6 Device 3.

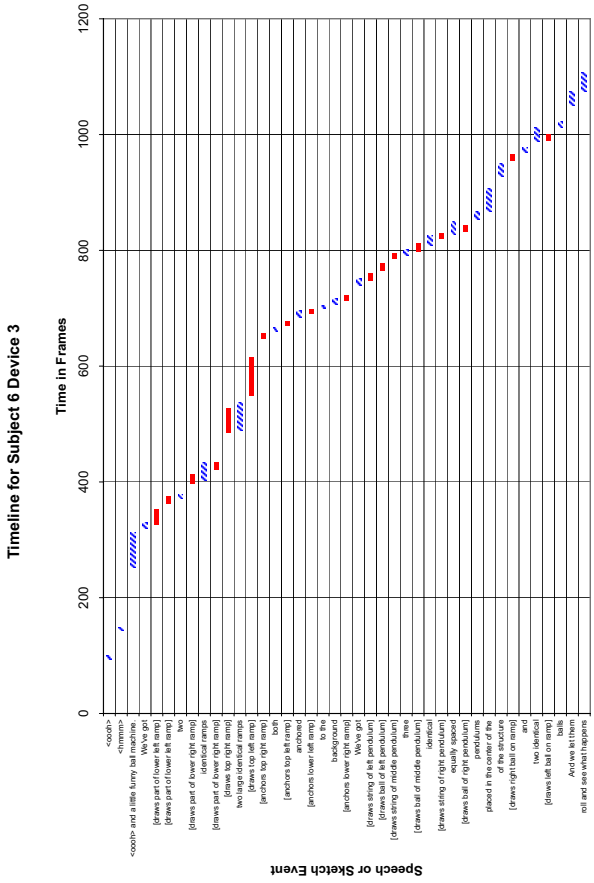


Figure B.28: The timeline graph for Subject 6 Device 3.

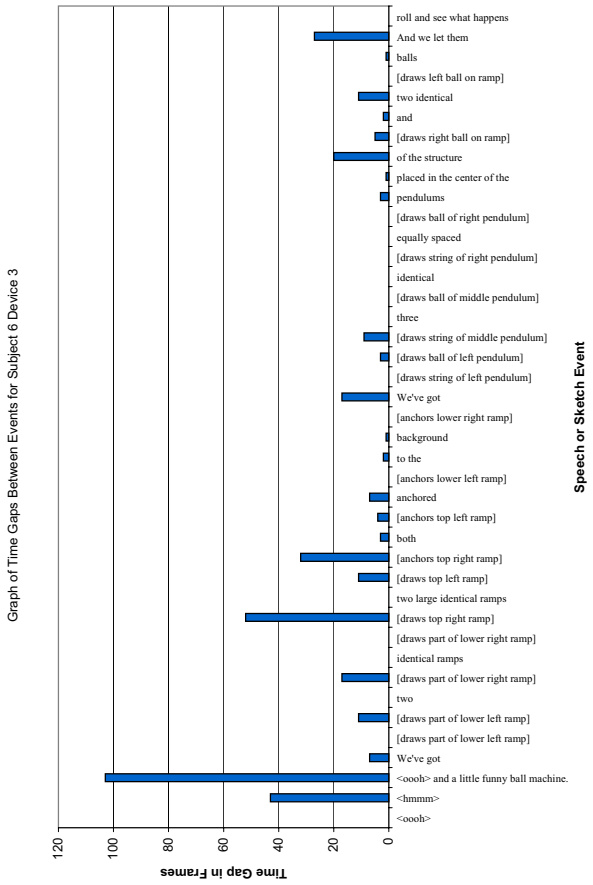


Figure B.29: The gap graph for Subject 6 Device 3.

B.4 Device 4

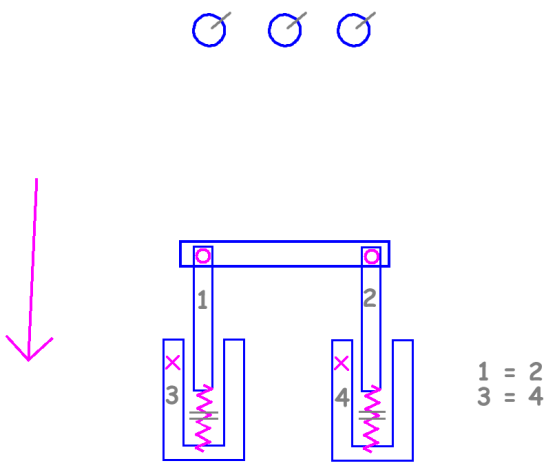


Figure B.30: This is Device 4.

B.4.1 Subject 1

Start		End		
sec	frame	sec	frame	Task Name
5	29	6	20	It looks like
7	15	10	5	two pulleys and a group of masses?
14	14	15	5	What are the <ahhh>
23	8	24	5	and the dotted lines?
29	14	29	21	Ok.
30	21	31	4	<ummm>
32	2	33	27	So first I'll draw gravity again
33	8	33	20	[gravity:g:draw downward line for gravity]
34	1	34	16	[gravity:g:draw draw arrowhead]
34	9	34	29	pointing downward.
36	18	37	2	I'll draw the
37	2	37	23	[pulley:p1:draw draw left pulley outside circle]
37	22	38	16	two pulleys.
38	2	38	13	[pulley:p1:draw draw left pulley inside circle]
38	29	39	19	[pulley:p2:draw draw right pulley outside circle]
39	26	40	8	[pulley:p2:draw draw right pulley inside circle]
42	21	44	6	There's a line coming off one pulley
43	10	43	25	"[line:l1:draw draw left pulley, left line]"
44	15	45	21	line coming off another pulley.
44	18	45	6	"[line:l2:draw draw left pulley, right line]"
46	17	47	0	"[line:l3:draw draw right pulley, left line]"
47	14	47	26	"[line:l4:draw draw right pulley, right line]"
49	9	49	22	and
51	17	52	20	[rectangle:r1:draw draw left block]
53	7	54	10	[rectangle:r2:draw draw middle block]
54	23	56	2	[rectangle:r3:draw draw right block]
55	4	55	26	three weights
56	28	57	14	plus
57	19	58	0	[line:l5:draw draw middle line down]
58	16	59	21	[rectangle:r4:draw draw bottom block]
58	29	60	2	a fourth weight in the middle.
62	9	62	26	So that's it.

Table B.14: Transcript of Video for Subject 1 Device 4.

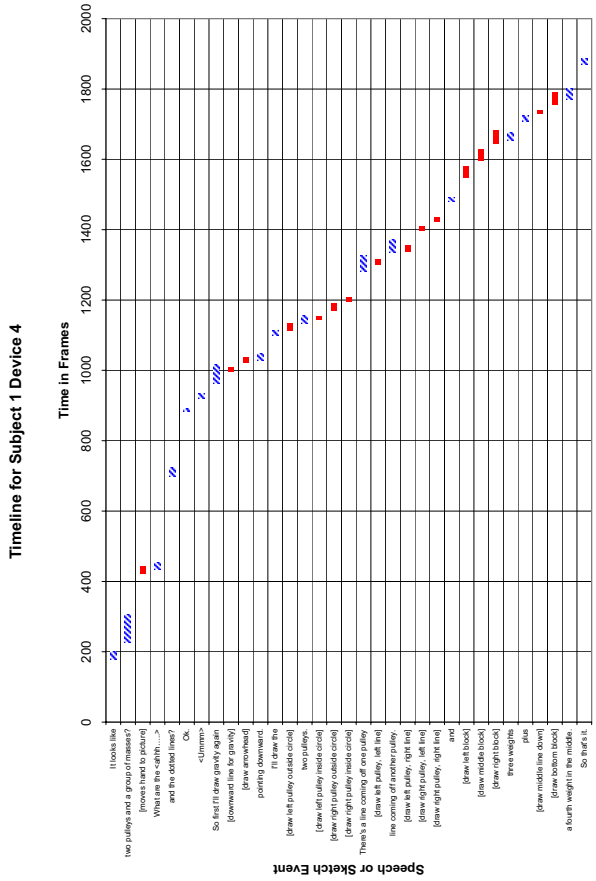


Figure B.31: The timeline graph for Subject 1 Device 4.

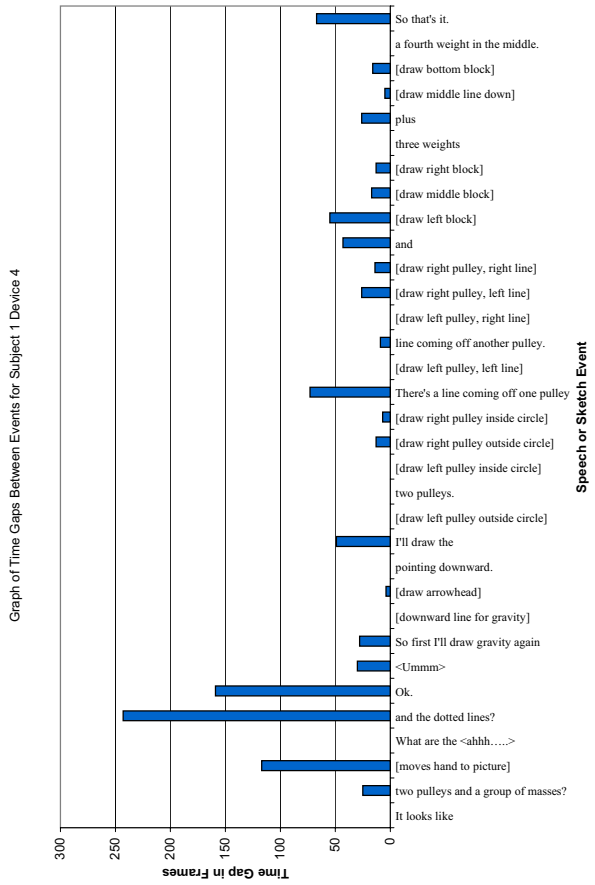


Figure B.32: The gap graph for Subject 1 Device 4.

B.4.2 Subject 4

Start		End		
sec	frame	sec	frame	Task Name
1	15	2	21	Next to it we have some <ahhh>
3	2	4	0	blocks and pulleys
7	7	7	27	[rectangle:r1:draw draws top left block]
8	9	9	2	[rectangle:r2:draw draws top middle block]
10	12	11	10	[rectangle:r3:draw draws top right block]
12	0	12	22	We've got these
12	29	13	22	[rectangle:r4:draw draws bottom block]
14	6	14	19	four blocks
15	1	15	18	and <ahhh>
17	1	17	13	"[pulley:p1:draw draws left pulley, outside circle]"
17	20	17	26	"[pulley:p1:draw draws left pulley, inside circle]"
17	21	18	19	I'll start with two pulleys
18	12	18	26	"[pulley:p2:draw draws right pulley, outside circle]"
19	5	19	14	"[pulley:p2:draw draws right pulley, inside circle]"
19	21	20	5	and
21	17	22	15	[line:l1:draw draws left rope]
21	20	22	9	these are connected
22	13	23	2	over the pulley
23	5	23	22	as are
24	0	25	1	[line:l2:draw draws right rope]
24	24	25	2	these
26	9	26	25	and then <ahhh>
28	0	29	2	this block is connected to this block.
28	5	28	17	[line:l3:draw draws middle rope]
30	16	31	5	"So,"
33	29	34	24	I guess when it starts
34	24	35	9	<ahhh>
36	13	37	2	these aren't connected
37	24	38	16	so these just fall down
47	13	48	18	"Oh, and gravity points down."

Table B.15: Transcript of Video for Subject 4 Device 4.

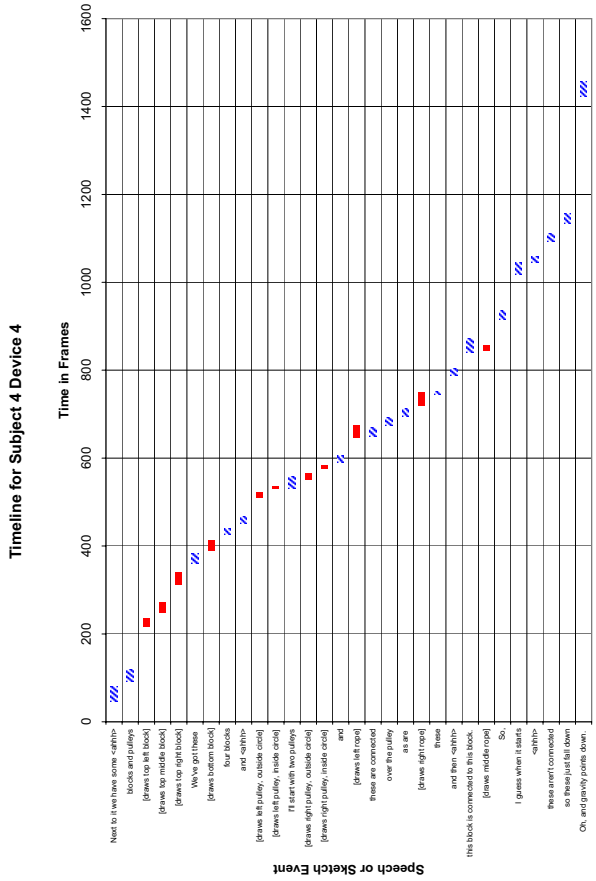


Figure B.33: The timeline graph for Subject 4 Device 4.

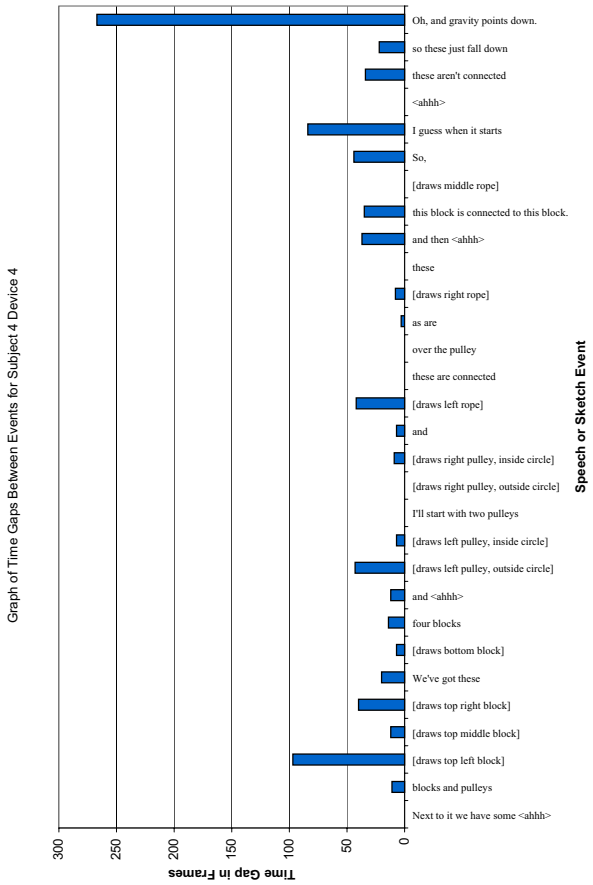


Figure B.34: The gap graph for Subject 4 Device 4.

