Pen-based Methods for Recognition and Animation of Handwritten Physics Solutions

by

SALMAN CHEEMA
B.Sc Computer Science. Lahore University of Management Sciences, 2005
M.S Computer Science. University of Central Florida, 2012

A dissertation submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy
in the Department of Electrical Engineering and Computer Science
in the College of Engineering and Computer Science
at the University of Central Florida
Orlando, Florida

Fall Term
2014

Major Professor: Joseph J. LaViola Jr.

ABSTRACT

There has been considerable interest in constructing pen-based intelligent tutoring systems due to the natural interaction metaphor and low cognitive load afforded by pen-based interaction. We believe that pen-based intelligent tutoring systems can be further enhanced by integrating animation techniques. In this work, we explore methods for recognizing and animating sketched physics diagrams. Our methodologies enable an Intelligent Tutoring System (ITS) to understand the scenario and requirements posed by a given problem statement and to couple this knowledge with a computational model of the student's handwritten solution.

These pieces of information are used to construct meaningful animations and feedback mechanisms that can highlight errors in student solutions. We have constructed a prototype ITS that can recognize mathematics and diagrams in a handwritten solution and infer implicit relationships among diagram elements, mathematics and annotations such as arrows and dotted lines. We use natural language processing to identify the domain of a given problem, and use this information to select one or more of four domain-specific physics simulators to animate the user's sketched diagram. We enable students to use their answers to guide animation behavior and also describe a novel algorithm for checking recognized student solutions. We provide examples of scenarios that can be modeled using our prototype system and discuss the strengths and weaknesses of our current prototype.

Additionally, we present the findings of a user study that aimed to identify animation requirements for physics tutoring systems. We describe a taxonomy for categorizing different types of animations for physics problems and highlight how the taxonomy can be used to define requirements for 50 physics problems chosen from a university textbook. We also present a discussion of 56 handwritten solutions acquired from physics students and describe how suitable animations could be constructed for each of them.

This dissertation is dedicated to my parents, Shaukat Ali Cheema and Rubina Shaukat

my wife Sana Aziz,

and my siblings Arslan, Hassaan and Anum

Thanks for being there.

Without your support and encouragement, this would not have been possible.

ACKNOWLEDGMENTS

I want to thank my advisor, Dr. Joseph J. LaViola Jr, for his continued support and advice. His guidance helped shape and clarify several important aspects of this work. I would also like to thank the members of my dissertation committee, Dr. Charles E. Hughes, Dr. Tracy Anne Hammond, and Dr. Gita Sukthankar for their feedback. I also want to thank Dr. Kurt VanLehn for his input on bayesian networks and deductive reasoning. I thank Dr. Sumit Gulwani for his guidance and support on the QuickDraw project, which was designed and developed under his supervision during a summer internship at Microsoft Research. I want to thank Clayton Andrews for his help. I also want to thank the members of the Interactive Systems and User Experience (ISUE) lab for their support during my stay at UCF.

My friends in Orlando deserve a special mention: Nazim Ashraf, Zain Masood, Sana Khosa, Mumtaz Sheikh, Adeel Ali, Asher Ahmad, Imran Saleemi, Nadia Imran, Kamran Sadiq, Rizwan Ashraf, Faisal Moinuddin, and Siddhant Mohapatra. Thank you guys for making my time in Florida fun and worthwhile.

This work is supported in part by NSF CAREER award IIS-0845921 and NSF awards IIS-0856045 and CCF-1012056.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1
INTRODUCTION

Intelligent Tutoring Systems (ITS) are an important area of research. In recent years, several researchers have attempted to use pen-based input to create better intelligent tutoring systems for a variety of domains such as Circuit Analysis [53, 72, 229], Chemistry [156, 198], Mechanical Design [5, 104, 153], Mathematics [14, 31, 48, 98, 120], Computer Science [33, 103], and Introductory Physics [17, 40, 121, 123, 174].

This interest in using pen interaction to build educational software is well founded because pen-based interfaces are natural and transparent [1], allow users to input mathematics faster than typing [12, 13] and may help reduce cognitive load on students compared to menu-based systems [194, 195]. A good discussion of the affordances of pen-based interfaces for educational software is presented in [14] and [17]. In summary, pen-based interaction closely mimics the ease of using pen and paper to take notes and communicate ideas, and is more natural than using a WIMP-based system or typing up solutions. These factors make pen-based interaction particularly suitable for educational software.

The act of drawing (or sketching)[1] is an important part of the design process [129, 106, 204, 205] and has important benefits in science education [3, 117, 208]. Larkin and Simon [117] examine the importance of visualization and drawing, concluding: "*Diagrams can group together all information that is used together, thus avoiding large amounts of search for the elements needed to make a problem solving inference*".

[1]For clarity, drawing or sketching is different from writing text or mathematics. It relates either to freeform sketching or using domain specific notation for diagramming.

Additionally, diagrams represent information in a spatial rather than a sequential manner (a.k.a textually), enabling easier spatial inferences [117]. Spatial skills have been found to correlate strongly with improved performance in STEM[2] disciplines [208]. Additionally, sketching diagrams has positive benefits for students in the form of increased engagement and improved learning [3]. In conclusion, drawing is an integral aspect of problem solving in science. Pen-based input provides the ability to leverage the natural problem solving behavior, including writing mathematics and sketching diagrams for constructing a unique class of intelligent tutoring systems.

4.10 •• A dockworker applies a constant horizontal force of 80.0 N to a block of ice on a smooth horizontal floor. The frictional force is negligible. The block starts from rest and moves 11.0 m in 5.00 s. (a) What is the mass of the block of ice? (b) If the worker stops pushing at the end of 5.00 s, how far does the block move in the next 5.00 s?

Figure 1.1: Friction problem statement taken from University Physics 13 Ed [226], Page 129.

An important goal is to understand the natural workflow that students employ in scientific problem solving, so that it can be supported in a tutoring system. Alvarado et al. [7] have done preliminary work in this vein with respect to drawing logic diagrams. For physics, it is instructive to examine some physics problems and their solutions. Figure 1.1 depicts the statement of a multipart kinematics problem that requires the application of $f = ma$ for a correct solution. Figure 1.2 shows a written solution acquired from a student for this problem. The solution demonstrates that writing mathematics and making diagrams are two integral aspects of problem solving. The solution in Figure 1.2 shows a sketched diagram that has been annotated with arrows, distance measurements and an equation for force. Both parts of the solution are clearly labeled. For part (a), the student uses the equation of motion ($S = v_0t + \frac{1}{2}at^2$) to reach an answer. For part (b), the student first uses the principle of work done to derive a value for the velocity which is used to reach the required answer. Alternatively, it is also possible to solve part (b) with just the equations

[2] Science, Technology, Engineering and Mathematics

of motion (Use $v_1^2 - v_0^2 = 2aS$ to derive velocity). When we examine the contents of this solution, the diagram was sketched to model the scenario of the assigned problem and it was annotated with the initial conditions provided in the statement. The student used physics and mathematics principles to write a series of equations which lead to an answer which was a numeric quantity in both these instances. Generally, the answer to a physics problem may be a function or a number, and in special cases, an identity or a statement.

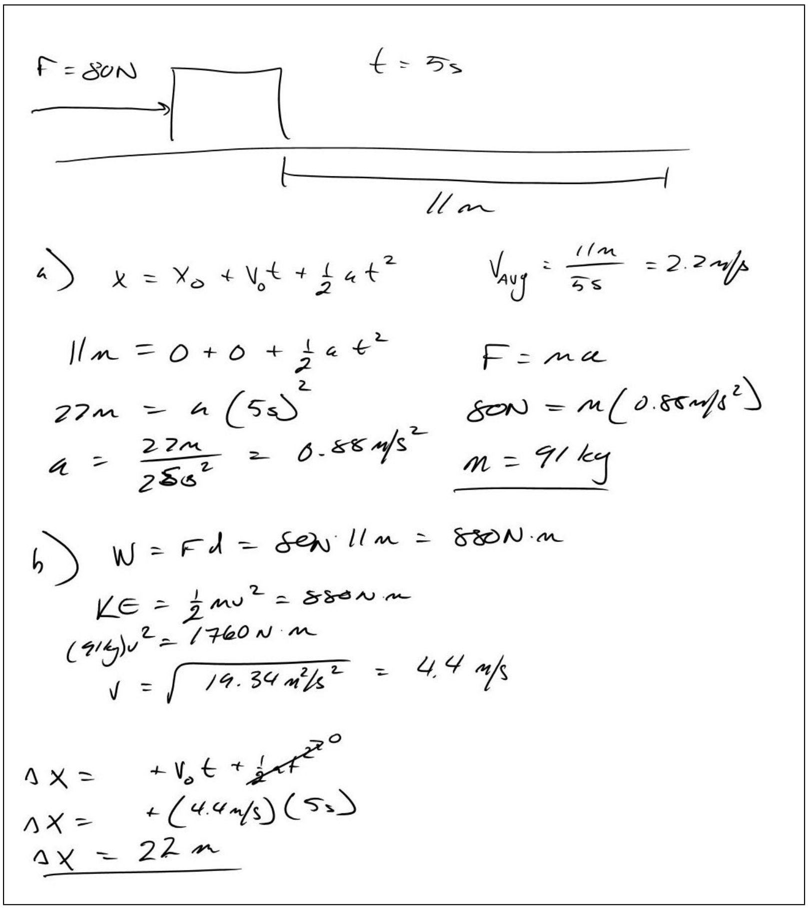


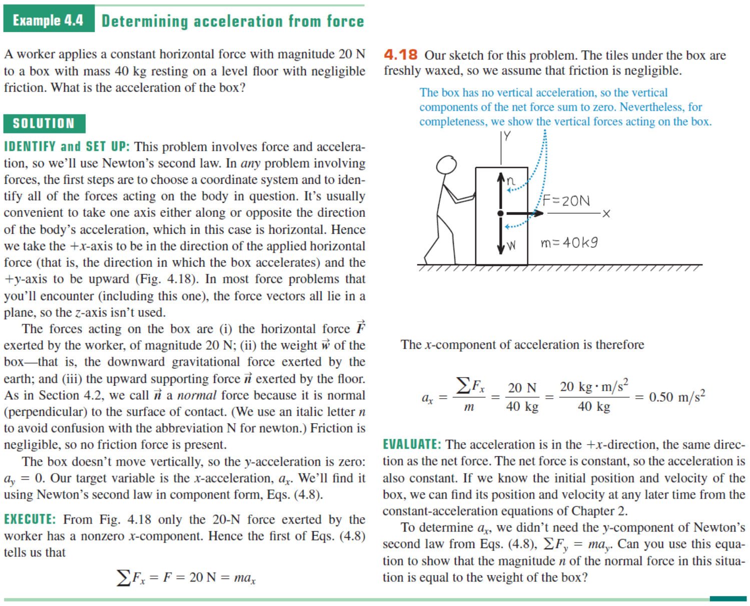
Figure 1.2: A handwritten solution for the kinematics problem presented in Figure 1.1.

On paper, the solution to a solved problem is static, providing no insight into how the answer would affect the scenario provided in the problem. Research indicates that when solving mechanical problems, many people report consciously simulating what would happen [44, 101, 181, 182]. Hegarty [87] presents a good introduction to various aspects of mental animation as applied to mechanical reasoning. We hypothesize that by providing the ability to animate student's diagrams, we will be able to construct better tutoring systems, because, on paper, the answer to an assigned problem in mathematics or science itself does little to impart intuitive knowledge to a student.

Our primary research goal, therefore, is to investigate methods and techniques to enhance the state-of-the-art for pen-based intelligent tutoring systems in the domain of physics, with an emphasis on supporting natural workflow and providing animation support for sketched diagrams. To support a natural workflow, students must be allowed to write in an unconstrained manner. An ITS must also develop a deep understanding of each student's problem solving process including what is asked in the statement, what has the student written down, and abilities to reason about sketched diagram(s). We hypothesize that given this level of understanding, an ITS will be better suited to check solutions for correctness, provide meaningful feedback, and aid knowledge scaffolding and student learning.

1.1 Motivating Examples

This section presents three example problems taken from Young and Freedman's University Physics 13th Edition [226] that demonstrate the desired functionality of our proposed system and also highlight inherent research challenges.



Example 4.4 Determining acceleration from force

A worker applies a constant horizontal force with magnitude 20 N to a box with mass 40 kg resting on a level floor with negligible friction. What is the acceleration of the box?

SOLUTION

IDENTIFY and SET UP: This problem involves force and acceleration, so we'll use Newton's second law. In *any* problem involving forces, the first steps are to choose a coordinate system and to identify all of the forces acting on the body in question. It's usually convenient to take one axis either along or opposite the direction of the body's acceleration, which in this case is horizontal. Hence we take the $+x$-axis to be in the direction of the applied horizontal force (that is, the direction in which the box accelerates) and the $+y$-axis to be upward (Fig. 4.18). In most force problems that you'll encounter (including this one), the force vectors all lie in a plane, so the z-axis isn't used.

The forces acting on the box are (i) the horizontal force $\vec{F}$ exerted by the worker, of magnitude 20 N; (ii) the weight $\vec{w}$ of the box—that is, the downward gravitational force exerted by the earth; and (iii) the upward supporting force $\vec{n}$ exerted by the floor. As in Section 4.2, we call $\vec{n}$ a *normal* force because it is normal (perpendicular) to the surface of contact. (We use an italic letter n to avoid confusion with the abbreviation N for newton.) Friction is negligible, so no friction force is present.

The box doesn't move vertically, so the y-acceleration is zero: $a_y = 0$. Our target variable is the x-acceleration, a_x. We'll find it using Newton's second law in component form, Eqs. (4.8).

EXECUTE: From Fig. 4.18 only the 20-N force exerted by the worker has a nonzero x-component. Hence the first of Eqs. (4.8) tells us that

$$\sum F_x = F = 20\ \text{N} = ma_x$$

4.18 Our sketch for this problem. The tiles under the box are freshly waxed, so we assume that friction is negligible.

The x-component of acceleration is therefore

$$a_x = \frac{\sum F_x}{m} = \frac{20\ \text{N}}{40\ \text{kg}} = \frac{20\ \text{kg} \cdot \text{m/s}^2}{40\ \text{kg}} = 0.50\ \text{m/s}^2$$

EVALUATE: The acceleration is in the $+x$-direction, the same direction as the net force. The net force is constant, so the acceleration is also constant. If we know the initial position and velocity of the box, we can find its position and velocity at any later time from the constant-acceleration equations of Chapter 2.

To determine a_x, we didn't need the y-component of Newton's second law from Eqs. (4.8), $\sum F_y = ma_y$. Can you use this equation to show that the magnitude n of the normal force in this situation is equal to the weight of the box?

Figure 1.3: Source: University Physics 13 Ed [226], Page 116. In this example, the textbook provides the answer and also indicates possible strategies for a student to approach such problems.

1.1.1 Example 1: Acceleration of a Moving Box

Figure 1.3 shows an worked example taken from a university level physics textbook. The task is to find the acceleration of a box moving under the action of a horizontal force. The scenario involves 1-Dimensional motion and requires the use of the $f = ma$ equation along the x-axis to find the answer. However, if considered from the perspective of an intelligent tutoring system, there are several complex issues. First, there is the problem of specification. Each problem statement assigned to students must be properly instrumented by an instructor or teaching assistant. This is necessary because the system needs to be aware of what information is provided to the student if it is to judge whether the student has used the information correctly. Second, if the problem was solved in an unconstrained manner (e.g., freeform input with pen and paper.), then sketch recognition principles need to be applied to:

- Disambiguate the diagram and its annotations from the equations in the solution.
- Construct a computational model of the diagram by recognizing its elements and correlating them with their annotations.
- Recognize the mathematical equations in the solution and infer their logical ordering.
- Determine if the solution is mathematically and logically sound.

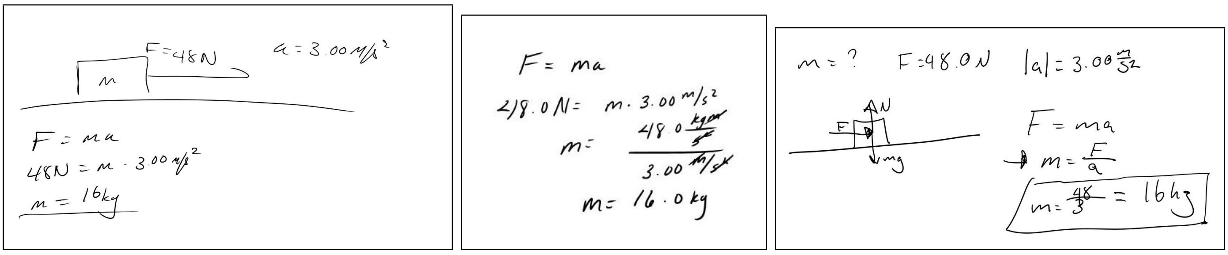
Finally, there is the need to animate the sketched diagram by using the derived answer. For this purpose, the ITS needs information about the domain of the problem in order to use the proper animation mechanisms. Additionally, knowledge of domain-specific mathematical notation is also required to understand and use the student's answer for animation.

1.1.2 Example 2: Deduce Mass Using Force Equation

4.9 • A box rests on a frozen pond, which serves as a frictionless horizontal surface. If a fisherman applies a horizontal force with magnitude 48.0 N to the box and produces an acceleration of magnitude 3.00 m/s^2, what is the mass of the box?

Figure 1.4: Source: University Physics 13 Ed [226], Page 129.

Figure 1.4 shows the statement of a simple kinematics problem that can be solved by the direct application of $f = ma$. Figure 1.5 depicts three solutions to this problem collected from students. All three solutions use the same physics principle to solve the problem yet there is significant variation in the notation used. Figure 1.5(a) annotates the diagram with the initial conditions, while Figure 1.5(c) writes down the initial conditions and only labels the forces on the diagram. Figure 1.5(c) also labels an arrow for the normal force acting on the box which is extraneous information for the problem at hand. Figure 1.5(b) does not draw a diagram and also shows how the units for mass are derived in his solution.



(a) Solution1 (b) Solution2 (c) Solution3

Figure 1.5: Three solutions to the physics problem presented in Figure 1.4 that use the same principle ($f = ma$) to solve the problem, yet show wide variation in notation and diagrams.

These solutions show that even for a simple problem, there is potentially huge variation in how students sketch diagrams and write down their solutions. This is in agreement with the findings of Alvarado et al. [7] one of which is explicitly stated as '*Symbol recognizers must incorporate a wide range of drawing styles (for most users)*'.

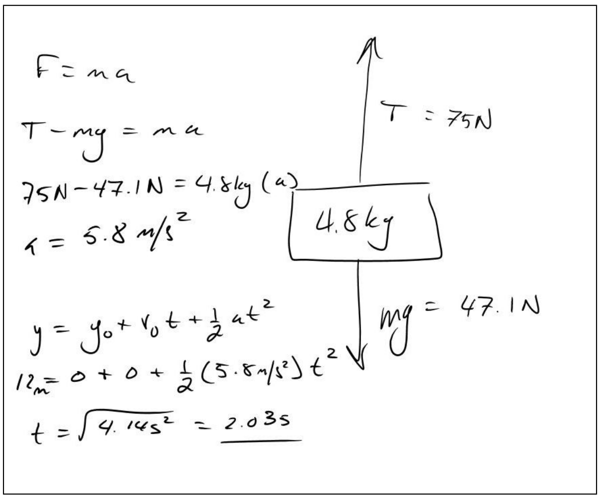
1.1.3 Example 3: Motion Under Constant Acceleration

4.33 CP A 4.80-kg bucket of water is accelerated upward by a cord of negligible mass whose breaking strength is 75.0 N. If the bucket starts from rest, what is the minimum time required to raise the bucket a vertical distance of 12.0 m without breaking the cord?

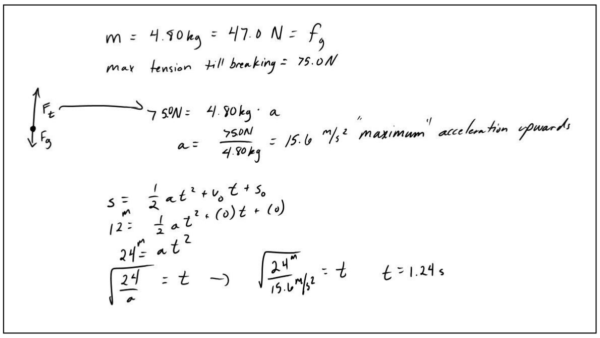
Figure 1.6: Source: University Physics 13 Ed [226], Page 130.

Figure 1.6 shows the statement of a another kinematics problem. Figure 1.7 depicts two solutions to this problem collected from students. Again, we observe significant variation in notation between both solutions, which use the same physics principle (equation of motion $S = v_0 t + \frac{1}{2}at^2$)

to solve the problem. The diagram in Figure 1.7(a) consists of a large box labeled with mass and annotated with arrows for weight and force. Figure 1.5(b), on the other hand, contains a free body diagram consisting of a point mass annotated with labeled arrows. Figure 1.7(b) also includes text phrases, interspersed within the solution that explain the student's reasoning.



(a) Solution1



(b) Solution2

Figure 1.7: Two solutions to the physics problem presented in Figure 1.6 that use the same principle ($S = v_0t + \frac{1}{2}at^2$) to solve the problem, yet show wide variation in notation and diagrams.

1.2 Proposed Features for Pen-based Intelligent Tutors

The solutions presented in Figures 1.2, 1.5, and 1.7 clearly demonstrate that mathematical equations and sketched diagrams are the most important aspects of written solutions. However, notation and problem solving strategy can vary widely, even for relatively simple problems. Diagrams contained within physics solutions contain complex, interrelated elements, such as shapes (denoting moving objects), annotation symbols, labels, equations denoting initial conditions, and possible text phrases explaining the student's reasoning.

We propose the following set of high-level features for pen-based physics tutoring systems:

1. *Natural Workflow*: The ability to solve a given problem in a natural manner. In mathematics and physics, the workflow begins with a text statement outlining a scenario. A student then provides a handwritten solution containing diagram(s) and mathematical equations. We believe that providing an approximation of unconstrained pen-input is the ideal interaction metaphor for this purpose.

2. *Deep Understanding*: Given a problem and its handwritten solution, the tutoring system should be able to:

   - Infer what information is provided to the student, e.g., what are the initial conditions in the problem statement?
   - Understand the objective of the question. Is the student asked to compute the value of some quantity or derive a function?
   - Construct a computational model for the diagram in the solution.
   - Construct a computational model for the mathematical steps in the solution.

3. *Multiple Sources of Input*: In order to achieve the deep understanding goal, sketch recognition techniques alone are not enough. The tutoring system must use different input sources in parallel: the text of a given problem, the mathematical steps in the solution, and possibly, contextual information provided by the instructor (problem authoring)[3].

4. *Integrated Domain Knowledge*: A tutoring system must necessarily incorporate knowledge about its target domain(s). This knowledge enables the system to understand the recognized solution to a given problem, in light of the notation and rules of a particular domain. Domain

[3]One of our guiding principles is to minimize the need for authoring and instrumenting by instructors and teaching assistants. This makes the deep understanding goal more difficult, as metadata for the assigned problem must now be inferred rather than specified by a power user. In this work, we utilize Natural Language Processing techniques to extract this information from the problem statement.

knowledge can be used to aid the recognition process by providing contextual clues for diagram elements. Additionally, it can be used to determine if the student has correctly used mathematical and physics rules, given the requirements inferred from the text of the problem.

5. *Animation Capabilities*: Given the importance of visual feedback and animation [44, 87, 101, 117, 181, 182], an tutoring system should be able to animate the diagram(s) in a student's solution. Students should be able to interact with and manipulate the animation of sketched diagrams, by use of hand-written mathematics and/or gestures. This functionality enables students to explore alternate answers to a given scenario. This is important because the ability to construct models of alternate possibilities leads to an improvement in reasoning capabilities [101].

6. *Feedback*: Feedback can take several forms. First, the ITS should be able to detect and highlight mistakes in a given solution. Second, students should be able to view how physics quantities vary over time. Graphing is an important tool in this respect. Step by step feedback and error highlighting may also be used for knowledge scaffolding [217].

1.3 Research Challenges

We need to address three fundamental challenges to reach out stated goals: 1) parsing a text statement via natural language processing, 2) recognition of a written solution to generate a computational model for animation and feedback, and 3) the ability to animate a recognized diagram. A sizable body of work exists for generating realistic physics animations in video games and simulation [22, 21, 61, 213, 140]. Such techniques can be adapted for tutoring purposes [40, 41, 42]. However, the task of making an animation system general enough to deal with a variety of physics concepts is non-trivial and poses significant design challenges. With respect to natural language processing, our needs are modest and can be met via simple heuristics built around existing frameworks such as SharpNLP [180] which provides a wrapper around the powerful WordNet lexical database [62, 139].

Therefore, unconstrained sketch recognition is the most challenging research question for our goals. Animation techniques are well-known for narrow domains but using them in a general purpose system presents a significant design challenge. Given a written solution, there is an initial need to distinguish the mathematical steps from diagram(s) and associated annotations. Ideally, sketched diagrams and mathematical equations in the solution ought to be two separate input modalities [4], but in unconstrained solutions, they must be separated by the use of sketch recognition techniques. The recognition problem is made harder by the fact that some mathematical equations are part of the sketched diagram in the form of annotations.

Reliable recognition of mathematical equations is another important challenge. With respect to handwriting and mathematics recognition, evidence suggests that adult users are not willing to accept accuracies lower than 97% [114] while children are willing to accept accuracies as low as 91% [167], allowing us to specify theoretical bounds on the required recognition accuracy for a tutoring system as between 91% to 97%. These figures also correspond well with human recognition of handwriting, which ranges from 94.9% to 96.5% [148].

Research challenges for understanding physics diagrams can be summarized as [40]:

1. <u>Unknown Elements</u>: The number of diagram elements in a target domain should be well-known, so that recognition systems can be customized for them. However, given the vast scale of physics problems, the number of potential diagram elements is potentially boundless.

[4] Indeed, some existing pen-based systems [98, 17, 123, 121] treat diagrams and solution steps as separate input modalities by forcing users to draw and write in separate panes on the same interface. We believe this approach detracts from the natural interaction afforded by pen-based systems and induces extraneous cognitive load.

2. Inference of Context: Oftentimes, there are multiple ways of sketching a diagram, which can be annotated and labeled with arrows, dotted lines, alphanumeric symbols and even equations [5]. Similar annotations can carry different meaning in different contexts.

3. Partial Specification: The information contained in sketched diagrams may be incomplete or even redundant in some cases. Such scenarios can be confusing from a recognition standpoint.

4. Imprecision: Sketched diagrams are approximate, by nature. Their purpose is to condense and clarify the provided information to aid the problem solving process. From an animation perspective, an imprecise diagram may lead to an incorrect animation, necessitating the use of sketch beautification techniques to properly represent user intent.

We envision the use of animation as a core feedback mechanism in future tutoring systems. Existing animation systems can be tailored to specific types of problems. However, for intelligent tutoring, it is unfeasible to write specific animation scenarios for particular problems. Therefore, intelligent tutoring systems must support general-purpose animation mechanisms which can be used in conjunction with pen-based input. Yet, any such animation mechanism must be limited by definition, i.e., it will always be possible to find examples of physics problems from the targeted domain that cannot be handled by it (due to generality). The key design challenge here is to describe general animation behaviors that will allow a student to build an animation for a representative sample of problems for a particular domain by using a small set of diagram elements.

[5] This set of annotations is not exhaustive. Depending on the chosen domain, different annotation symbols may need to be supported.

1.4 Reader's Guide

This document is outlined in the following manner. Chapter 2 presents a review of existing work related to sketch understanding and physics tutoring. We outline the high-level design and the user interface of our prototype tutoring system in Chapter 3. Chapters 4 and 5 describe our strategies for 1) extracting information from the problem statement and 2) recognizing and beautifying written solutions and diagrams. Chapter 6 describes our framework for animating recognized physics diagrams. In Chapter 7, we outline a method for modeling recognized solution steps and describe rules for verifying the correctness of a written solution. Next, Chapter 8 presents examples of physics problems taken from a textbook and modeled using our prototype system. This chapter also discusses the limitations of our methodology and identifies categories of animations that need to be supported for physics solutions. In Chapter 9, we analyze 50 physics problems selected from a university level textbook [226] for animation requirements. Chapter 9 also discusses how a corpus of 56 student solutions may be animated. Chapter 10 concludes this work by summarizing its contributions and outlining avenues of future work.

A brief description of our earlier prototypes is given in Appendix A. Appendix B contains IRB approval and closure letters for our user study which was used to construct the dataset examined in Chapter 9.

CHAPTER 2
LITERATURE REVIEW

5.16 •• **CP** A 8.00-kg block of ice, released from rest at the top of a 1.50-m-long frictionless ramp, slides downhill, reaching a speed of 2.50 m/s at the bottom. (a) What is the angle between the ramp and the horizontal? (b) What would be the speed of the ice at the bottom if the motion were opposed by a constant friction force of 10.0 N parallel to the surface of the ramp?

Figure 2.1: Two-part kinematics problem from University Physics 13 Ed [226], Page 165.

2.1 Sketch Recognition and Understanding

Pen-based systems provide the ability to manipulate a computer system via a stylus. Sophisticated methods and techniques are needed to parse and understand sketches drawn by users. The goal of sketch recognition systems is to construct models of input sketches in order to do something interesting. Sketch recognition systems are either interactive (update model after each ink stroke), work in batch mode (process entire sketch in one go, triggered by either a gesture or a button press) or mixed mode (can work in both batch and interactive modes). Gestures can be employed to control various aspects of the user experience.