B.4.3 Subject 5

Start		End		
sec	frame	sec	frame	Task Name
0	25	1	13	"Right, and the"
1	14	1	28	the fourth
2	4	2	25	in the fourth case
2	25	3	7	we have
3	15	3	26	<ummm>
4	1	4	7	two
4	9	4	24	pulleys
4	23	5	18	"[pulley:p1:draw draws left pulley, outside circle]"
5	24	6	6	"[pulley:p1:draw draws left pulley, inside circle]"
6	26	7	17	"[pulley:p2:draw draws right pulley, outside circle]"
7	25	8	5	"[pulley:p2:draw draws right pulley, inside circle]"
11	18	11	24	[rectangle:r1:draw draws part of top middle block]
11	22	12	11	which are
12	4	12	23	[rectangle:r1:draw draws part of top middle block]
13	14	14	20	attached to
15	1	16	5	immediately attached
16	5	16	10	to
16	8	16	14	[rectangle:r2:draw draws part of top right block]
16	17	17	10	three blocks
16	20	17	8	[rectangle:r2:draw draws part of top right block]
18	3	18	9	[rectangle:r3:draw draws part of top left block]
18	15	19	7	[rectangle:r3:draw draws part of top left block]
19	29	20	8	And
20	8	20	20	this is
20	16	20	27	[arrow:a1:draw draws downward line for gravity]
21	4	21	16	[arrow:a1:draw draws arrowhead for gravity]
21	25	22	23	it's the y axis
22	24	23	17	so it's suspended
25	5	25	18	"So, we have"
25	21	26	8	ropes
25	21	27	24	[line:l1:draw draws left rope]
26	14	27	7	two ropes
28	22	30	24	[line:l2:draw draws right rope]
29	3	29	28	which are attached
30	21	31	7	which go aro–
31	7	31	27	which are <ahhh>
32	9	32	28	attached to the pull–
32	28	33	23	which are suspended
33	23	34	13	over the pulleys
34	22	35	18	and attached
36	5	36	11	<ummm>
36	15	37	5	each to
37	8	38	2	an outside block
38	4	38	26	and they're both attached
38	26	39	12	to the center block.
40	9	41	2	and the center block
41	5	41	19	has
42	6	42	11	<uhhh>
42	18	42	28	[line:l3:draw draws middle rope]
42	19	43	0	another
43	16	44	21	[rectangle:r4:draw draws bottom block]
44	15	44	22	rope

Table B.16: Transcript of Video for Subject 5 Device 4, Part 1.

Start		End		
sec	frame	sec	frame	Task Name
46	8	46	17	which is
46	18	47	7	connected to
47	10	47	18	a block
47	21	48	11	hanging beneath it
48	21	49	20	of the same size
50	20	51	15	these are all the same size
51	22	52	16	and the pulleys
52	16	53	14	are obviously the same size
54	10	55	6	"<ahhh>, so"
55	28	56	14	the question is
56	14	56	23	this is
56	23	57	15	this is an equilibrium
57	15	57	25	problem
58	14	59	1	where is it gonna
59	1	59	8	where
59	8	60	7	where is this <ahhh>
61	18	62	8	where are these blocks
62	8	62	28	gonna be relative
62	28	63	15	to each other
63	18	63	22	when the
63	28	64	16	when the system becomes
64	16	64	26	stable

Table B.17: Transcript of Video for Subject 5 Device 4, Part 2.

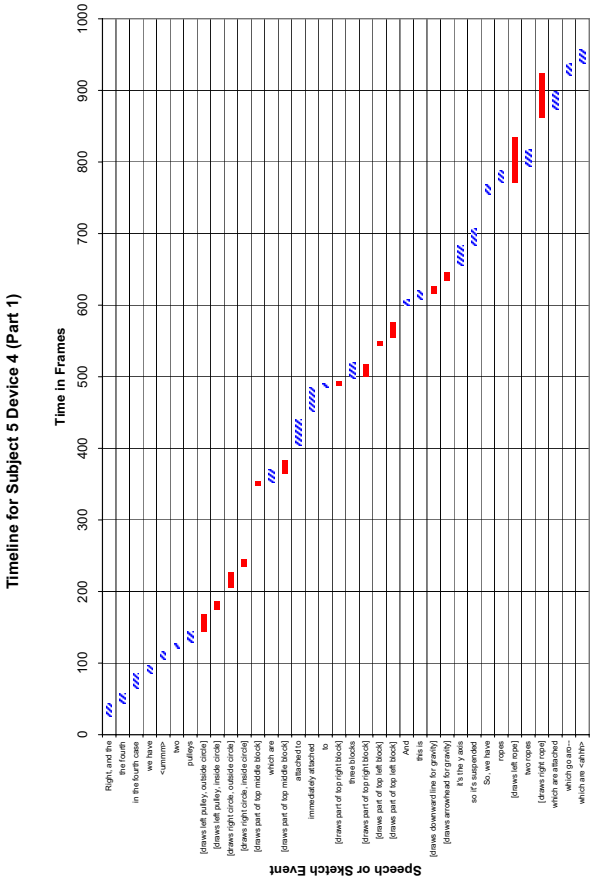


Figure B.35: The timeline graph for Subject 5 Device 4, Part 1.

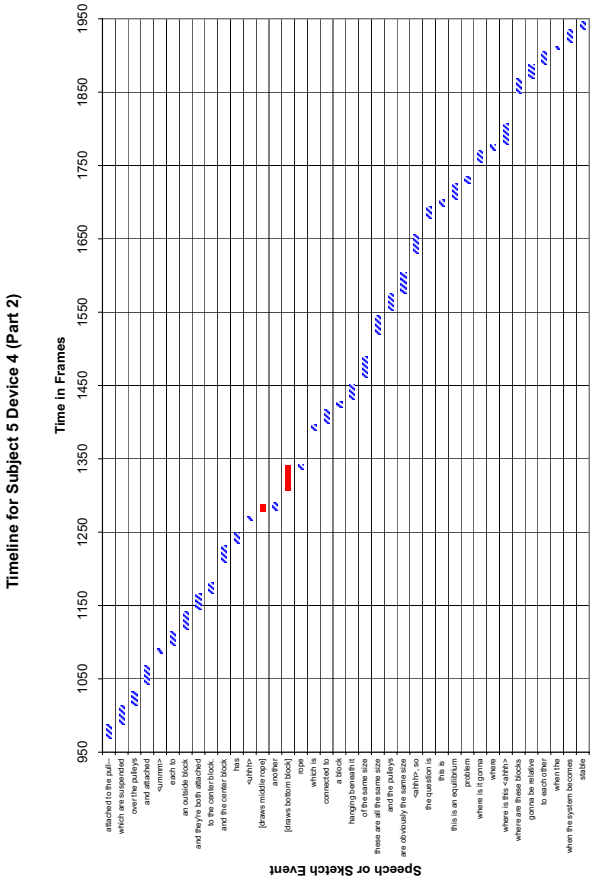


Figure B.36: The timeline graph for Subject 5 Device 4, Part 2.

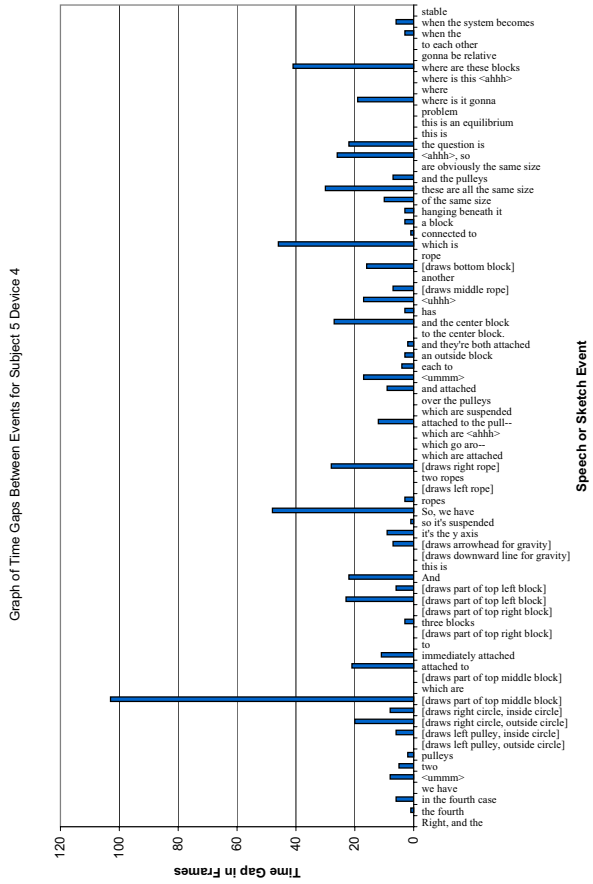


Figure B.37: The gap graph for Subject 5 Device 4.

B.4.4 Subject 6

Start		End		
sec	frame	sec	frame	Task Name
0	14	1	17	The problem is that I can make
2	0	3	0	educated guesses
3	0	4	5	as to what you are looking for.
8	13	9	9	In this structure
12	1	12	15	We've got
13	9	13	25	[rectangle:r1:draw draws part of top left block]
14	3	14	16	[rectangle:r1:draw draws part of top left block]
15	7	15	20	[rectangle:r2:draw draws part of top middle block]
15	15	15	21	four
15	24	16	6	cubes
15	29	16	11	[rectangle:r2:draw draws part of top middle block]
16	25	17	28	[rectangle:r3:draw draws top right block]
17	23	18	8	three like this
18	8	19	3	equally spaced
19	7	19	15	And the
19	18	19	24	fourth
19	20	20	8	[rectangle:r4:draw draws part of bottom block]
19	26	20	4	one
20	17	21	3	[rectangle:r4:draw draws part of bottom block]
21	3	21	17	attached
21	20	22	7	to this one
21	22	22	5	[line:l1:draw draws middle rope]
23	11	23	29	Then we've got
24	5	24	22	"[pulley:p1:draw draws left pulley, outside circle]"
24	29	25	5	two pulleys
25	7	25	23	"[pulley:p2:draw draws right pulley, outside circle]"
25	10	25	25	pulleys
25	29	26	8	"[pulley:p2:draw draws right pulley, inside circle]"
26	23	27	5	"[pulley:p1:draw draws left pulley, inside circle]"
28	11	28	16	half
28	17	28	24	way
29	8	29	11	half
29	13	30	3	way inbetween
30	3	30	24	these two pairs of
30	26	31	8	guys
31	9	31	23	[line:l2:draw draws left part of left rope]
32	7	32	21	[line:l3:draw draws right part of left rope]
32	20	33	11	supporting
33	6	33	19	[line:l4:draw draws left part of right rope]
34	4	34	20	[line:l5:draw draws right part of right rope]
34	15	35	6	these three
35	16	36	15	cubes like this

Table B.18: Transcript of Video for Subject 6 Device 4.

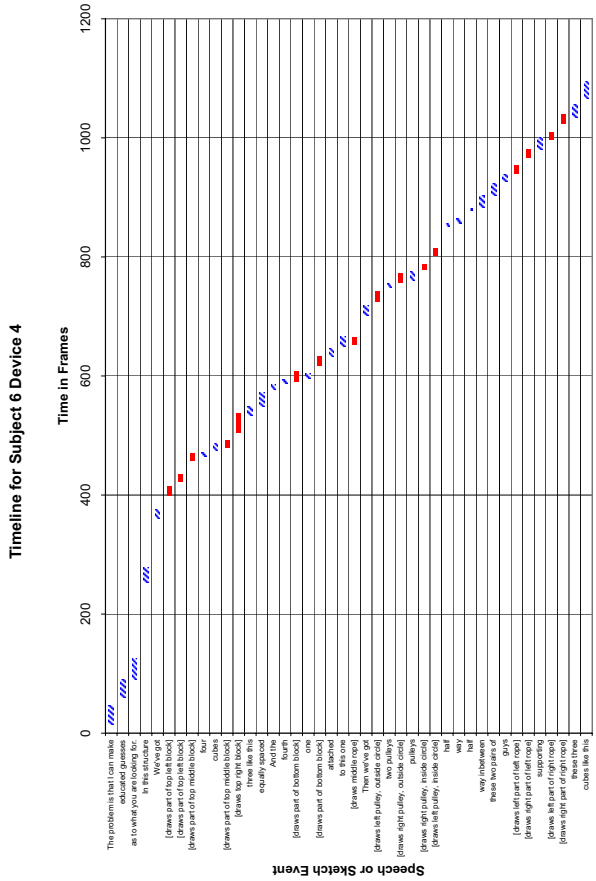


Figure B.38: The timeline graph for Subject 6 Device 4.

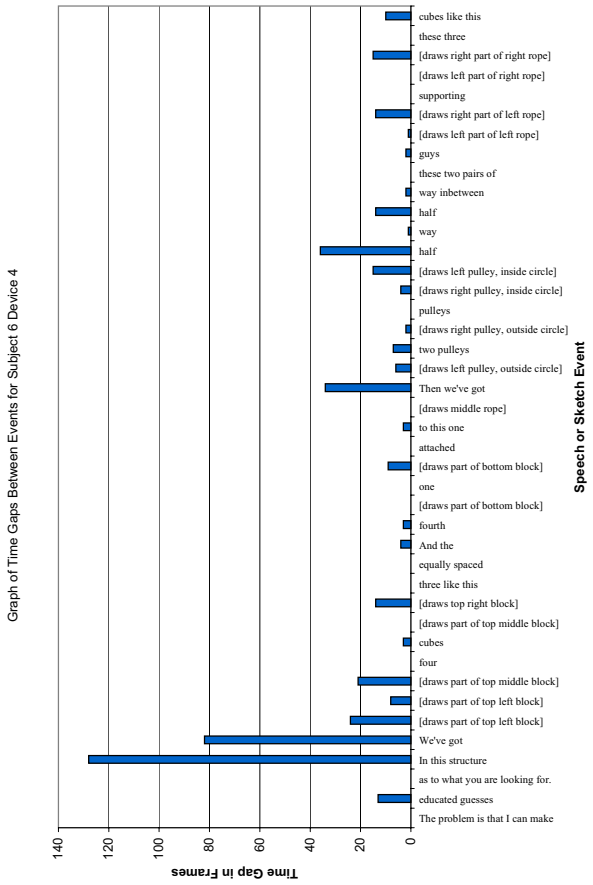


Figure B.39: The gap graph for Subject 6 Device 4.

B.5 Device 5

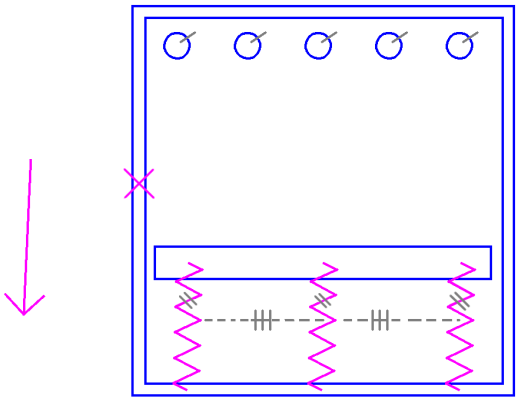


Figure B.40: This is Device 5.

B.5.1 Subject 2

Start		End		
sec	frame	sec	frame	Task Name
7	27	8	7	Ok
8	14	9	16	"In the fifth one,"
9	16	9	28	there's a
10	3	10	15	big
10	18	11	4	box
10	19	11	8	[rectangle:r1:draw draws part of outside box]
11	23	13	27	[rectangle:r1:draw draws part of outside box]
14	16	17	19	[rectangle:r2:draw draws inside box]
16	18	17	9	That actually has a
17	12	17	27	thickness to it.
18	1	18	20	And it's fixed.
18	10	18	16	"[anchor:a1:draw draws anchor, top left to bottom right]"
18	22	18	28	"[anchor:a1:draw draws anchor, top right, to bottom left]"
19	26	20	29	And we have
21	18	22	2	[circle:c1:draw draws left most ball]
22	8	22	22	a bunch of
22	11	22	23	[circle:c2:draw draws second ball from left]
23	2	23	14	[circle:c3:draw draws middle ball]
23	23	24	5	[circle:c4:draw draws second ball from right]
23	26	24	25	circles <unitelligible>
24	13	24	27	[circle:c5:draw draws right most ball]
25	24	26	15	probably balls.
26	20	27	13	And then we have
27	16	27	26	this
28	3	30	9	[rectangle:r3:draw draws platform inside box]
29	8	29	18	<ummm>
30	22	31	5	table
31	11	32	3	that's suspended
32	5	32	24	by springs
32	9	33	12	[spring:s1:draw draws left spring]
33	14	33	29	on the bottom.
33	28	35	1	[spring:s2:draw draws middle spring]
35	17	36	26	[spring:s3:draw draws right spring]
37	19	38	5	and
39	3	39	19	we have gravity
39	10	39	23	[gravity:g:draw draws downward line for gravity]
39	20	40	2	that pulls
40	0	40	10	[gravity:g:draw draws arrowhead for gravity]
40	3	40	19	the balls down.
41	11	41	24	That's about it.
42	23	44	25	And the springs are equidistant
45	19	46	5	from each other.