These choices depend on the chosen objectives of the system and have direct design and usability implications [30, 209]. User perception of sketch recognition has been explored by Wais et al. [209], who conclude that 1) Users prefer to trigger recognition once they have finished drawing

2) The method of triggering recognition must be reliable[1]. Additionally, they also tested methods for indicating annotations and receiving recognition feedback. Similarly, Bott et al. [30] have explored user preferences for batch and realtime recognition feedback for math recognition results via a Wizard of Oz study. Their findings indicate that recognition accuracy has little impact on user preference of feedback mechanism which is instead dictated by the amount of mathematical equations a user is expected to write. In particular, Bott et al [30] found that users prefer realtime feedback when they are expected to write more than a one expression.

Each sketch consists of a collection of digital ink strokes representing different categories of data: text (letters, dots, words, phrases, sentences), mathematics (numbers and equations), and drawings (art, scientific diagrams, tables, etc). There is considerable variation in writing styles and notation between different domains and even between individual users, leading to immense differences in sketched symbols in each of the above categories [131]. Symbols can be written in different sizes, positions, and orientations (rotation and tilting). Users can speed up or slow down their writing while tracing different parts of the same ink strokes. This variation in writing speed is measurable and can serve as an important feature for recognition. Similarly, pause time between ink strokes can also be an important feature for recognition [7]. Users sometimes finish their ink strokes with flourishes[2]. Furthermore, users can sometimes add ink strokes that are intended to correct something drawn earlier[3]. From a user's perspective, a sketch is logically organized into regions containing different types of data. Identification of such regions is a difficult research problem in itself. Some of these issues are highlighted in the two-part solution shown in Figure 2.2, which corresponds to the statement in Figure 2.1.

[1]In their experiments, users preferred the button press method over gesture based recognition, because it was consistently more reliable.

[2]Such flourishes at or near the terminal points of ink strokes are called hooks [119] or tails [84] in sketch recognition literature.

[3]Such extraneous strokes are usually intended to fix or extend existing ink and include behaviors such as overtracing, continuation, touching up, and unintentional strokes [84].

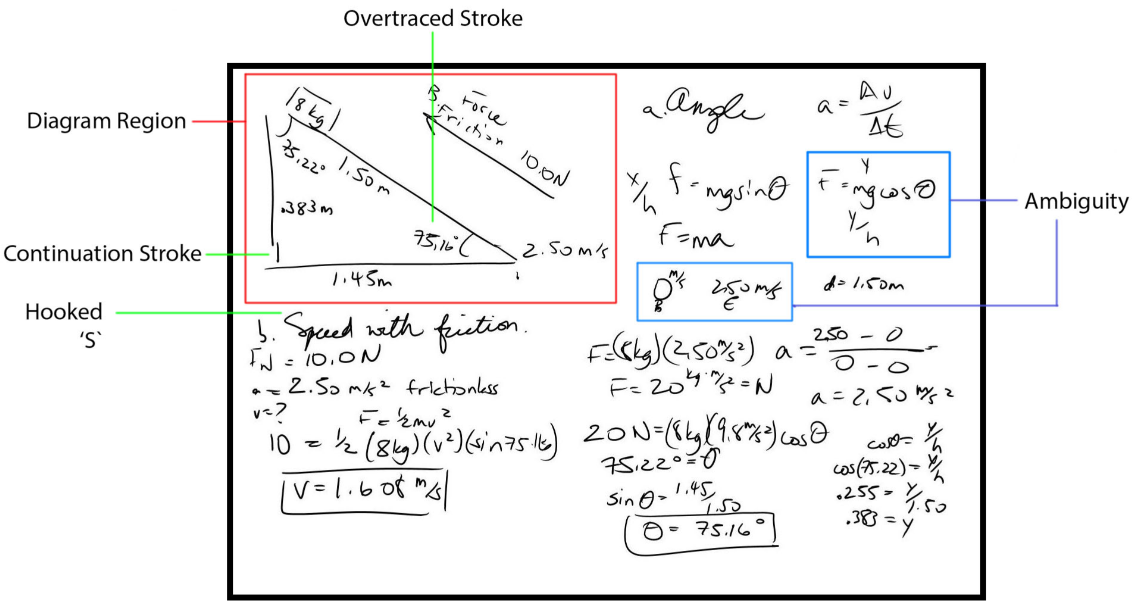


Figure 2.2: A handwritten solution to the problem presented in Figure 2.1. The interesting (from a sketch understanding perspective) elements of the solution are marked for demonstrative purposes.

In conclusion, sketch understanding is a complex problem at a low level. Extracting high-level information from a sketch is yet harder. As mentioned before, there is wide variation in domain-specific notation and individual handwriting. Domain symbols (unistroke or multistroke) must be considered in groups to extract interesting information. Additionally, contextual information, typically encoded as a set of domain-specific rules assumes a critical importance at this stage. For example, in Figure 2.2, a low level recognition engine may recognize the numbers and letters in the equations and text. However, a high-level recognizer must group recognized letters and numbers with shapes and use physics domain knowledge to understand the units and formulae. Resolving ambiguity is a difficult problem, and in our opinion, may not always be possible (e.g., see the regions marked 'ambiguous' in Figure 2.2).

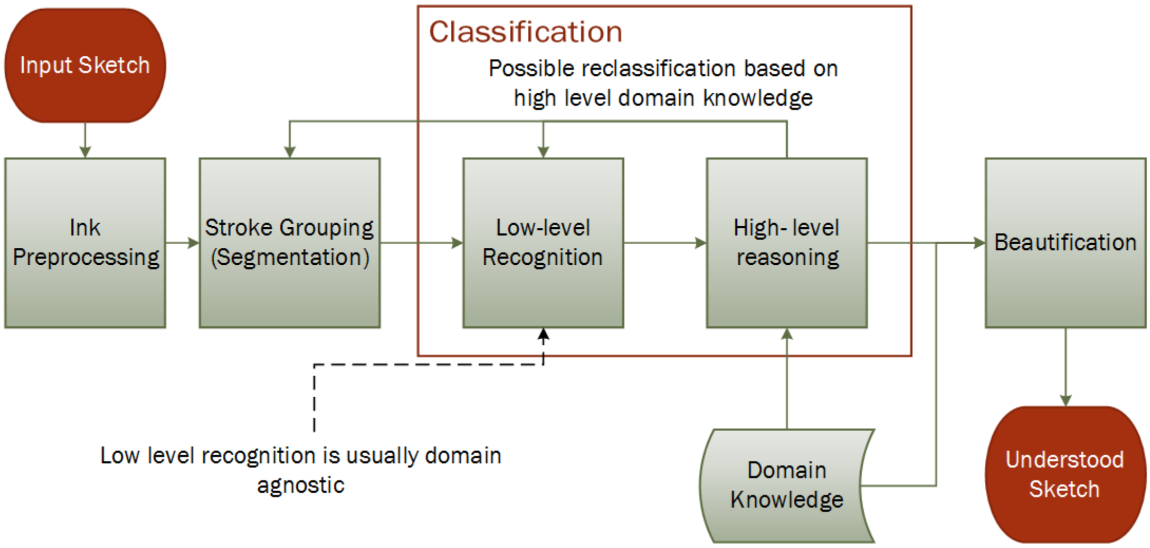


Figure 2.3: A typical sketch recognition workflow is depicted here consisting of ink preprocessing, stroke grouping/segmentation, a classification step subdivided into a low and high level recognition steps, and a final beautification step.

2.1.1 The Ideal Sketch Recognition Pipeline

Sketch understanding systems are usually organized as a sequential pipeline (depicted in Figure 2.3. Initially, ink strokes are preprocessed to remove noise and prepare them for recognition, followed by an optional grouping (also called segmentation) step[4]. Then, low-level domain-agnostic recognizer(s) are employed to extract sketched symbols. Recognized symbols are examined in the context of domain-specific rules (which may possibly trigger reclassification if something illogical is detected) to construct a model of the input sketch. At this point, the job of the

[4]Grouping or Segmentation is an interesting and non-trivial problem. It is inextricably linked to low- and high-level recognition. Deciding which strokes are to be combined to create a single symbol is a partial solution to the symbol recognition problem. Similarly, at a higher level, deciding which groupings of domain symbols are interesting requires the knowledge about notation and domain context. Common methods for grouping ink strokes rely on spatial proximity, temporal order, and encoded context rules. There is considerable flexibility in this step. It can be a part of low and high level recognition or can be performed as a separate preprocessing step. Sometimes, ink strokes must be subdivided during this step to create sub-strokes that correspond to unique domain symbols. For example, the text fragments in Figure 2.2 are written in joined handwriting and need to be segmented. Yet, the F in the mathematics is written with 3 different ink strokes, which need to be grouped.

sketch understanding system is over and the model is passed onto some other module in the system for further computation and/or interaction.

2.1.2 Ink Preprocessing

An input sketch is a collection of digital ink strokes, each of which is represented by a sequence of 2D points with position (x,y), timestamp, pen pressure (usually as a floating point number in the 0-1 range) and tilt information[5]. Different pen-input systems (on a hardware level) may support different subsets of these raw features. However, position (x,y) and timing are always available. During preprocessing, ink strokes in the input sketch are cleaned and prepared for classification. Sezgin et al. [176] proposed a recognition methodology consisting of approximation, beautification, and basic recognition. This strategy is useful for low-level recognition in simple domains where the number of symbols and contextual rules for beautifying them are well known. However for complex domains, this approach is inadequate. Preprocessing can consist of several tasks:

- Dehooking [119]: Removal of areas of sharp curvature near the terminal points of each ink stroke. Hooks are usually artifacts of the sketching process and can confuse cusp detectors by adding false positives.
- Noise Filtering: Use filtering methods (from signal processing) to smooth the points in each ink stroke. This remove kinks and constructs a more uniform view for classification.
- Resampling: Users draw ink strokes with variable speed, leading to a high density of points in some areas of the stroke. During resampling, interpolation methods are used to reduce each stroke to a fixed number of equidistant points. This strategy is used in several corner finding and stroke classification methods [88, 89, 215, 216, 219].

[5] In the past, ink strokes have also been represented by 2D pixel matrices but such representations are quite uncommon nowadays.

- Cusp (Corner) Detection: Cusps or Corners are points of sharp curvature in the ink stroke. They play an important role in stroke segmentation and low level recognition [176, 161][6]. Strategies for cusp detection usually examine geometric and temporal features of stroke points. Some recent reliable methods for cusp detection include [88, 89, 215, 216, 219], each of which defines a preprocessing methodology for noise filtering and resampling. Of these, ClassySeg [88] is one the most recent methods and uses AdaBoosted C4.5 decision trees to achieve 94.6% 'all-or-nothing' cusp recognition accuracy, when trained in a user-optimized configuration.

2.1.3 Region Segmentation and Page Layout Analysis

Identification of interesting regions in a sketch can guide a recognition system to select appropriate algorithms for the class of data in each region. This avenue of research is not entirely related to sketch-based interaction. Instead, it was initially motivated by the need to extract structure from scanned pages and documents. However, there are several insights which are useful for free-form sketch understanding. Some popular techniques in this domain include the Docstrum system [151] (based on bottom-up, nearest neighbor clustering), CLIDE [187] (based on Kruskal's Algorithm), Kise et al. [109] (Approximated Area Voronoi Diagrams) and Lee et al. [122] (Pyramidal Quadtree Representation coupled with Texture Analysis), the RXYC algorithm [145] (Recursive X-Y Cut, top down approach similar to space partitioning algorithms), Morphological Smearing [86], and Whitespace Analysis [20]. Good comparative surveys of early techniques for document analysis have been published by Cattoni et al. [37] and Mao et al. [134].

[6] Cusp detection is not necessarily a preprocessing step. Systems such as Paleosketch [161] and Hammond et al. [84] employ it during the low-level recognition step.

Wang et al. [211] attempted to use machine learning methods (decision trees, HMMs, etc) to identify the contents of known regions, with a reported accuracy of 98.5%. More recently, comparative performance analyses were conducted by Shafait et al. [178] and Indermühle et al. [95] to benchmark classification performance and identify scenarios where different document analysis algorithms prove more suitable. Their results indicate that for non-Manhattan layouts (layouts with text in different orientations), the voronoi diagram based approach by Kise et al. [109] and morphological smearing [86] perform best. This has direct bearing on the design on sketch-based interfaces as user's often write and draw in non-Manhattan layouts.

2.1.3.1 A Note on Text/Non-Text Division Strategies

Deciding if an ink stroke belongs to a text or a non-text symbol is an important question for sketch understanding systems, helping to optimize recognition algorithms. Therefore, it is quite common to treat stroke grouping as part of the text/non-text classification process [184].

Jain et al. [97] did early work in this domain, developing a hierarchical approach based on 2-feature classification to identify regions containing either text or text and graphics, followed by use of a distance-based minimum spanning tree to group non-text strokes into regions containing either tables or diagrams. They report a text/non-text classification accuracy of 99%. However, in practice, this figure is useful as a theoretical bound as their dataset mostly included figures with text in Manhattan layouts. Other methods have since been investigated, notably Spatio-temporal heuristics [184], Probabilistic Feature Grammars coupled with Genetic Programming [28], Hidden Markov Models (HMMs) [25], Stochastic Context Free Grammars [146], Data Mining [160], AdaBoost [224] (with decision trees as a weak learner), Markov Random Fields [233] (with Support Vector Machines used for individual stroke classification, with an accuracy of 96%), Support Vector Machines with Spatio-Temporal Graphs [212], BLSTM Neural Networks [96] (Bidirectional

Long-Short Term Memory) . Zhou et al. [234], Matsushita et al. [137] and Mochida et al. [143] have adapted similar techniques for distinguishing text in Japanese from non-text strokes. Almost all of these methods include a stroke grouping or recognition step after the text/non-text classifier has labeled each ink stroke.

Bhat and Hammond [24] have developed a lightweight technique using a single feature measuring entropy based on Shannon's formula. An experimentally determined threshold is used to classify ink strokes as either text or non-text. Bhat and Hammond report a accuracy of 92% for their entropy-based technique but it is worthwhile to note that their test dataset consisted of examples of two unique diagrams collected from only seven users. More recently, Blagojevic et al. [26] have also investigated the entropy metric and found that in practice, the entropy based classifier yields roughly 83% accuracy. Additionally, Blagojevic et al. [27] have conducted extensive benchmarks experiments to determine the performance of data mining techniques for the text/non-text classification task. Using 114 ink features for training, they found that the LogitBoost[7] [69] and LADTree[8] [90] classifiers yield the best accuracy for this task, at roughly 97%. In similar vein, Delaye et al. [56, 55] have developed methods based on Conditional Random Fields which seem to corroborate the findings of Blagojevic et al. [27] with overall classification accuracies at around 93.5% and 97%.

2.1.4 Symbol Recognition

Methods for recognizing sketched symbols can be categorized in several ways such as by type of symbols recognized (unistroke[9] or multistroke symbols), by target domain (handwriting, chemical drawings, math recognition, etc), by granularity (low-level, high-level, segmentation, etc), by

[7] Additive Logistic Regression.
[8] Alternating decision tree using the LogitBoost strategy.
[9] Methods for recognizing unistroke symbols are sometimes referred to as gesture recognition methods.

choice of representation (structural methods vs image-based methods vs stroke or trajectory based methods) or by type of classification method used (machine learning, heuristics, ensembles, etc). Of these, listing by symbol type fails to capture the variety and nuances of approaches used for symbol recognition (by reducing methods to an either/or choice). Listing methods by domain or by granularity can make it confusing to identify trends. Similarly, categorization by choice is representation is too limited to account for hybrid approaches such as recognizer fusion or ensemble based methods. Therefore, we prefer to categorize symbol recognition methods by choice of classification method:

2.1.4.1 Heuristic or Rule-based Methods

Heuristic approaches attempt to apply hand-coded rulesets to determine domain classifications for ink strokes. Early work in this domain includes CALI [66], a multistroke recognition algorithm for 12 different symbols. CALI was presented in both heuristic and machine learning forms[10], with its heuristic formulation providing the better recognition accuracy at around 95%. Yu and Cai [227] also created a domain independent, low-level unistroke recognizer. A popular recent system is the PaleoSketch [161], which describes rules for recognizing and beautifying 9 low-level unistroke primitives with high accuracy (over 98%). The PaleoSketch system was modified to use a Neural Network based classifier [84, 162] and a novel clustering algorithm, in order the support the recognition of multistroke symbols.

Hammond and Davis have also developed LADDER [83] which is a shape description language that allows a user to specify sketch recognition rules in a domain independent manner. Users can describe sketch primitives and some high level semantics for individual domains in

[10]The machine learning flavor of CALI was tested with K-Nearest Neighbor, Naive Bayes, and Decision Tree formulations, with the Naive Bayes providing best accuracy at around 93.5%.

text form, which is then used to generate domain specific recognizers. LADDER is a powerful and expressive system, yet it slows down considerably for sketches containing a large number of strokes. This is fundamentally a design problem, as it is performing close to exhaustive search for grouping primitive elements (which is known to be exponential [124, 131, 132, 177]). Hammond and Davis [85] present an interesting approach based on indexing recognized primitives that helps reduce the size of the search space.

2.1.4.2 Graph-based Methods

Graph-based recognizers usually attempt to model ink strokes in an input sketch by constructing graphs with ink strokes as labeled vertices and assign edges based on some interesting relationships between ink strokes. Lee et al. [124] present a good overview of graph-based recognition techniques. Graph-based methods have been applied to multistroke symbol recognition [34, 49, 80, 112, 124, 135, 225], handwritten numerals [43], and recognition of stick figures [132] with varying degrees of success. Symbol recognition with graph-based techniques requires identifying subsets of input graphs that fit the description of domain symbols. Framing sketch recognition in this way poses the problem of stroke grouping and multistroke symbol recognition as a Subgraph Isomorphism problem (which is NP-Complete) [124, 131, 132, 177]. Hall, Pomm and Widmayer [80] present a very nice formalism that frames sketch recognition as a combinatorial optimization problem, reducing it to the well-known Hamiltonian Circuit problem (which is known to be NP-Hard).

Graph-based methods are interesting for two reasons. First, they model recognition in a manner that can be visualized and understood intuitively[11]. Second, graph theoretic constructions

[11] It is difficult to visualize the operation of machine learning and statistical approaches such as Support Vector Machines and Hidden Markov Models in a similar manner.

enable researchers to reason about the overall difficulty of sketch recognition, showing that the stroke grouping problem (which is an integral part of the sketch recognition pipeline) is at least NP-Complete, if not NP-Hard. In practice, most sketch recognition systems include workarounds that avoid exhaustive search through all subsets of ink strokes by either limiting ways in which symbols can be drawn or by approximation.

2.1.4.3 Statistical, Template-based and Machine Learning Methods

Template matching methods use the notion of visual similarity to train simple classification algorithms that use various distance metrics (e.g. Hausdorff distance, Yule Coefficient, and Tanimoto Coefficient) to determine the label for new symbols. A very popular and lightweight template matching algorithm is the $1 Recognizer [214] which uses a 'path distance' to classify unistroke symbols with very good accuracy (reported accuracy of 99% with 5 training samples per symbol). The $1 recognizer has been extended to yield the $N recognizer [10] (generalization of $1 to multistroke symbols), Protractor[126] (Nearest Neighbor classifier with angular distance measure) and the $N-Protractor [11] systems. Similarly, Ouyang and Davis [155] use a deformable template matching (using an image-based stroke representation in conjunction with a custom image deformation model) to recognize handwritten symbols, Powerpoint shapes and circuit diagrams with over 95% accuracy.

Statistical methods and traditional machine learning formulations have been another popular source of sketch recognition methods. Some representative examples include diagram recognition with neural networks [2, 71, 70, 84], multi-domain sketch understanding with Bayesian networks [6], circuit diagram recognition with Naive Bayes-based classifiers [72], a Bayesian classification approach using visual language models [185] for the SILK system [116], various recognition methods based on Hidden Markov Models (HMMs) [8, 16, 175, 177, 186], Conditional Random Fields [165, 223], Support Vector Machines (SVMs) [32, 102, 128, 147, 154, 222], Constellation Models [179], Zernike Moments [92], and AdaBoost-based Classifiers [118, 183]. Most of the machine learning methods described above seem to perform fairly well on their target domain or dataset, yielding accuracies in the 90-95% range.

However, comparing the results between different classification algorithms reveals instances where one algorithm performs better vs another. With this view, hybrid or ensemble methods that fusing the strengths of individual classifiers hold the promise for increased recognition performance (See Section 2.1.4.4). With respect to machine learning and data mining based methods, exhaustive feature libraries [160, 54] have been developed that can be used for classification algorithms on a variety of domains. One such library [160] lists 114 stroke-based features (categorized using grounded theory [192]). Blagojevic et al. [26] have examined strategies for automatically selecting subsets of these features using the WEKA framework [81], and have also used it in conjunction with Rubine's classifier [169] to establish its effectiveness. This same feature library has been used to identify good text/non-text classifiers [27] (mentioned in Section 2.1.3.1). Stahovich et al. [190] have also developed a 2-step method to group and recognize multistroke symbols in logic diagrams. They first use a single stroke classifier (AdaBoost with c4.5 decision trees as weak learners) to assign a class (gate, wire, text) to each stroke (features adapted from [160]. For clustering, they compare two methods based on threshold-based and AdaBoost-based grouping as a second step. Delaye et al. [54] have constructed a competing feature library, called the HBF49 feature set, consisting of 49 different ink features that have been tested using a Support Vector Machine (SVM) classifier, with good accuracy.

2.1.4.4 Hybrid Methods

Hybrid sketch recognition methods aim to combine the strengths of different techniques, in order to yield an improvement in overall accuracy. Cates [36] presents a terminology for describing multiple sketch representations as spatial, temporal and conceptual, and describes methods that combine multiple representations for recognition to yield better accuracy than a single representation. Kara and Stahovich [105] describe an ensemble-based approach that combines four template matching classifiers[12] (input is represented as a pixel array). The method for combining these template matching classifiers is based on [110]), with a polar-coordinate based representation to mitigate issues with rotational invariance. Using this method, Kara and Stahovich report a recognition accuracy of around 98% on a dataset containing graphic symbols and an accuracy of around 95% on a dataset containing digits.

Tumen et al. [201] present another method for achieving feature-level fusion between several image-based methods, including Zernike Moments, IDM (Image Deformation Models), and Shape Context. Additionally, they introduce the Extended Trace Transform (ETT) as novel method for calculating features from sketch data. Their results show that ETT results in more compact and representative feature vectors. They describe methods for extracting and fusing features, as well as methods for identifying relative importance. Additionally, they tested their extracted features using 10 different classifiers and demonstrated that using combined features from different methods yields a significant boost in recognition accuracy.

In a similar vein, Arandjelovic and Sezgin [15] describe a method for fusing an image-based sketch recognition method (Zernike Moments) with a recognition method usually employed for temporal representations of sketches (Hidden Markov Models). Three methods for fusing the results of both classifiers were tested: Naive Bayes, Mean Combination Rule [110], and Dempster-

[12] Distance metrics used are Hausdorff Distance [170], Modified Hausdorff Distance [60], Tanimoto Coefficient [65] and Yule Coefficient [200].

Shafer [220]. Additionally, the outputs of the HMM and Zernike moment based classifiers was also used to train Support Vector Machines in 'One-against-All' and 'One-against-One' configurations. Their findings indicate that the fused classifiers yield better performance than single classifiers.

2.1.4.5 Miscellaneous Methods

In addition to the methods described the previous sections, Agent-based recognition frameworks have also been used for sketch recognition [63, 130, 35].

2.2 Pen-based Systems

2.2.1 Pen-based Systems for Physics

As stated earlier, our overarching goal is to construct pen-based intelligent tutors for physics, particularly classical mechanics. While no sketch-based tutoring system of sufficient capability exists for this target domain, several researchers have made strides toward one in recent years. Alvarado [5], Oltmans [153] and Kara [104] have constructed systems for sketch understanding in the domains of computer aided design, mechanical design and vibratory systems. These tools can recognize and animate relevant diagrams but none of these allow users to write down mathematics that can influence animation. Scott and Davis [174] have constructed PhysInk that allows users to specify physically correct behaviors via sketching techniques.

The MathPad^2 [120] system is an interesting tool because (1) it is domain independent (2) allows for unconstrained sketch input and (3) allows users to explicitly associate equations with diagrams to guide animation behavior. However, MathPad^2 is difficult to use as a tutoring system, because it includes no domain knowledge and lacks deep reasoning mechanisms about sketched solutions. Additionally, users must specify all aspects of animation through hand-written mathematics. CogSketch [67] is another sketch-based system that emphasizes conceptual labeling of sketches by its users. While CogSketch permits unconstrained sketch input and aims to aid development of sketch-based educational software, its focus is very broad, ranging from investigating cognitive aspects to running simulations. This makes it unsuitable as a tutoring system, as it lacks the focus on understanding student solutions automatically and providing visual feedback.

Newton's Pen [123], Newton's Pen II [121], and Mechanix [17] are important sketch-based systems targeted at the domain of Statics. Newton's Pen [123] and Newton's Pen II [121] focus on drawing free body diagrams and writing equilibrium equations. Both do not use animation or allow unconstrained sketch input. Instead, users must adhere to a particular workflow while sketching diagrams. Additionally, diagrams and equations must be drawn/written in separate pre-defined areas. Newton's Pen II [121] also describes an interesting method based on Hidden Markov Models (HMM) to correct errors in recognized equations. Mechanix [17] is another tutoring system for statics that allows students to sketch truss and free-body diagrams. Mechanix is a deployed system and has been thoroughly user-tested, with encouraging results. It allows instructors to easily author and assign new problems to students via a web-based interface. Students solve assigned problems via a sketch-based interface. If a diagram is incorrect, the student can make corrections until it matches the instructor's correct solution [64]. While Mechanix allows students to sketch drawings in an unconstrained manner, it is focused on a very narrow type of diagram, and does not support handwriting recognition. Students must enter the values for different labeled forces via a text box. While mechanix provides useful feedback and hints, it does not provide any animation of sketched diagrams.

We have previously made strides toward our stated goals with a prototype physics tutoring system titled PhysicsBook [40], which is an extension of our older work [41, 42]. Our initial prototype [42] was a simple proof of concept that could recognize unconstrained written solutions containing both diagrams and mathematics and used a customized physics engine to animate diagrams. Users could associate their answers with components of a diagram to guide animation behavior. This was extended by incorporating sketch beautification methods and additional animation capabilities [41]. At this stage, the animation parameters were limited to physics quantities directly related to motion such as the use of position, velocity, acceleration and force variables defined as functions of time [41]. By providing support for diagram annotations such as arrows and dotted lines, and by using realtime data transformations, we constructed PhysicsBook [40], which was able to provide additional animation support for select cases of pulley systems, work done, kinetic and gravitational potential energy.

In summary, very few existing sketch-based systems for physics provide animation capabilities or allow for unconstrained input. Additionally, most are focused on particular sub-domains of physics. We have made initial strides toward these goals with our previous prototype PhysicsBook, but our approach has so far been limited by reliance on a single monolithic physics engine for animation. Additionally, we have focused exclusively on the answer step in the solution, not using any of the information provided in the problem statement or the rest of the solution. In this paper, we describe an extensible framework that overcomes the shortcomings of our previous approach, in order to better meet our stated goals.

2.2.2 Pen-based tools for other Domains

The previous section highlighted a few prominent examples of recent pen-based tools constructed to augment some aspect of physics education. However, the domain of pen-based interaction is quite old, beginning with Sutherland's seminal work [193]. In addition to physics, researchers have also leveraged pen-based interaction for educational tools targeting other STEM (Science, Technology, Engineering and Mathematics) disciplines. Some recent examples include pen-based tools for Circuit Analysis [53, 72, 229], Chemistry [156, 198], Anatomy [164], Mathematics [14, 31, 39, 48, 98, 120], and Computer Science [33, 103, 125, 159, 228].

There has also been significant work on using pen-based interaction for other creative and design tasks. Table 2.1 provides a representative listing of such tools from the last two decades and briefly describes their intended use.

Table 2.1: Listing of Pen-based Systems by Year

Year	**Name**	**Purpose**
1994	PerSketch [172]	Perceptual Image Editor
1995	Silk [116]	User Interface Design
1996	Sketch [232]	3D Modeling
1996	Electronic Cocktail Napkin [77]	Design Diagrams
1996	Gross & Do [78]	Diagramming Tool based on Electronic Cocktail Napkin [77]
1997	Pegasus [93]	Rapid Geometric Design
1997	QuickSet [46]	Multimodal Training Tool
1998	Music Notepad [68]	Musical Score Creation
1999	Teddy [94]	3D Modeling
2000	Harold [45]	3D Modeling
2000	SATIN [91]	Plush Toy Design
2002	Tahuti [82]	UML Class Diagrams
2003	Scanscribe [171]	Perceptual Text and Graphics Editor

Continued on next page

Year	Name	Purpose
2003	Denim [149]	Website Design
2003	SketchPoint [127]	Informal Presentations
2004	GIDeS++ [163]	3D Modeling
2005	Okabe et al. [152]	Tree Modeling
2005	HHReco [92]	Sketching and Beautifying Symbols On Power-point Slides
2005	Yang et al. [221]	3D Modeling
2006	Motion Doodles [199]	Animation
2007	Plushie [144]	Plush Toy Design
2007	MaramSketch [79]	Software Design & Diagramming
2007	SketchUML [166]	UML Diagrams
2008	AgentSketch [35]	UML Diagrams
2008	Matisse [23]	3D Modeling
2008	K-Sketch [51]	Animation
2008	Lineogrammer [231]	Diagramming Tool with Beautification Support
2008	ILoveSketch [18]	3D Modeling
2009	EverybodyLovesSketch [19]	3D Modeling
2010	Inkus [133]	Business Process Models
2010	ICanDraw [58]	Sketching Faces
2010	LAMPS [196]	Teaching Mandarin Phonetic Symbols
2012	QuickDraw [39]	Geometry Diagrams
2012	Vignette [108]	Artwork and Texturing

Continued on next page

Year	Name	Purpose
2012	Concepture [59]	Regular language based framework for Repetitive Drawings
2012	Dragimation [210]	Technique for Manipulating Animations
2012	Sketch It, Make It(SIMI) [99]	Design tool for fabrication
2013	SimSketch [29]	Simulation
2014	Draco [107]	Animation
2014	EulerSketch [52]	Euler Diagrams
2014	PatternSketch [38]	Structured Drawings

2.3 Traditional Tutoring Systems

VanLehn [206] describes Tutoring Systems as:

"...having two loops. The outer loop executes once for each task, where a task usually consists of solving a complex, multi-step problem. The inner loop executes once for each step taken by the student in the solution of a task. The inner loop can give feedback and hints on each step. The inner loop can also assess the students evolving competence and update a student model, which is used by the outer loop to select a next task that is appropriate for the student."

This is, perhaps, the simplest and most intuitive way to describe the function of an intelligent tutoring system. VanLehn [206] presents an excellent introduction to the domain of intelligent tutoring systems and summarizes different approaches for interacting with students, providing coarse and fine grained feedback and for student modeling using cognitive methods. Additionally, a listing of tutoring systems is also provided for laymen. Representative examples of Intelligent

Tutoring Systems presented in [206] include Steve [100, 168], Andes [207], Algebra Cognitive Tutor [9], AutoTutor [74, 75, 76], SQL-Tutor [141, 142].

Of these, Steve [100] is an animated agent to aid training in physical procedural tasks, in a virtual reality environment. Steve includes mechanisms for monitoring students and providing proactive assistance, and is also capable of demonstrating entire tasks. Andes [207] is a physics tutoring system, which provides step by step guidance (by pre-computing solution steps) in problem solving, and has been extensively tested in the classroom. AutoTutor [74, 75, 76] is another tutoring system that emphasizes natural language dialogue with the student for feedback. AutoTutor uses Latent Semantic Analysis (LSA) [115] to compare the semantic content of a student's step with encoded learning events to generate feedback. It has been shown to correlate with improved learning in the domains of Newtonian physics and computer literacy. SQL-Tutor [141, 142] is a constraint-based tutoring system for teaching the basics of the SQL query language.

The Algebra Cognitive Tutor [9] belongs to the family of Cognitive Tutors, which are based around the idea that instruction and interventions by the tutoring system should be designed with reference to a cognitive model. Cognitive tutors [111] have been developed for teaching LISP, geometry and algebra, with the Algebra Cognitive Tutor being one of the most popular and well-known systems. More recently, Anthony et al. [14] have coupled the ideas behind Cognitive Tutors with a pen-based interaction methodology to create a tutoring system for algebraic equation solving. Their system uses the math parser behind the Freehand Formula Entry System (FFES) [189] for recognizing handwritten input. The system presented by Anthony et al. [14] is nice because it 1) presents a unified view and workflow of a pen-based tutoring system 2) is grounded in sound cognitive principles and 3) is well tested by students. However, it does not utilize animation, and also requires the students to type in their final answer, to mitigate recognition errors. We believe these two factors detract somewhat from the overall natural interaction metaphor provided by using a pen-based interface.

Tutoring systems usually require a student model to measure learning progress, and to determine if intervention and/or feedback is needed. Bayesian Networks are very popular for this purpose [47, 57, 73, 138]. Traditionally, tutoring systems have relied on using WIMP (Windows, Icons, Menus, Pointers) interfaces, which impose cognitive load [194] on the student. Ideally, the process of learning new concepts in an education setting should not be synonymous with learning a new interface. With pen-based interfaces, students can write their solutions in a natural manner. While pen-based systems may also involve some learning curve, we conjecture that the cognitive load is significantly less when using pen-based interaction.

Tutoring systems in general must utilize some understanding of a given problem and its solution, in order to provide feedback. An advantage of pen-based tutoring systems is the promise of a more natural mode of interaction. Compared to WIMP based interfaces, they can also incorporate sketch understanding techniques to provide better feedback/animation mechanisms.

2.4 Commercial Tools and Teaching Aids

Lastly, several commercial tools are also available that let users construct animations for physics concepts. Representative examples include Algodoo [4], Working Model 2D [218], Newton's Playground [150] and Crayon Physics [50]. Such tools can create animations but they do not let students work out a given problem and then directly associate the answer with a diagram to do the animation. In such tools, a student would solve the problem in a notebook and then have to separately reconstruct the diagram using the tool to perform the animation. Other commercial tools such as Maple and Matlab allow experienced users to construct physics animations programmatic ally, but are difficult and complicated to use.

CHAPTER 3
PROTOTYPE PEN-BASED PHYSICS TUTORING SYSTEM

Figure 3.1 depicts the high level architecture of our prototype ITS which is logically divided into four modules: User Interface, Recognition, Animation Runtime and Solution Checking.

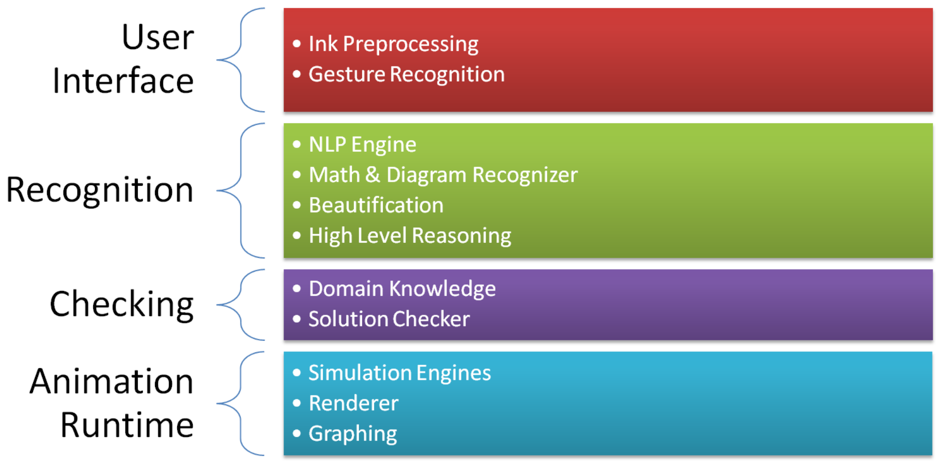


Figure 3.1: System Architecture

In our idealized workflow, students load up a problem statement and solve it in a natural manner. After completing the solution, they can tell the system to recognize and animate any diagrams in the solution, view graphs of interesting quantities, or ask for verification. Our prototype system requires is aimed purely at students. We aim to provide students with powerful animation tools that they can use to experiment and see physics concepts in action. The key feature of our prototype ITS is to use the student's own answer to a problem and their sketched diagrams to generate an animation. Our prototype ITS requires no input/instrumentation from an instructor and instead relies on natural language processing techniques to infer contextual information about each problem.

Animation is a core mechanism in our prototype ITS. It can be difficult to provide animation support for all types of physics problems due to the requirement of modeling the vast number of physics concepts. We have designed our animation system to allow a student to model elements of diagrams via a simple set of shapes (circles and polygons) that may be attached to pulleys, wires or springs. Shapes can be free falling or constrained to move along a surface (represented by a line segment or a polyline). Initial conditions can be indicated via mathematical expressions and supplemented with annotations such as arrows and dotted lines. Dotted lines and intervals (parallel dotted lines) also enable students to indicate a particular event in the simulation or to define a displacement range. We demonstrate that these behaviors provide good animation support for a variety of physics problems in kinematics.

3.1 User Interface

The key feature of our prototype ITS is to use a student's answer to a given problem to animate the diagram that was drawn as part of the solution. We allow students to write in an unconstrained manner and place no explicit restrictions on how or what to draw. The user interface (Shown in Figure 3.2) is simple with a big writing area and an always visible toolbar for system functions. The toolbar contains the text of the problem and allows a user to trigger recognition, animation, and solution checking. A stylus to write down the solution to a given problem on the writing area. We impose no constraints on the user's input. For editing, the 'Scribble' gesture can be used to erase parts of the solution which can then be rewritten correctly. When the writing area is filled, users can scroll down for more space. The system menu can also be used to save and load solutions (along with problem statements) from disk.

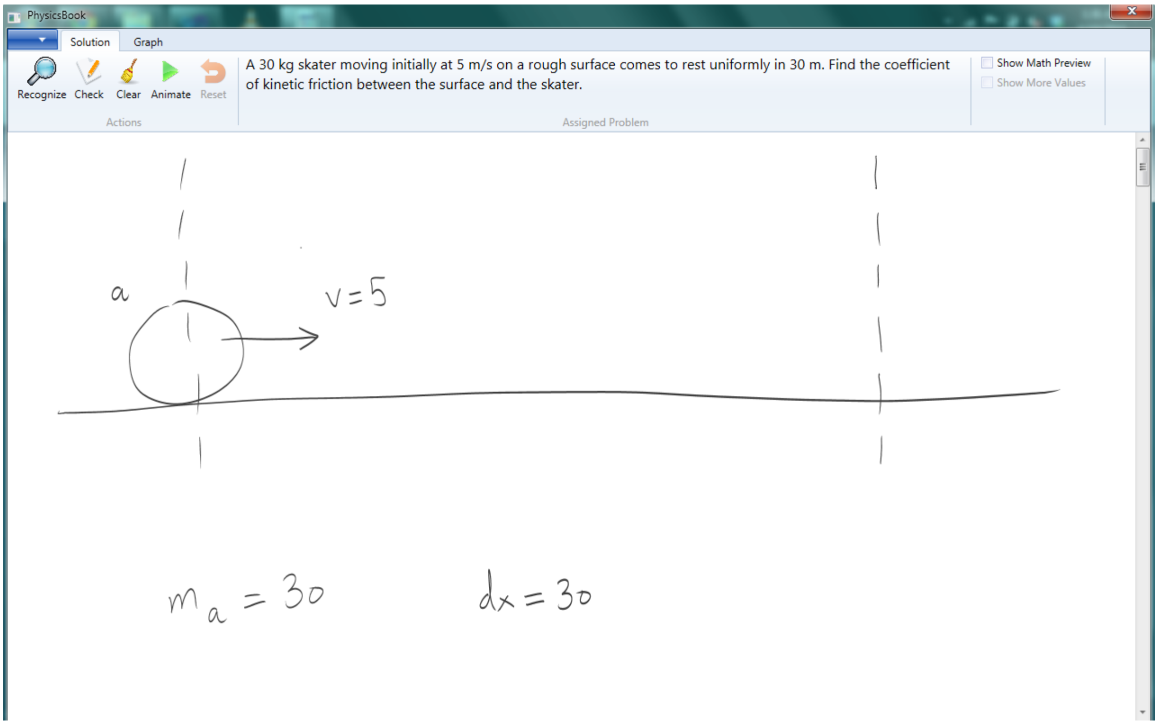


Figure 3.2: A screenshot of our prototype ITS

Once a solution is complete, the ‘Recognize’ button can be used to trigger sketch recognition. This interaction method is chosen because earlier studies by [209] have indicated that users prefer to trigger sketch recognition after they finish solving a problem. However, experiments by [30] have found that users prefer realtime recognition feedback for mathematics when they are expected to write more than one expression. To balance both of these concerns, the math recognition engine in our prototype is continuously active and provides realtime feedback as the user is writing. If needed, this behavior can be disabled via a checkbox on the system menu.

During the recognition phase, the solution is parsed to produce 1) physics simulator(s) for the diagram, and 2) a series of ordered mathematical steps. During recognition, our prototype uses spatial proximity and label matching to implicitly associate initial conditions written by the user with recognized diagram elements. Implicitly associated equations can be viewed by hovering the stylus over recognized diagram elements. Any associations missed during recognition can be explicitly indicated by the user. To make an explicit association, users select mathematical expression(s) with the ‘Lasso’ gesture and use the ‘Tap’ gesture to link them with a diagram element.

Hitting the ‘Animate’ button triggers the animation runtime, which uses domain-specific physics simulator(s) to animate the recognized diagram. Our prototype also generates a list of ordered solution steps during recognition, which can be used to check if the solution is correct, according to the rules of physics. To trigger solution checking, a user hits the ’Check’ button on the system menu, which displays the results of analysis on a separate window. The system menu can also be used to view realtime graphs of physics quantities during animation.

3.2 Ink Stroke Preprocessing

A written solution in our prototype system is a collection of digital ink strokes, each of which is a sequence of 2D points. After each ink stroke is completed, it is passed through a preprocessing step that performs the following operations to prepare it for eventual recognition:

1. Remove hooks near stroke endpoints using method described in [119]
2. Filter stroke to remove kinks and to ensure equally spaced points using method described in [219]
3. Compute bounding box, centroid and average radius
4. Enumerate cusps using the IStraw algorithm [219]
5. Count self-intersections
6. Check if this stroke is a gesture (Tap, Lasso, or Scribble)

If the stroke denotes a gesture, the gesture's effect is applied and the stroke is discarded. Each non-gesture ink stroke is immediately added to the math recognizer after preprocessing which yields realtime recognition results. Realtime mathematics recognition also allows us to speed up the recognition phase which can then be customized to only recognize diagram elements.

CHAPTER 4
UNDERSTANDING HANDWRITTEN PHYSICS SOLUTIONS

A written solution to a given problem needs to be parsed in order to construct a computational model that can be animated and checked to provide feedback. As we focus purely on the student experience, our prototype ITS does not provide support for an instructor/teacher to author individual problems for students. Therefore, important information about each problem (such as its domain, initial conditions, objective) needs to be inferred from the problem statement. Each solution is logically divided into two parts: an annotated diagram and a series of mathematical steps, which must be disambiguated by a recognition system. To complicate matters, our mathematics recognition engine is continuously active, incorporating each new ink stroke and providing real-time recognition feedback. Taken together, these form three distinct modes of input which must be considered in unison to yield a good animation.

Figure 4.1 depicts the workflow of our sketch recognition system. The three inputs to the system are: 1) the text of the current problem, 2) the entire handwritten solution and 3) the results of the math recognizer. With the problem statement, we use a heuristic-based approach built around the SharpNLP framework [180] and WordNet [139, 62] to infer the problem domain (e.g. momentum, linear motion, etc), to extract the initial conditions for objects in the described scenario, and to identify the objective of the question (what quantity is the student asked to work out?). The solution is parsed to identify low-level diagram elements which are then linked implicitly with initial conditions via a set of heuristics. The ink for the recognized diagram is removed from the math recognizer which outputs a list of ordered solution steps. Finally, information gleaned from the text of the problem is used to identify simulation engine(s) which are then added to the animation runtime , along with beautified diagram elements.

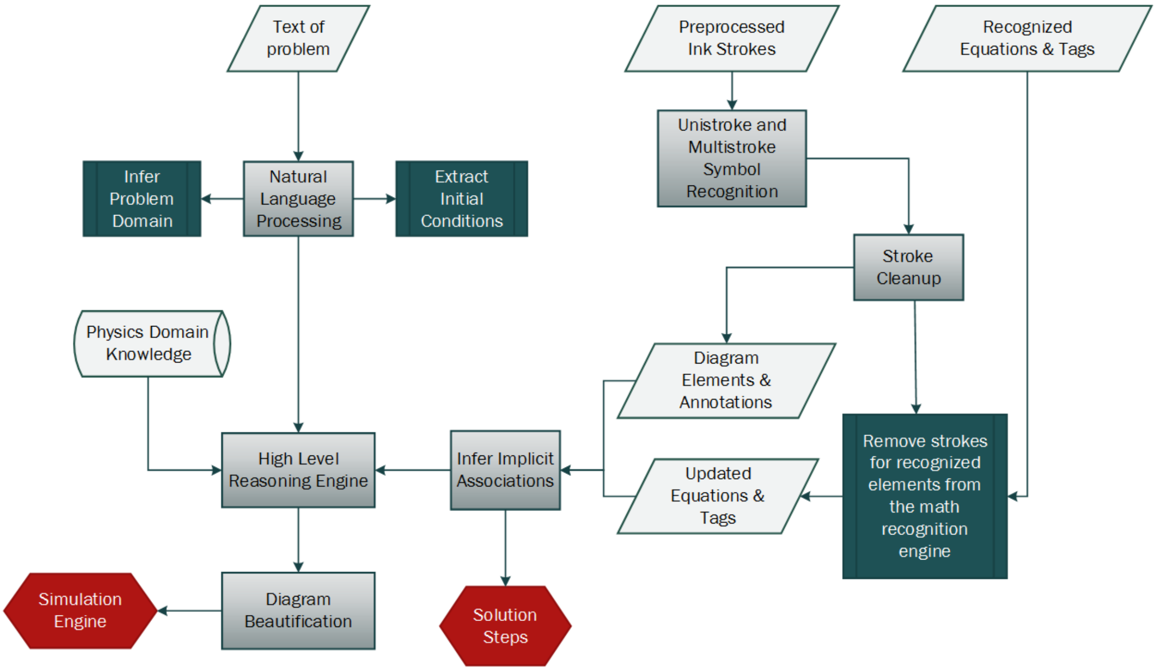


Figure 4.1: A problem statement, a solution, and recognized mathematics are inputs to the recognition system. These are used to 1) recognize and beautify the diagram, 2) identify and set up simulation engine(s) for the student's diagram and 3) output a list of ordered solution steps.

4.1 Natural Language Processing

The input to the NLP subsystem is the text of the problem. We use the SharpNLP framework [180] to parse the problem statement into sentences, which are in turn parsed into tagged chunks. Chunks corresponding to noun phrases are clustered together to yield fragments. Using regular expressions, we filter each fragment to identify patterns that contain names of physical quantities followed by a number followed by physics units. Each such fragment denotes a possible initial condition or information about the scenario. Fragments that do not conform to this pattern are compared with a known list of names for physics objects to construct a list of entities in the diagram. We have examined a set of 50 physics problems [1] selected at random from Young's University Physics 13th Ed [226] to compile this list of common object names:

Names = { *circle, ball, sphere, orange, skater, baseball, puck, skaters, truck, car, monkey, square, crate, box, bananas, supertanker, sled, wagon, rock, building, floor, table, surface, ice, pond, ramp, horizontal, spring, wire, pulley* }.

Our prototype contains four simulation engines for free fall kinematics, friction and sliding-contact problems, equilibrium problems and for momentum problems. Based on our analysis of 50 representative problems, we have identified a set of key words that help identify each problem's domain:

Freefall Kinematics {*moving, falling, spring, dives, falls, hits, dropped, pulley, hangs, rope* }.

Friction { *friction, rough, surface, rests, rest, slides, downhill, inclined plane, stops, ramp, inclined, pulley, hangs, rope* }.

Momentum { *momentum, impulse, collision, collides, elastic, head-on, rest* }.

Equilibrium { *tension, rope, hangs, wire, chain, suspended, weights, break*}.

Given the text of a new physics problem, we simply count all the occurrences of each domain's keywords in the statement to get a score for each domain. The highest scoring domain is picked as the primary candidate. The secondary options are passed to the high level reasoning engine along with the primary candidate. If two domains acquire the same score, we use the following precedence order to break the tie: Free fall Kinematics, Friction, Momentum, and Equilibrium to decide the primary domain. There is a good reason for considering secondary candidate

[1] The selected problems are listed in Table 9.1 in Chapter 9.

simulators. In some cases (See Section 8.4.2), it is possible that the primary simulator can only animate part of the recognized diagram, forcing the animation runtime to use the secondary simulator to animate the remaining parts of the diagram.

We analyzed the statements of 50 physics problems (See Table 9.1) to measure the accuracy for this keyword-based recognition scheme. Using this method, the primary domain was correctly identified for 78% of the problems in Table 9.1. This is not a discouraging result, because some of the test problems are out of scope for our prototype system. For the sample problems presented in Chapter 8, the domain is correctly identified every time. A confusion matrix for the problem statements is presented in Table 4.1:

Table 4.1: Confusion Matrix for NLP Heuristics

	Kinematics	Equilibrium	Friction	Momentum
Kinematics (13 problems)	9	1	2	1
Equilibrium (11 problems)	0	8	3	0
Friction (18 problems)	2	0	16	0
Momentum (8 problems)	1	0	1	6

4.2 Mathematics Recognition Engine

Our math recognition system is built around the StarPad 0.1.3 system [191], which in turn is based on techniques presented in MathPaper [230]. This system maintains a list of ink strokes which are analyzed to yield realtime recognition results. The StarPad system uses rule-based symbol recognition with the Microsoft Ink Analyzer as a fallback for unknown symbols and is capable of recognizing and updating multiple mathematical expressions simultaneously, and also provides support for limited forms of matrices and algorithmic notation. The MathPaper [230] system which underlies StarPad was tested in a user evaluation and received favorable feedback, yet recognition accuracy numbers were not recorded. However, the recognition performance appears to be quite robust given its use in several recent systems [125, 103, 48].

The Math Recognizer in our prototype is always active and maintains a list of ink strokes that are analyzed to provide realtime recognition results. Each ink stroke, after it is completed and preprocessed, is immediately added to this list, causing the recognition results to be updated. We manipulate the recognition results to 1) generate a set of alphanumeric tags, which are numbers and/or strings of characters in isolation, and can denote either labels for diagram elements or the value of some quantity. 2) Mathematical equations recognized by StarPad are split up into one of two categories: a constant expression or a variable expression. Constant expressions have a single variable on the left hand side of the equation and either an real number or fraction on the right hand side. Variable expressions are equations that contain one or more variables in the right hand side.

4.3 Diagram Recognition

Our prototype ITS supports the animation of diagrams from different domains of physics. Therefore, physics solutions are processed in two separate stages, initially using domain-agnostic heuristics to detect diagram elements and group related annotations in a bottom-up manner. This step explicitly separates the diagram from the mathematical steps in the solution, and also performs some initial beautification. The second recognition pass works top-down, assigning meaning and context to the recognized diagram by using physics domain knowledge. The domain of the problem is inferred from the text by our NLP engine. The recognized diagram elements are beautified and added to the animation runtime. Figure 4.2 shows the steps in our diagram recognition pipeline.

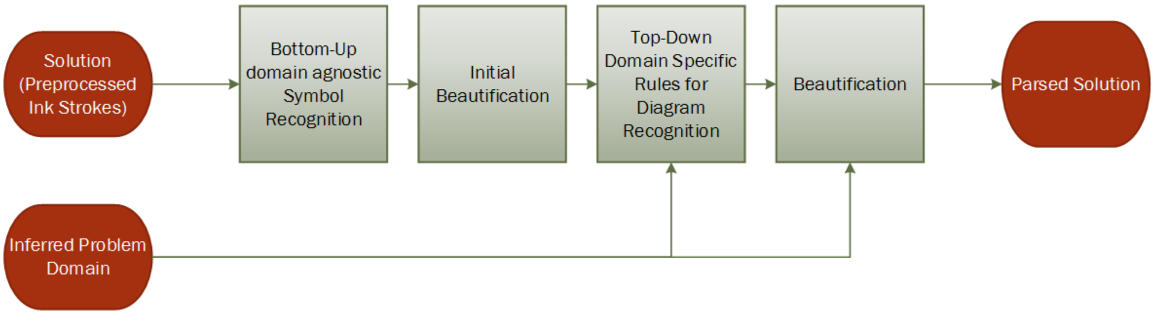


Figure 4.2: An overview of the Diagram Recognition Pipeline in our prototype ITS.

4.3.1 Text/Non-Text Division Strategy

It should be noted that we do not use stroke-based text/non-text divider strategies (See Section 2.1.3.1. While existing methods for text/non-text stroke division can promise very robust performance (up to 97.5% [27]), they have some shortcomings. The best performance in this regard is usually achieved via machine learning algorithms which require training from large datasets of labeled examples. Additionally, while the time cost of classifying each ink stroke as text or non-text is quite fast (usually under a second), the computational cost of training the best performing classification algorithms can be quite prohibitive [27]. Rule-based methods have also been developed for this purpose [24] but they do not perform as well as machine learning approaches. At the current stage of our system, we do not have a large dataset of labeled physics problems to train a good text/non-text classifier.

Our solution is to apply the unistroke and multistroke recognizers to the entire solution in a particular order (Shown in Figure 4.3). This approach yields a considerable number of false positives (FP). After each recognizer is finished, we apply a threshold to the size of its results to prune the false positives. This method works quite well at this stage of our system but eventually, we may need to incorporate a more robust text/non-text divider strategy.

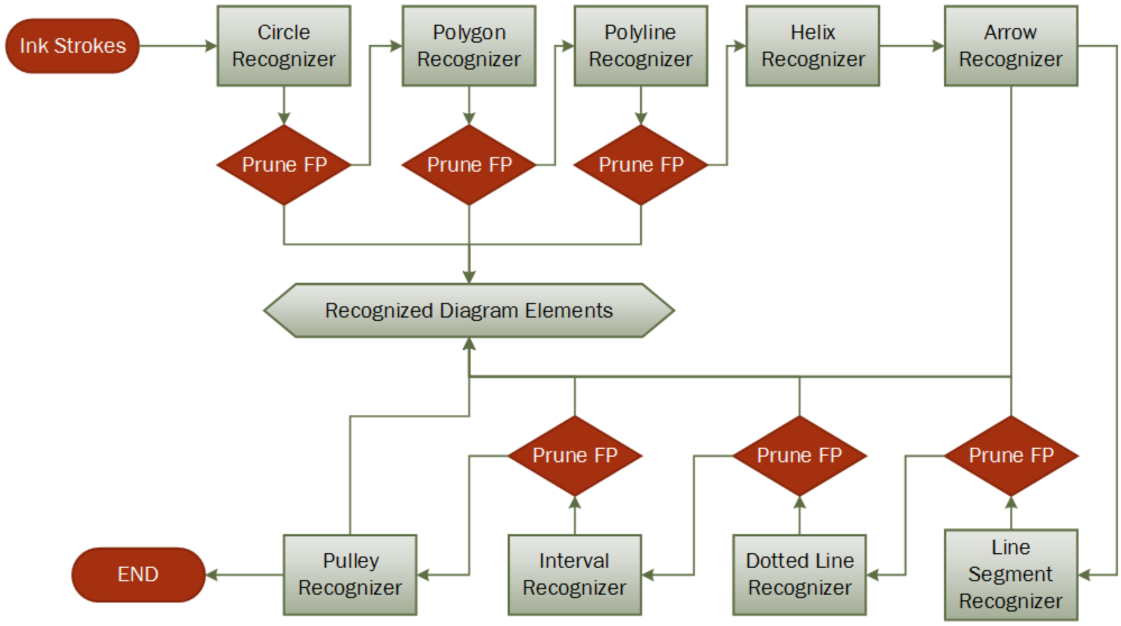
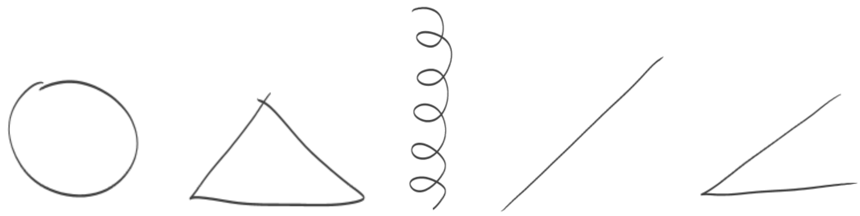


Figure 4.3: Bottom-up Sketch Recognition Workflow showing the order in which low-level recognizers are applied. After each recognizer is finished, a pruning step removes false positives (FP) in the recognition results.

4.3.2 Bottom-up Recognition Phase

During the first (bottom-up) recognition step, domain-agnostic recognizers work in a bottom-up manner to identify unistroke (circles, polygons, polylines, helixes, and line segments) and multi-stroke diagram elements (arrows, dotted lines, intervals and pulleys). Figure 4.3 shows the order in which low-level recognizers are applied to the solution. This order of the recognizers is chosen to enable a feed-forward architecture where initial results are used in conjunction with raw ink data for succeeding recognizers. Each recognizer works on the entire set of ink strokes. Recognition results are pruned for false positives (See Section 4.3.4 for details), and the ink strokes corresponding to successful recognition results are labeled. Unlabeled strokes and results from earlier recognizers are passed onto the next recognizer in the pipeline. Section 4.3.4 gives details of heuristics used by each recognizer in Figure 4.3.



(a) Circle (b) Polygon (c) Helix (d) Line Segment (e) Polyline

Figure 4.4: Examples of unistroke diagram elements supported by our prototype ITS.

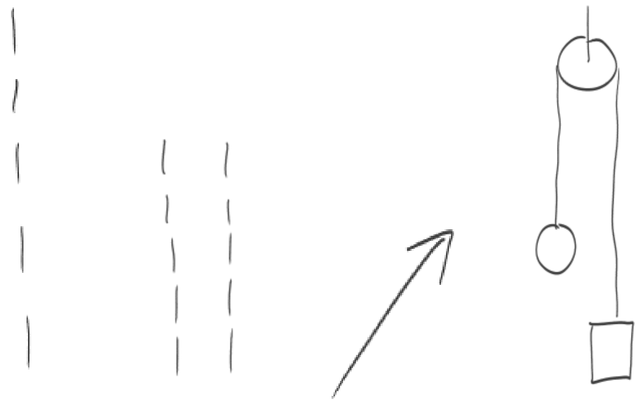
After unistroke and multistroke symbol recognition, the ink strokes corresponding to recognized elements are removed from the solution, updating the math recognition results which now only include text labels and equations. Some multistroke symbols (arrows, dotted lines, and intervals) denote annotations for diagram elements. Bottom-up recognition is finalized by inferring relationships between diagram elements, annotations, labels, and solution steps based on spatial proximity (clustering) and label matching (implicit association). As a result of the cleanup process, the ink strokes pertaining to the diagram are removed from the math recognizer, leaving only the solution steps, which are assigned a logical ordering based on their spatial position (for details, see Chapter 7).