Table B.19: Transcript of Video for Subject 2 Device 5.

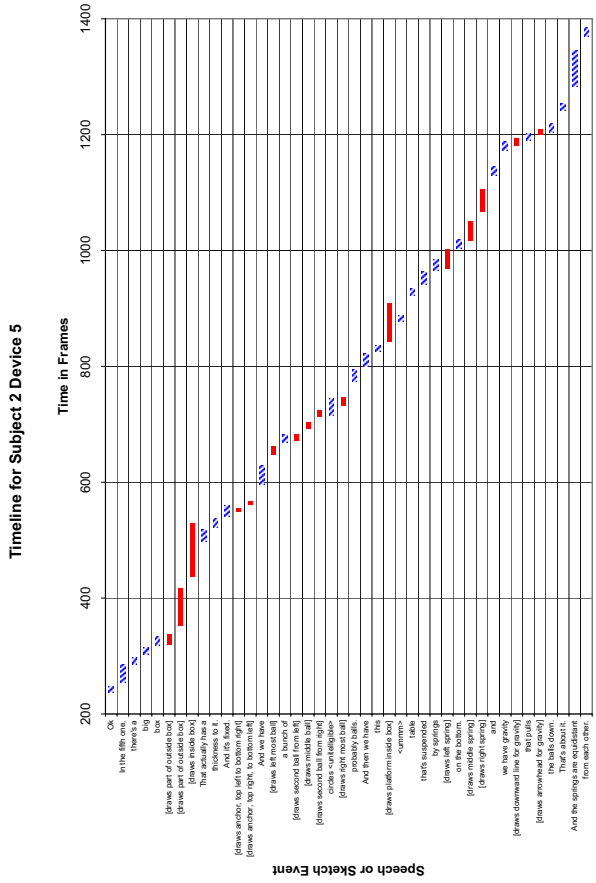


Figure B.41: The timeline graph for Subject 2 Device 5.

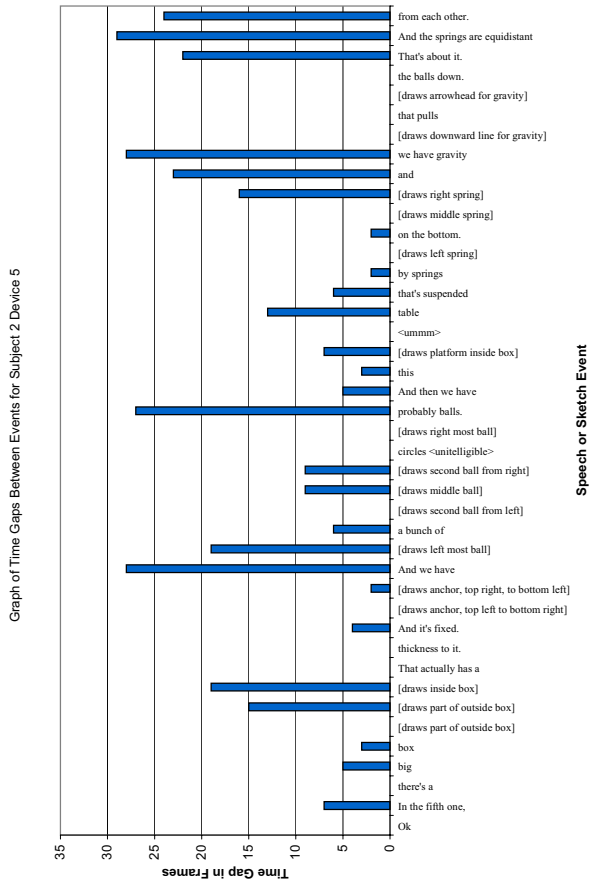


Figure B.42: The gap graph for Subject 2 Device 5.

B.5.2 Subject 3

Start		End		
sec	frame	sec	frame	Task Name
11	12	11	26	So now
11	29	12	8	we have a
12	9	12	26	box
12	23	13	26	[rectangle:r1:draw draws part of outside box]
14	16	15	16	[rectangle:r1:draw draws part of outside box]
16	28	20	19	[rectangle:r2:draw draws inside box]
22	13	23	12	with five circles
23	25	24	8	[circle:c1:draw draws left most ball]
24	5	24	26	inside
25	10	25	27	on the top
26	5	26	15	[circle:c2:draw draws second ball from left]
27	5	27	16	[circle:c3:draw draws middle ball]
28	1	28	11	[circle:c4:draw draws second ball from right]
28	25	29	5	[circle:c5:draw draws right most ball]
33	2	33	16	And
34	7	34	29	Then we have
36	9	36	19	like a
36	10	36	20	[rectangle:r3:draw draws part of platform inside box]
37	12	39	25	[rectangle:r3:draw draws part of platform inside box]
37	25	39	8	divider in that box.
40	28	42	11	[rectangle:r3:draw draws part of platform inside box]
42	28	43	20	towards the bottom
43	27	44	15	which is
44	22	45	23	attached to the bottom
46	1	46	22	[spring:s1:draw draws left spring]
46	17	47	10	via three
47	15	48	6	[spring:s2:draw draws middle spring]
48	10	48	28	equally spaced
48	22	50	2	[spring:s3:draw draws right spring]
49	22	50	9	springs
51	9	51	23	And then
52	21	54	0	with gravity in this equation
52	28	53	5	[gravity:g:draw draws downward line for gravity]
53	7	53	22	[gravity:g:draw draws arrowhead for gravity]

Table B.20: Transcript of Video for Subject 3 Device 5.

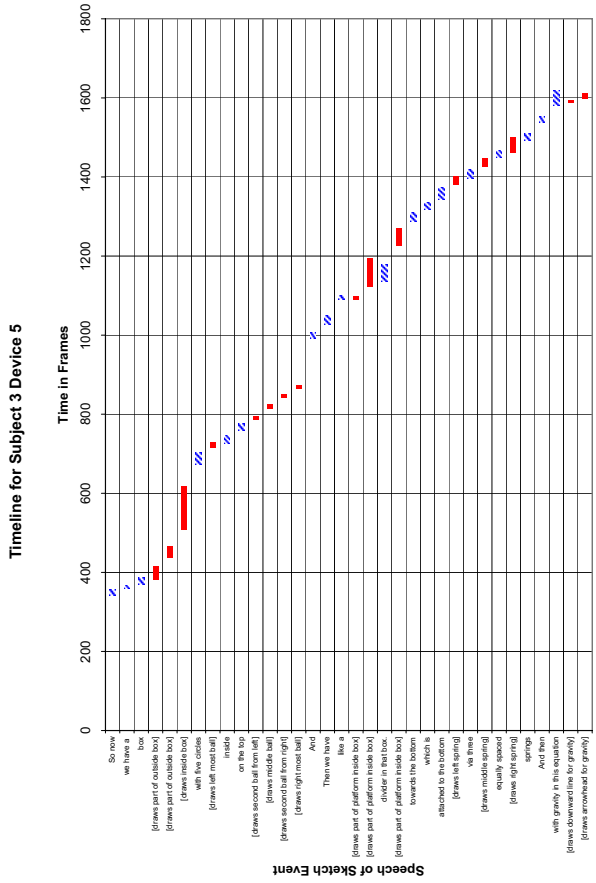


Figure B.43: The timeline graph for Subject 3 Device 5.

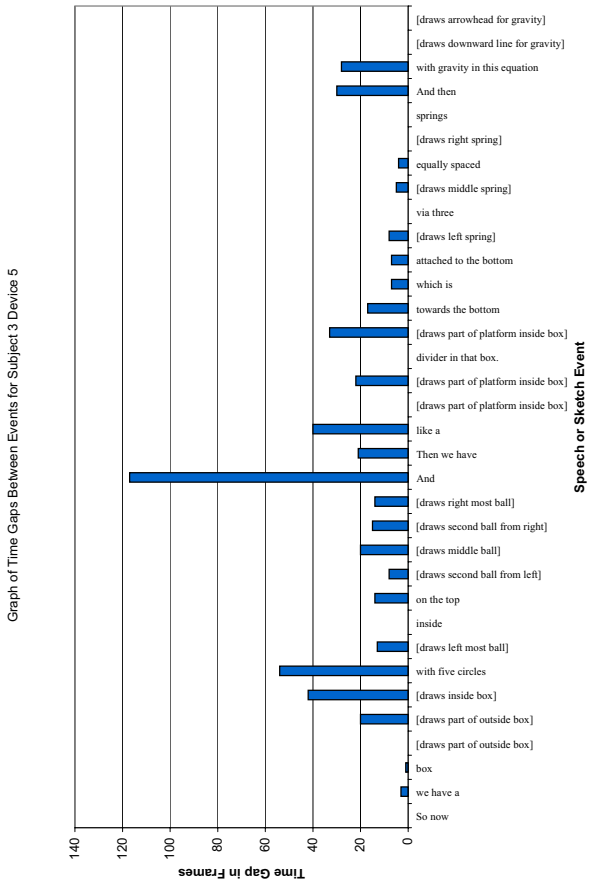


Figure B.44: The gap graph for Subject 3 Device 5.

B.5.3 Subject 4

Start		End		
sec	frame	sec	frame	Task Name
1	11	2	7	"Next, <ahhh>"
3	12	4	3	similar thing I guess
4	4	4	12	where the
4	10	6	7	[rectangle:r1:draw draws outside box]
6	5	7	0	<unitelligible> inside a box
6	25	9	24	[rectangle:r2:draw draws inside box]
11	1	11	20	And we have balls
11	20	11	24	up
11	22	12	0	[circle:c1:draw draws left most ball]
11	25	12	11	here again
13	10	13	18	[circle:c2:draw draws second ball from the left]
13	27	14	5	[circle:c3:draw draws middle ball]
14	14	14	23	[circle:c4:draw draws second ball from the right]
15	1	15	10	[circle:c5:draw draws right most ball]
16	17	17	13	"And,"
17	26	19	16	[rectangle:r3:draw draws platform inside box]
19	2	20	3	something to fall on
20	13	21	5	And that's <ahhh>
21	14	22	4	[spring:s1:draw draws left spring]
22	27	23	28	[spring:s1:erase erases left spring]
23	14	24	5	filled with springs.
24	12	25	18	[spring:s1:draw draws left spring]
26	17	27	19	[spring:s2:draw draws middle spring]
28	10	29	16	[spring:s3:draw draws right spring]
30	29	31	20	Gravity's down
31	2	31	16	[gravity:g:draw draws gravity arrow]
31	22	32	7	This is fixed
31	25	31	28	"[anchor:a1:draw draws anchor, top right to bottom left]"
32	1	32	3	"[anchor:a1:draw draws anchor, top left to bottom right]"
32	13	32	26	So
33	18	33	28	<aaah>
34	11	35	5	similar to the last one where
35	5	36	18	"when it starts, the balls will ball and push the"
38	6	38	18	thing down.

Table B.21: Transcript of Video for Subject 4 Device 5.

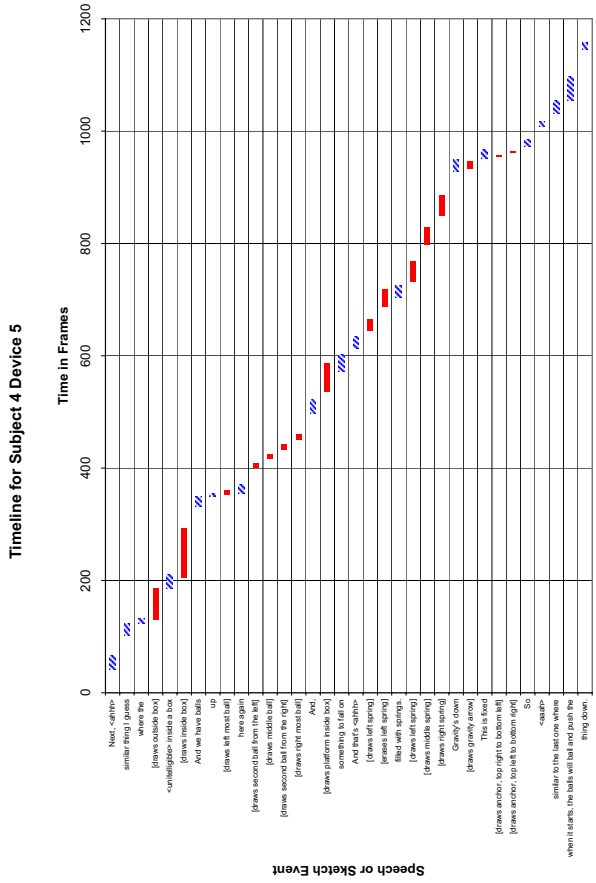


Figure B.45: The timeline graph for Subject 4 Device 5.

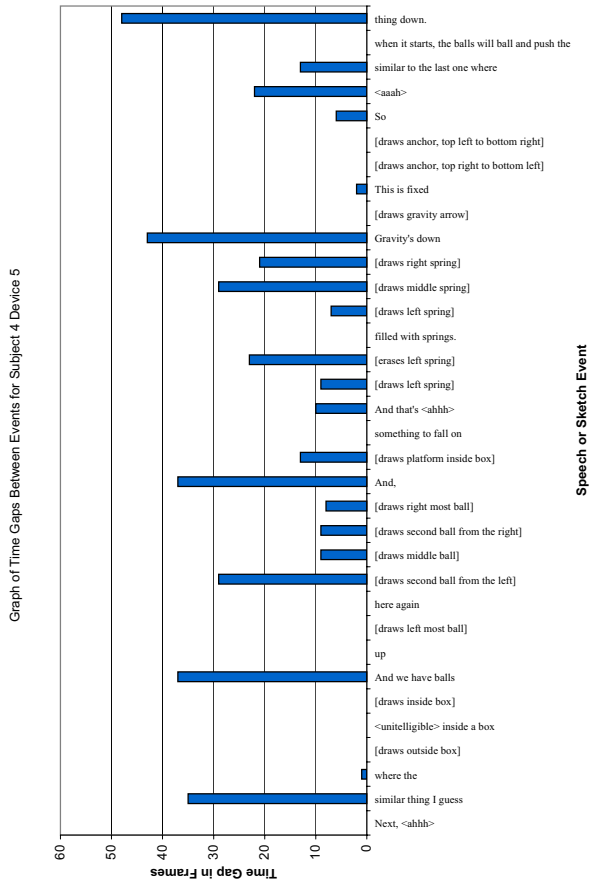


Figure B.46: The gap graph for Subject 4 Device 5.