4.3.3 Top-Down Recognition Phase

The second recognition pass assigns meaning to the recognized diagram and its annotations by leveraging the information extracted from the text of the problem. This functionality is logically contained in the 'High Level Reasoning Engine' that accepts the results of diagram recognition and information from the problem statement, and uses the two together to construct one or more domain-specific simulators for animation. Each simulator incorporates specific rules for physics quantities and unit conversions, allowing it to extract information from associated equations and properly apply initial conditions to recognized diagram elements. As an example, during this step, reasonable values are assigned for the physical properties of diagram elements, such as mass, weight, moment of inertia, etc.

4.3.4 Unistroke and Multistroke Recognition Heuristics

Our prototype supports the recognition of 5 unistroke diagram elements: Circles, Helixes, Line Segments, Polylines and Polygons (Examples shown in Figure 4.4) and 4 multistroke diagram elements: vertical and horizontal dotted lines, intervals, arrows, and pulleys (Examples shown in Figure 4.5). We do not support alternative classifications. Therefore, once each stroke is classified as one of the 5 unistroke elements, it is removed from both the pool of available strokes and from the math recognizer. Our heuristics for detecting these basic elements are adapted from [40, 39] and are summarized in Table 4.2.



(a) Dotted Line (b) Interval (c) Arrow (d) Pulley

Figure 4.5: Examples of multistroke diagram elements supported by our prototype ITS.

Table 4.2: Recognition Heuristics for Unistroke and Multistroke Symbols. The last column lists the rule for pruning false positives of each type.

Symbol	**Heuristic**	**Pruning Rule**
Line Segment	Two distant[2] cusps, at beginning and end of ink stroke. $Linearity < 0.05$, where $$Linearity = \|1.0 - \frac{\sum_{i=1}^{n-1} \|p_i, p_{i+1}\|}{\|p_1, p_n\|}\|$$	Length $>$ 20% of screen height

Continued on next page

[2] 'distant' implies the two points are not within touch distance. *Touch Distance* = 2.5% of screen width.

Symbol	Heuristic	Pruning Rule
Circle	Two nearby[3] cusps at the beginning and end of the ink stroke. Uniform curvature, where 'Curvature = standard deviation in distance from each stroke point to stroke centroid'. Threshold value of 'Curvature $<$ 2% of screen width' is used	Area $>$ 750 square pixels
Polygon	$\|Cusps\| > 3$, First and last cusp must be nearby. Stroke segment between each consecutive pair of cusps must be a line[4]	Same as Circle
Polyline	$\|Cusps\| > 2$, First and last cusp must be distant. Stroke segment between each set of cusps must be a line	Same as Line Segment
Helix	Two distant cusps, Number of self intersections ≥ 3	Same as Line Segment
Arrow	Two stroke corresponding to shaft and arrowhead. Arrow shaft must pass linearity test. Arrow head must have 3 cusps. Middle cusp of arrowhead must be near (within *Touch Distance*) of one of the endpoints of an arrow shaft	Shaft must be longer than 15% of screen width

Continued on next page

[3]Two points are 'nearby' if they are within *Touch Distance*.

[4]For polygons, pen strokes must finish at the initial point, yielding 4 points for a triangle (with the first and last vertex repeated).

Symbol	Heuristic	Pruning Rule
Dotted Line	Set of 4 or more ink strokes that each pass the linearity test, are vertical or horizontal, and have length between 1-6% of screen height. The entire set is clustered around a certain x- or y-value and does not deviate more than 1.5% of the screen width. Maximum inter-segment distance is approximately half of maximum segment length	Dotted lines must be longer than 30% of screen height
Interval	Set of parallel dotted lines with an overlapping projection along the major axis.	None required
Pulley	A recognized circle to act as hinge, two recognized line segments (tangent to hinge) to act as wires, and one small line segment (that passes linearity test) to act as anchor. Anchor must touch hinge center with one end. Methods adapted from QuickDraw [39]	None required

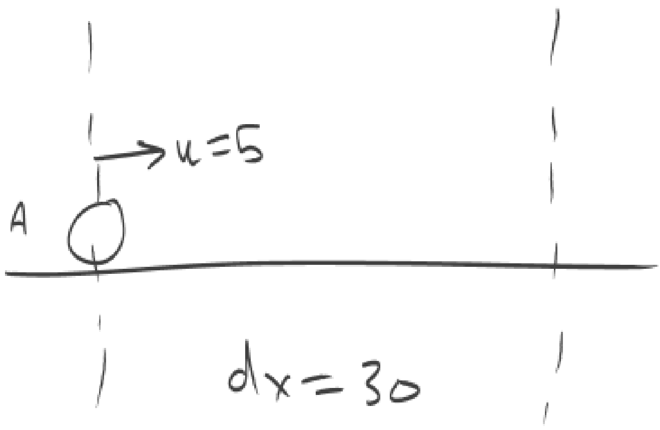


Figure 4.6: An example diagram of a ball resting on a surface, with a tag and two annotations (an arrow and an interval).

4.4 Clustering and Implicit Associations

With a list of recognized diagram elements, tags and equations, our prototype tries to infer implicit relationships between them. Some diagram elements (arrows, dotted lines, and intervals) are annotations for other elements and do not depict objects in an animation, e.g., arrows (annotated with equations) denote the direction and magnitude of vector quantities to be associated with some simulation object. Figure 4.6 presents an example where a tag and two annotations need to be clustered with the ball and surface. We use the following rules for clustering tags, equations and annotations with recognized diagram elements:

1. Associate Tags: User defined tags are associated with diagram elements based on a euclidean distance check. If a tag is contained with a circle or polygon, it is associated. If the tag is not fully contained but is written close to a diagram element[5], an association is created[6]. In case

[5]Measure of proximity = Distance threshold equal to 0.05% of screen width.
[6]this rule is applied to all diagram elements, not merely circles and polygons

of a tie or multiple possible associations, the tag is associated with the diagram element that was drawn first. An example is shown in Figure 4.7.

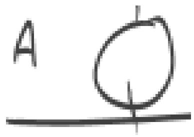


Figure 4.7: Spatial proximity of a tag and the ball necessitates clustering.

2. Associate Equations with Nearby Arrows: If an equation is written close to the head of an arrow, an association is created. A distance threshold $D = 75\%$ of current arrow's shaft length is used for this purpose. An example is shown in Figure 4.8. Arrows are especially tricky for making implicit associations because it is possible to specify the quantity associated with an arrow in several ways. Figure 4.9 demonstrates some ways in which a velocity of $50m/s$ at an angle of $\pi/4$ with the horizontal may be represented by an arrow using our prototype ITS.

Figure 4.8: A mathematical expression written close to the head of an arrow necessitates clustering.

3. Tag Matching Using user defined tags, we associate equations depicting initial conditions with recognized diagram elements in the following way:

   (a) Equations and Arrows: As mentioned previously, Arrows are a complex case for implicit associations because their tags can denote a physical quantity rather than a symbolic name. To associate the proper equation with a tagged arrow, it is sufficient to examine the left hand side of each equation to see if it matches the arrow's tag.

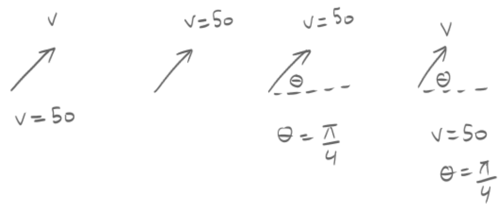


Figure 4.9: Different ways to specify a velocity of $50m/s$ at an angle of $\pi/4$ to the horizontal using an arrow

(b) Equations and Diagram Elements: If an equation has only one variable on its left hand side, and its subscript matches the tag for a diagram element, an association is made. For example, a ball depicted by a circle may be tagged 'B' by the user. if there is an equation in the solution such as $m_B = 15$, it is logical to assume that the ball tagged 'B' by the user has a mass of 15 kg.

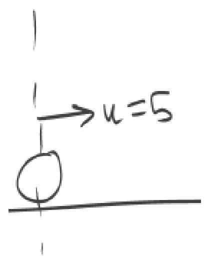


Figure 4.10: In this example, the arrow with its associated equation will be clustered with the dotted line of the interval which itself will be clustered with the ball.

4. Cluster Annotations with Diagram Elements: Our prototype supports three annotations: dotted lines, intervals and arrows. First, all arrows are examined to see if their shaft is partially contained within or in close spatial proximity[7] to a diagram element (including intervals and dotted lines). If so, the arrow with all its implicit associations is itself associated with the

[7]distance threshold = 3% of screen width.

diagram element. Second, dotted lines and intervals are examined to check if they intersect a diagram element (excluding annotations). If an intersection is found, an association is made. Figure 4.10 depicts a scenario where an arrow denoting a horizontal velocity of $5m/s$ will initially be clustered with an interval. The interval itself is clustered with the ball resting on a surface. The order of clustering is not important because rules for inferring associated values from connected annotations are built into each simulation engine in our system.

Our method for inferring implicit relationships between diagram elements has a time complexity of $O(n^2)$, where n is the number of diagram elements. This is not an optimal strategy for performing comparisons, and may possibly see improvement by incorporating spatial partitioning methods (grids, quadtrees, etc). However, given the small number of elements, tags and equations usually found in physics diagrams, this is not a significant bottleneck at this stage.

Inferring associations in this manner eliminates the need for users to associate all required initial conditions with the diagram manually. For each association (either inferred implicitly or indicated explicitly by the user), the system needs to extract the actual value of the physical quantity denoted by the association. Annotations denoted by arrows indicate vector quantities. Arrows can have dotted lines associated with them to indicate the angle with vertical or horizontal axes. By examining arrow labels, in combination with associated angle and axis annotation, the system is able to infer any of the following vector quantities: velocity, force, momentum, acceleration.

4.5 High Level Reasoning Engine

This phase of the recognition process merges the information extracted from the text of the problem with the recognized diagram elements. First, an appropriate primary simulator is chosen and assigned to the animation runtime. A series of secondary simulators is also instantiated, if the NLP

Engine indicates a need to do so, and assigned to the animation runtime. If no problem statement is provided, then the Free Fall Kinematics simulator is chosen as the primary simulator. We anticipate that this scenario can arise when the user wants to doodle or to quickly prototype an idea using our system. The Free Fall Kinematics simulator is the most general purpose of our simulation engines and allows for multiple granularities of input, making it ideal as the default choice.

Second, the high level reasoning engine passes the list of initial conditions to the Solution Checking module for later use. Once the correct simulator(s) are identified and instantiated, they are used to construct the simulation from the recognized diagram elements. Recognized elements are passed via the Animation Runtime to the correct simulators which match equation variables to domain-specific physics quantities, and assign initial conditions to each element based on either user defined annotations/equations or the element's spatial appearance. The responsibility for handling communication between the different simulators and for performing the overall animation lies with the Animation Runtime.

CHAPTER 5
BEAUTIFICATION OF RECOGNIZED DIAGRAMS

In this chapter, we first describe QuickDraw [39], a diagram beautification framework based on geometric constraint solving, that we have developed independently of our physics tutoring prototype. QuickDraw provides a powerful and robust method of globally beautifying an entire geometric diagram. it is able to work iteratively or in batch mode and is robust to recognition errors where some constraints are not properly recognized. .

We then describe the beautification procedures employed in our prototype ITS, some of which are based off of our earlier prototypes [40, 41] and some which have been adapted from the quickdraw system.

5.1 The QuickDraw Framework

QuickDraw [39] is a pen-based diagramming tool for geometry diagrams containing line segments and circles. Key contributions of this work include a novel diagram beautification algorithm based on realtime geometric constraint solving. QuickDraw first infers a series of geometric constraints between recognized drawing elements. Then, the notion of iterative refinement is applied to successively assign values to attributes of each diagram element such that the inferred constraints are obeyed. Table 5.1 lists all the constraints recognized by the QuickDraw system.

Table 5.1: List of constraints used by QuickDraw for beautification.

Primitive Types	**Constraint**
Line Segments	Vertical line segment Horizontal line segment Collinear line segments Parallel line segments Perpendicular line segments Equidistant line segments Touching line segments Intersecting line segments Line segments with same length Line segments with endpoint(s) at same horizontal level Line segments with endpoint(s) at same vertical level
Circles	Circles with same radius Concentric circles Circles touching at their circumference Intersecting Circles Circle passing through the center of another circle
Circles & Line Segments	Line segment tangent to circle Line segment intersecting circle Line segment passing through center of circle Line segment touching circumference with an endpoint

Continued on next page

Primitive Types	Constraint
	Line segment touching circle center with an endpoint

5.1.1 Recognition and Constraint Inference in QuickDraw

A sketched diagram is first parsed into primitive diagram elements (line segments and circles). The recognition heuristics are described in [39]. Recognized primitives are assigned a canonical ordering $\mathcal{O}$, from left to right followed by top to bottom. This ensures a deterministic view of each diagram, independent of the order in which its elements were drawn.

5.1.2 Constraint-based Beautification Framework

Once primitive elements C (lines and circles) are recognized, QuickDraw infers the intended constraints between them (See Table 5.1 for supported constraints). After inferring constraints, a novel beautification algorithm (Algorithm 1) is used to processes the recognized primitives and inferred constraints to generate a precise geometric diagram whose elements satisfy the intended constraints.

Each primitive diagram element C has attributes s such that it can be uniquely determined after some appropriate subset of its attributes are known. For example, the attributes of a line segment are its slope, intercept (y-intercept if slope is not vertical; otherwise x-Intercept), length, and x- and y- coordinates of its two end points. A line segment can be uniquely determined from the x- and y- coordinates of the two end points, or alternatively from its slope, intercept and y

Algorithm 1 Beautification Algorithm

$\{\tilde{C}$ = Set of Primitives; α = Set of Constraints; $\}$

Require: $\tilde{C}$, α

`A` := Set of all attributes of primitives in $\tilde{C}$;

`B` := $\emptyset$;

while `B` $\neq$ `A` **do**

if `A` $-$ `B` contains an attribute s that is computable from attributes in `B` because of α (using V)

then

Compute s;

else

s := attribute from `A` $-$ `B` with the highest rank.

Read value of s from sketch.

end if

`B` := `B` $\cup \{s\}$;

C := parent primitive of s.

if C is determined from its attributes in `B` (using U) **then**

Beautify **C**; `B` := `B` $\cup$ attributes of C;

end if

end while

coordinates of its two end points (if the slope is not vertical). Similarly, the attributes of a circle are its radius and the coordinates of its center. A circle is uniquely determined if all of its attributes are known. This knowledge is captured as an extensible set of rules U.

The idea behind defining constituent attributes for each diagram primitive is that constraints between primitives uniquely identify some of their attributes. This knowledge is captured as an extensible set of rules V, each of which specifies how to compute the value of some attribute from values of some other attributes under appropriate constraints. For example, if a line L is tangent to a circle C, C.radius can be computed from C.center, L.slope, and L.intercept. Specifically, C.radius can be computed as the perpendicular distance between C.center and the line determined by L.slope and L.intercept. As another example, under the same constraint that a line L is tangent to a circle C, L.intercept can be computed from L.slope, C.center, and C.radius.

The beautification framework in QuickDraw maintains a worklist B that holds the set of all attributes whose values have been computed. B is initialized to the empty set. The main loop of the beautification algorithm is repeated until B is equal to set A that holds all attributes of all recognized primitives. Each iteration of the main loop attempts to identify an attribute $s \in (\mathtt{A} - \mathtt{B})$ (of some diagram primitive C) whose value can be computed from the attributes in B using any of the rules V : s is then added to B. If the primitive C is uniquely determined from its attributes in B using any of the rules U, then C is beautified and all of its attributes are added to B.

If no such attribute s exists, then to maintain progress, the algorithm identifies an attribute $s \in (\mathtt{A} - \mathtt{B})$ with the highest rank. The rank of an attribute s of a diagram primitive C is given by lexicographic ordering on the following tuple:

$$\mathtt{Rank}(s) \quad = \quad \left(\mathtt{Max}_S \left\{ \frac{\sum\limits_{s' \in S \cap \mathtt{B}} W(s')}{\sum\limits_{s' \in S} W(s')} \right\}, \mathcal{O}(C), \frac{1}{W(s)} \right)$$

An interesting element of the above rank tuple is a weight function W that maps each attribute type to some score between 0 and 1 and is used to assert the relative importance of knowing some attribute over another. More specifically, the relative weight ordering reflects the order of observing any visual discrepancies (thereby avoiding the need to edit diagram primitives unless really required), and also the order of ease of editing diagram primitives if QuickDraw didn't get it right.

QuickDraw uses the following relative weights: (a) Attributes of a line-segment: x-y coordinates of the two end-points and length (0.5 each), intercept (0.75), slope (1). (b) Attributes of a circle: x-y coordinates of the two end-points (0.5 each), radius (1).

The first element of the rank tuple identifies a diagram primitive C that is closest to being determined. This is estimated by computing the maximum of the weighted ratio of the attributes that are known from among some minimal set of attributes S of C that uniquely determine the primitive C. The second element of the rank tuple breaks any ties among C by using the canonical ordering $\mathcal{O}(C)$ assigned to the primitive at recognition time. The third element of the rank tuple identifies an attribute s of component C that has the lowest weight. The value of the attribute s with the highest rank is then read off from the sketch. C is then beautified and all of its attributes are added to `B`.

This beautification framework has two interesting characteristics: robustness and interactive support. The algorithm is robust due to the powerful deductive reasoning enabled by an extensible set of rules U and V over a saturated set of constraints inferred using sketch understanding techniques. This allows it to correctly beautify diagrams when the recognition engine misses out on some constraints. The algorithm also allows for interactive drawing. The main loop of the algorithm can be run in an incremental fashion after adding attributes of any new diagram primitives sketched by the user to `A` and updating the set of constraints.

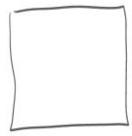


Figure 5.1: A roughly sketched square.

5.1.3 Example: A Sketched Square

Given the sketched square in Figure 5.1, QuickDraw should ideally infer the following constraints between four recognized line segments:

1. Two line segments are horizontal and two are vertical.
2. The horizontal line segments are parallel, and are both perpendicular to the vertical line segments.
3. The vertical line segments are parallel, and are both perpendicular to the horizontal line segments.
4. All the line segments in the sketch form a connected path, and are all equal in length.
5. The perpendicular distance between horizontal line segments is the same as that between vertical line segments.

Beautification now proceeds as follows. (i) After computing the slope of all the line-segments, the algorithm reads off the x-y coordinates of the top-left corner and the y-coordinate of the bottom-left corner from the sketch and then beautifies the left line-segment. (ii) Next, the algorithm computes the y-coordinate of the top-right corner from the y-coordinate of the top-left corner (because of the top segment having horizontal slope constraint), and then the x-coordinate of the top-right corner from the two left corners (because of the equal length constraint between the top and left segments), and then beautifies the top line-segment. (iii) In a manner similar to the previous case, the algorithm computes the y-coordinate of the bottom-right corner from the y-coordinate of the bottom-left corner (because of the bottom line-segment having horizontal slope constraint), and then the x-coordinate of the bottom-right corner from the two left corners (because of the equal length constraint between the bottom and left line-segments), and then beautifies the bottom line-segment.

Alternatively, suppose that the system had failed to infer any equal length constraint involving the bottom line-segment. The algorithm can still compute the x-coordinate of the bottom-right corner from the x-coordinate of the top-right corner (because of the right line-segment having vertical slope constraint). Let us also suppose that the system failed to infer the vertical slope constraint for the right line-segment. The algorithm can still compute the slope of the right line-segment from the slope of the top line-segment (because of the perpendicular constraint between the top and right line-segments) followed by computing the intercept of the right line-segment from the x coordinate of the top-right corner. The algorithm can then compute the x-coordinate of the bottom-right corner from the two top corners (because of the equal length constraint between the right and top line-segments).

These instances of missing constraints highlight the robustness of the QuickDraw beautification framework, which is able to make up for the missing constraints by making effective use of other (logically equivalent) constraints. Some examples of diagrams that can be beautified using QuickDraw are shown in Figure 5.2.

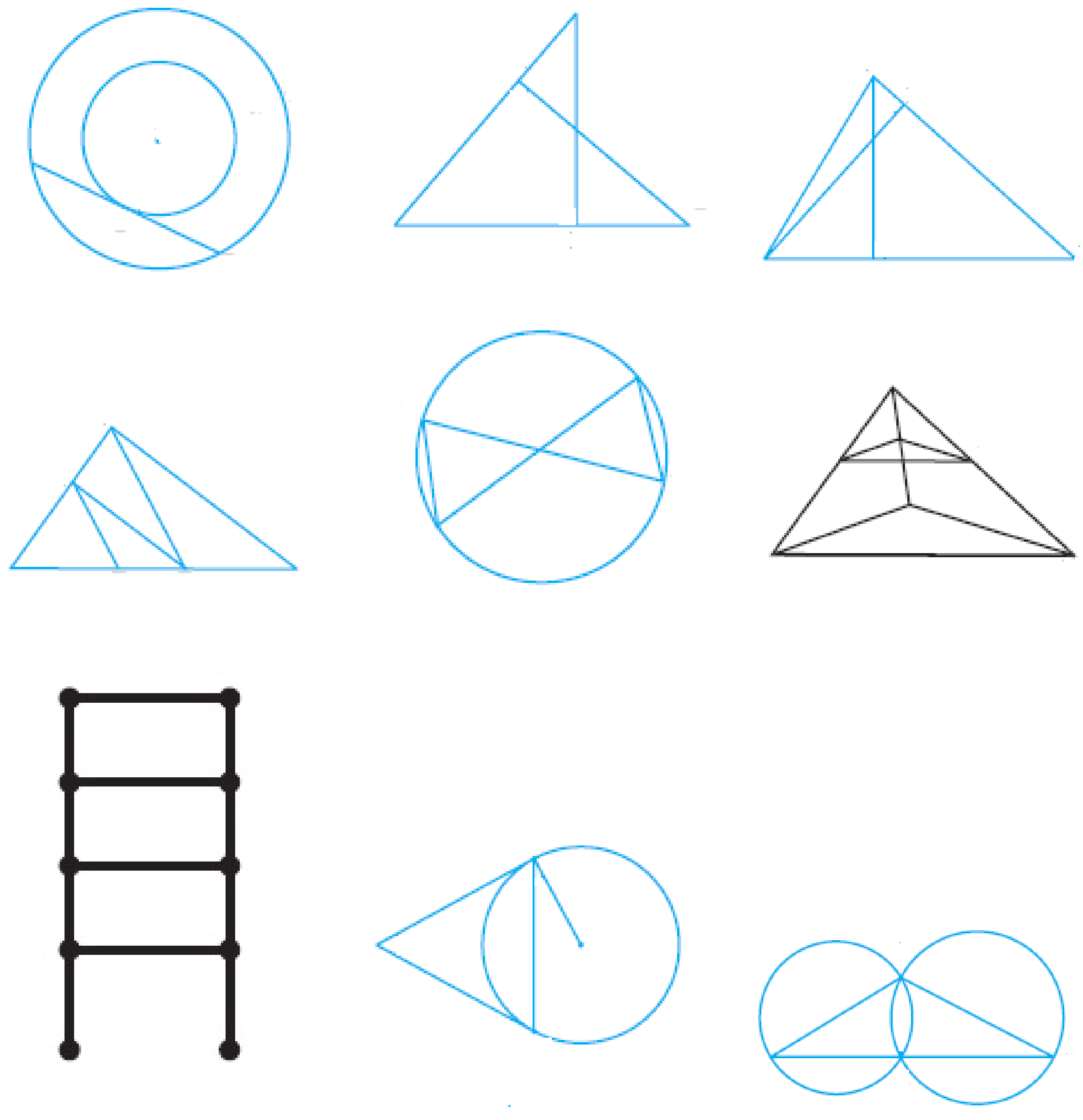


Figure 5.2: Examples of geometric diagrams that can be precisely beautified using the QuickDraw beautification framework.

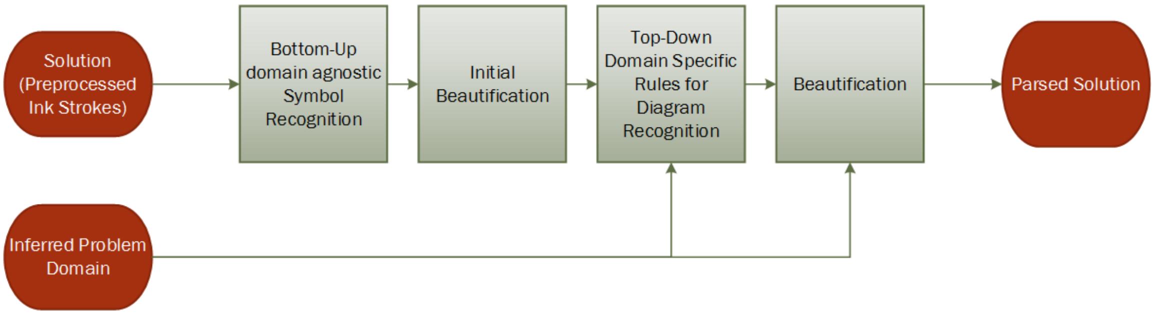


Figure 5.3: The diagram recognition pipeline in our prototype ITS. Beautification occurs at two points during recognition, and may also occur when a user makes an explicit association.

5.2 Beautification in our Prototype ITS

Sketch beautification is extremely important for constructing a physically correct animation [41]. Hand drawn diagrams are approximate in nature, and must be precisely beautified in order to generate a plausible and correct animation. For our prototype system, we have adapted beautification techniques from QuickDraw [39] and our earlier prototypes [41, 40]. Figure 5.3 depicts the diagram recognition pipeline in our prototype. In some instances, recognized diagram elements can be beautified in isolation, immediately after they are recognized. For other instances of beautification, the relationship between diagram elements needs to be considered, possibly within the context of the problem domain. This is handled by a beautification post-processing step that is invoked after the entire recognition process is complete. Lastly, beautification may be needed when an equation is explicitly associated with an element by a user.

5.2.1 Initial Beautification: Individual Elements

The first beautification pass is conducted immediately after unistroke and multistroke elements have been recognized, and only affects each element in isolation. This comprises the following steps:

1. Line Segments, Helixes, and Arrows are aligned vertical or horizontal if their slope falls within a specified range (within approximately 8 degrees of the x- or y-axes).

2. Polygon vertices are checked to see if they are in counter-clockwise order, and assigned a counter-clockwise winding if not so. The ordering is checked by taking the 2D cross product of direction vectors of each pair of adjacent edges[1].

3. Each edge in polygons is aligned vertical or horizontal if its slope falls within a specified range (within 8 degrees of x- and y-axes).

4. Each segment of polylines is aligned vertical or horizontal if its slope falls within a specified range (within 8 degrees of x- and y-axes).

5. Dotted lines and intervals are aligned vertical or horizontal (within 8 degrees of x- and y-axes).

6. Pulleys are beautified to ensure that the constraints specifying their appearance are ensured precisely: One end of anchor touches the center of the hinge and the wires touch at tangent points to the hinge. We use the QuickDraw [39] framework for beautifying pulleys, because pulleys are described purely in terms of geometric constraints and are therefore natural candidates for constrained-based beautification.

[1]The 2D cross product is defined as $v_1 \times v_2 = x_1y_2 - y_1x_2$.

5.2.2 Secondary Beautification: Annotations and Groups of Elements

The initial beautification pass examines each diagram element in isolation. Yet, in physics diagrams, there is sometimes a need to examine interesting groups of elements simultaneously and ensure their correct positions if a realistic animation is to be constructed. Therefore, we conduct a second pass after the recognition process is complete, comprising the following tasks:

1. Arrow directions are altered if an equation specifying an angle with the horizontal or vertical axis is associated.

2. If a shape (Circle or polygon) rests on a surface (line segment or polyline), their positions are manipulated to ensure touch constraints. Resting contact is determined by checking spatial proximity, with a threshold = 5% of screen width.

3. For springs and pulleys that are connected to shapes, the connection point is manipulated to ensure that it exists on the shape boundary. This mitigates imprecise drawing where the endpoints are usually drawn inside the shape.

Finally, explicit associations can also alter the appearance of diagram elements. For such associations, we utilize the method described in [41].

CHAPTER 6
ANIMATION RUNTIME

In our earlier prototypes, we used a single physics engine [42, 41, 40] for animating student diagrams. This approach was limited to modeling scenarios where the student's answers could be morphed to concepts related to $f = ma$ which formed the core position update mechanism for the physic engine (See Appendix A for a listing of earlier prototypes). Student answers resulting in equations for position, displacement, velocity, acceleration, force and spring stiffness could be used directly [42, 41] while equations for concepts such as work done, kinetic and gravitational potential energy needed to be transformed during animation [40]. This design was difficult to extend to new domains of physics problems and was prone to instability as new features were added.

Our current prototype uses separate, domain-specific simulators to provide animation support for diagrams belonging to linear motion, free fall, projectile motion, equilibrium, momentum, and friction. This design is more modular and easier to extend. Each simulator is entirely self-contained, thus localizing logical bugs and minimizing impact on other system areas. Figure 6.1 depicts a high-level overview of the design of our animation module. The fundamental unit of animation in our new architecture is a shape (circle or polygon). Each simulator defines different rules for modeling forces, resolving collisions and updating the position of shapes. Each simulator also integrates rules for understanding domain-specific quantities and equations and for performing unit conversions. Additionally, we have abstracted out pulleys and springs as an optional add-on to all simulators[1], allowing for reuse of these components between diagrams in different domains.

[1]However, at this time, pulleys and springs are only supported in the context of Freefall Kinematics and Friction simulators.

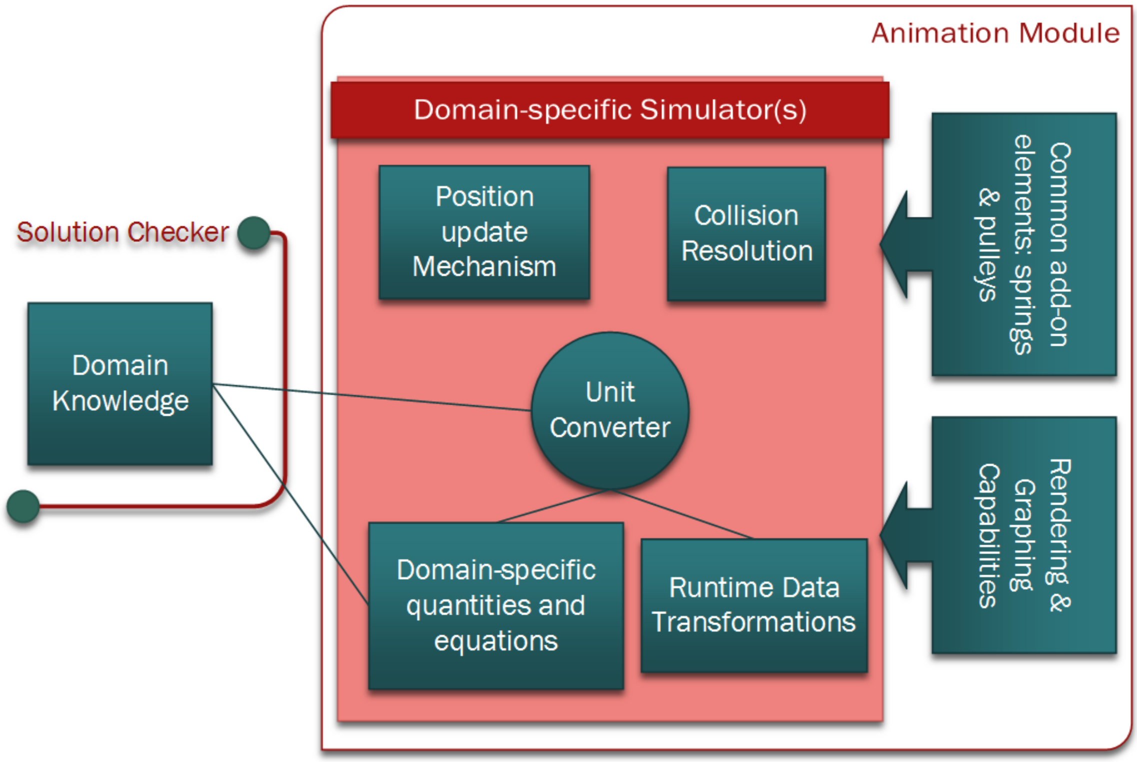


Figure 6.1: High-level overview of the animation module showing components of each domain-specific simulator, the common elements shared between simulators and the interdependence between the animation module and parts of the Solution Checker.

Each simulator provides entry points for mathematical equations that can be explicitly associated with physical properties of simulation elements, altering their animation behavior. Associated equations are evaluated at association time or during animation to yield scalar or vector values that are used to guide the animation in accordance with the user's intent. It should be noted that the modified behavior caused by associated mathematics may or may not be intuitively correct. This serves as a feedback mechanism indicating whether there is a mistake in the user's solution. Each simulator also exposes a list of domain-specific physics quantities for shapes that can be used to generate realtime graphs.

Our animation framework provides a possibility for runtime data transformations. This affords us the choice to extend one or more simulators at a later date to include animation support for new types of problems, without writing entirely new simulators. Table 6.1 summarizes the differences in notations and functionality supported by the three simulators in our prototype ITS:

Table 6.1: Simulation Configuration

Simulator	**Supported Elements**	**Supported Annotations**	**Target Problem Domains**
Free Fall Kinematics	Circles, Polygons, Pulleys, Springs	Arrows, Dotted Lines	Free Fall, Free-hanging Springs, Projectiles, Pulleys, Doodling
Friction	Circles, Polygons, Pulleys, Springs, Line Segments, Polylines	Arrows, Dotted Lines, Intervals	Sliding Contact, Inclined Planes, Kinetic and Static Friction
Momentum	Circles, Polygons, Pulleys, Springs, Line Segments	Arrows, Dotted Lines	Momentum problems involving elastic collisions in 1-dimension
Equilibrium	Circles, Polygons, Wires	Arrows, Dotted Lines	Simple Equilibrium Problems, Tension Problems, Objects held with breakable wires

6.1 Free Fall Kinematics Simulator

This simulator is based on game physics engine concepts [140, 61]. Its core functionality includes collision detection and resolution, constant and drag forces and an incremental position update mechanism based on numerical integration. Sketched shapes (circles and polygons) denote rigid bodies. Users can associate equations for forces, velocity, position, and acceleration which can augment or replace the standard position update mechanism. Additionally, equations may be assigned to alter mass, spring stiffness, work done, kinetic energy and gravitational potential energy. Equations for time limits may be associated with the simulator itself by using 'Lasso' to select an equation (usually of the form $t = x$, where x is the time limit) and 'tapping' anywhere on the writing area[2].

This simulator is able to provide support for problems that involve bodies in free fall or are attached to springs, or for pulley systems released from rest. Within these domains, it is able to animate diagrams whose corresponding solutions yield an answer in the form of one of the physics quantities mentioned above.

6.2 Friction Simulator

The friction simulator models kinetic and static friction between two surfaces. This allows it to animate diagrams where an object is in sliding contact with a fixed surface. It does not have any collision detection. For this simulator, users must have drawn one fixed surface denoted by a long line segment or a polyline, upon which a shape rests. Users may sketch intervals to denote a range of displacement, and may associate equations for displacement with the interval. Users

[2]This mechanism is supported by all simulators in our Animation Runtime to denote time limits for a particular animation.

may also associate equations for mass, velocity, acceleration, constant force, kinetic friction, static friction, coefficients of kinetic and static friction, work done, kinetic and potential energy, and spring stiffness with various recognized diagram elements.

This simulator is able to handle problems from the domain of friction where a single body is moving along a rough surface, and may or may not be constrained to move within a specified distance. Additionally, by setting the friction forces to zero, this simulator can be used to provide animation support for inclined plane problems or problems involving 1-D motion along a frictionless surface.

6.3 Equilibrium Simulator

The equilibrium simulator provides support for problems where a body is held in equilibrium by a number of breakable wires. It does not have any collision detection. For this simulator, wires can be drawn as line segments connected to shapes. Users may associate equations for mass, force, and tension with diagram elements. At recognition time, the length of each recognized wire is taken as allowed length. If any wire is stretched beyond 20% of its allowed length, it is tagged as broken, and its effect is removed from the simulation. Broken wires are not rendered.

This functionality enables us to model a small set of equilibrium problems where systems in equilibrium are released from rest and the student's answer determines if equilibrium is maintained. Possible avenues of future work include providing support for angular effects such as moment of force, and enabling users to mark or test the center of turning.

6.4 Momentum Simulator

This simulator models elastic collisions between two bodies moving in 1-dimension along a fixed surface. Again, users must draw one fixed surface denoted by a long line segment or a polyline, upon which two shapes rest. Users may sketch a single dotted line to denote the event of a collision and may associated equations or arrows denoting velocity with the collision event. For this simulator, we have adopted the convention that all velocities prior to collision are denoted by u and all velocities after the collision are denoted by v. Users may only associate equations for mass and velocity. For collision problems, a coordinate system is usually required, where one direction is assumed as the positive direction, with any vector quantities that lie in the opposite direction having a negative sign. In our Momentum simulator, the sign of direction is automatically inferred from arrow annotations (arrow direction implies direction of associated vector quantity). This also means that while $+$ and $-$ signs are ignored when matching equations to arrow labels. For an example, please refer to Section 8.2.7.

This simulator is able to handle limited instances of momentum problems, and serves merely as a proof-of-concept at the moment. Possible extensions include support for collisions in 2-dimensions by enabling a top-down viewing volume.

CHAPTER 7
SOLUTION CHECKER

Our solution checker is based on building a graph structure to represent the mathematical steps in a student's solution. In principle, this scheme is similar to the standard representation for a deductive proof [136, 173, 188], where one starts with a set of givens and uses logic to reach a conclusion. In our case, the initial conditions specified in the problem statement are the givens, and the overall solution is a mathematical (rather than deductive) proof. The steps in the solution can be linked in a graph-structure where each link signifies an inference based on a mathematical or physical principle. The initial conditions and the answer then become the leaf nodes for the graph.

As our prototype system does not contain any prior knowledge or instrumentation about each problem, it does not contain a known model (which can be coupled with model tracing techniques for tutoring). Instead, we extract pertinent information from the text of a given problem, and generate the graph-model at runtime. Existing ITS's incorporate sophisticated facilities for pre-computing the solutions to given problems (Andes [207] is a relevant example). Our methodology is different. We construct a model of a written solution and then use path searching methods to determine if the student has used some path to arrive at an answer. Using a database of physics and mathematical equations (containing common algebraic variations[1]), we infer relationships between solution steps and verify the soundness of each step.

The solution checker can be invoked by hitting the 'Check' button from the system menu. This module contains knowledge of physics concepts in the form of a database of physics and

[1]In our current prototype, the size of the database is quite small. It is limited to the equations used for solving the problems in Chapter 8. However, the database is easy to extend by providing examples of new equations in an xml format.

mathematics equations, as well as their variations. Additionally, information extracted from the text of the problem is also utilized here.

Checking can only happen once the solution has been recognized and parsed into diagram and solution steps. The solution checker also assumes that the mathematical steps in the solution are in correct logical order. We infer the ordering by sorting the y-coordinate for the centroid of each step in ascending order. This enables us to order solutions written vertically in a single column format. We currently do not have a method to assign ordering to solutions steps written in two-column format. Lastly, the solution must not contain chained mathematical expression, e.g., $f = ma = (2)(1.5) = 3N$.

7.1 A Graph Model for Solution Steps

We use two key insights to check solutions. First, each step in the solution must be mathematically sound, i.e., it must use physics and mathematics formulae correctly. Second, the solution must be sound as a whole, i.e., each step should use well-known mathematics and physics knowledge and/or be dependent on the preceding steps. Our solution checker assigns a label to each solution step. Each solution step falls into one of four categories: Initial Condition, Formula Statement, Manipulation, and Answer.

Initial Condition Usually written in the early stages of a solution. It is also possible for the student to write them in the middle. Also, sometimes, students may not write them at all. They can also be identified by examination of the problem statement, as it contains all initial conditions.

Formula Statement Well-known equation that expresses a mathematics or physics principle. Our domain knowledge database contains a list of equations as well as their variations. These steps can be identified by a simple lookup.

Manipulation These are steps where the student either substitutes values for quantities, manipulates the equation in some manner (e.g. multiplying/dividing/adding some value to both sides), simplifies an expression or computes values. In order to speed up the labeling process, any solution step that is not an initial condition, a formula, or the answer is labeled a manipulation step.

Answer Last step in the solution. Based on the analysis of the problem statement, our system can infer the quantity the student is asked to solve for. We can check the last solution step to see if it is an expression for the relevant physical quantity.

Labeled solution steps can be used to verify if the solution is correct. A trivial rule is to check the existence of the answer, indicating if the solution is complete. We then construct a directed graph data structure $\overline{S}$, where each vertex s_i corresponds to a solution step. The solution step labeled 'Answer' is not added to $\overline{S}$. Edges in $\overline{S}$ are assigned based on the following rules:

R1. If a vertex s denotes a formula statement, an edge is assigned from the list of physics/mathematics equations $\overline{E}$ to that vertex.

R2. If the mathematical step in a vertex s_j is a manipulated form of another mathematical step s_i, an edge is assigned from s_i to s_j. This can include substitution of values into a formula, arithmetic manipulation or computation.

R3. If a vertex s_m denotes a computed value or an initial condition e.g. $a = 1.5m/s^2$, and there exists a subset $\overline{S}_{dependent} \subset \overline{S}$ such that each vertex $s_k \in \overline{S}_{dependent}$ uses the value in s_m, then an edge is assigned from s_m to each $s_k \in \overline{S}_{dependent}$.

R4. If the vertex corresponding to the last solution step s_n denotes a computation for the quantity asked for in the question, an edge is assigned from s_n to the answer A (which exists as a lone vertex outside the graph).

7.2 Rules for Checking a Solution Graph

For our purposes, domain knowledge in the form of physics and mathematics formulae and the answer A are modeled as vertices outside $\overline{S}$ which are connected to vertices inside the graph. The solution represented by $\overline{S}$ is valid if the following conditions are true:

1. $\overline{S}$ contains no cycles.
2. $\forall i = 1...n$, s_i is valid, i.e., uses mathematical and physics formulae correctly, substitutes values correctly and only uses mathematically valid manipulations.
3. $\forall i = 1..n,$ If s_i is not labeled 'Initial Condition', $Indegree(s_i) \in \overline{S} \geq 1$
4. $\forall i = 1..n, Outdegree(s_i) \in \overline{S} \geq 1$

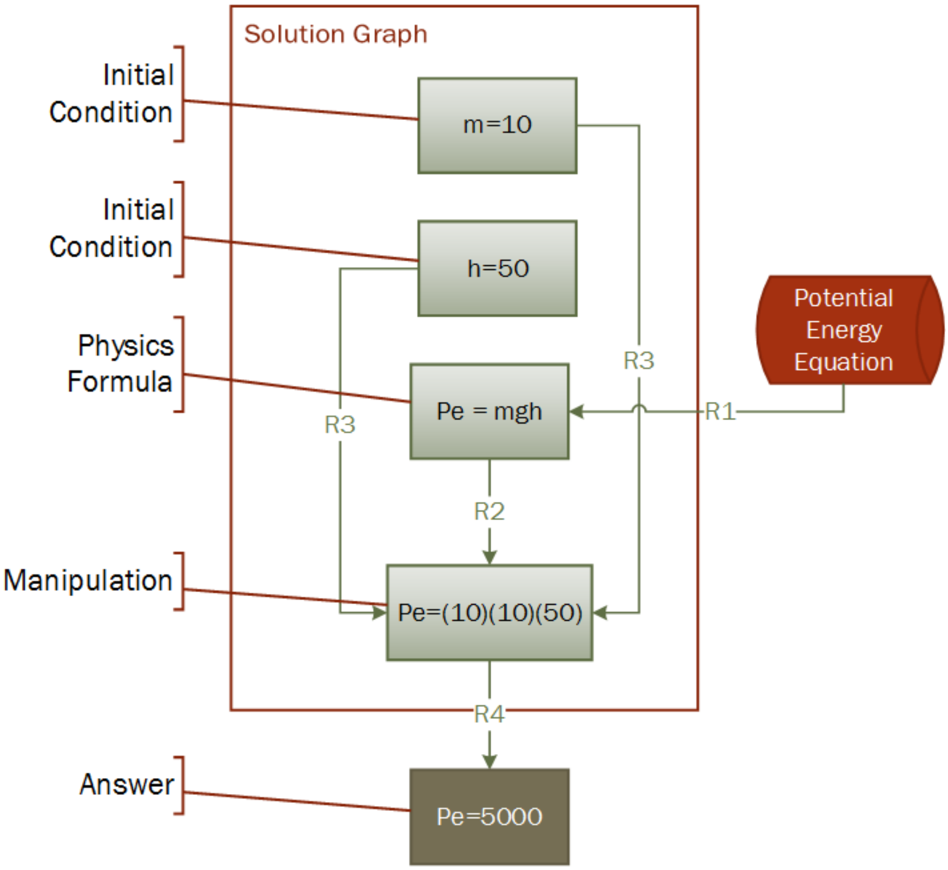


Figure 7.1: Figure showing the solution graph constructed for a gravitational potential energy problem where a ball of mass 10 kg falls a distance of 50 meters.

7.3 A Worked Example

Figure 7.1 depicts a physics problem where a student is asked to work out the change in gravitational potential energy when a ball of mass 10 kg falls a distance of 50 meters. This solution contains two initial conditions ($m = 10$ and $h = 50$), two solution steps ($P_e = mgh$ and $P_e = (10)(10)(50)$), and a final answer step $P_e = 5000J$. The solution graph shown in Figure 7.1 consists of 4 vertices, two initial conditions, one formula. and a manipulation step where the values of the initial condition are being substituted into the formula. The manipulation step is linked to the answer step, which lies outside the solution graph. The edges of the graph in Figure 7.1 have been labeled with the appropriate rule (See Section 7.1) that was invoked to construct each edge.

7.4 Discussion

Our solution checking algorithm is user independent, can handle complex solutions and works fairly fast. The labeling process works in linear time $O(n)$, for an n-step solution[2]. Constructing the directed graph representation takes longer, because for each solution step, we need to potentially examine all the steps prior to it, in order to determine dependencies, giving the time cost as $O(n^2)$. Checking the validity of the solution graph is slightly more nuanced. Checking for cycles can be accomplished in $O(|Vertices| + |Edges|)$ time (Tarjan's Algorithm [197]) [3].

The time required to verify the validity of each vertex s_i in the graph is either constant or dependent on its total degree, which in the worst case is $O(n)$[4] Counting the in- and out-degrees of each vertex is an $O(1)$ operation per vertex, giving overall time cost as $O(n)$. Combining all these factors together, the worst case time complexity for our solution checker is $O(n^2)$, for a solution containing n steps. However, as our mathematics recognizer is continuously active, the graph structure $\overline{S}$ can be constructed and modified while the student is writing or editing his/her solution, mitigating the time required for setting up the solution checker.

However, it may be difficult for a handwriting-based system to fulfill the prerequisites for our algorithm. First, perfect handwriting recognition accuracy is required, because any recognition

[2]It can be assumed that any lookup from the database of physics and mathematics formulae is an $O(1)$ operation, given suitable choice of storage and lookup mechanisms.

[3]In the worst case, we can envision the entire solution as consisting of interrelated formulae and manipulation steps, yielding $|Vertices| = n$ and $|Edges| = n^2$.

[4]In practice, we expect to validate most steps in constant time. The theoretical worst case implies that a solution step exists that is linked to all the steps preceding it. The 'Manipulation' step in Figure 7.1 is an example of this. However, the solution in Figure 7.1 is a simple one comprising only a few steps. Steps involving such gestalten computations are likely to be a very small subset of the entire solution.

errors would impact the step-labeling process. Second, the logical ordering of the solution steps must also be known exactly. We require this in order to mitigate ambiguity in assigning dependency relationships between solution steps. Third, an external library of well-known mathematical and physics formulae is required, which may grow quite large based on which domains are targeted by the tutoring system. With a large database, searching for a matching equation may become a bottleneck.

CHAPTER 8
ANIMATION CAPABILITIES

This chapter lists several physics problems modeled using our prototype system. In each instance, we describe how an animation may be constructed from the written solution. In this way, we highlight the range of capabilities of our prototype system and discuss supported animation mechanisms.

8.1 Animations for Toy Examples

Toy scenarios are usually a first step for any recognition system. The aim is to work with a small set of driving examples that enable one to identify and construct core functionality that may be generalized to a range of real world scenarios. This section presents three toy examples (either adapted from real physics problems or interesting in their own right).

8.1.1 Doodling: 3-Spring System

In Figure 8.1, a box is connected to 3 springs of different stiffness. This is an example of prototyping or doodling where deriving the closed-form solution for the box's motion trajectory is non-trivial. Instead, it is easier to sketch the system and generate realtime graphs showing how different quantities change in realtime.

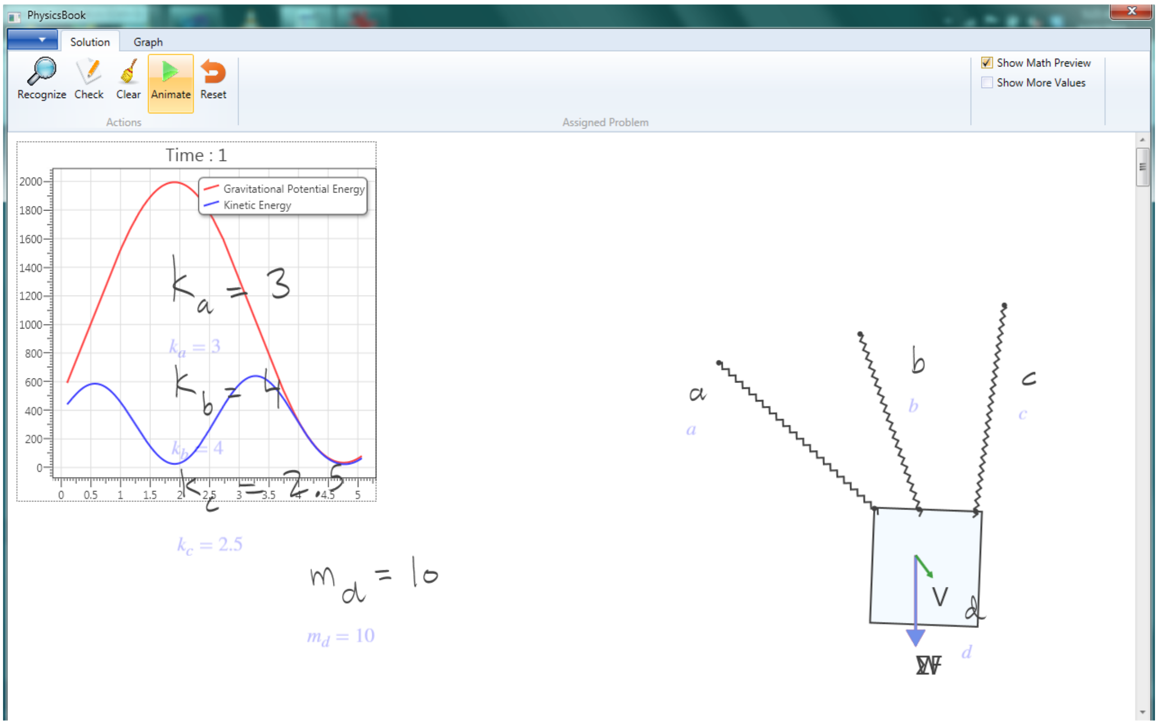


Figure 8.1: A box suspended using three springs. Deriving a closed-form solution for the behavior of this system can be very difficult whereas generating a simulation for this scenario is very easy within our prototype system.

The three springs are labeled 'a', 'b', and 'c' by the user, while the box is labeled 'd'. Upon recognition, the labels are associated implicitly with each diagram element based on proximity checking. Similarly, the equations for stiffness and mass are also associated implicitly using label-matching. The graph in Figure 8.1 demonstrates the interplay between the kinetic and potential energy of the box once the system is released from rest. This example highlights the use of graphing as an important aspect of animation.

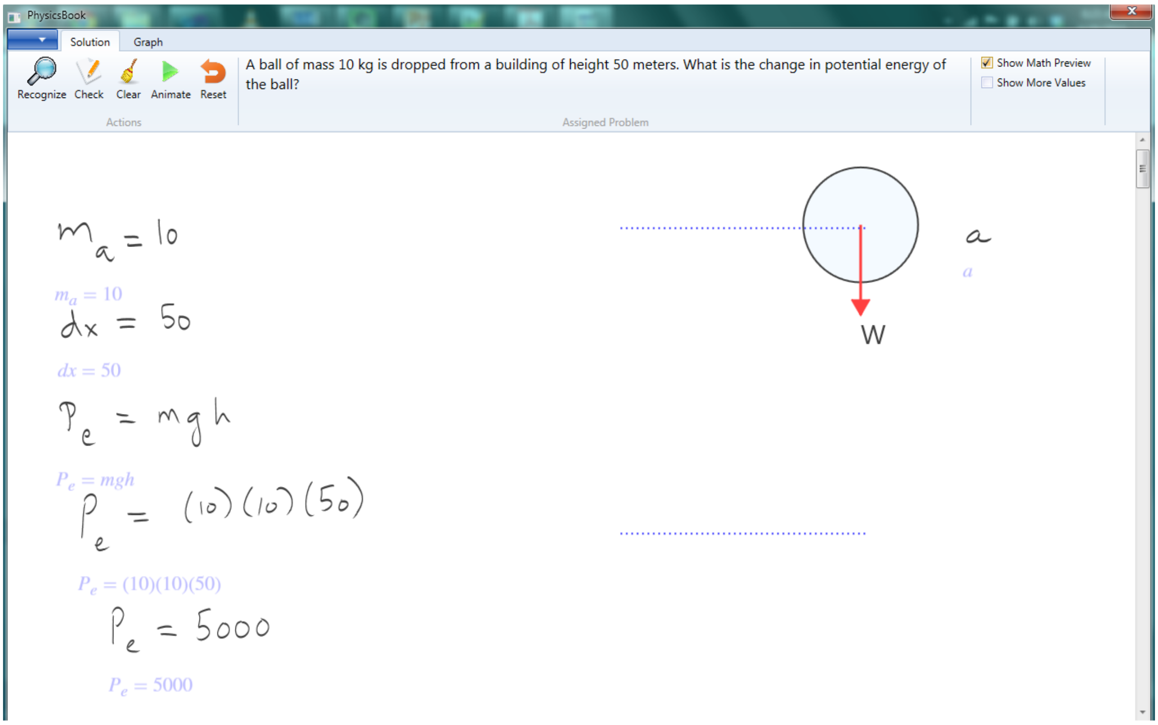


Figure 8.2: Example scenario where a ball of mass 10kg is dropped from a the roof of a building of height 50m. The student is asked to work out the change in gravitational potential energy of the ball.

8.1.2 Change in Gravitational Potential Energy During Free-Fall

Figure 8.2 shows a contrived free fall problem. A ball of mass $m = 10$kg is dropped from a known height. The student is asked to determine the change in gravitational potential energy after the ball has fallen 50m. The student sketches a circle to represent the ball, and annotates the diagram with an interval (two parallel, horizontal dotted lines) to indicate limits of displacement. From the given information, the student computes the change in potential energy of the ball to be $P_e = 5000J$.

To verify the answer, the student animates the sketch with the solution ($P_e = 5000$) as input. Upon recognition, the ball is replaced by a circle. The dotted lines are interpreted as an interval. The circle is labeled ‘a’, and the equation indicating mass ($m_a = 10$) is associated implicitly.

The student first selects the expression $dx = 50$ and associates it with the interval to indicate the displacement limit. The student then selects the answer, $P_e = 5000$ with the 'Lasso' gesture, associates it by tapping the circle, and triggers the animation. The system uses the association $P_e = 5000$ to derive the magnitude of required displacement. The simulation stops when the required displacement has been covered. If the computed change in potential energy is correct, the ball will move exactly 50m. If it is incorrect, then the ball will come to rest either before or after reaching the end level. This example highlights the usefulness of being able to denote the starting and ending points of motion by using dotted lines.

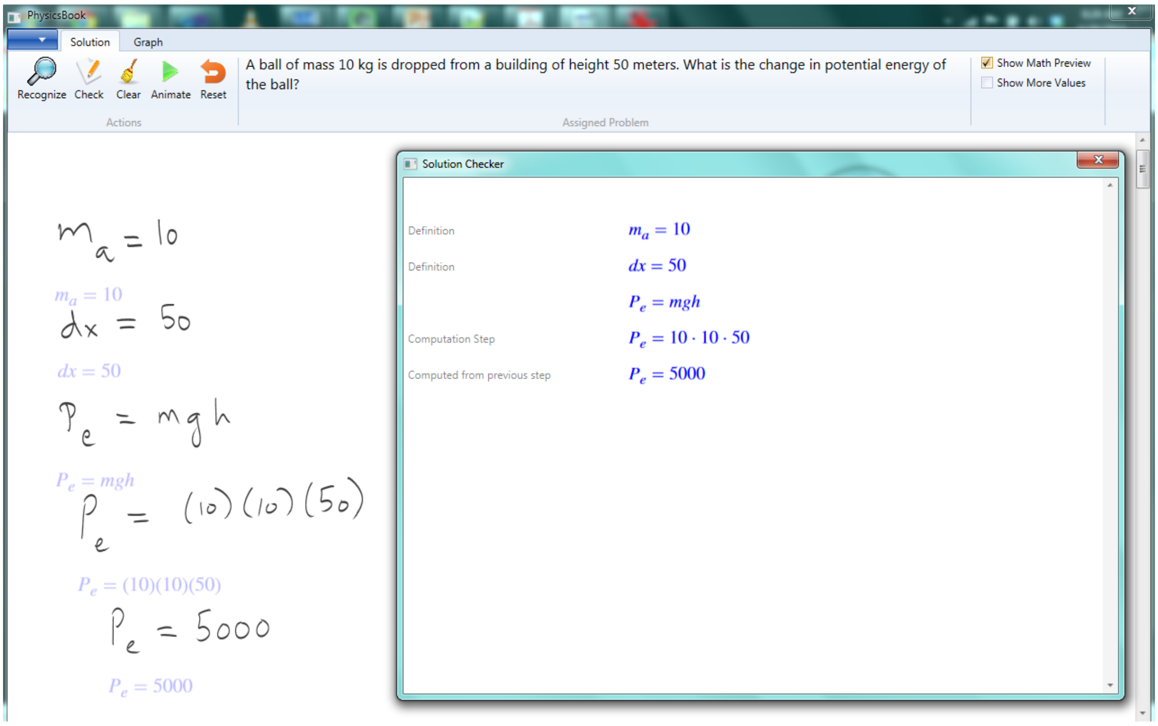


Figure 8.3: An example of the solution checker in action.