B.5.4 Subject 5

Start		End		
sec	frame	sec	frame	Task Name
2	21	4	12	The fifth problem we have is
4	15	5	8	we have some kind of
5	11	6	9	we have a box
5	13	5	24	[rectangle:r1:draw draws left vertical line of inside box]
6	8	7	2	[rectangle:r1:draw draws top line of inside box]
7	2	7	20	[rectangle:r1:draw draws right vertical line of inside box]
7	20	8	17	[rectangle:r1:draw draws bottom line of inside box]
10	26	11	18	which again has a
11	18	12	13	has a platform
12	0	13	27	[rectangle:r2:draw draws platform inside box]
14	3	15	1	supported by
14	18	15	4	[spring:s1:draw draws left spring]
15	10	16	8	three springs
16	26	17	18	[spring:s2:draw draws middle spring]
18	5	18	20	[spring:s3:draw draws right spring]
19	17	19	28	<ummm>
20	1	20	11	of equal
20	11	21	7	of equal length
22	4	22	8	<ummm>
22	8	23	7	they are equally spaced
24	8	25	10	And we have five
25	19	25	28	[circle:c1:draw draws left most ball]
26	8	26	17	[circle:c2:draw draws second ball from the left]
27	0	27	11	[circle:c3:draw draws middle ball]
27	26	28	8	[circle:c4:draw draws second ball from the right]
28	17	28	27	[circle:c5:draw draws right most ball]
29	4	29	21	balls
30	5	31	0	that are going to
30	25	31	13	[gravity:g:draw draws gravity arrow]
31	5	31	19	fall
32	16	32	26	down
32	26	33	18	to this platform
34	10	35	4	<ahhh> <ahhh>
36	1	37	10	I don't know what's going to happen now
38	3	39	19	presumably they're either going to bounce around
39	19	40	21	or they are going to stay where they are
40	24	41	23	depending on their elasticity.
42	20	42	27	this
42	27	43	2	this
43	4	43	26	this box
43	28	44	8	of course is
44	9	44	15	is
44	15	45	6	is fixed.
45	20	46	5	"So,"
46	16	47	5	presumably it's
47	5	48	5	any collisions with it
48	10	49	2	<ahhh> <errr>
49	27	51	7	depending on the elasticity of the balls
51	10	52	20	it's going to completely stop them
52	23	53	21	or make them bounce around more.

Table B.22: Transcript of Video for Subject 5 Device 5.

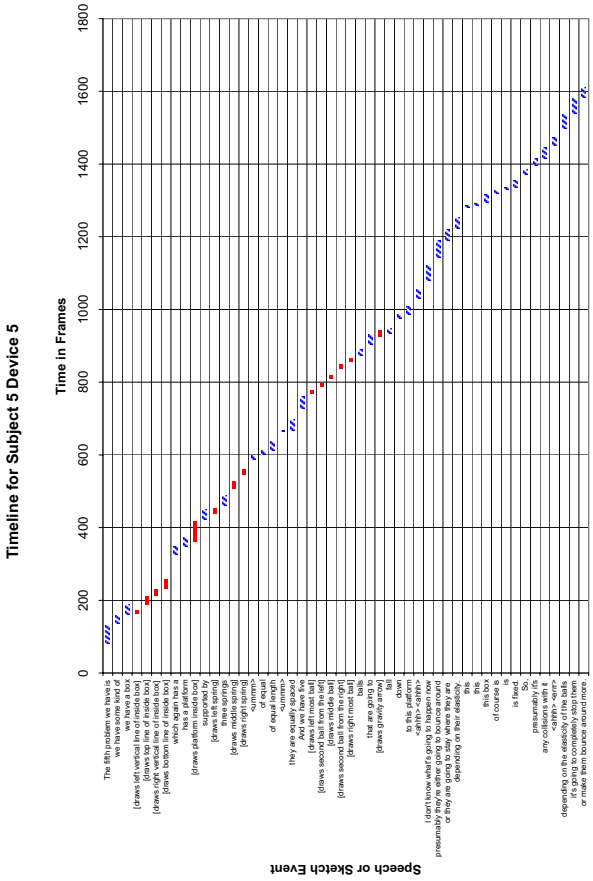


Figure B.47: The timeline graph for Subject 5 Device 5.

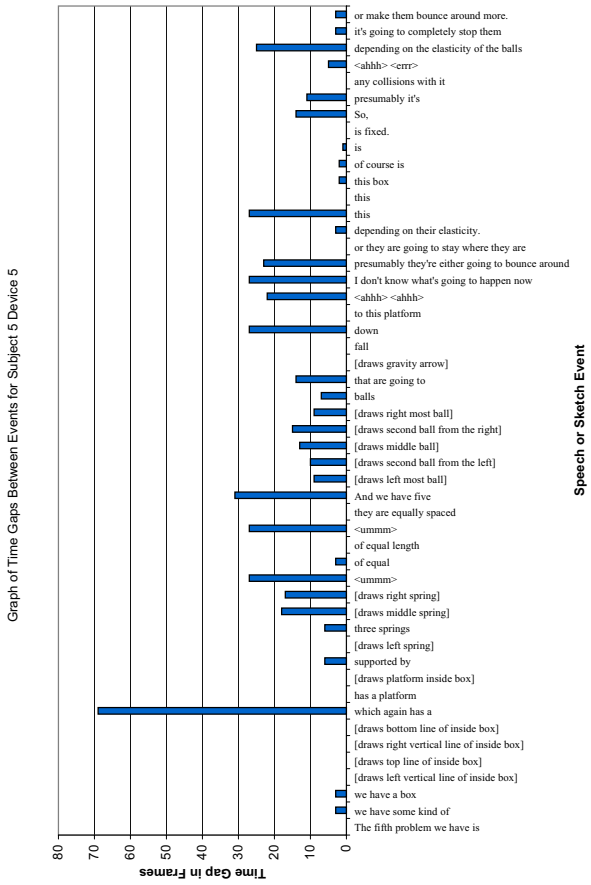


Figure B.48: The gap graph for Subject 5 Device 5.

B.6 Device 6

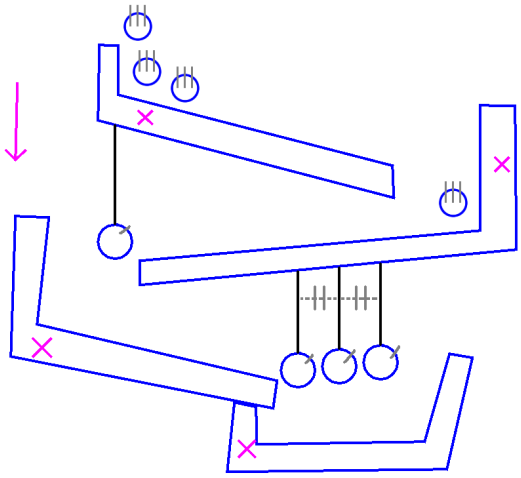


Figure B.49: This is Device 6.

B.6.1 Subject 2

Start		End		
sec	frame	sec	frame	Task Name
0	15	1	9	so
2	2	2	28	“In the sixth case,”
2	28	3	27	we have a bunch of balls
3	29	4	19	falling down
5	8	6	6	a couple of slopes
6	26	7	2	[polygon:p1:draw draws part of top ramp]
7	10	8	6	[polygon:p1:draw draws part of top ramp]
8	22	10	9	[polygon:p1:draw draws part of top ramp]
10	22	11	1	“[anchor:a1:draw draws top ramp anchor, top left to bottom right]”
11	5	11	9	“[anchor:a1:draw draws top ramp anchor, top right to bottom left]”
11	11	12	28	The slopes are fixed in position.
12	6	12	11	[polygon:p2:draw draws part of middle ramp]
12	18	15	6	[polygon:p2:draw draws part of middle ramp]
15	19	15	24	“[anchor:a2:draw draws middle ramp anchor, top left to bottom right]”
15	27	16	2	“[anchor:a2:draw draws middle ramp anchor, top right to bottom left]”
17	0	17	7	[polygon:p3:draw draws part of bottom ramp]
17	14	20	13	[polygon:p3:draw draws part of bottom ramp]
18	0	18	9	slope
20	26	21	1	“[anchor:a3:draw draws bottom ramp anchor, top left to bottom right]”
21	4	22	20	“And then at the end, they fall into”
21	5	21	10	“[anchor:a3:draw draws bottom ramp anchor, top right to bottom left]”
22	20	22	26	a bucket
23	0	23	16	of some sort.
23	20	26	12	[polygon:p4:draw draws bucket]
26	21	26	25	“[anchor:a4:draw draws bucket anchor, top left to bottom right]”
26	27	27	27	The bucket is also fixed.
27	0	27	4	“[anchor:a4:draw draws bucket anchor, top right to bottom left]”
28	10	29	7	And you start off with
29	20	30	3	[circle:c1:draw draws top ball on top ramp]
30	11	30	23	[circle:c2:draw draws middle ball on top ramp]
30	29	31	11	[circle:c3:draw draws bottom ball on top ramp]
31	3	32	0	three balls at the top.
32	20	34	2	And another ball here.
33	10	33	24	[circle:c4:draw draws ball on middle ramp]
34	20	35	2	and a ball
35	4	35	20	suspended
35	11	35	24	[pendulum:d1:draw draws string of top pendulum]
35	27	36	7	here
35	28	36	10	[pendulum:d1:draw draws ball of top pendulum]
37	3	37	18	<uhhh>

Table B.23: Transcript of Video for Subject 2 Device 6, Part 1.

Start		End		
sec	frame	sec	frame	Task Name
38	3	38	18	three balls
38	21	39	15	suspended over here
38	25	39	6	[pendulum:d2:draw draws string of left bottom pendulum]
39	7	39	22	[pendulum:d2:draw draws ball of left bottom pendulum]
40	3	40	14	[pendulum:d3:draw draws string of middle bottom pendulum]
40	18	41	3	[pendulum:d3:draw draws ball of middle bottom pendulum]
41	16	41	28	[pendulum:d4:draw draws string of right bottom pendulum]
42	2	42	16	[pendulum:d4:draw draws ball of right bottom pendulum]
43	22	44	1	<hmmm>
44	14	45	8	And we have gravity
45	9	45	18	[gravity:g:draw draws downward line for gravity]
45	14	46	6	pulling the balls down.
45	25	46	4	[gravity:g:draw draws arrowhead for gravity]
49	23	50	1	<hmmm>
50	6	51	25	I'm puzzled as to how to indicate that
54	10	55	4	equal size of
55	10	56	14	the suspended balls
57	3	58	4	and that it is not the same as
58	4	59	1	the falling balls

Table B.24: Transcript of Video for Subject 2 Device 6, Part 2.

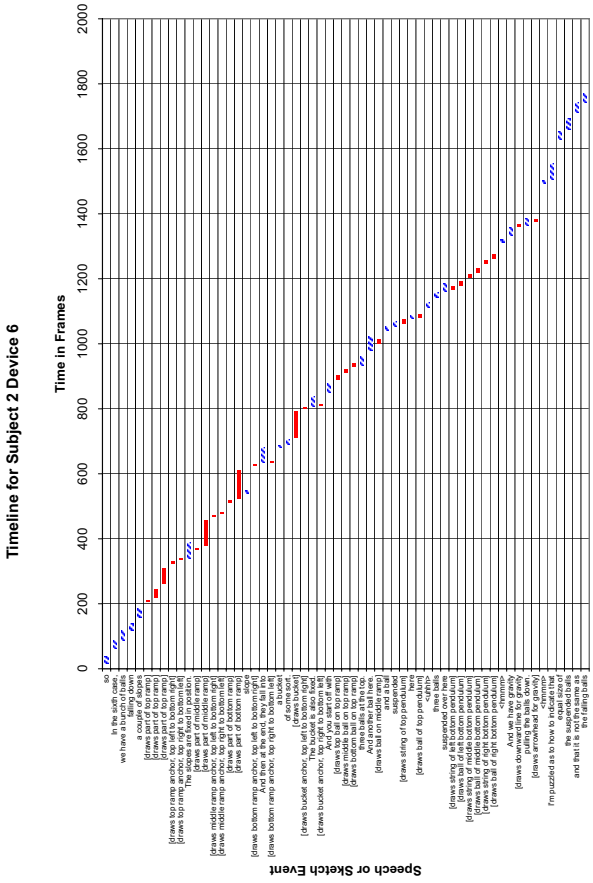


Figure B.50: The timeline graph for Subject 2 Device 6.

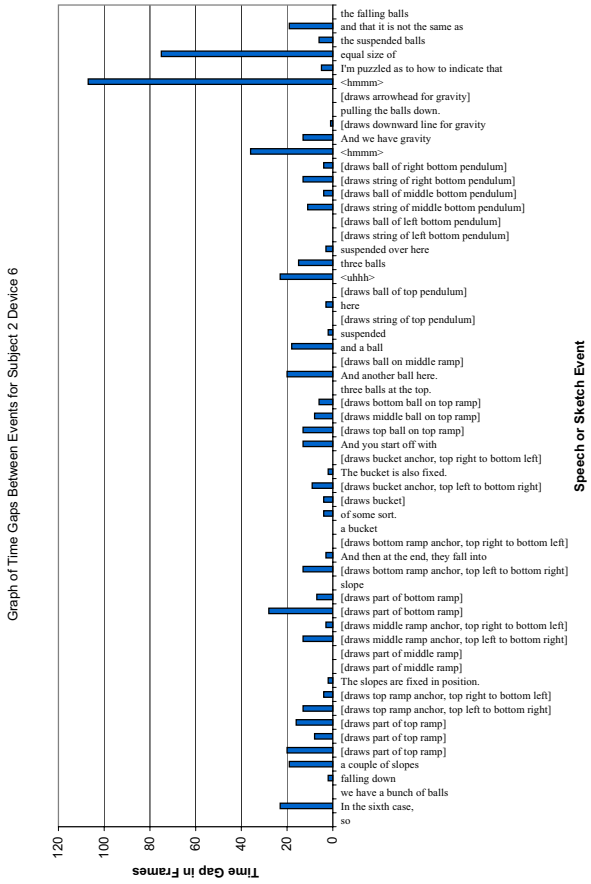


Figure B.51: The gap graph for Subject 2 Device 6.

B.6.2 Subject 3

Start		End		
sec	frame	sec	frame	Task Name
5	26	6	23	This is the last one.
6	23	7	21	<unintelligible>
9	3	10	9	We have this weird
11	24	13	17	[polygon:p1:draw draws part of top ramp]
15	5	17	3	like a sideways L-shaped ramp.
15	19	16	7	[polygon:p1:draw draws part of top ramp]
16	14	17	6	[polygon:p1:draw draws part of top ramp]
17	7	17	16	[polygon:p1:draw draws part of top ramp]
19	11	19	18	with
20	18	21	0	[circle:c1:draw draws top ball on top ramp]
22	2	22	26	three balls
22	29	23	11	[circle:c2:draw draws middle ball on top ramp]
25	12	25	29	[circle:c3:draw draws bottom ball on top ramp]
31	0	31	18	And then
33	9	33	27	there's another one
34	2	34	25	[polygon:p2:draw draws part of middle ramp]
35	23	36	26	[polygon:p2:draw draws part of middle ramp]
37	6	37	24	over here
38	9	38	19	[polygon:p2:draw draws part of middle ramp]
38	20	39	19	[polygon:p2:draw draws part of middle ramp]
40	7	41	0	[polygon:p2:draw draws part of middle ramp]
43	1	43	22	there's another ball
43	20	44	4	[circle:c4:draw draws ball on middle ramp]
44	12	44	21	there.
48	1	49	18	[pendulum:d1:draw draws string of top pendulum]
49	28	50	21	We also have a
50	23	52	11	ball suspended by a string.
51	17	52	6	[pendulum:d1:draw draws ball of top pendulum]
53	12	53	27	right there.
57	18	58	1	[polygon:p3:draw draws part of bottom ramp]
58	7	58	22	"Then,"
59	27	61	5	at the end of the second ramp
61	10	61	26	we have
63	8	63	29	another ramp
64	25	65	12	[polygon:p3:draw draws part of bottom ramp]
67	16	68	13	shaped like and L
69	2	69	17	[polygon:p3:draw draws part of bottom ramp]
70	0	70	17	[polygon:p3:draw draws part of bottom ramp]
70	25	71	11	[polygon:p3:draw draws part of bottom ramp]
72	2	73	1	[polygon:p3:draw draws part of bottom ramp]
76	5	76	28	[polygon:p3:erase erases part of bottom ramp]
77	25	78	22	[polygon:p3:draw draws part of bottom ramp]
79	4	79	15	[polygon:p3:draw draws part of bottom ramp]
80	24	81	10	"And then,"
81	24	82	17	we also have like a
83	16	84	6	[polygon:p4:draw draws part of bucket]
84	7	86	2	another divider that acts like a basin
84	18	85	3	[polygon:p4:draw draws part of bucket]
85	9	86	16	[polygon:p4:draw draws part of bucket]
86	21	87	2	a basket.
87	7	89	11	[polygon:p4:draw draws part of bucket]

Table B.25: Transcript of Video for Subject 3 Device 6, Part 1.