Figure 8.3 shows what happens if the student hits 'Check' after the diagram has been recognized. Each solution step is typeset and displayed to the user, along with the results of the analysis, i.e., if each step is an initial condition, formula or a computation.

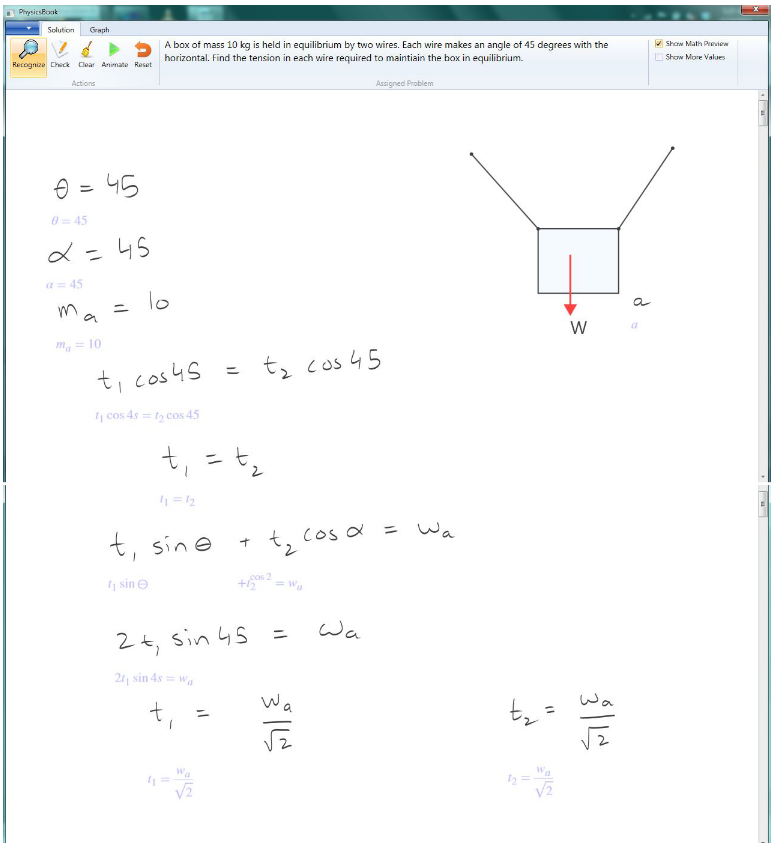


Figure 8.4: A contrived equilibrium problem based on the real world scenario presented in Section 8.2.4.

8.1.3 A Contrived Equilibrium Problem

The problem in Figure 8.4 is a simplified version of the equilibrium problem depicted in Section 8.2.4. Here, a box is suspended using two wires, each at an angle of 45^o with the horizontal direction. The student is asked to work out the tension in each wire such that the box is held in equilibrium. As the angle with the horizontal is same for each wire, the magnitude of the tension in each wire is also the same. The student derives a value of $\frac{w_a}{\sqrt{2}}$ for the tension in each wire. After associating the tension with each wire, the student can hit ‘Animate’. As the tension in each wire is correct, the box is held in equilibrium. If an incorrect answer for the tension had been derived, the box would move in the direction of the net force. This would cause one or both wires to stretch (depending on the direction and magnitude of the net force). If any wire is stretched beyond 20% of its initial length, it is broken, causing the box to potentially fall freely.

This scenario illustrates three important issues. First, the answer is given in symbolic variables rather than in a numeric form. The animation system must be able to find and extract the value of w_a at runtime for it to work correctly[1]. Second, sketched wires may not touch the box at its exact corners and may also not make an angle of 45^o with the horizontal direction. Precise beautification after recognition and explicit association is required to mitigate these problems. Third, this problem highlights an important feedback mechanism for animation-based tutoring: ”the presence of motion where there should be none”.

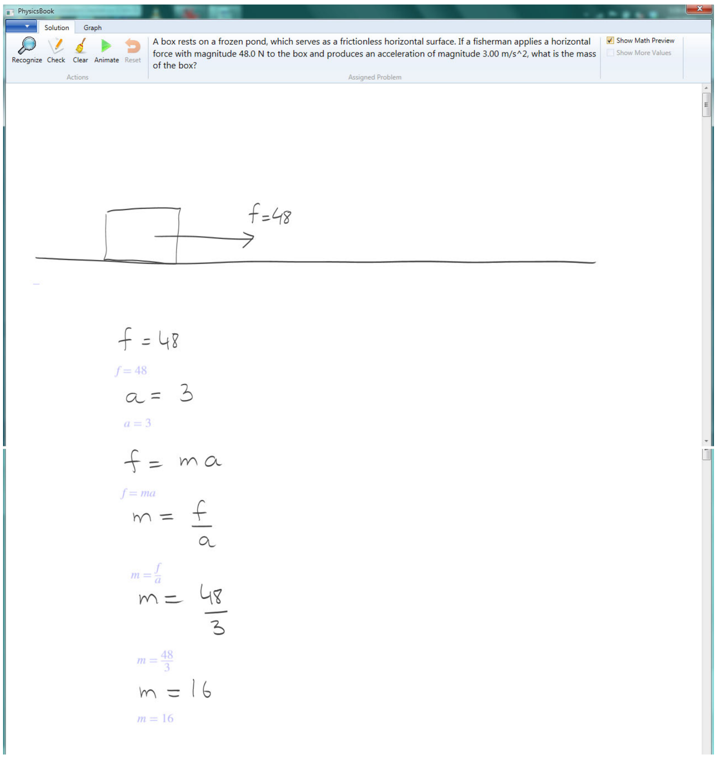


Figure 8.5: Simple problem that requires the use of $f = ma$.

[1]This can generally be stated as: Given a variable and its subscript, find out 1) the physical quantity represented 2) the simulation element to which this physical quantity belongs and 3) the value of said physical quantity at this instant in time.

8.2 Animations for Real Physics Problems

8.2.1 A Simple Force and Acceleration Problem

The scenario in Figure 8.5 is a very simple one that can be solved with the direct application of $f = ma$. A force and acceleration are provided, and the mass is required. However, in this instance, the answer cannot be visually verified in a simple way. If the box moves with a constant but wrong acceleration, a user may not be able to tell the difference. However, if graphing functionality is employed (shown in Figure 8.6), then the student can observe that the acceleration is precisely that which is predicted.

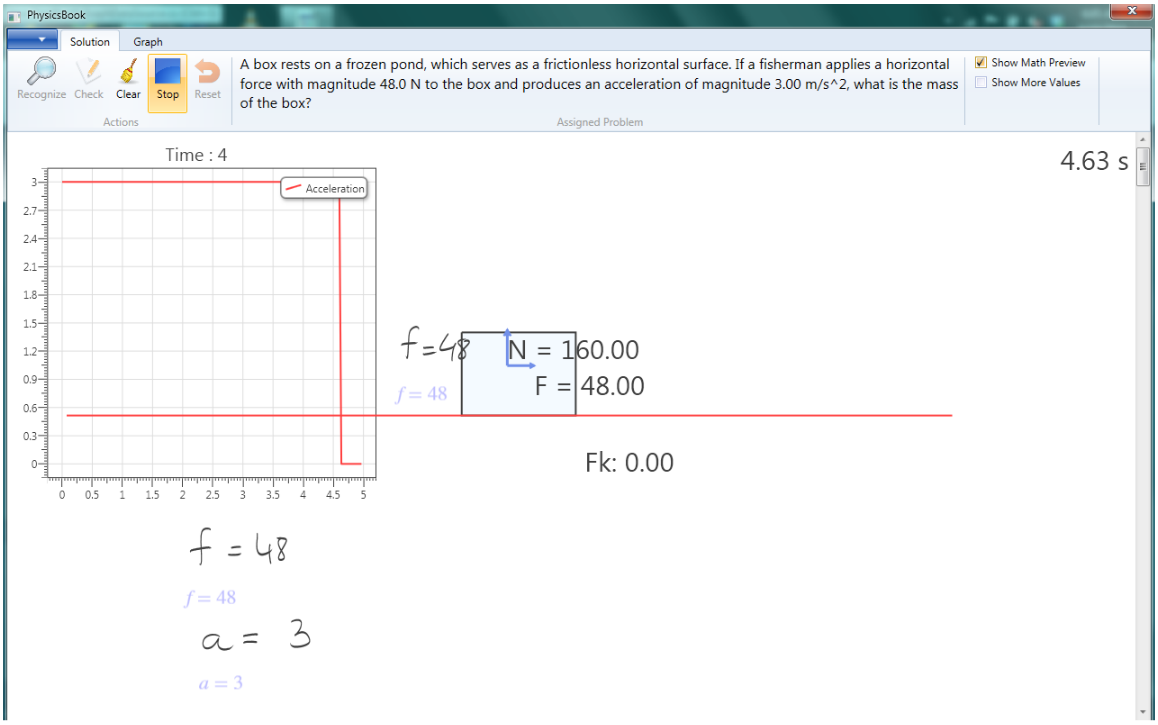
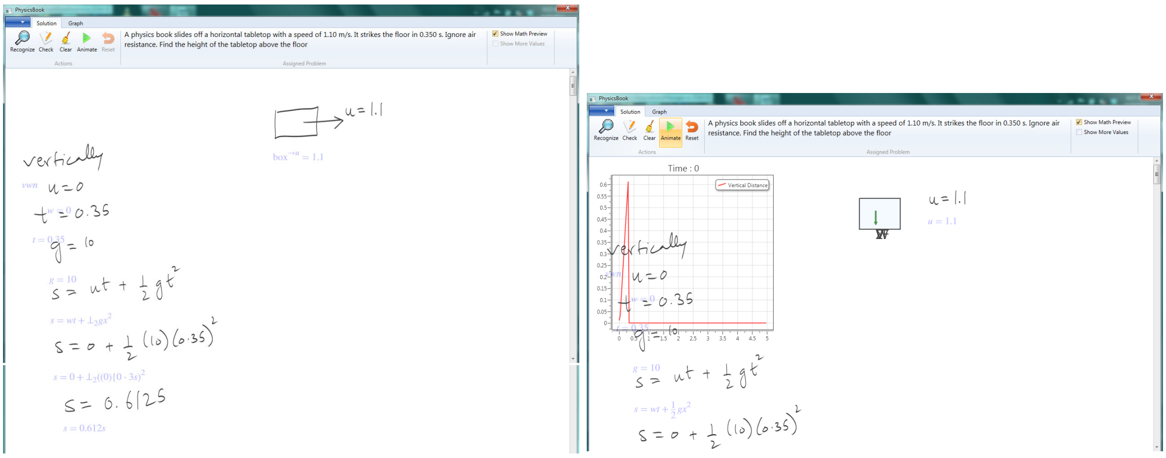


Figure 8.6: Animation for the scenario in Figure 8.5.

This scenario is not very different from doodling, from the animation point of view. A set of initial conditions are provided that yield an open-ended animation (no time or distance constraints). The result can be visually or graphically verified.



(a) Projectile Example

(b) Animation and Graph

Figure 8.7: Simple projectile scenario with limits on vertical movement. The recognized diagram and graph are also shown.

8.2.2 A Projectile Constrained in 1-dimension

The solution shown in Figure 8.7(a) requires the computation of vertical distance traveled in a given amount of time. This can be computed using motion equation $S = ut + \frac{1}{2}gt^2$. This example shows an interesting scenario where the answer $S = 0.6125$ is not used for animation at all. Instead, the diagram can be animated using only the information provided in the problem statement: initial velocity, time interval, and gravity. Figure 8.7(b) shows that when the book's vertical distance traveled is graphed, it comes out to a little over 0.6 meters. Graphing is again shown to be an important feedback mechanism.

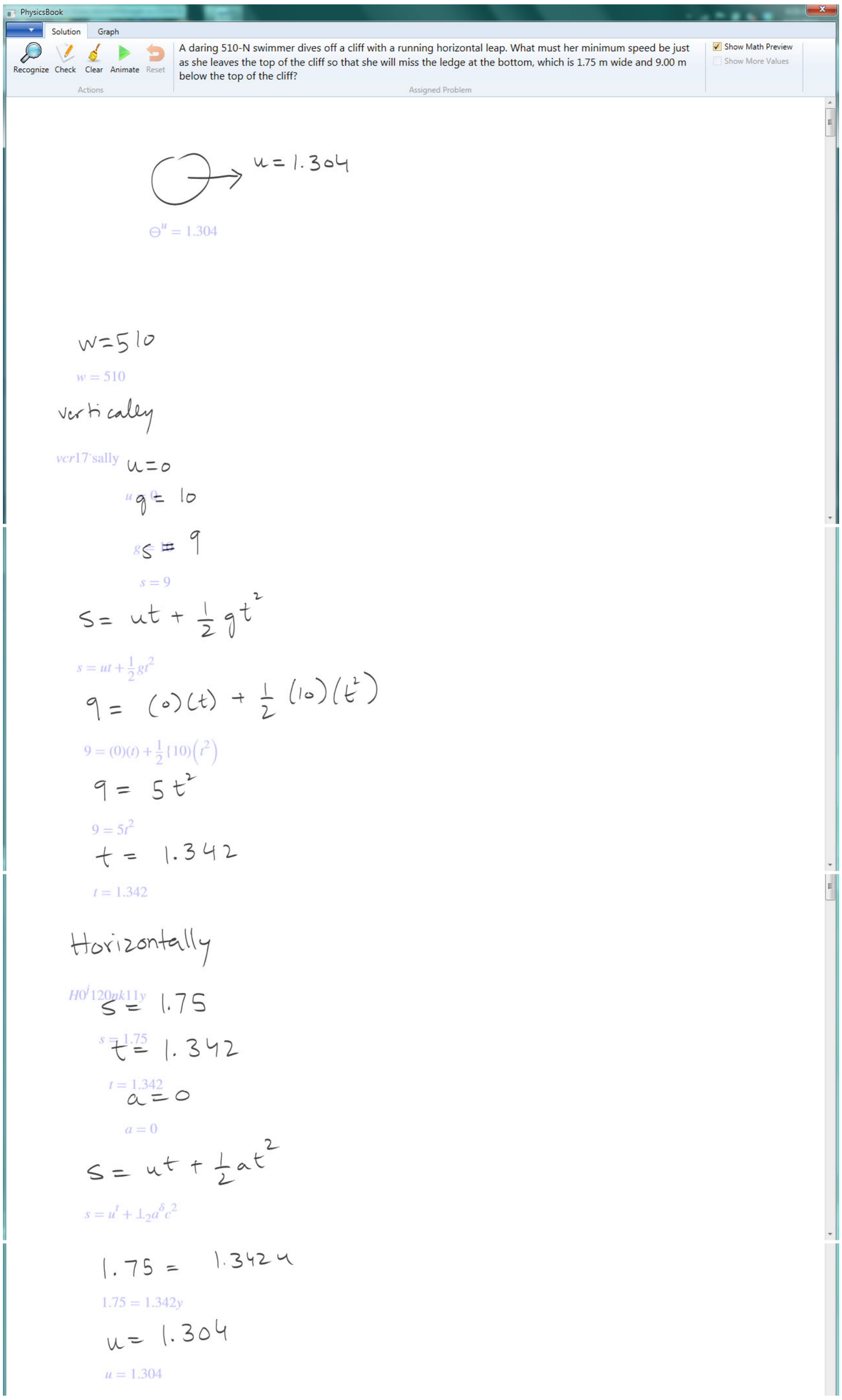


Figure 8.8: Projectile problem with limits on vertical and horizontal movement.

8.2.3 A Projectile Constrained in 2-dimensions

Figure 8.8 depicts another projectile problem. Here, both the vertical and horizontal distance traveled are constrained and the student is asked to solve for a speed such that both constraints are overcome. By applying the equations of motion in the vertical and horizontal directions, a minimum speed can be worked out. On the face of it, this scenario doesn't seem too different from Figure 8.7. However, from an animation perspective, there are important differences. Here, the answer is used for animation. Again graphing is used for feedback (Shown in Figure 8.9) but this scenario indicates that graphing is indispensable for cases where more than one quantities need to be observed simultaneously as functions of time. An alternative method for animation might have been to show the vertical and horizontal motion trails, but this approach would not show the comparison between the two quantities as functions of time.

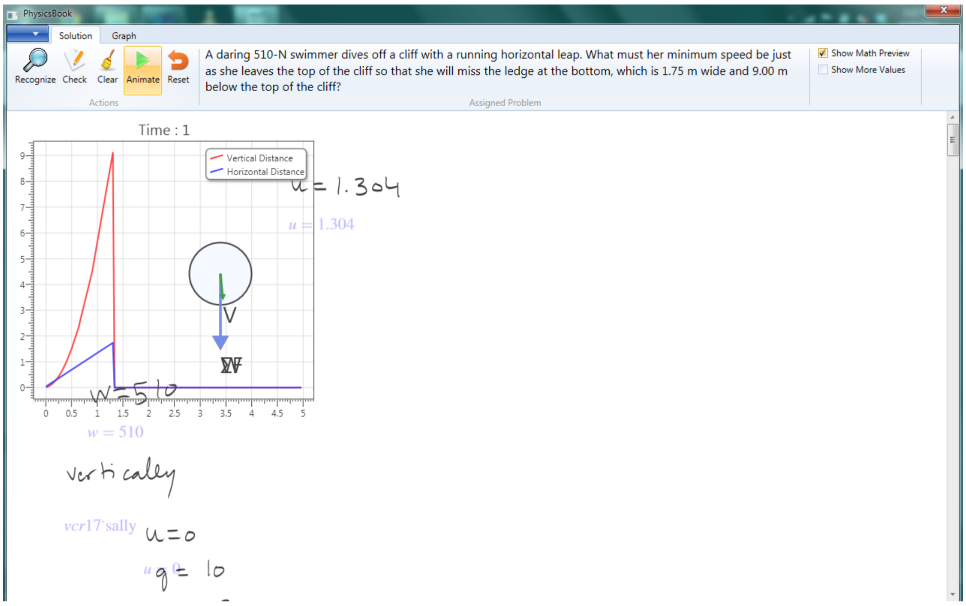


Figure 8.9: Animation and Graph for solution shown in Figure 8.8.

8.2.4 A Box Held in Equilibrium

The problem in Figure 8.10 is a slightly more complicated version of the problem presented in Section 8.1.3. Here, due to the different angle with the horizontal direction, the tension in both wires is not equal. Again, the diagram is beautified upon recognition, and each wire's angle with the horizontal is corrected upon association. The derived value for tension keeps the box in equilibrium. For incorrect values, the box would eventually break the wires and drop freely.

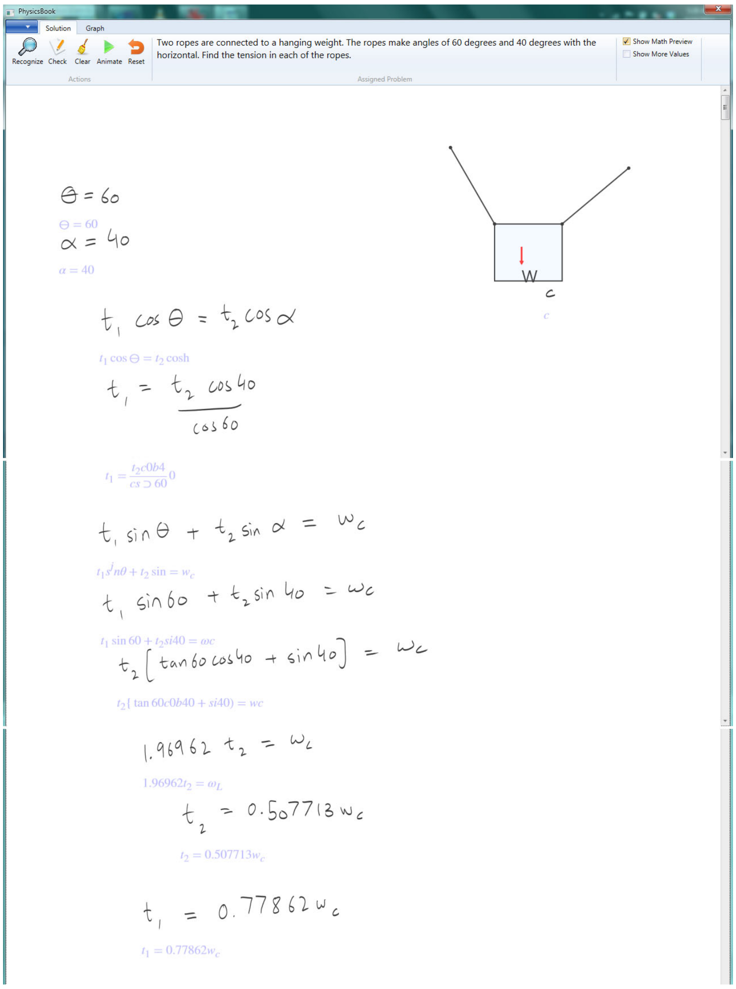


Figure 8.10: Equilibrium problem taken from Young's University Physics [226] and modeled using our system.

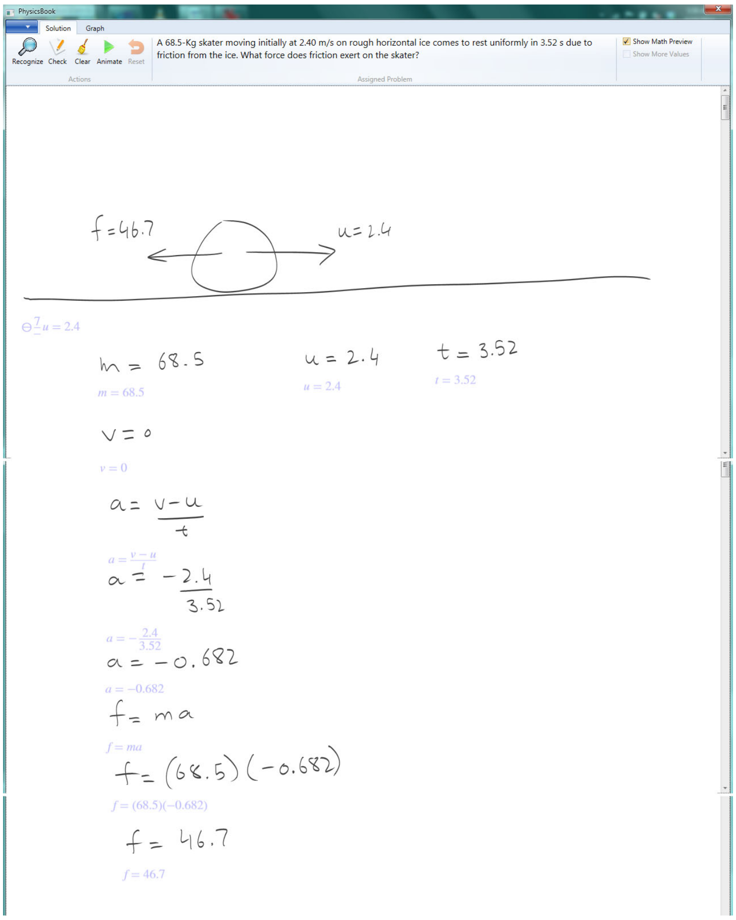


Figure 8.11: Sample Problem taken from Young's University Physics [226].

8.2.5 A Box Sliding on a Rough Surface

For Figure 8.11, the student must work out the friction force such that a moving ball comes to rest in a specified amount of time. Several factors affect the recognition and beautification of this scenario. The line denoting the surface is beautified so that it is perfectly horizontal. The circle is moved so that it is tangent to the line (to satisfy the touch constraint). In this scenario, the answer ($f = 46.7$) is not associated with the circle using the 'Lasso' + 'Tap' method. Instead, it is simply written down by the user as an arrow annotation. The user can indicate the time limit by using 'Lasso' to select $t = 3.52$ and tapping anywhere on the draw area. Please note that this is not necessary. If the value for the force is correct, the circle will stop in exactly 3.52 seconds. However, the moment may pass too quickly for a user to notice. To remedy this, graphing can be used to show that the circle comes to rest in the specified time. Alternatively, the time limit may be indicated, causing the system to stop animation when the limit is reached. In either case, the user will be able to check if he computed the initial conditions for the scenario correctly. Figure 8.12 shows the case where graphing is used for this purpose.

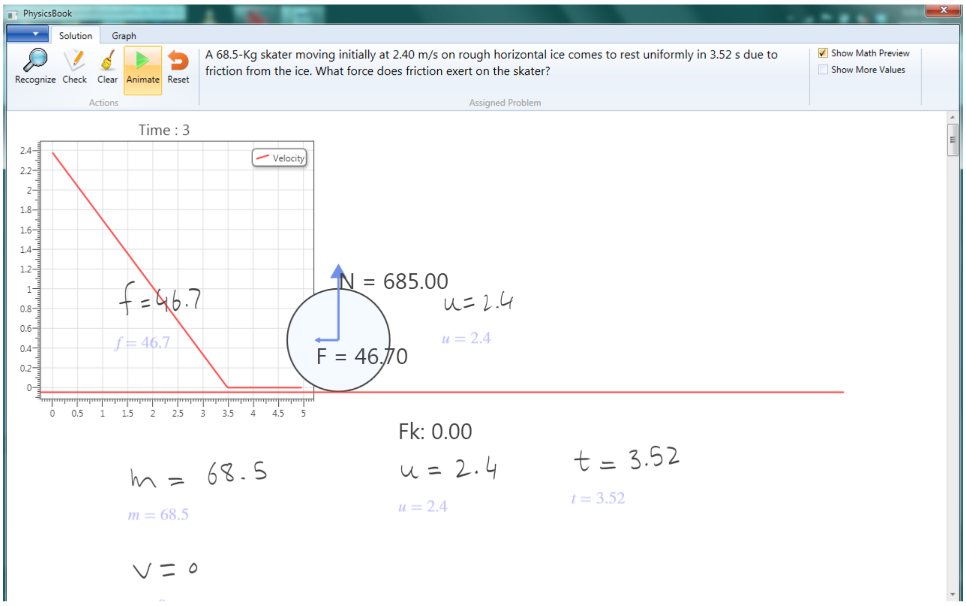


Figure 8.12: An animation of the friction problem in Figure 8.11, highlighting the use of graphing.

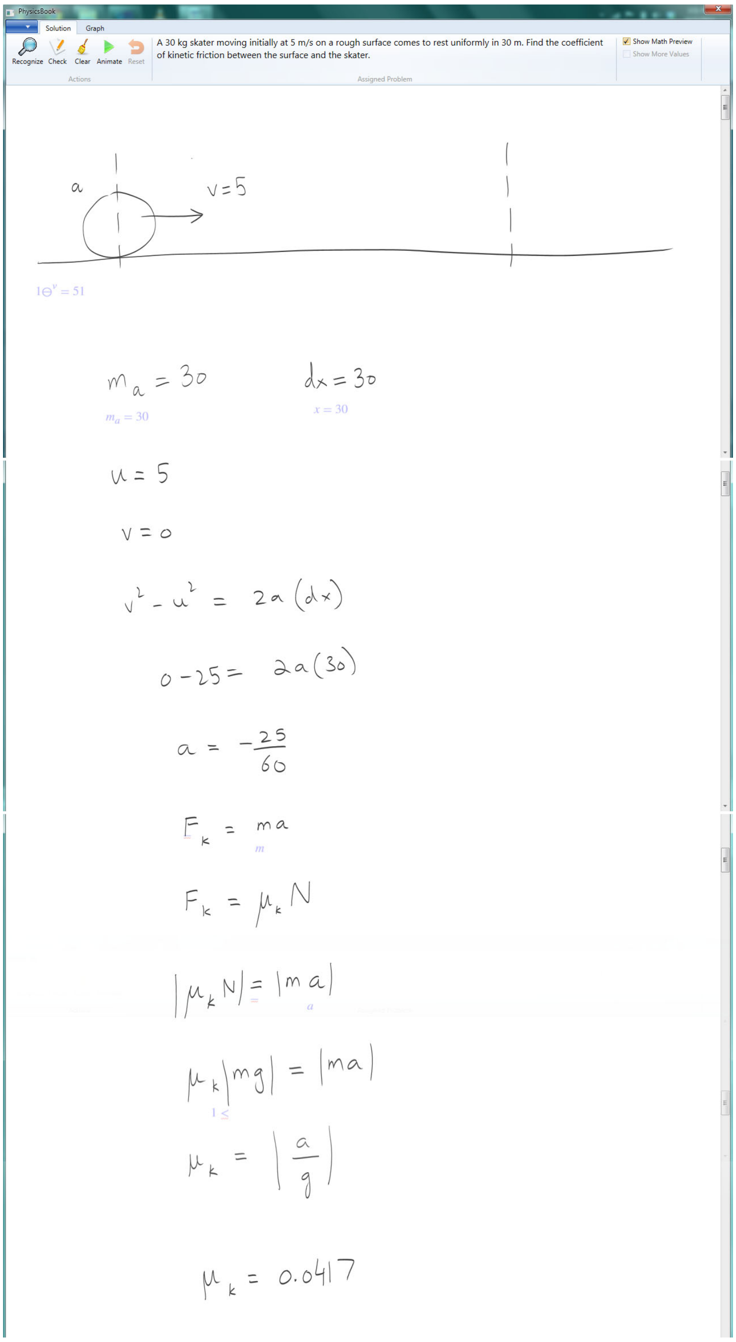


Figure 8.13: A student is asked to work out the coefficient of kinetic friction μ_k such that a moving ball comes to rest after traveling a given distance.