Start		End		
sec	frame	sec	frame	Task Name
91	5	92	13	[polygon:p4:erase erases part of bucket]
92	22	93	17	[polygon:p4:draw draws part of bucket]
95	11	95	24	[polygon:p4:erase erases part of bucket]
96	16	97	22	[polygon:p4:draw draws part of bucket]
98	9	99	1	[polygon:p4:erase erases part of bucket]
100	16	101	2	"And,"
101	10	101	21	we have
101	13	101	28	[pendulum:d2:draw draws string of left bottom pendulum]
101	24	103	21	three more balls suspended by ropes.
103	7	103	18	[pendulum:d2:draw draws string of left bottom pendulum]
103	20	104	5	[pendulum:d2:draw draws ball of left bottom pendulum]
104	28	105	22	[pendulum:d3:draw draws string of middle bottom pendulum]
105	24	106	15	[pendulum:d3:draw draws ball of middle bottom penulum]
107	12	108	7	[pendulum:d4:draw draws string of right bottom pendulum]
108	9	109	1	[pendulum:d4:draw draws ball of right bottom pendulum]
112	12	112	23	"And then,"
112	12	114	10	[gravity:g:draw draws gravity arrow]
113	9	114	7	the force of gravity
114	14	115	3	is here.
116	5	117	11	"All of the balls start falling,"
117	11	118	9	"roll down the ramps, 'n"
119	2	120	8	put the balls on the strings in motion.

Table B.26: Transcript of Video for Subject 3 Device 6, Part 2.

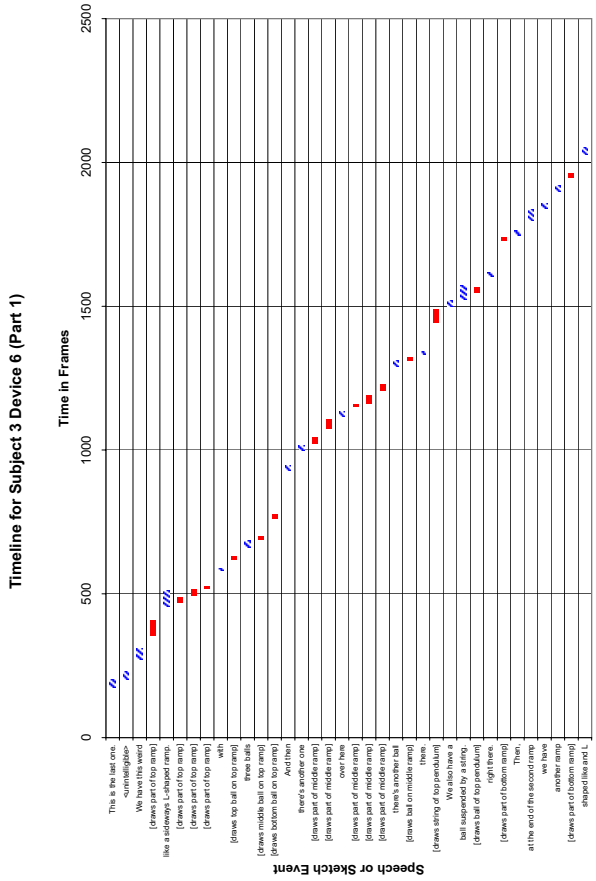


Figure B.52: The timeline graph for Subject 3 Device 6, Part 1.

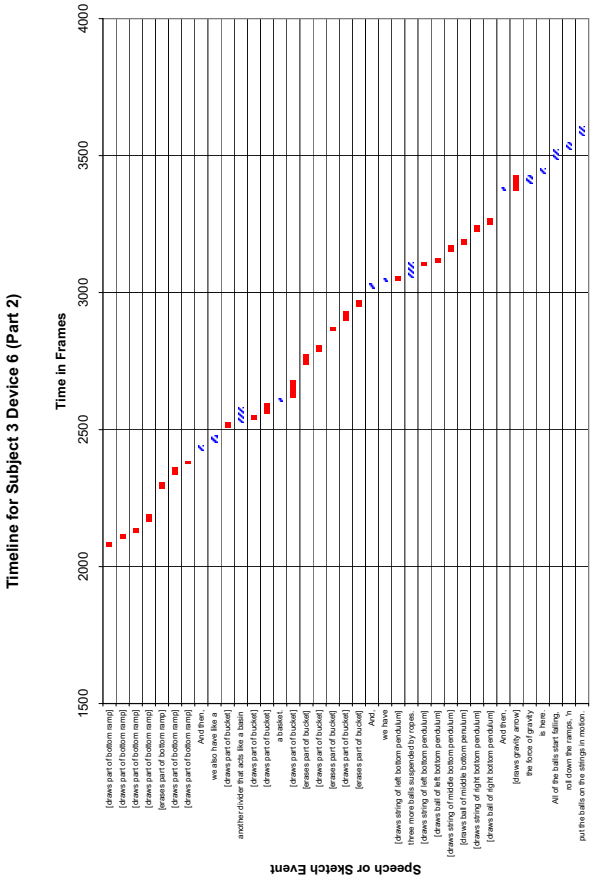


Figure B.53: The timeline graph for Subject 3 Device 6, Part 2.

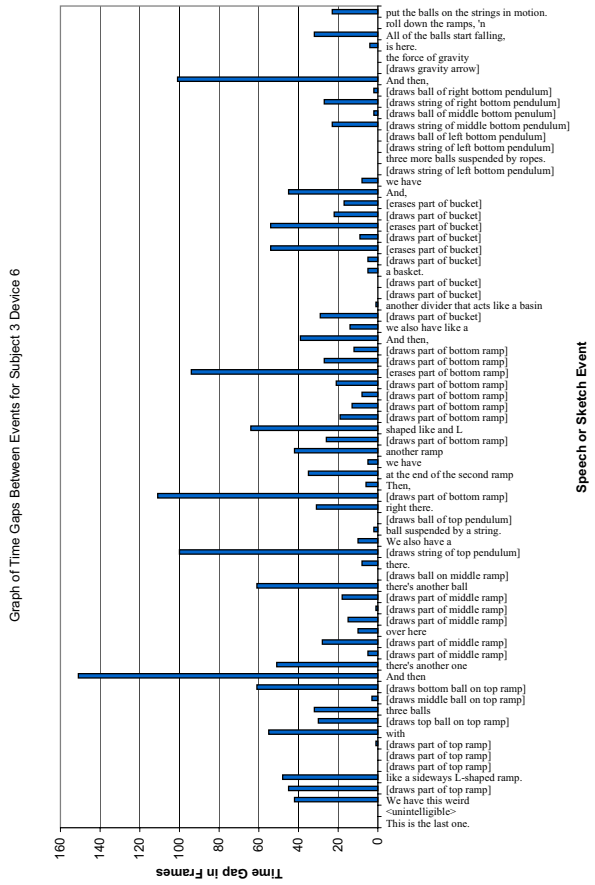


Figure B.54: The gap graph for Subject 3 Device 6.

***

B.6.3 Subject 4

Start		End		
sec	frame	sec	frame	Task Name
0	12	1	21	And lastly <aaah>
2	27	3	23	more complicated ramps.
4	4	4	14	[polygon:p1:draw draws part of top ramp]
6	0	7	26	[polygon:p1:draw draws part of top ramp]
8	26	10	8	[polygon:p1:erase erases part of top ramp]
9	8	9	14	It's not that steep.
10	27	11	9	[polygon:p1:draw draws part of top ramp]
11	20	12	18	[polygon:p1:erase erases part of top ramp]
13	6	14	1	[polygon:p1:draw draws part of top ramp]
15	4	15	15	[polygon:p1:draw draws part of top ramp]
17	3	20	12	[polygon:p2:draw draws middle ramp]
22	13	25	1	[polygon:p3:draw draws bottom ramp]
27	29	31	25	[polygon:p4:draw draws bucket]
33	0	33	11	[anchor:a1:draw anchors bucket]
33	6	34	1	These are all fixed.
33	23	34	2	[anchor:a2:draw anchors bottom ramp]
35	20	36	1	[anchor:a3:draw anchors middle ramp]
36	27	37	9	[anchor:a4:draw anchors top ramp]
39	13	39	24	[pendulum:d1:draw draws string of top pendulum]
40	29	41	11	[pendulum:d1:draw draws ball of top pendulum]
41	23	42	17	Ball hanging there
44	15	44	22	[pendulum:d2:draw draws string of left bottom pendulum]
45	0	45	6	[pendulum:d3:draw draws string of middle bottom pendulum]
45	15	45	21	[pendulum:d4:draw draws string of right bottom pendulum]
45	27	47	9	And <ahhh>
47	11	47	23	[pendulum:d2:draw draws ball of left bottom pendulum]
48	11	48	24	[pendulum:d3:draw draws ball of middle bottom pendulum]
49	4	49	16	[pendulum:d4:draw draws ball of right bottom pendulum]
49	25	49	29	[pendulum:d4:draw draws string of right bottom pendulum]
50	10	50	15	[pendulum:d3:draw draws string of middle bottom pendulum]
50	21	51	13	three hanging here.
50	26	51	0	[pendulum:d2:draw draws string of left bottom pendulum]
52	15	52	26	"So,"
53	20	54	9	[gravity:g:draw draws gravity arrow]
53	28	54	17	gravity down.
56	24	57	6	[circle:c1:draw draws ball on middle ramp]
57	2	57	27	Ball starts here.
59	0	59	10	[circle:c2:draw draws top ball on top ramp]
59	9	60	7	Three start up here.
59	20	59	28	[circle:c3:draw draws middle ball on top ramp]

Table B.27: Transcript of Video for Subject 4 Device 6, Part 1.

Start		End		
sec	frame	sec	frame	Task Name
60	11	60	21	[circle:c4:draw draws bottom ball on top ramp]
61	16	63	17	"So when it runs, these– these balls will start falling"
64	7	65	29	they'll all hit this once they go down
65	29	67	21	and hit these three and they'll all collect here

Table B.28: Transcript of Video for Subject 4 Device 6, Part 2.

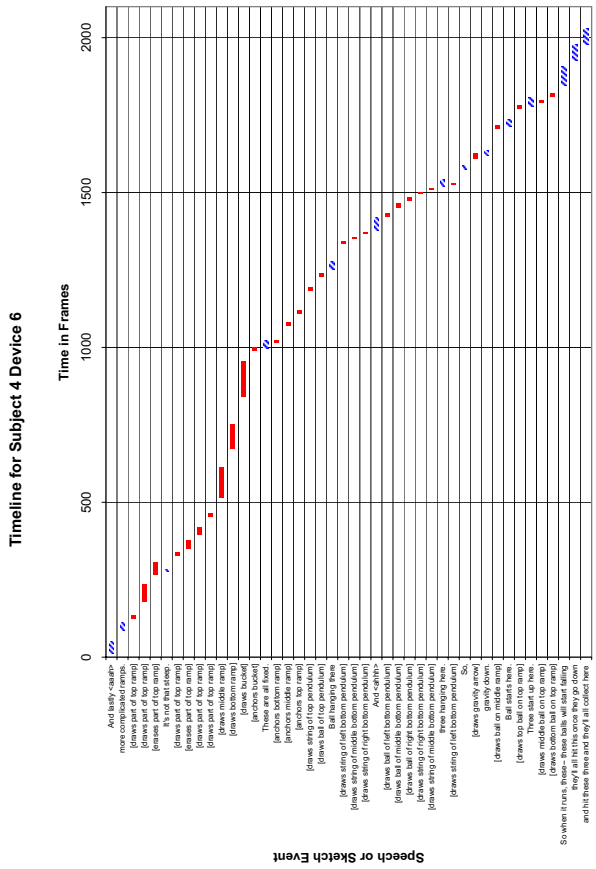


Figure B.55: The timeline graph for Subject 4 Device 6.

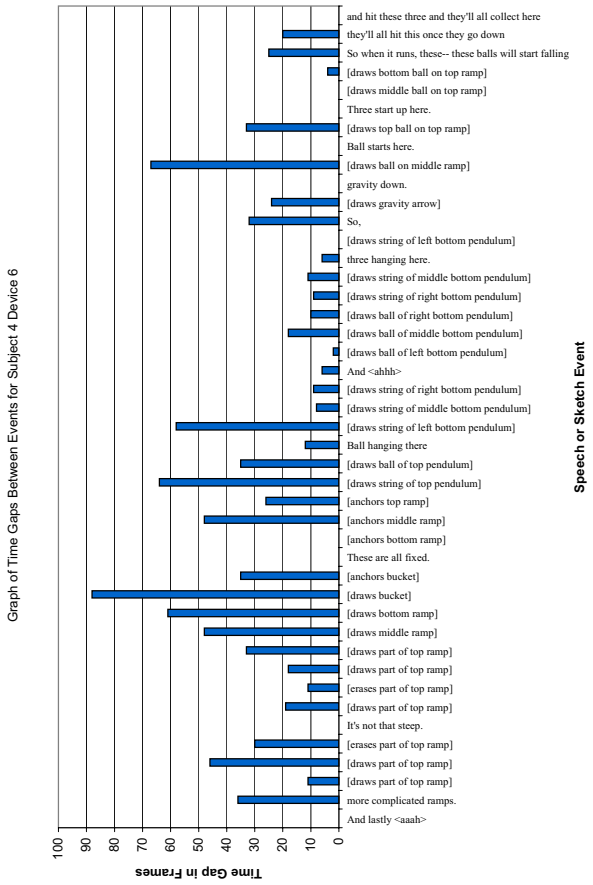


Figure B.56: The gap graph for Subject 4 Device 6.

Appendix C

Watch Rules

This Appendix contains all the rules that are used in the WATCH system.

C.1 functions.clp

```
;;; defining functions

;;; a function that computes if region 1 (r1start to r1end) is
;;; contained in region 2 (r2start to r2end). This does not
;;; test if region 2 is contained in region 1. It checks that
;;; r1start <= r1end and r2start <= r2end
(deffunction contains (?r1start ?r1end ?r2start ?r2end) (
  if (and (<= ?r1start ?r1end)
          (<= ?r2start ?r2end)
          (<  ?r1start ?r2end)
          (>= ?r1start ?r2start)
          (>  ?r1end   ?r2start)
          (<= ?r1end   ?r2end))
  then TRUE
  else FALSE)
)

;;; a function that computes if region 1 (r1start to r1end) has
;;; and end that is in region 2 (r2start to r2end). This does not
;;; test if region 2 has an end in region 1. It checks that
;;; r1start <= r1end and r2start <= r2end
(deffunction simple-overlap (?r1start ?r1end ?r2start ?r2end) (
  if (and (<= ?r1end   ?r2end)
          (>  ?r1end   ?r2start)
          (<  ?r1start ?r2start)
          (<= ?r1start ?r1end)
          (<= ?r2start ?r2end))
  then TRUE
  else FALSE)
)

;;; a function that computes if region 1 (r1start to r1end) touches
;;; region 2 (r2start to r2end) - meaning that the start of one region
```

```
;;; is the end of the other. It checks that r1start <= r1end and
;;; r2start <= r2end
(deffunction touches (?r1start ?r1end ?r2start ?r2end) (
  if (and (<= ?r1start ?r1end)
          (<= ?r2start ?r2end)
          (or (= ?r1end ?r2start)
              (= ?r2end ?r1start)))
  then TRUE
  else FALSE)
)

;;; a function that computes if two regions overlap. This function
;;; does not consider 2 regions that just "touch" to be overlapping.
(deffunction overlap (?r1start ?r1end ?r2start ?r2end) (
  if (or (contains ?r1start ?r1end ?r2start ?r2end)
         (contains ?r2start ?r2end ?r1start ?r1end)
         (simple-overlap ?r1start ?r1end ?r2start ?r2end)
         (simple-overlap ?r2start ?r2end ?r1start ?r1end))
  then TRUE
  else FALSE)
)

;;; a function that computes if two regions overlap including just
;;; "touching"
(deffunction anyOverlap (?r1start ?r1end ?r2start ?r2end) (
  if (or (overlap ?r1start ?r1end ?r2start ?r2end)
         (touches ?r1start ?r1end ?r2start ?r2end))
  then TRUE
  else FALSE)
)

;;; this function returns the first multifield from the input multifield
;;; the first item in the multifield indicates the length of the first
;; multifield.
(deffunction get-first-multifield ($?m)
  (bind ?length-first (nth$ 1 $?m))
  ;(bind $?first (subseq$ $?m 2 (+ 1 ?length-first)))
  (return (subseq$ $?m 2 (+ 1 ?length-first)))
  ;(return $?first)
)

;;; this function returns the second multifield from the input multifield
;;; the first item in the multifield indicates the length of the first
;; multifield.
(deffunction get-second-multifield ($?m)
  (bind ?length-first (nth$ 1 $?m))
  (bind $?second (subseq$ $?m (+ 2 ?length-first) (length$ $?m)))
  (return $?second)
)

;;; a function that computes if 2 multifields contain consequtive ids
(deffunction consecutive-ids ($?i)
  (bind $?i1 (get-first-multifield $?i))
  (bind $?i2 (get-second-multifield $?i))
  (foreach ?x $?i1
    ; for each id in i1 see if it matches an id one larger or one
    ; smaller in i2. Not that not eq FALSE is not the same as TRUE
    ; because member returns an integer or FALSE

    (if (or (not (eq FALSE (member$ (+ ?x 1) $?i2)))
            (not (eq FALSE (member$ (- ?x 1) $?i2))))
     then (return TRUE)))
```

```
  (return FALSE)
)


;;; a function that determines if a sketch unit is the first one since
;;; the most recent break. If it is returns true otherwise false
(deffunction first-sketch-unit-since-break (?start-of-sketch-unit $?i)
  (bind $?breaks (get-first-multifield $?i))
  (bind $?sketch-units (get-second-multifield $?i))

  (bind ?closest-break 0)

  ; for each break see if it is the latest break that is still before
  ; the start-of-sketch-unit
  (bind ?list-length (length$ $?breaks))
  (bind ?n 1)
  (while (< ?n ?list-length) do
    (bind ?num (nth$ ?n $?breaks))
    (bind ?t (nth$ (+ ?n 1) $?breaks))
    (if (and (<= ?num ?start-of-sketch-unit)
             (> ?num ?closest-break))
     then (bind ?closest-break ?num))
    (bind ?n (+ 2 ?n))
  )

  ; see if there is a sketch unit between closest-break and start-of-sketch-unit
  ; if there is return false
  (foreach ?y $?sketch-units
    (if (and (< ?y ?start-of-sketch-unit)
             (>= ?y ?closest-break))
     then
       (return FALSE)
     )
    )

  ; otherwise no sketch-units since the last break
  (return TRUE)
)


;; returns the next sketch-unit after current or 0 if there isn't one
(deffunction next-sketch-unit (?current $?sketch-units)
  (bind ?next 0)
  (foreach ?x $?sketch-units
    ; for each sketch unit see if is is after the current sketch-unit
    ; but before the current next sketch unit
    (if (> ?x ?current)
    then
      (if (or (= ?next 0) (< ?x ?next))
      then (bind ?next ?x))
    )
  )

  (return ?next)
)
```

C.2 template.clp

```
;;; defining templates
(deftemplate sketch
  "This template is for items that were sketched"
```

```
  (slot text)       ; Text from Excel
  (slot objType)    ; i.e. Rectangle
  (slot objID)      ; an identifying string
  (slot startTime)  ; start time
  (slot endTime)    ; end time
  (slot actionType) ; drawing or erasing
  (slot idNum)      ; # indicating sketch order
)

(deftemplate speech
  "This template is for items that were spoken"
  (slot text)       ; Text from Excel
  (slot startTime)  ; start time
  (slot endTime)    ; end time
  (slot idNum)      ; # indicating speech order
)

(deftemplate sketch-group
  "This template is for a group of sketched components that are
       part of the same object"
  (slot startTime)   ; start time
  (slot endTime)     ; end time
  (slot objType)     ; i.e. Rectangle
  (slot objID)       ; an identifying string
  (slot groupID)     ; This is to easily identify a group
  (multislot idNums) ; idNum of component sketches
)
```

C.3 gap_rules.clp

```
;;; Define what a long-gap is
(bind ?*long-gap* 25)

;;; Define what a and-gap is
(bind ?*and-gap* 7)

;;; defining rules

;;; this rule says you can't have a gap within a sketch event
(defrule sketch-time
  (sketch (startTime ?x) (endTime ?y))
  =>
  (assert (no-gap ?x ?y))
)

;;; this rule says you can't have a gap within a speech event
(defrule speech-time
  (speech (startTime ?x) (endTime ?y))
  =>
  (assert (no-gap ?x ?y))
)

;;; This rule says you can't have a gap where speech and sketching events overlap
(defrule sketch-speech-overlap
  (sketch (startTime ?a) (endTime ?b))
  (speech (startTime ?c) (endTime ?d))
  (test (eq (overlap ?a ?b ?c ?d) TRUE))
  =>
  (assert (no-gap (min ?a ?c) (max ?b ?d)))
)
```

```
;;; This rule combines existing overlapping no-gap facts into one fact
(defrule combine-no-gap
  ?g1 <- (no-gap ?a ?b)
  ?g2 <- (no-gap ?c ?d)
  (test (eq (anyOverlap ?a ?b ?c ?d) TRUE))
  (test (neq ?g1 ?g2))
  =>
  (retract ?g1)
  (retract ?g2)
  (assert (no-gap (min ?a ?c) (max ?b ?d)))
)

;;; This rule adds a no-gap between pieces of the same sketched item
(defrule same-object
  ?o1 <- (sketch (objType ?a) (objID ?b) (idNum ?c) (startTime ?s1)
      (endTime ?e1))
  ?o2 <- (sketch (objType ?a) (objID ?b) (idNum ?d) (startTime ?s2)
      (endTime ?e2))
  (test (neq ?o1 ?o2))
  (test (< ?c ?d)) ;;insures rule runs once, not twice
  =>
  (assert (no-gap (min ?s1 ?s2) (max ?e1 ?e2))))

;;; Rules for adding gaps

;;; need to indicate start of timeline so add a no-gap there
(assert (no-gap 0 0))

;;; this rule adds lots of gaps
(defrule gap
  (no-gap ?a ?b)
  (no-gap ?c&:(> ?c ?a) ?d)
  (not (no-gap ?e&:(and (> ?e ?a) (< ?e ?c)) ?f))
  =>
  (assert (gap ?b ?c))
)

;;; occasionally bad gaps maybe added because of no-gaps that are added.
;;; This rule fixes this problem by taking out incorrect gaps
(defrule remove-gap
  ?r <- (gap ?a ?b)
  (no-gap ?c ?d)
  (test (eq (overlap ?a ?b ?c ?d) TRUE))
  =>
  (retract ?r)
)

;;; This rule asserts that there is a long-pause when
;;; a gap is larger than ?*long-gap*
(defrule long-pause
  (gap ?start ?end)
  (test (< ?*long-gap* (- ?end ?start)))
  =>
  (assert (long-pause ?start ?end))
)

;;; Sometimes a long-pause may get asserted but later a supporting gap
;;; may be removed. Fix this by checking the long-pause to make sure
;;; that if there is no gap there it is removed.
(defrule remove-long-pause
  ?r <- (long-pause ?a ?b)
  (not (gap ?a ?b))
```

```
  =>
  (retract ?r)
)

;;; We shouldn't really look at the long pauses that occur prior to
;;; erasures. So take those out.
(defrule remove-erase-pause
  ?r <- (long-pause ?start ?end)
  (sketch (startTime ?end) (actionType erase))
  =>
  (retract ?r)
)
```

C.4 text_rules.clp

```
;;; defining global variables

;;; The list of words to look for
(bind $?*key-words* (create$ "then" "so" "we have" "there is" "there
   are" "next" "we've" "possibly" "ok" "it's" "it is" "with" "there's"
   "which" "plus" "i'll" "i will" "let's" "let us" "that's" "that is"
   "this is" "these are"))

;;; "and" words which right now is just "and"
(bind $?*and-words* (create$ "and"))

;;; "the" words which right now is just "the"
(bind $?*the-words* (create$ "the"))

;;; mumbled words
(bind $?*mumbled-words* (create$ "<ahhh>" "<ummm>" "<uhhh>" "<ahhh>" "<hmmm>"))

;;; This rule adds statements for speech text containing and-words
(defrule speech-and-words
  (speech (startTime ?a) (endTime ?e) (text ?b))
  (test (> (num-contains-strings ?b $?*and-words*) 0))
  =>
  (assert (and-words-at ?a ?e num (num-contains-strings ?b $?*and-words*)))
)

;; This rule adds statements for speech text containing key-words
(defrule speech-key-words
  (speech (startTime ?a) (endTime ?e) (text ?b))
  (test (> (num-contains-strings ?b $?*key-words* ) 0))
  =>
  (assert (key-words-at ?a ?e num (num-contains-strings ?b $?*key-words*)))
)

;;; This rule adds statements for speech text containing a the
;;; at the beginning of the speech
(defrule speech-the-words
  (speech (startTime ?a) (endTime ?e) (text ?b))
  (test (> (start-of-string ?b $?*the-words*) 0))
  =>
  (assert (the-words-at ?a ?e))
)

;;; This rule adds statements for speech text containg mumbled words
(defrule mumbled-words
```

```
  (speech (startTime ?start) (endTime ?end) (text ?t))
  (test (> (end-of-string ?t $?*mumbled-words*) 0))
  =>
  (assert (mumbled-words-at-end ?start ?end))
)

;;; If and is surrounded by gaps on both sides there is likely a break
(defrule and-with-gaps
  (and-words-at ?gap1-end ?e num ?num)
  (speech (startTime ?gap1-end) (endTime ?gap2-start))
  (gap ?gap1-start ?gap1-end)
  (gap ?gap2-start ?gap2-end)
  (test (< ?*and-gap* (- ?gap1-end ?gap1-start)))
  (test (< ?*and-gap* (- ?gap2-end ?gap2-start)))
  =>
  (assert (and-gaps ?gap1-end ?e))
)

;;; Rule for linking mumbled words to the next speech utterance
(defrule link-mumbled-words
  (mumbled-words-at-end ?start-1 ?e)
  (speech (startTime ?start-1) (endTime ?end-1) (idNum ?id))
  (speech (startTime ?start-2) (endTime ?end-2) (idNum =(+ 1 ?id)))
  =>
  (assert (linked-by-mumbles ?start-1 ?end-2))
)

;;; Rule for indicating the number of words in one speech
;;; utterance that also occur in the following speech utterance
(defrule repeat-words
  (speech (text ?t1) (startTime ?s) (idNum ?i))
  (speech (text ?t2) (endTime ?e)   (idNum =(+ ?i 1)))

  ; test the number of overlap in words from the first speech to the
  ; second and fire the rule if there is an overlap
  (test (> (num-words-overlap ?t1 ?t2) 0))
  =>
  (assert (repeat-words ?s ?e num (num-words-overlap ?t1 ?t2) ))
)
```

C.5 group_rules.clp

```
;;; We need to have a counter that keeps track of group IDs to uniquely
;;; identify the groups
(bind ?*group_id* 1) ;initialize the group id counter

;;; This rule creates a group for a sketched item that does not already
;;; have a group
(defrule create-sketch-group
  (sketch (startTime ?s) (endTime ?e) (objType ?t) (objID ?o) (idNum ?i))
  ; this first line finds a sketch event
  (not (sketch-group (objType ?t) (objID ?o)))
  ; this line tests to make sure there is not an existing group that
  ; this event should belong in.
  =>
  ; the following line creates the new group
  (assert (sketch-group (groupID ?*group_id*) (startTime ?s) (endTime ?e)
           (objType ?t) (objID ?o) (idNums (create$ ?i))))
  ;the following line increments the groupID counter
  (bind ?*group_id* (+ ?*group_id* 1))
```

```
)

;;; This rule adds additional parts of a sketched item to its group
(defrule add-to-sketch-group
  ; the following line finds a sketch event
  (sketch (startTime ?s) (endTime ?e) (objType ?t) (objID ?o) (idNum ?i))
  ; this line finds a group that matches the sketch event
  ; note that a match has the same objType and objID
  ?g <- (sketch-group (startTime ?g-s) (endTime ?g-e) (objType ?t) (objID ?o)
      (idNums $?n))
  ; the next line tests to see if the sketch event is already a member
  ; of this group so that it can't be added twice
  (test (eq FALSE (member$ ?i $?n)))
  =>
  ; This line modifies the existing group to update the start and end
  ; times as well as add the new idNum of the new sketch event to the
  ; idNums multislot
  (modify ?g (startTime (min ?s ?g-s)) (endTime (max ?e ?g-e))
      (idNums (union$ $?n (create$ ?i))))
)

;; rule for linking similar shapes drawn in sequence
(defrule time-and-shape-similar-objects
  ; Find two sketch-group of the same type
  (sketch-group (groupID ?g1) (startTime ?s1) (endTime ?e1) (objType ?t)
      (idNums $?i1))
  (sketch-group (groupID ?g2) (startTime ?s2) (endTime ?e2) (objType ?t)
      (idNums $?i2))

  ; check to make sure that the two sketch groups are not the same group
  ; by using less than, we also ensure that this rule will only run once
  ; not twice
  (test (< ?g1 ?g2))

  ; see if the two sketch-groups contain consecutive ids
  ; if they do this means that the sketches occured near each other
  (test (eq TRUE (consecutive-ids (create$ (length$ $?i1) $?i1 $?i2))))

  =>
  (assert (time-and-shape-similar (min ?s1 ?s2) (max ?e1 ?e2)))
)

;; rule for taking out out-of-date time-and-shape-similar shapes
;; based on start times
(defrule remove-time-and-shape-similar-objects-1
  ; find the time-and-shape assertion
  ?tss <- (time-and-shape-similar ?start ?end)
  ; if there is not a sketch-group that has the same start time
  (not (sketch-group (startTime ?start)))
  =>
  ;; then retract the assertion
  (retract ?tss)
)

;; rule for taking out out-of-date time-and-shape-similar shapes
;; based on end times
(defrule remove-time-and-shape-similar-objects-2
  ; find the time-and-shape assertion
  ?tss <- (time-and-shape-similar ?start ?end)
  ; if there is not a sketch-group that has the same end time
  (not (sketch-group (endTime ?end)))
  =>
```

```
  ;; then retract the assertion
  (retract ?tss)
)

;;; we need to set up a tolerance on the overlap of
;;; sketching groups and speech. This is a leeway
;;; such that they can overlap a little bit and still
;;; be considered a strong-group-link
(bind ?*strong-group-link-tolerance* 5)

(defrule strong-group-link
  ; this rule adds an assertion if the time-and-shape-similar-objects
  ; occur with no overlapping speech (within the
  ;?*strong-group-tolerance* limit)
  (time-and-shape-similar ?start ?end)

  ; for a given time-and-shape-similar assertion there should not be
  ; a speech event that overlaps with it
  (not (speech (startTime ?s)
               (endTime ?e&:(overlap ?start
                                     ?end
                                     (+ ?s ?*strong-group-link-tolerance*)
                                     (- ?e ?*strong-group-link-tolerance*) ))
  ))
  =>
  (assert (strong-group-link ?start ?end))
)
```