8.2.6 Computing the Coefficient of Kinetic Friction

The scenario in Figure 8.13 is more involved than the scenario presented in Figure 8.11. Here, instead of simply computing the friction force, the student must work out the correct coefficient of kinetic friction that will cause the ball to stop moving in a given distance. The processes of recognition and beautification are similar to Section 8.2.5. The recognized diagram is depicted in Figure 8.14. In this scenario, the initial velocity and mass are associated implicitly while the student must associate the distance and the coefficient of kinetic friction manually. To do this, they can Lasso the $dx = 30$ and tap on of the dotted lines. Similarly, selecting $\mu_k = 0.0417$ and tapping the ball associates the coefficient of kinetic friction. When the animation is run, the ball comes to rest in the correct distance. A different value for μ_k would cause the ball to either stop before the appropriate distance or to overshoot the distance limit.

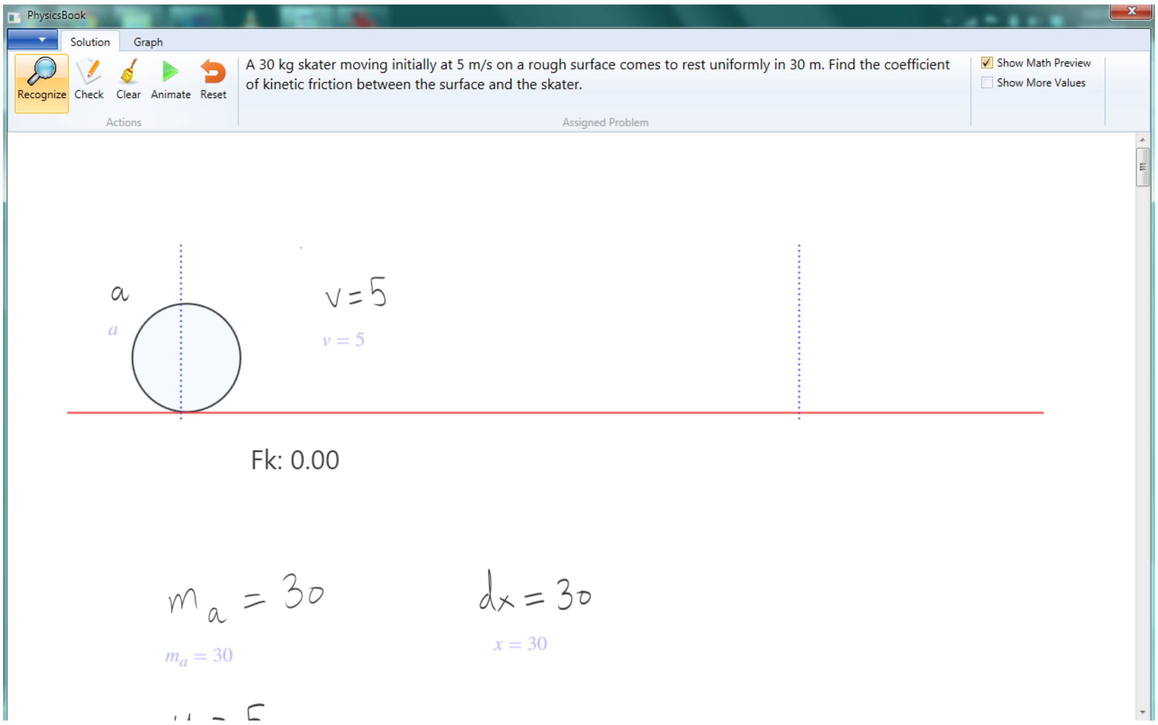


Figure 8.14: An animation of the friction problem in Figure 8.13, highlighting the use of graphing.

Modeling this type of physics problem requires the modeling of static and kinetic friction forces. The Friction simulator in our Animation Runtime is capable of modeling these interactions. This enables it to generate animations for cases where either an object moving on a rough surface starts from rest or is already moving with a steady velocity.

8.2.7 Elastic Collisions

As mentioned previously, the momentum simulator in our prototype can handle limited cases of elastic collisions in 1-Dimension. Figure 8.15. Here, two marbles (labeled ‘a’ and ‘b’) are moving toward each other at different velocities. In this instance, the event of collision is denoted by a single vertical dotted line. The before- and after- collision velocities of both marbles are written using horizontal arrows, with labels. Using physics formulae for perfectly elastic collisions, the student derives the after collision velocities as $v_a = -1.02$ and $v_b = 0.88$. When associated the equations for before- and after- collision velocities, our system ignores the $-$ sign in the equations, instead inferring the direction from the recognized arrow. This process is invisible to the user.

This problem is another example of a scenario where the answer itself need not be associated with the recognized diagram. Here, the initial conditions are sufficient to run the simulation. After the collision occurs, the student can visually check the velocities of both marbles to see if they match the derived answer.

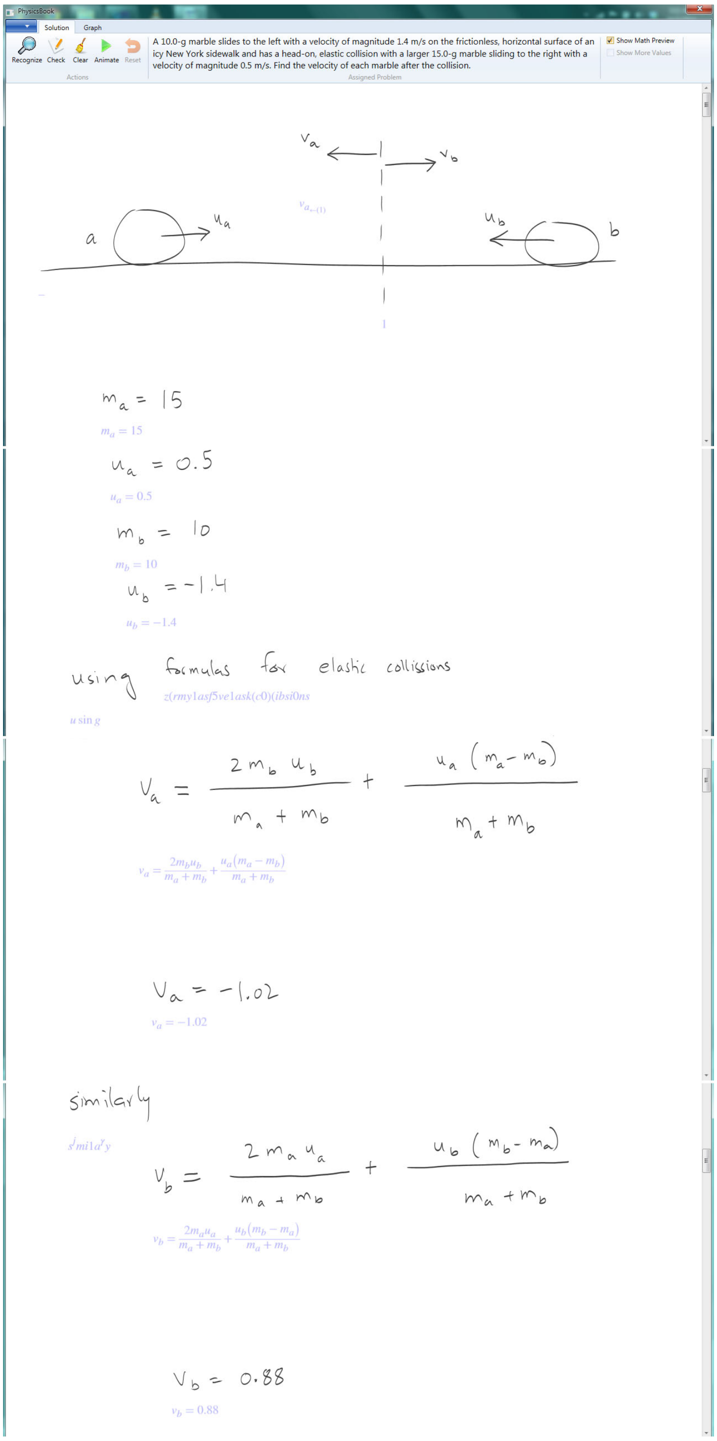


Figure 8.15: An example of a scenario with perfectly elastic collisions in 1-Dimension.

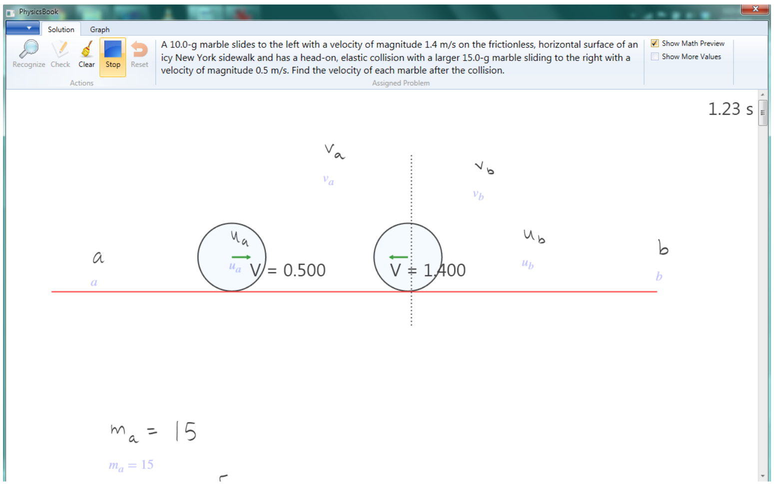


Figure 8.16: A snapshot of the animation for Figure 8.15, before collision happens.

8.2.8 Using the Work-Energy Theorem to Calculate Initial Velocity

The scenario presented in Figure 8.18 uses the work-energy theorem to compute the initial velocity of a projectile fired vertically. While the physics concepts tested are different from all the examples presented in this section, the animation mechanism is very similar. Here, again, we have a system of moving objects where we can indicate initial conditions and are required to verify the value of some quantity at a known point in time, starting from time $t = 0$. This can be accomplished by using the graphing functionality.

8.3 Observed Animation Patterns in Modeled Solutions

By analyzing the examples in the previous section, we can identify some common patterns. It appears that, from an animation perspective, physics problems and solutions tend to fall into three broad categories:

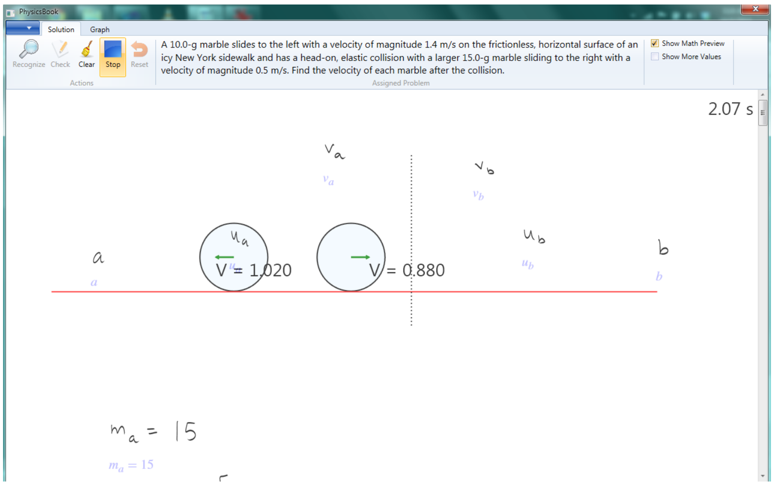


Figure 8.17: An after-collision snapshot of the animation for Figure 8.15, showing that the student derived the correct values for velocity.

S1. Open-Ended Animation: Typically, in such scenarios, the initial conditions are either known or have been computed, in part, by the student. The student is expected to reason about or assert the correctness of some condition throughout the time frame of the simulation. In some instances, the student is expected to reason about the properties of an event during the course of the simulation. An example is inelastic and elastic collision problems. Predicting the time and place of such events is a non-trivial problem.

S2. Time-of-Interest Animation: Again, the initial conditions for the simulation are known or have been computed by the student. In time-of-interest simulations, the student is expected to reason about or assert the correctness of some condition at one or more points in time during the simulation. If only one time value $t = t_1$ is of interest, then this scenario is reducible to running a simulation from time $t = 0 \rightarrow t_1$.

S3. Point-of-Interest Animation Once again, the initial conditions are either known or computed. In such simulation, one or more points in the trajectory of a simulation object are deemed interesting. Some condition is held to be true or false, or a question about the value of some

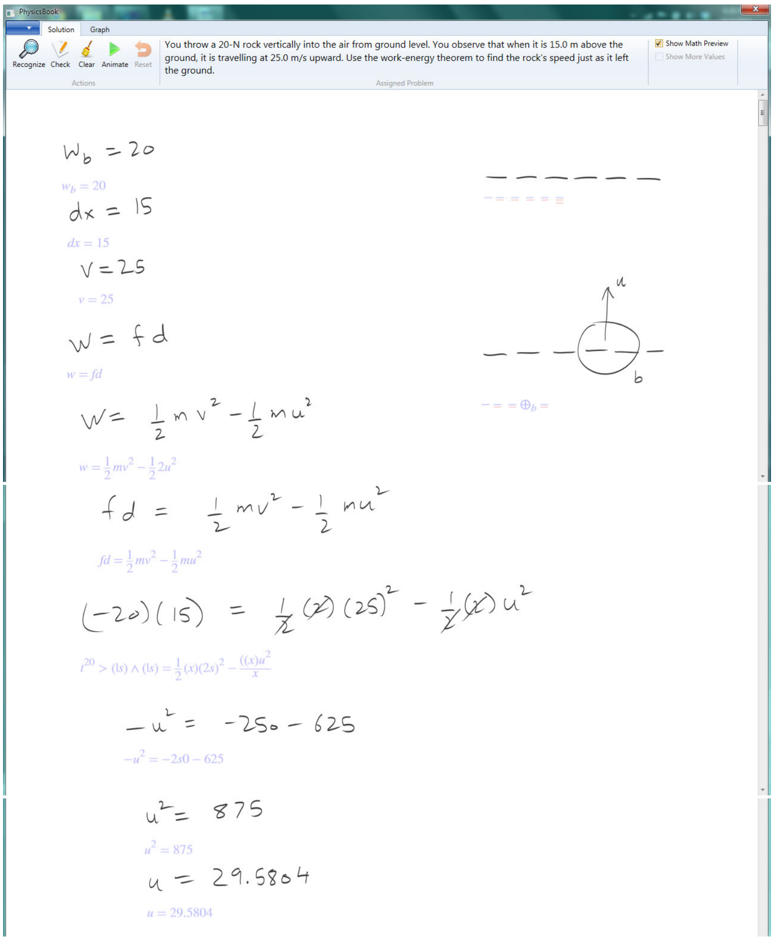


Figure 8.18: An Example of a scenario that uses the Work-Energy Theorem.

physical quantity is asked at each point of interest. A representative example is ”At what two points in its path, is a projectile above x meters?”

For time-of-interest and point-of-interest simulations, it is important to know the interesting points and time values for each simulation element. Our prototype allows a user to write an expression for time, e.g. $t = 5$ and associate it with the simulation by tapping the canvas. Similarly, the use of parallel dotted lines allows a user to define a range of motion. Our prototype can infer some of the limits in time- and space-bounded simulations by the use of annotations and spatial reasoning. If we assume that an easy method can be devised to enable the user to indicate all interesting points in time and trajectory, then we are faced with the immense challenge of tailoring the simulation with respect to each interesting point (because the same physical property need not be tested at each interesting point).

8.4 Known Cases That Cannot Be Modeled Using our Approach

Some types of physics problems are obviously outside the scope of our prototype system, because we have no simulation engine to deal with them. Examples include circuit analysis, fluid dynamics, gravitation, etc. A more interesting question is:

What are scenarios from the chosen domains that cannot be handled by our current prototype?

This question is important for understanding the effectiveness of our overall approach. We can gain insight into this question by examining it in the view of the categories of animation identified in Section 8.3. This section discusses particular mechanisms that are not supported in our current prototype.

8.4.1 Kinematics Problems

Our prototype system allows users to associate any kind of quantity with a diagram element. Scalar quantities require no special processing while direction for vector quantities can be indicated via arrows. Some quantities such as mass, velocity, time, weight, force, etc are well-known and can be processed directly by our system. For some others, realtime data transformations can be used.

The main limitation for our kinematics simulator is defining points of interest. Currently, we may only define a single point in time (some $t = t_1$) and can only start the simulation at $t = 0$.

We cannot handle scenarios where the simulation starts at some $t = t_1$, while the initial conditions are provided for some other point in time $t = t_2$, where $t_2 < t_1$. Our system always assumes that the initial conditions indicated only hold at beginning of the simulation. This behavior can be extended by providing support for modeling progress from initial conditions, $t = 0$, to some other state, $t = t_1$ internally, and then allowing a user to view the remainder of the simulation. There is also no support for defining multiple points of interest (either in trajectory or time). This inability of defining multiple points of interest extends to all simulators in our prototype system. To overcome this weakness, two things are needed. First, a robust interaction metaphor must be devised that allows a user to indicate the interesting points in trajectory or time to the simulator. Second, the simulation must be able to proceed in segments, with possibly support needed for handling some segments internally.

8.4.2 Friction and Sliding Contact Problems

Figure 8.19 shows a friction problem that involves pulley systems. A box labeled 'A' of weight $45N$ rests on a rough table and is attached by a wire running over a pulley to a free-hanging block labeled 'B' of mass $25N$. When the system is set in motion, the free hanging block descends at a constant velocity. The student is asked to work out the coefficient of kinetic friction between the table and box 'A'. Using the principle that constant velocity implies zero acceleration and in turn, zero force, the student computes an expression for $\mu_k = 0.555$.

Our prototype system can accurately recognize and beautify this system, as shown in Figure 8.19. However, the simulation engine is not sophisticated enough to model this system properly. In our implementation, the system would remain at rest, starting from $t = 0$, because kinetic friction would prevent any net force, resulting in zero motion.

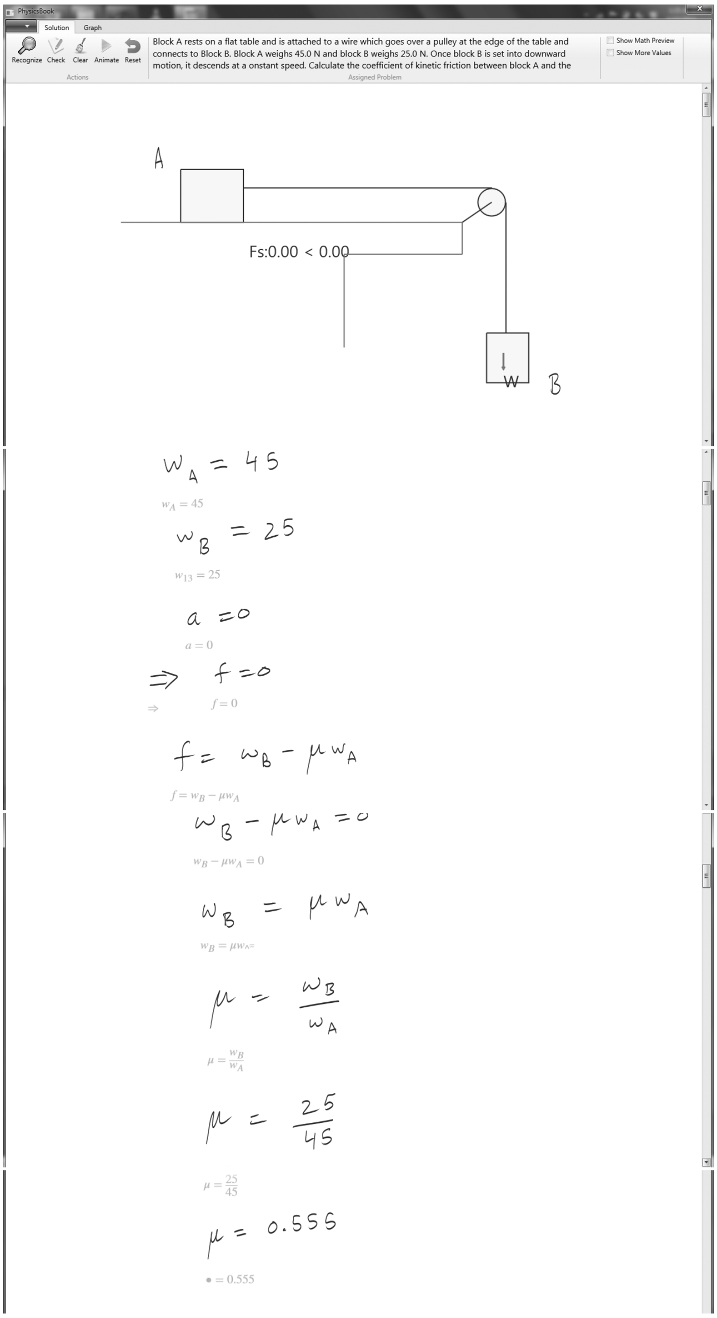


Figure 8.19: A weight hanging from a pulley pulls a box resting on a rough surface. The student is asked to work out the coefficient of kinetic friction that will cause the box to move at a constant speed.

This limitation is grounded in the design of our simulation engine for such problems, which is set up only to deal with instances where a system is either released from rest[2] or is in a steady state[3]. In this particular instance, the problem statement specifically states that *the system is set into motion*, implying an outside influence, which our prototype cannot model. If the problem were modified to give a value for the velocity at which either box is moving, then this can be animated using our prototype ITS.

8.4.3 Momentum Problems

The most important limitation of the momentum simulator is lack of functionality. It is currently set up only to deal with limited instances of 1-D elastic collisions, and provides no support for inelastic collisions, 2D collisions of any sort, or for impulse problems. Figure 8.20 shows a simple 1-dimensional scenario about a collision between two pucks 'A' and 'B'. The after-collision velocities of both 'A' and 'B' are provided. Additionally, the problem stipulates that 'B' is initially at rest, asking the student to work out the initial velocity of 'A'. Using the principle of conservation of momentum, the correct value of the velocity of 'A' can be derived. However, in this case, the problem statement does not indicate if the collision was elastic or inelastic. As our momentum simulator can properly animate only perfectly elastic collisions, this scenario may yield an incorrect looking simulation. Such instances are particularly problematic because they require prior knowledge, usually provided by the teacher or some domain expert. Automatic identification of missing information such as this is a difficult problem. The momentum simulator in our prototype is just a proof-of-concept. However, missing prior knowledge can pose problems for mature systems as well, because it requires the system to essentially solve the problem and reason about the solution, in order to identify the missing information.

[2] Equivalent to saying that simulation begins at $t = 0$.

[3] For example, a ball moving at a constant velocity comes to rest under the influence of kinetic friction.

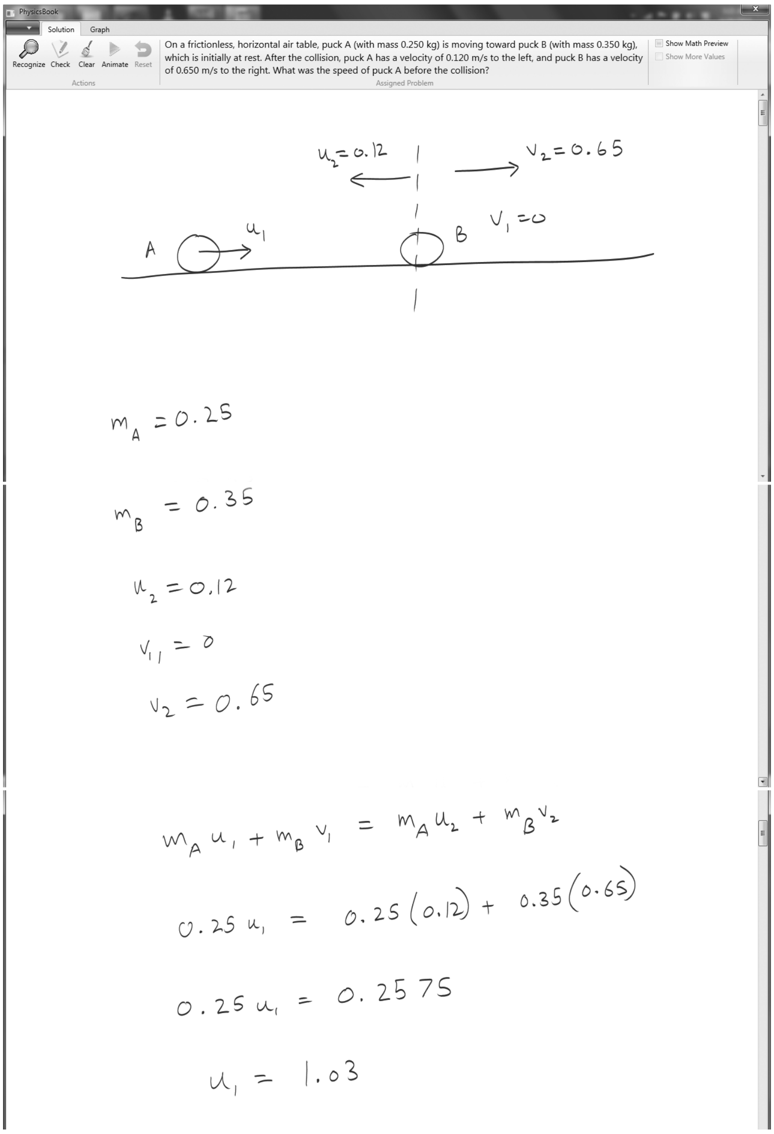


Figure 8.20: Example scenario depicting an elastic collision between two objects moving on a 1-dimensional surface.

8.4.4 Equilibrium Problems

The limitations highlighted for the other simulators also apply to the equilibrium simulator. Like the momentum simulator, this is a proof of concept and only provides limited support for a small subset of equilibrium problems.

8.5 Summary of Supported Animations

Animation in our prototype begins when the user hits the 'Animate' button and ends either when the 'Stop' button is hit or a user specified time limit is reached. Our prototype provides support for the following types of animation:

1. Reasoning about Motion: This can happen in two ways. First, a simulation element that is meant to be immobile moves as a result of the student's answer. Second, some interesting aspect of the movement of a simulation element is wrong. For example, an object that is supposed to move with a constant velocity develops an acceleration. In this way, a student can judge if his or her answer has some mistake. The method is particularly useful for modeling equilibrium scenarios where a system is released from rest. A simple visual test or a graph-based view can be used to identify anomalies. These behaviors can be applied to all categories of physics problems (open-ended, time- and space-bounded).

2. Defining Time Limits: Each simulation starts at time $t = 0$ and can be stopped either on explicit command or by hitting a user defined time limit. Allowing a user to specify time limits is useful for modeling two different types of scenarios: a) student is asked to compute some initial value that affects some aspect of motion in a defined way at a particular time or b) student is asked to compute the value of some quantity that should result from the initial conditions specified in the problem statement. Again, in both of these cases, visual testing and graphing can be used to identify anomalous animation behavior.

3. Defining Distance Limits: Similar to previous point. Here, the start and stop points of motion are identified using vertical or horizontal intervals (denoted by parallel dotted lines).

4. Reasoning about Events: An interesting event is supposed to happen at a particular time. This behavior is applicable when predicting the time of the event beforehand is a non-trivial

operation. In particular, this method is useful for collision problems, where it is usually mentioned that a collision happens (without time and location information[4]).

[4] It should be noted that although, the time and location can be computed, this is an extraneous operation that would not add too much value to the problem statement.

CHAPTER 9
ANALYSIS OF STUDENT SOLUTIONS

The previous chapter described how real-world physics problems could be modeled and animated using our prototype system. We also discussed its limitations. During the analysis, it became apparent that the animations for physics problems could be categorized in three distinct ways (See Section 8.3). We believe that these three categories of animation are sufficient to model a wide variety of physics solutions[1]. The three animation categories identified earlier were:

- Open-Ended: No time limits defined, Student answer effects scenario for undefined time frame. Reasoning-about-motion type problems are a subset of this category.
- Time-of-Interest: One or more moments in time are interesting. At each such moment, the student must either compute a quantity or reason about some physical aspect of specified scenario. If only one time value $t = t_1$ is of interest, then this category is reducible to running a simulation from time $t = 0 \rightarrow t_1$. Otherwise, it can be generalized as a series of simulations ($t = t_0 \rightarrow t_1 \rightarrow t_2 \cdots \rightarrow t_n$).
- Point-of-Interest In this category, one or more points in the trajectory of a simulation object are deemed interesting. Some condition is held to be true or false, or a question about the value of some physical quantity is asked at each point of interest.

[1]There is a caveat to this claim. The answer to a physics problem can either be a decision (e.g., given this scenario and some initial conditions, does something happen?) or a computation (e.g. what is the value of "x" at $t = t_1$ seconds or after traveling "k" meters). For computation problems involving some form of motion, the categories identified above are obviously adequate. Decision problems are trickier. Yet, they also conform to the same three categories, because the student must do some computation to support the decision. The answers computed for decision problems can therefore be used for animation in the same way as for computation problems: by defining initial conditions, and observing the resulting animation to see if it conforms to the decision.

Our current prototype provides limited support for these types of animations. The key challenges include 1) lack of interaction metaphors to allow users to fully specify required parameters of animation and 2) insufficient understanding of what feedback mechanisms are contextually appropriate for each animation category. For specifying the parameters of the animation in a natural way, our prototype uses implicit associations and the 'Lasso' + 'Tap' gesture command (itself leveraged from Mathematical Sketching [119] and our earlier prototypes [40, 41, 42]). This method appears suitable for specifying small sets of animation parameters but may be cumbersome for defining all required parameters (which can be quite large in complex scenarios). In fact, we currently do not know of a good interaction metaphor to allow a user to define a large number of parameters for a simulation.

The question of contextually-appropriate feedback is similarly hard to address. Our prototype provides feedback in three ways. First, the values for interesting quantities are shown during animation (as arrows and floating text). Second, there is visual confirmation. This enables users to judge the correctness of the animation (and thus the answer) by intuition about physically correct behavior. Third, we provide users with the ability to generate runtime graphs of various quantities, allowing them to customize feedback as they see fit. These methods of providing feedback seem appropriate for the problems discussed in Chapter 8, but we do not know how well they generalize or if they are adequate.