C.6 sketch_unit_rules.clp

```
(bind $?*sketch-units* (create$))

;; This time should be the START time of the sketch unit!
(deffunction insert-sketch-unit (?time)
  (bind $?*sketch-units* (insert$ $?*sketch-units*
                                  (+ 1 (length$ $?*sketch-units*)) ?time))
  (bind ?next (next-sketch-unit ?time $?*sketch-units*))
  (if
    (eq FALSE (first-sketch-unit-since-break ?next
                               (create$ (length$ $?*breaks*)
                                        $?*breaks* $?*sketch-units*)))
  then
    (assert (break-if-gap ?next))
  )
)

(defrule add-break-if-gap
  ?bg <- (break-if-gap ?time)
  (gap ?st ?time)
  =>
  (retract ?bg)
  (insert-break sketchUnit ?time)
  (assert (break sketchUnit ?time))
)

;; get rid of the bad break-if-gap assertions
(defrule remove-break-if-gap
  ?bg <- (break-if-gap ?time)
  (no-gap ?s ?e&:(and (<= ?time ?e) (>= ?time ?s)))
  =>
  (retract ?bg)
```

```
)

;; This time should be the START time of the sketch unit!
(deffunction delete-sketch-unit (?time)
  (bind $?*sketch-units* (delete$ $?*sketch-units*
          (member$ ?time $?*sketch-units*) (member$ ?time $?*sketch-units*)))
)

;;; a sketch unit is an overlap between a sketch group and a no-gap
(defrule create-sketch-unit
  (sketch-group (groupID ?g1) (startTime ?s1) (endTime ?e1) (objType ?t)
        (idNums $?i1))
  (no-gap ?gs ?ge)
  (test (eq (overlap ?s1 ?e1 ?gs ?ge) TRUE))
  =>
  (insert-sketch-unit (min ?s1 ?gs))
  (assert (sketch-unit (min ?s1 ?gs) (max ?e1 ?ge)))
)

;; remove erroneous sketch units
(defrule remove-sketch-unit-1
  ?u <- (sketch-unit ?start ?end)
  (not (no-gap ?gs ?ge&:(contains ?gs ?ge ?start ?end)))
  =>
  (delete-sketch-unit ?start)
  (retract ?u)
)

(defrule remove-sketch-unit-2
  ?u <- (sketch-unit ?start ?end)
  (not (sketch-group (groupID ?g1)
                     (startTime ?s1)
                     (endTime ?e1&:(contains ?s1 ?e1 ?start ?end))
                     (objType ?t)
                     (idNums $?i1)))
  =>
  (delete-sketch-unit ?start)
  (retract ?u)
)
```

C.7 break_rules.clp

```
;;; Define what a sig-gap is
(bind ?*sig-gap* 5)

(bind $?*breaks* (create$))

(deffunction contains-break (?type ?time)
  ;; this function returns true if the type and time of the break are in the
  ;; break list and false if they are not

  (bind ?list-length (length$ $?*breaks*))
  (bind ?n 1)
  (while (< ?n ?list-length) do
    (bind ?num (nth$ ?n $?*breaks*))
    (bind ?t (nth$ (+ ?n 1) $?*breaks*))

    (if (numberp ?num)
     then
       (if (and (eq ?num ?time)
                (eq ?t ?type))
```

```
       then
         (return TRUE)
       ))
    (bind ?n (+ 2 ?n))
  )
  (return FALSE)
)

(deffunction remove-break-from-list (?time ?type $?brks)
  (bind $?ret-brks (create$ $?brks))
  (bind ?list-length (length$ $?ret-brks))
  (bind ?n 1)
  (while (< ?n ?list-length) do
    (bind ?num (nth$ ?n $?ret-brks))
    (bind ?t (nth$ (+ ?n 1) $?ret-brks))

    (if (numberp ?num)
     then
       (if (and (eq ?num ?time)
                (eq ?t ?type))
        then
          (bind $?ret-brks (delete$ $?ret-brks ?n (+ 1 ?n)))
          ;; now 2 fewer elements in the list
          (bind ?list-length (- ?list-length 2))
        else
          ;; only increment the counter if we haven't just deleted
          (bind ?n (+ 2 ?n))
       ))
  )
  (return $?ret-brks)
)

(deffunction insert-break (?type ?time)
  (if (eq FALSE (contains-break ?type ?time))
   then
     (bind $?*breaks* (insert$ $?*breaks* (+ 1 (length$ $?*breaks*)) ?time))
     (bind $?*breaks* (insert$ $?*breaks* (+ 1 (length$ $?*breaks*)) ?type))
  )
)

(deffunction delete-break (?type ?time)
  (bind ?list-length (length$ $?*breaks*))
  (bind ?n 1)
  (while (< ?n ?list-length) do
    (bind ?num (nth$ ?n $?*breaks*))
    (bind ?t (nth$ (+ ?n 1) $?*breaks*))
    (if (numberp ?num)
     then
       (if (and (eq ?num ?time)
                (eq ?t ?type))
        then
          (bind $?*breaks* (delete$ $?*breaks* ?n (+ 1 ?n)))
          ;; now 2 fewer elements in the list
          (bind ?list-length (- ?list-length 2))
        else
          ;; only increment the counter if we haven't just deleted
          (bind ?n (+ 2 ?n))
       ))
  )

  ; check to see if taking out a break means adding a new break for sketch-unit
  (bind ?next (next-sketch-unit ?time $?*sketch-units*))
```

```
  (if
    (eq FALSE (first-sketch-unit-since-break ?next
                                             (create$ (length$ $?*breaks*)
                                                  $?*breaks* $?*sketch-units*)))
  then
    (assert (break-if-gap ?next))
  )
)

(defrule long-pause-break
  (long-pause ?start ?end)
  =>
  (insert-break longPause ?end)
  (assert (break longPause ?end))
)

(defrule retract-long-pause-break
  ?r <- (break longPause ?end)
  (not (long-pause ?start ?end))
  =>
  (delete-break longPause ?end)
  (retract ?r)
)

(defrule sketch-unit-break
  (gap ?gs ?start)
  (sketch-unit ?start ?end)
  (test (eq FALSE (first-sketch-unit-since-break ?start
                                                 (create$ (length$ $?*breaks*)
                                                      $?*breaks* $?*sketch-units*)
        )))
  =>
  (insert-break sketchUnit ?start)
  (assert (break sketchUnit ?start))
)

(defrule retract-sketch-unit-break-1
  ?b <- (break sketchUnit ?start)
  (not (sketch-unit ?start ?end))
  =>
  (delete-break sketchUnit ?start)
  (retract ?b)
)

(defrule retract-sketch-unit-break-2
  ?b <- (break sketchUnit ?start)
  (not (gap ?gs ?start))
  =>
  (delete-break sketchUnit ?start)
  (retract ?b)
)

(defrule retract-sketch-unit-break-3
  ?b <- (break sketchUnit ?start)
  (gap ?gs ?start)
  (test (eq TRUE (first-sketch-unit-since-break ?start
         (create$ (length$ (remove-break-from-list ?start sketchUnit $?*breaks*))
                          (remove-break-from-list ?start sketchUnit $?*breaks*)
                          $?*sketch-units*)
        )))
  =>
  (delete-break sketchUnit ?start)
```

```
  (retract ?b)
)

(deffunction after (?i ?a)
  ;; if there is any break between ?i and ?a return false
  ;; otherwise return true
  (bind ?list-length (length$ $?*breaks*))
  (bind ?n 1)
  (while (< ?n ?list-length) do
    (bind ?num (nth$ ?n $?*breaks*))

    (if (numberp ?num)
     then
       (if (and (> ?num ?i) (< ?num ?a))
        then
          (return FALSE)
        ))
    (bind ?n (+ 2 ?n))
  )

  ;; didn't find any inbetween
  (return TRUE)
)

(defrule retract-sketch-unit-break-4
  ?b <- (break sketchUnit ?before)
  ?a <- (break ?typeB ?after)
  (test (< ?before ?after))
  (test (eq TRUE (after ?before ?after)))
  (not (speech (startTime ?s) (endTime ?e&:(or (and (< ?e ?after) (> ?e ?before))
                                               (and (< ?s ?after) (> ?s ?before))
       ))))
  ; need to check to make sure that the breaks do NOT overlap
  ; with a strong-group-link
  (not (strong-group-link ?s-start
                          ?s-end&:(or
                                   (and (> ?before ?s-start) (< ?before ?s-end))
                                   (and (> ?after ?s-start) (< ?after ?s-end)))))
  (gap ?before-s ?before)
  (gap ?after-s ?after)
  =>
  (if (< (- ?before ?before-s) (- ?after ?after-s))
   then
     (delete-break sketchUnit ?before)
     (retract ?b)
   else
     (delete-break sketchUnit ?after)
     (retract ?a)
  )
)

(defrule retract-sketch-unit-break-similar-shapes
  ?b <- (break sketchUnit ?brk)
  (time-and-shape-similar ?start ?end)
  (test (eq (overlap ?brk ?brk ?start ?end) TRUE))
  =>
  (delete-break sketchUnit ?brk)
  (retract ?b)
)

;;; When there is a pause and a "the" there might be a break.
(defrule the-and-pause
```

```
  (long-pause ?start ?end)
  (the-words-at ?end)
  =>
  ; break should be at end
  (insert-break thePause ?end)
  (assert (break thePause ?end))
)

;;; Sometimes a the-and-pause might be erroneously inserted because of
;;; a long-pause that shouldn't be there.
(defrule remove-the-and-pause
  ?b <- (break thePause ?end)
  (not (long-pause ?start ?end))
  =>
  (delete-break thePause ?end)
  (retract ?b)
)

;;; Breaks shouldn't overlap mumbled words.
(defrule remove-link-by-mumble-break
  ?b <- (break ?type ?end)
  (linked-by-mumbles ?m-start ?m-end)
  (test (< ?m-start ?end))
  (test (> ?m-end ?end))
  =>
  (delete-break ?type ?end)
  (retract ?b)
)

;;; When there is a gap followed by a key word there might be a break
;;; the gap should be of a significant length
(defrule gap-and-keyWords
  (gap ?start ?end)
  (key-words-at ?end ?end2 num ?num)
  (test (> (- ?end ?start) ?*sig-gap*))
  =>
  ; break should be between them
  (insert-break gapKeyWords ?end)
  (assert (break gapKeyWords ?end))
)

;;; Sometimes a gap-and-keyWords might be erroneously inserted because of
;;; a gap that shouldn't be there.
(defrule remove-gapKeyWords-1
  ?b <- (break gapKeyWords ?end)
  (not (gap ?start ?end))
  =>
  (delete-break gapKeyWords ?end)
  (retract ?b)
)

;;; Sometimes a gap-and-keyWords might be erroneously inserted because of
;;; a gap that is too short
(defrule remove-gapKeyWords-2
  ?b <- (break gapKeyWords ?end)
  (gap ?start ?end)
  (not   (test (> (- ?end ?start) ?*sig-gap*)))
  =>
  (delete-break gapKeyWords ?end)
  (retract ?b)
)
```

```
;;; Breaks shouldn't overlap strong-group-link
(defrule remove-strong-group-link
  ?b <- (break ?type ?end)
  (strong-group-link ?s-start ?s-end)
  (test (< ?s-start ?end))
  (test (> ?s-end ?end))
  =>
  (delete-break ?type ?end)
  (retract ?b)
)
```

Appendix D

Hand Segmentation Graphs

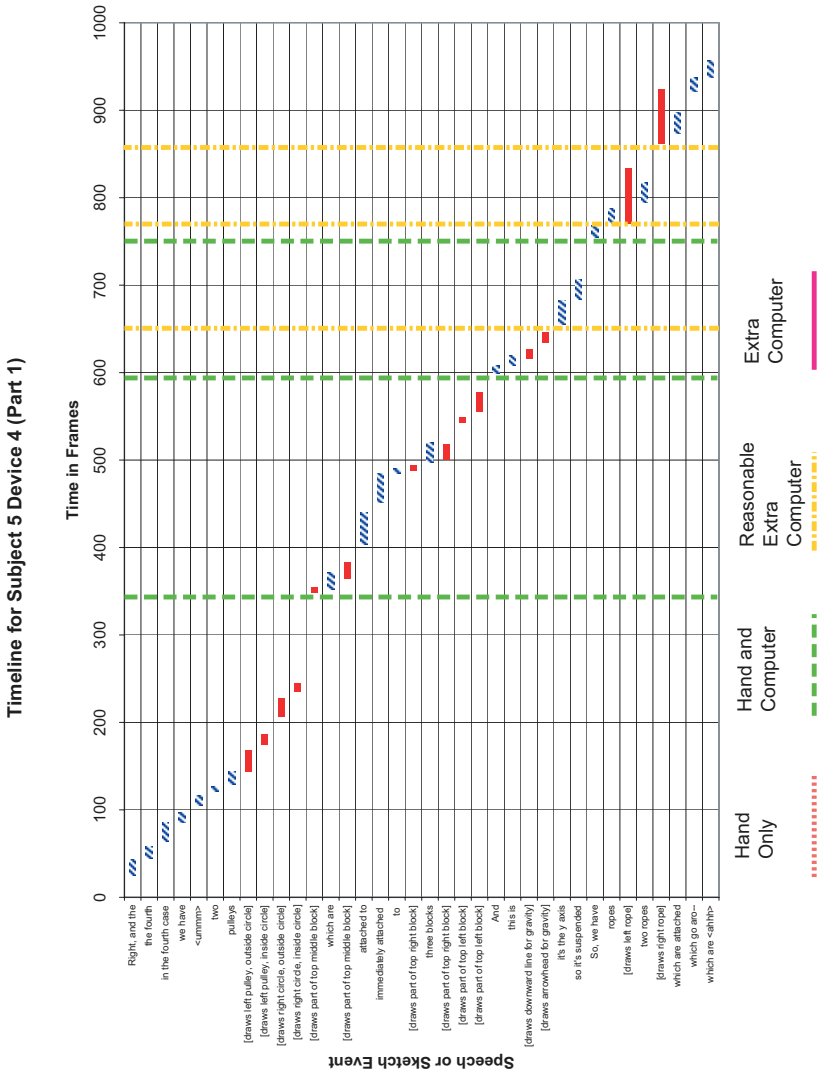


Figure D.1: Part 1 of Subject 5 Device 4 with lines indicating the segmentation points of both hand segmentation and computer segmentation.

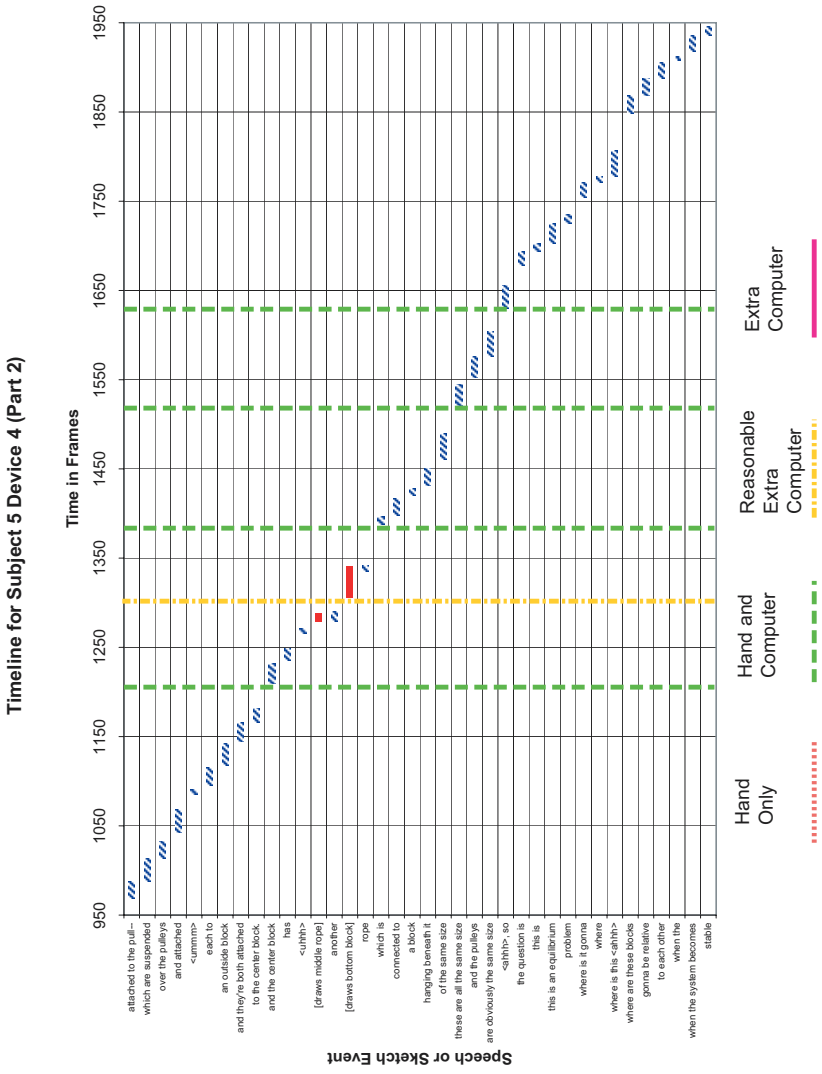


Figure D.2: Part 2 of Subject 5 Device 4 with lines indicating the segmentation points of both hand segmentation and computer segmentation.

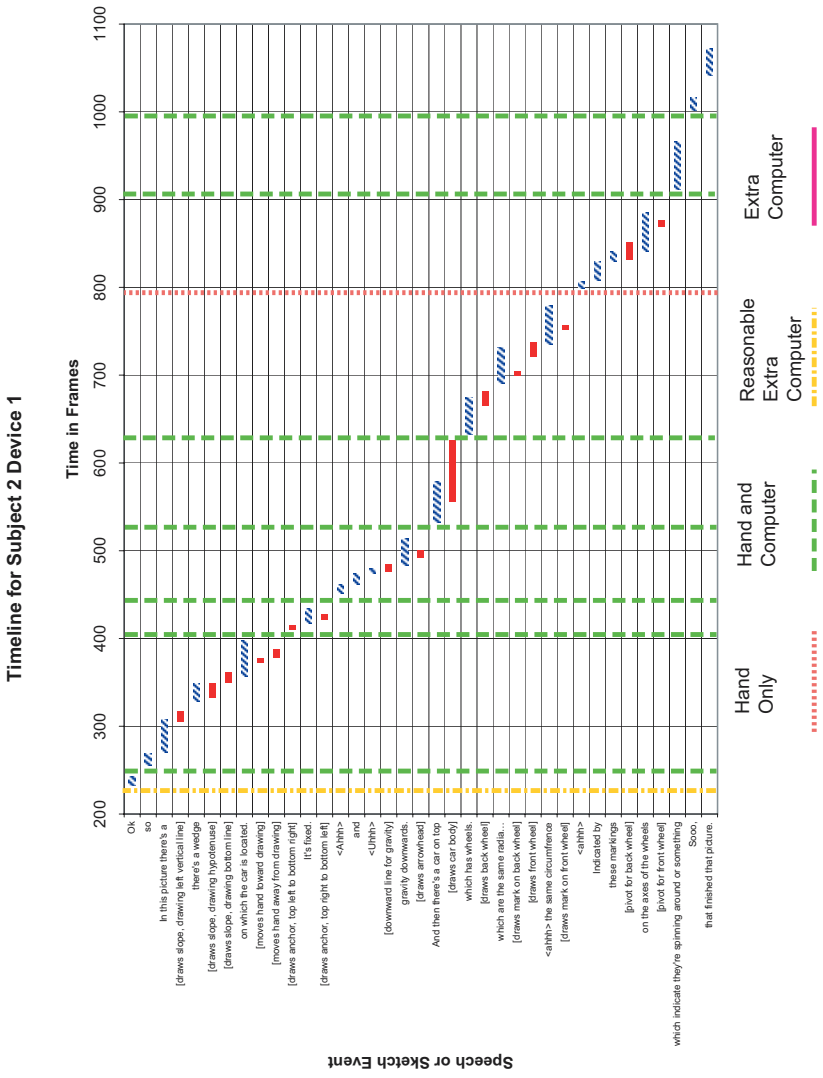


Figure D.3: Subject 2 Device 1 with lines indicating the segmentation points of both hand segmentation and computer segmentation.

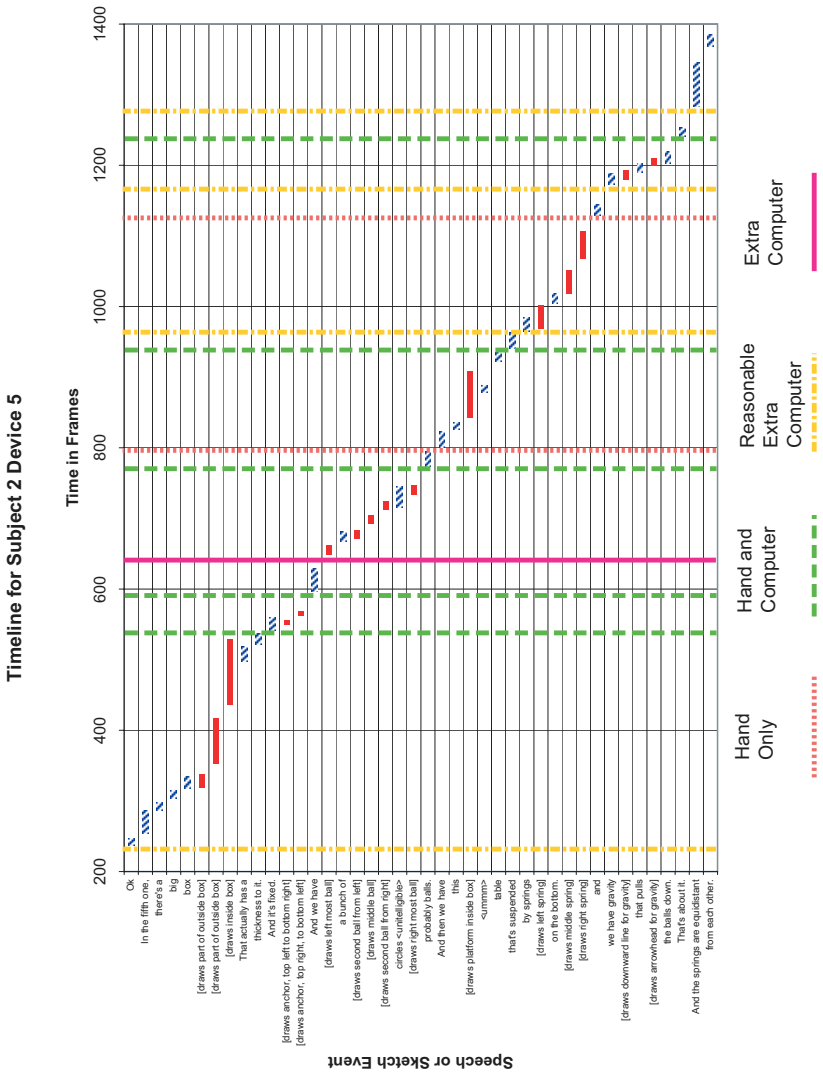


Figure D.4: Subject 2 Device 5 with lines indicating the segmentation points of both hand segmentation and computer segmentation.

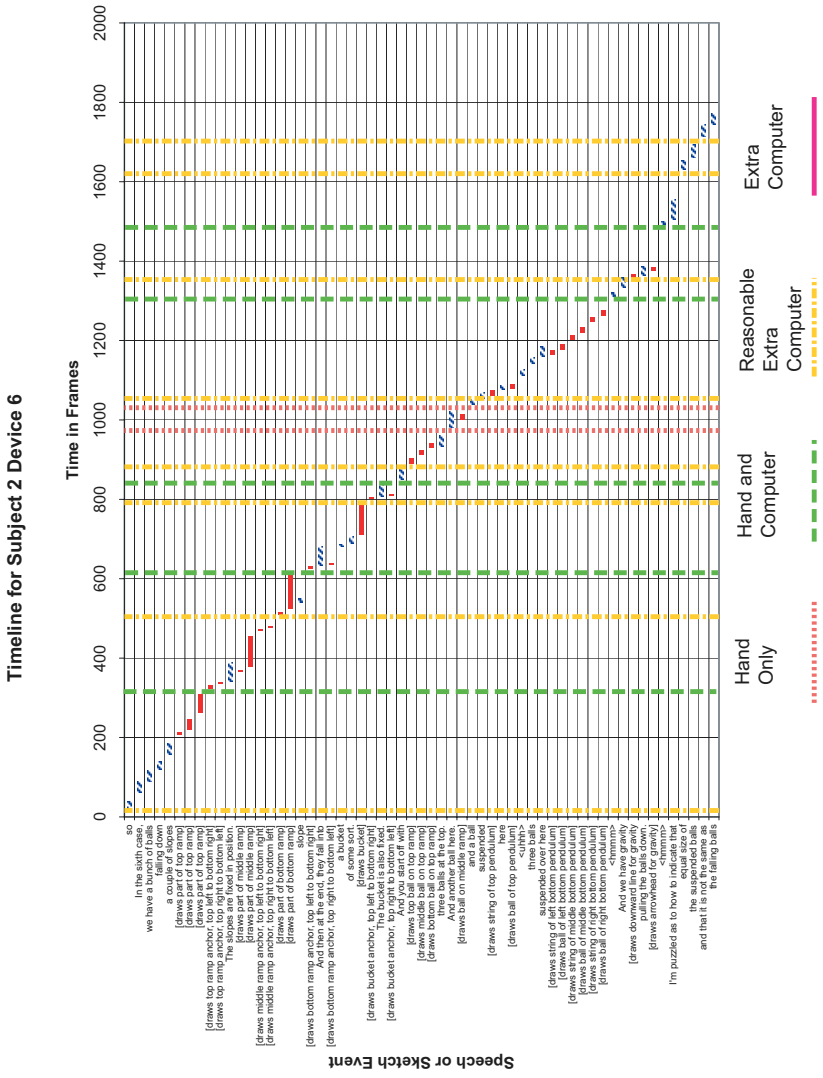


Figure D.5: Subject 2 Device 6 with lines indicating the segmentation points of both hand segmentation and computer segmentation.

Bibliography

[1] Christine Alvarado. A Natural Sketching Environment: Bringing the Computer into Early Stages of Mechanical Design. Master's Thesis, Massachusetts Institute of Technology, 2000.

[2] Christine Alvarado and Randall Davis. Preserving the freedom of paper in a computer-based sketch tool. In *Proceedings of HCI International 2001*, 2001.

[3] Christine Alvarado and Randall Davis. Resolving ambiguities to create a natural computer-based sketching environment. In *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence*, pages 1365–1374, 2001.

[4] Richard A. Bolt. "Put-That-There": Voice and gesture at the graphics interface. In *Proceedings of the 7th Annual Conference on Computer Graphics and Interactive Techniques*, pages 262–270, 1980.

[5] Norman Carver and Victor Lesser. The evolution of blackboard control architectures. Technical report, University of Massachusetts, 1992.

[6] Kenneth D. Forbus, Ronald W. Ferguson, and Jeffrey M. Usher. Towards a computational model of sketching. In *Proceedings of the 6th International Conference on Intelligent User Interfaces*, pages 77–83. ACM Press, 2001.

[7] Kenneth D. Forbus and Jeffrey Usher. Sketching for knowledge capture: A progress report. In *Proceedings of the 7th International Conference on Intelligent User Interfaces*, pages 71–77. ACM Press, 2002.

[8] Michael Johnston and Srinivas Bangalore. Finite-state multimodal parsing and understanding. In *Proceedings of COLING-2000*, 2000.

[9] Michael Johnston, Srinivas Bangalore, Gunaranjan Vasireddy, Amanda Stent, Patrick Ehlen, Marilyn Walker, Steve Whittaker, and Preetam Maloor. MATCH: An architecture for multimodal dialogue systems. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 376–383, 2002.

[10] James V. Mahoney and Markus P. J. Fromherz. Three main concerns in sketch recognition and an approach to addressing them. In *AAAI Spring Symposium on Sketch Understanding*, 2002.

[11] Manoj Muzumdar. ICEMENDR: Intelligent Capture Environment for Mechanical Engineering Drawing. Master's Thesis, Massachusetts Institute of Technology, 1999.

[12] Penny Nii. The blackboard model of problem solving. *AI Magazine*, 7(2):38–53, 1986.

[13] Penny Nii. Blackboard systems part two: Blackboard application systems. *AI Magazine*, 7(3):82–106, 1986.

[14] Gordon S. Novak and William C. Bulko. Diagrams and text as computer input. *Journal of Visual Languages and Computing*, 4(2):161–175, June 1993.

[15] Michael Oltmans. Understanding Naturally Conveyed Explanations of Device Behavior. Master's Thesis, Massachusetts Institute of Technology, 2001.

[16] Michael Oltmans and Randall Davis. Naturally conveyed explanations of device behavior. In *Workshop on Perceptive Use Interfaces*. ACM Digital Library, November 2001.

[17] Sharon Oviatt. Mutual disambiguation of recognition errors in a multimodal architecture. In *Proceeding of the CHI 99 Conference on Human Factors in Computing Systems: the CHI is the Limit*, pages 576–583. ACM Press, 1999.

[18] Sharon Oviatt. Ten myths of multimodal interaction. *Communications of the ACM*, 42(11):74–81, 1999.

[19] Sharon Oviatt, Phil Cohen, Lizhong Wu, John Vergo, Lisbeth Duncan, Berhard Suhm, Josh Bers, Thomas Holzman, Terry Winograd, James Landay, Jim Larson, and David Ferro. Designing the user interface for multimodal speech and pen-based gesture applications: State-of-the-art systems and future research directions. *Human Computer Interaction*, 15(4):263–322, 2000.

[20] Sharon Oviatt and Philip Cohen. Multimodal interfaces that process what comes naturally. *Communications of the ACM*, 43(3):45–53, March 2000.

[21] Sharon Oviatt, Antonella DeAngeli, and Karen Kuhn. Integration and synchronization of input modes during multimodal human-computer interaction. In *Conference Proceedings on Human Factors in Computing Systems*, pages 415–422. ACM Press, 1997.

[22] Metin Sezgin. Feature Point Detection and Curve Approximation for Early Processing of Free-Hand Sketches. Master's Thesis, Massachusetts Institute of Technology, 2001.

[23] Luke Weisman. A Foundation for Intelligent Multimodal Drawing and Sketching Programs. Master's Thesis, Massachusetts Institute of Technology, 1999.