9.1 A Database of Physics Problems and Student Solutions

We conducted a user study[2] to construct a database of solutions to physics problems that would help us identify usage scenarios and discover animation mechanisms for pen-based intelligent tutors. A set of 50 physics problems was chosen from Young's University Physics 13th Edition [226] for this

[2]IRB documentation for the user study is listed in Appendix B.

purpose. The chosen problems are listed in Table 9.1. We initially aimed to collect 3-5 solutions for each problem. However, due to unavailability of participants, our database currently contains 56 solutions, collected from 8 students.

9.1.1 Subjects and Apparatus

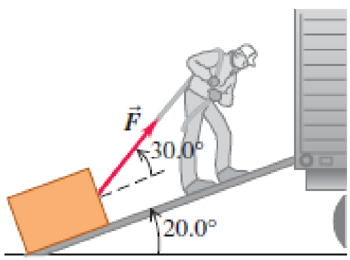
We recruited 8 students (7 male and 1 female) from the University of Central Florida for participation in the study. The ages of participants were between 19 and 25 years. Each participant was asked to write solutions to 7 randomly selected problems from Table 9.1. Each participant took 90-120 minutes to complete the experiment and was paid $10 for their time. Participants were provided with a Physics textbook to look up concepts and a scientific calculator to aid in computations. They were instructed to write numeric quantities correct to four decimal places (where appropriate). Additionally, they were informed that they could voluntarily end the experiment at any time.

The experiment was conducted on an HP EliteBook 2760p multi-touch tablet computer equipped with an Intel Core-i5-2410M processor and four gigabytes of memory. The screen resolution was set at 1200x800 pixels. We disabled multi-touch interaction on the tablet which was placed on a table for the experiment. A separate computer was used to display problem statements to the participants. We recorded all ink strokes written by the participant for each problem, including position (x,y), pressure and timing data. The dataset of student solutions thus constructed can be downloaded from the ISUElab website (`http://eecs.ucf.edu/isuelab/downloads.php`).

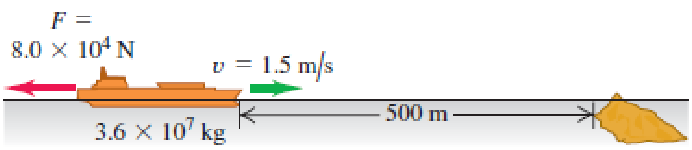
9.2 Animation Requirements for Chosen Physics Problems

The 50 problems chosen for the experiment are listed in Table 9.1. The third column of the table describes the category of animation that can be used to model each problem. The categories are: Open-Ended, Point-of-Interest and Time-of-Interest. Some problems are multi-category, i.e., they can be animated using mechanisms from more than one categories.

Table 9.1: List of 50 problems selected from Young's University Physics 13th Edition [226].

ID & Page No	**Physics Problem**	**Animation Requirement**
K-01 (129)	**4.4** • A man is dragging a trunk up the loading ramp of a mover's truck. The ramp has a slope angle of 20.0°, and the man pulls upward with a force $\vec{F}$ whose direction makes an angle of 30.0° with the ramp (Fig. E4.4). (a) How large a force $\vec{F}$ is necessary for the component F_x parallel to the ramp to be 60.0 N? (b) How large will the component F_y perpendicular to the ramp then be? Figure **E4.4** 	Open-Ended
K-02 (129)	**4.7** •• A 68.5-kg skater moving initially at 2.40 m/s on rough horizontal ice comes to rest uniformly in 3.52 s due to friction from the ice. What force does friction exert on the skater?	Time-of-Interest
K-03 (129)	**4.9** • A box rests on a frozen pond, which serves as a frictionless horizontal surface. If a fisherman applies a horizontal force with magnitude 48.0 N to the box and produces an acceleration of magnitude 3.00 m/s^2, what is the mass of the box?	Open-Ended

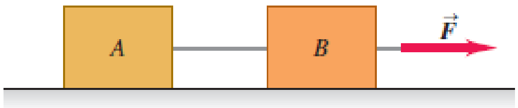
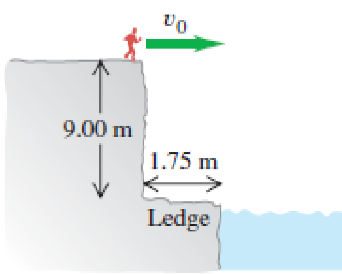
Continued on next page

ID (Page No)	Physics Problem	Animation Requirement
K-04 (129)	**4.10** •• A dockworker applies a constant horizontal force of 80.0 N to a block of ice on a smooth horizontal floor. The frictional force is negligible. The block starts from rest and moves 11.0 m in 5.00 s. (a) What is the mass of the block of ice? (b) If the worker stops pushing at the end of 5.00 s, how far does the block move in the next 5.00 s?	Time-of-Interest, Point-of-Interest
K-05 (130)	**4.33** CP A 4.80-kg bucket of water is accelerated upward by a cord of negligible mass whose breaking strength is 75.0 N. If the bucket starts from rest, what is the minimum time required to raise the bucket a vertical distance of 12.0 m without breaking the cord?	Time-of-Interest, Point-of-Interest
K-06 (130)	**4.38** • CP An oil tanker's engines have broken down, and the wind is blowing the tanker straight toward a reef at a constant speed of 1.5 m/s (Fig. P4.38). When the tanker is 500 m from the reef, the wind dies down just as the engineer gets the engines going again. The rudder is stuck, so the only choice is to try to accelerate straight backward away from the reef. The mass of the tanker and cargo is 3.6×10^7 kg, and the engines produce a net horizontal force of 8.0×10^4 N on the tanker. Will the ship hit the reef? If it does, will the oil be safe? The hull can withstand an impact at a speed of 0.2 m/s or less. You can ignore the retarding force of the water on the tanker's hull. Figure **P4.38**  	Open-Ended

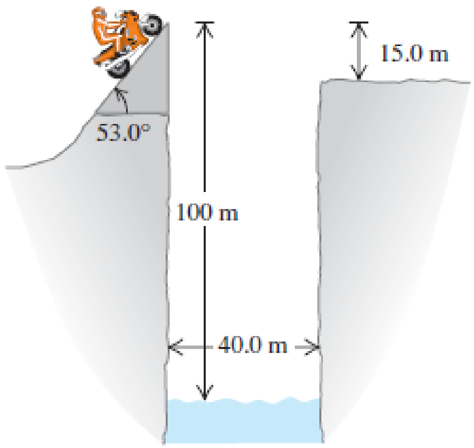
Continued on next page

ID (Page No)	Physics Problem	Animation Requirement
K-07 (165)	**5.16** •• **CP** A 8.00-kg block of ice, released from rest at the top of a 1.50-m-long frictionless ramp, slides downhill, reaching a speed of 2.50 m/s at the bottom. (a) What is the angle between the ramp and the horizontal? (b) What would be the speed of the ice at the bottom if the motion were opposed by a constant friction force of 10.0 N parallel to the surface of the ramp?	Point-of-Interest
K-08 (165)	**5.22** •• **CP** **CALC** A 2540-kg test rocket is launched vertically from the launch pad. Its fuel (of negligible mass) provides a thrust force so that its vertical velocity as a function of time is given by $v(t) = At + Bt^2$, where A and B are constants and time is measured from the instant the fuel is ignited. At the instant of ignition, the rocket has an upward acceleration of 1.50 m/s^2 and 1.00 s later an upward velocity of 2.00 m/s. (a) Determine A and B, including their SI units. (b) At 4.00 s after fuel ignition, what is the acceleration of the rocket, and (c) what thrust force does the burning fuel exert on it, assuming no air resistance? Express the thrust in newtons and as a multiple of the rocket's weight. (d) What was the initial thrust due to the fuel?	Time-of-Interest
K-09 (166)	**5.28** •• A box of bananas weighing 40.0 N rests on a horizontal surface. The coefficient of static friction between the box and the surface is 0.40, and the coefficient of kinetic friction is 0.20. (a) If no horizontal force is applied to the box and the box is at rest, how large is the friction force exerted on the box? (b) What is the magnitude of the friction force if a monkey applies a horizontal force of 6.0 N to the box and the box is initially at rest? (c) What minimum horizontal force must the monkey apply to start the box in motion? (d) What minimum horizontal force must the monkey apply to keep the box moving at constant velocity once it has been started? (e) If the monkey applies a horizontal force of 18.0 N, what is the magnitude of the friction force and what is the box's acceleration?	Open-Ended

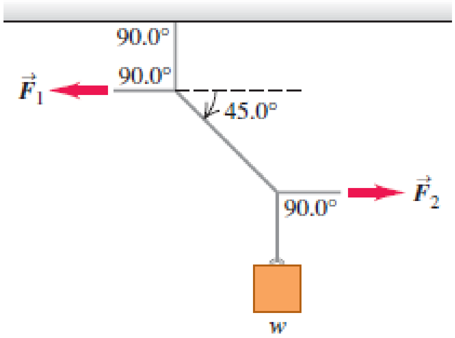
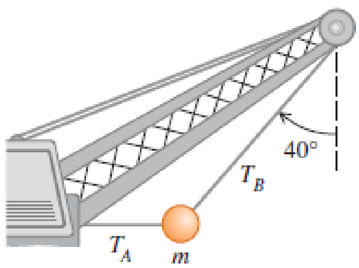
Continued on next page

ID (Page No)	Physics Problem	Animation Requirement
K-10 (166)	**5.35** • Two crates connected by a rope lie on a horizontal surface (Fig. E5.35). Crate A has mass m_A and crate B has mass m_B. The coefficient of kinetic friction between each crate and the surface is μ_k. The crates are pulled to the right at constant velocity by a horizontal force $\vec{F}$. In terms of m_A, m_B, and μ_k, calculate (a) the magnitude of the force $\vec{F}$ and (b) the tension in the rope connecting the blocks. Include the free-body diagram or diagrams you used to determine each answer. Figure **E5.35** 	Open-Ended
K-11 (96)	**3.9** • A physics book slides off a horizontal tabletop with a speed of 1.10 m/s. It strikes the floor in 0.350 s. Ignore air resistance. Find (a) the height of the tabletop above the floor; (b) the horizontal distance from the edge of the table to the point where the book strikes the floor; (c) the horizontal and vertical components of the book's velocity, and the magnitude and direction of its velocity, just before the book reaches the floor. (d) Draw x-t, y-t, v_x-t, and v_y-t graphs for the motion.	Time-of-Interest
K-12 (96)	**3.10** •• A daring 510-N swimmer dives off a cliff with a running horizontal leap, as shown in Fig. E3.10. What must her minimum speed be just as she leaves the top of the cliff so that she will miss the ledge at the bottom, which is 1.75 m wide and 9.00 m below the top of the cliff? Figure **E3.10** 	Point-of-Interest

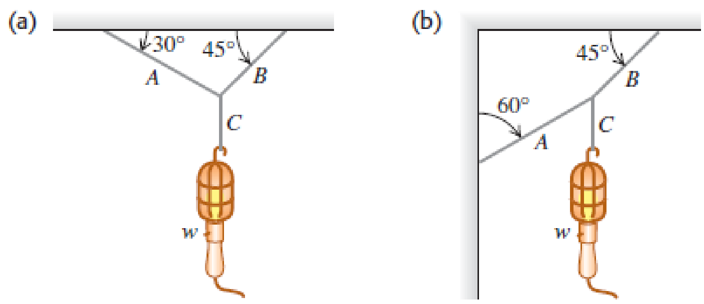
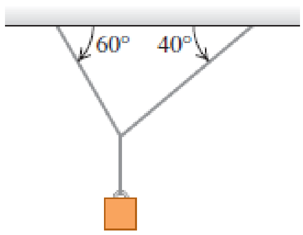
Continued on next page

ID (Page No)	Physics Problem	Animation Requirement
K-13 (97)	**3.17** • A major leaguer hits a baseball so that it leaves the bat at a speed of 30.0 m/s and at an angle of 36.9° above the horizontal. You can ignore air resistance. (a) At what *two* times is the baseball at a height of 10.0 m above the point at which it left the bat? (b) Calculate the horizontal and vertical components of the baseball's velocity at each of the two times calculated in part (a). (c) What are the magnitude and direction of the baseball's velocity when it returns to the level at which it left the bat?	Point-of-Interest
K-14 (101)	**3.67** •• **Leaping the River II.** A physics professor did daredevil stunts in his spare time. His last stunt was an attempt to jump across a river on a motorcycle (Fig. P3.67). The takeoff ramp was inclined at 53.0°, the river was 40.0 m wide, and the far bank was 15.0 m lower than the top of the ramp. The river itself was 100 m below the ramp. You can ignore air resistance. (a) What should his speed have been at the top of the ramp to have just made it to the edge of the far bank? (b) If his speed was only half the value found in part (a), where did he land? Figure **P3.67** 	Point-of-Interest

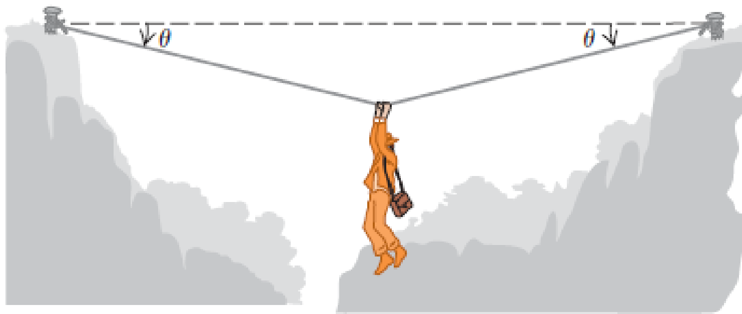
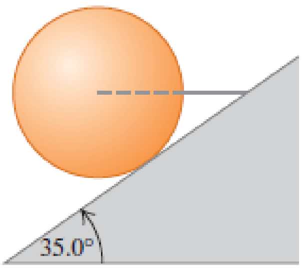
Continued on next page

ID (Page No)	Physics Problem	Animation Requirement
E-01 (163)	**5.1** • Two 25.0-N weights are suspended at opposite ends of a rope that passes over a light, frictionless pulley. The pulley is attached to a chain that goes to the ceiling. (a) What is the tension in the rope? (b) What is the tension in the chain?	Open-Ended
E-02 (164)	**5.10** •• In Fig. E5.10 the weight w is 60.0 N. (a) What is the tension in the diagonal string? (b) Find the magnitudes of the horizontal forces $\vec{F}_1$ and $\vec{F}_2$ that must be applied to hold the system in the position shown. Figure **E5.10** 	Open-Ended
E-03 (164)	**5.6** •• A large wrecking ball is held in place by two light steel cables (Fig. E5.6). If the mass m of the wrecking ball is 4090 kg, what are (a) the tension T_B in the cable that makes an angle of 40° with the vertical and (b) the tension T_A in the horizontal cable? Figure **E5.6** 	Open-Ended

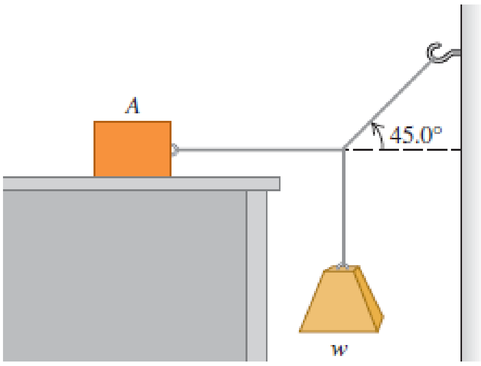
Continued on next page

ID (Page No)	Physics Problem	Animation Requirement
E-04 (164)	**5.7** •• Find the tension in each cord in Fig. E5.7 if the weight of the suspended object is w. Figure **E5.7** 	Open-Ended
E-05 (168)	**5.57** ••• Two ropes are connected to a steel cable that supports a hanging weight as shown in Fig. P5.57. (a) Draw a free-body diagram showing all of the forces acting at the knot that connects the two ropes to the steel cable. Based on your force diagram, which of the two ropes will have the greater tension? (b) If the maximum tension either rope can sustain without breaking is 5000 N, determine the maximum value of the hanging weight that these ropes can safely support. You can ignore the weight of the ropes and the steel cable. Figure **P5.57** 	Open-Ended

Continued on next page

ID (Page No)	Physics Problem	Animation Requirement
E-06 (168)	**5.56** •• An adventurous archaeologist crosses between two rock cliffs by slowly going hand over hand along a rope stretched between the cliffs. He stops to rest at the middle of the rope (Fig. P5.56). The rope will break if the tension in it exceeds 2.50×10^4 N, and our hero's mass is 90.0 kg. (a) If the angle θ is 10.0°, find the tension in the rope. (b) What is the smallest value the angle θ can have if the rope is not to break? Figure **P5.56** 	Open-Ended
E-07 (168)	**5.60** ••• A horizontal wire holds a solid uniform ball of mass m in place on a tilted ramp that rises 35.0° above the horizontal. The surface of this ramp is perfectly smooth, and the wire is directed away from the center of the ball (Fig. P5.60). (a) Draw a free-body diagram for the ball. (b) How hard does the surface of the ramp push on the ball? (c) What is the tension in the wire? 	Open-Ended

Continued on next page

ID (Page No)	Physics Problem	Animation Requirement
E-08 (169)	**5.72** •• Block A in Fig. P5.72 weighs 60.0 N. The coefficient of static friction between the block and the surface on which it rests is 0.25. The weight w is 12.0 N and the system is in equilibrium. (a) Find the friction force exerted on block A. (b) Find the maximum weight w for which the system will remain in equilibrium. Figure **P5.72**  	Open-Ended
W-01 (198)	**6.6** •• Two tugboats pull a disabled supertanker. Each tug exerts a constant force of 1.80×10^6 N, one 14° west of north and the other 14° east of north, as they pull the tanker 0.75 km toward the north. What is the total work they do on the supertanker?	Point-of-Interest
W-02 (199)	**6.14** •• A 1.50-kg book is sliding along a rough horizontal surface. At point A it is moving at 3.21 m/s, and at point B it has slowed to 1.25 m/s. (a) How much work was done on the book between A and B? (b) If −0.750 J of work is done on the book from B to C, how fast is it moving at point C? (c) How fast would it be moving at C if +0.750 J of work were done on it from B to C?	Point-of-Interest

Continued on next page

ID (Page No)	Physics Problem	Animation Requirement
W-03 (199)	**6.18** • A 4.80-kg watermelon is dropped from rest from the roof of a 25.0-m-tall building and feels no appreciable air resistance. (a) Calculate the work done by gravity on the watermelon during its displacement from the roof to the ground. (b) Just before it strikes the ground, what is the watermelon's (i) kinetic energy and (ii) speed? (c) Which of the answers in parts (a) and (b) would be *different* if there were appreciable air resistance?	Point-of-Interest
W-04 (199)	**6.20** •• You throw a 20-N rock vertically into the air from ground level. You observe that when it is 15.0 m above the ground, it is traveling at 25.0 m/s upward. Use the work–energy theorem to find (a) the rock's speed just as it left the ground and (b) its maximum height.	Time-of-Interest, Point-of-Interest
W-05 (200)	**6.23** • A sled with mass 8.00 kg moves in a straight line on a frictionless horizontal surface. At one point in its path, its speed is 4.00 m/s; after it has traveled 2.50 m beyond this point, its speed is 6.00 m/s. Use the work–energy theorem to find the force acting on the sled, assuming that this force is constant and that it acts in the direction of the sled's motion.	Point-of-Interest
W-06 (200)	**6.24** •• A soccer ball with mass 0.420 kg is initially moving with speed 2.00 m/s. A soccer player kicks the ball, exerting a constant force of magnitude 40.0 N in the same direction as the ball's motion. Over what distance must the player's foot be in contact with the ball to increase the ball's speed to 6.00 m/s?	Point-of-Interest

Continued on next page

ID (Page No)	Physics Problem	Animation Requirement
W-07 (200)	**6.26** • A batter hits a baseball with mass 0.145 kg straight upward with an initial speed of 25.0 m/s. (a) How much work has gravity done on the baseball when it reaches a height of 20.0 m above the bat? (b) Use the work–energy theorem to calculate the speed of the baseball at a height of 20.0 m above the bat. You can ignore air resistance. (c) Does the answer to part (b) depend on whether the baseball is moving upward or downward at a height of 20.0 m? Explain.	Point-of-Interest
W-08 (200)	**6.27** • A little red wagon with mass 7.00 kg moves in a straight line on a frictionless horizontal surface. It has an initial speed of 4.00 m/s and then is pushed 3.0 m in the direction of the initial velocity by a force with a magnitude of 10.0 N. (a) Use the work–energy theorem to calculate the wagon's final speed. (b) Calculate the acceleration produced by the force. Use this acceleration in the kinematic relationships of Chapter 2 to calculate the wagon's final speed. Compare this result to that calculated in part (a).	Point-of-Interest
W-09 (200)	**6.28** •• A block of ice with mass 2.00 kg slides 0.750 m down an inclined plane that slopes downward at an angle of 36.9° below the horizontal. If the block of ice starts from rest, what is its final speed? You can ignore friction.	Point-of-Interest
W-10 (201)	**6.52** •• A 20.0-kg rock is sliding on a rough, horizontal surface at 8.00 m/s and eventually stops due to friction. The coefficient of kinetic friction between the rock and the surface is 0.200. What average power is produced by friction as the rock stops?	Open-Ended
W-11 (232)	**7.1** • In one day, a 75-kg mountain climber ascends from the 1500-m level on a vertical cliff to the top at 2400 m. The next day, she descends from the top to the base of the cliff, which is at an elevation of 1350 m. What is her change in gravitational potential energy (a) on the first day and (b) on the second day?	Point-of-Interest

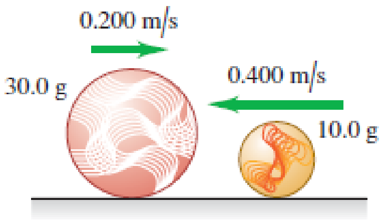
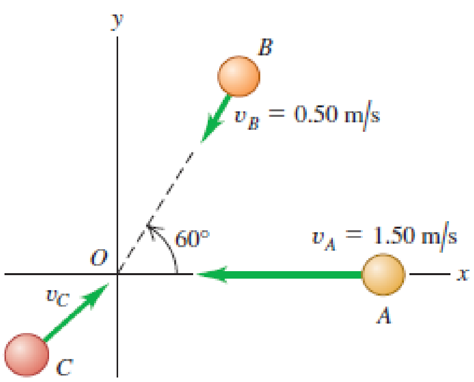
Continued on next page

ID (Page No)	Physics Problem	Animation Requirement
W-12 (202)	**6.65** ••• **CP** A 20.0-kg crate sits at rest at the bottom of a 15.0-m-long ramp that is inclined at 34.0° above the horizontal. A constant horizontal force of 290 N is applied to the crate to push it up the ramp. While the crate is moving, the ramp exerts a constant frictional force on it that has magnitude 65.0 N. (a) What is the total work done on the crate during its motion from the bottom to the top of the ramp? (b) How much time does it take the crate to travel to the top of the ramp?	Time-of-Interest, Point-of-Interest
W-13 (232)	**7.11** •• You are testing a new amusement park roller coaster with an empty car of mass 120 kg. One part of the track is a vertical loop with radius 12.0 m. At the bottom of the loop (point A) the car has speed 25.0 m/s, and at the top of the loop (point B) it has speed 8.0 m/s. As the car rolls from point A to point B, how much work is done by friction?	Point-of-Interest
M-01 (269)	**8.8** • **Force of a Baseball Swing.** A baseball has mass 0.145 kg. (a) If the velocity of a pitched ball has a magnitude of 45.0 m/s and the batted ball's velocity is 55.0 m/s in the opposite direction, find the magnitude of the change in momentum of the ball and of the impulse applied to it by the bat. (b) If the ball remains in contact with the bat for 2.00 ms, find the magnitude of the average force applied by the bat.	Open-Ended
M-02 (269)	**8.11** • **CALC** At time $t = 0$, a 2150-kg rocket in outer space fires an engine that exerts an increasing force on it in the $+x$-direction. This force obeys the equation $F_x = At^2$, where t is time, and has a magnitude of 781.25 N when $t = 1.25$ s. (a) Find the SI value of the constant A, including its units. (b) What impulse does the engine exert on the rocket during the 1.50-s interval starting 2.00 s after the engine is fired? (c) By how much does the rocket's velocity change during this interval?	Time-of-Interest

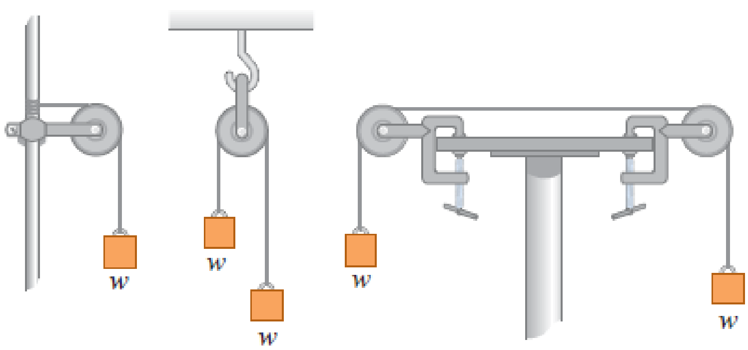
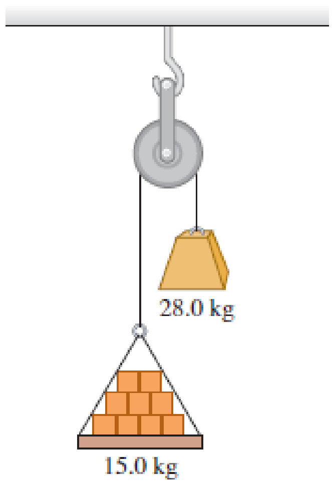
Continued on next page

ID (Page No)	Physics Problem	Animation Requirement
M-03 (269)	**8.21** •• On a frictionless, horizontal air table, puck A (with mass 0.250 kg) is moving toward puck B (with mass 0.350 kg), which is initially at rest. After the collision, puck A has a velocity of 0.120 m/s to the left, and puck B has a velocity of 0.650 m/s to the right. (a) What was the speed of puck A before the collision? (b) Calculate the change in the total kinetic energy of the system that occurs during the collision.	Open-Ended
M-04 (270)	**8.25** •• A hunter on a frozen, essentially frictionless pond uses a rifle that shoots 4.20-g bullets at 965 m/s. The mass of the hunter (including his gun) is 72.5 kg, and the hunter holds tight to the gun after firing it. Find the recoil velocity of the hunter if he fires the rifle (a) horizontally and (b) at 56.0° above the horizontal.	Open-Ended
M-05 (270)	**8.27** •• Two ice skaters, Daniel (mass 65.0 kg) and Rebecca (mass 45.0 kg), are practicing. Daniel stops to tie his shoelace and, while at rest, is struck by Rebecca, who is moving at 13.0 m/s before she collides with him. After the collision, Rebecca has a velocity of magnitude 8.00 m/s at an angle of 53.1° from her initial direction. Both skaters move on the frictionless, horizontal surface of the rink. (a) What are the magnitude and direction of Daniel's velocity after the collision? (b) What is the change in total kinetic energy of the two skaters as a result of the collision?	Open-Ended
M-06 (270)	**8.36** • A 1050-kg sports car is moving westbound at 15.0 m/s on a level road when it collides with a 6320-kg truck driving east on the same road at 10.0 m/s. The two vehicles remain locked together after the collision. (a) What is the velocity (magnitude and direction) of the two vehicles just after the collision? (b) At what speed should the truck have been moving so that it and the car are both stopped in the collision? (c) Find the change in kinetic energy of the system of two vehicles for the situations of part (a) and part (b). For which situation is the change in kinetic energy greater in magnitude?	Open-Ended

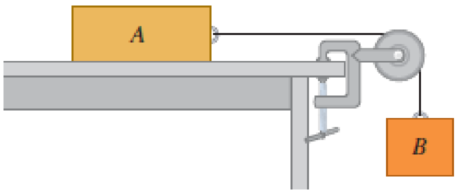
Continued on next page

ID (Page No)	Physics Problem	Animation Requirement
M-07 (271)	**8.48** • A 10.0-g marble slides to the left with a velocity of magnitude 0.400 m/s on the frictionless, horizontal surface of an icy New York sidewalk and has a head-on, elastic collision with a larger 30.0-g marble sliding to the right with a velocity of magnitude 0.200 m/s (Fig. E8.48). (a) Find the velocity of each marble (magnitude and direction) after the collision. (Since the collision is head-on, all the motion is along a line.) (b) Calculate the *change in momentum* (that is, the momentum after the collision minus the momentum before the collision) for each marble. Compare the values you get for each marble. (c) Calculate the *change in kinetic energy* (that is, the kinetic energy after the collision minus the kinetic energy before the collision) for each marble. Compare the values you get for each marble. Figure **E8.48** 	Open-Ended
M-08 (273)	**8.73** • Spheres *A* (mass 0.020 kg), *B* (mass 0.030 kg), and *C* (mass 0.050 kg) are approaching the origin as they slide on a frictionless air table (Fig. P8.73). The initial velocities of *A* and *B* are given in the figure. All three spheres arrive at the origin at the same time and stick together. (a) What must the x- and y-components of the initial velocity of *C* be if all three objects are to end up moving at 0.50 m/s in the $+x$-direction after the collision? (b) If *C* has the velocity found in part (a), what is the change in the kinetic energy of the system of three spheres as a result of the collision? Figure **P8.73** 	Open-Ended

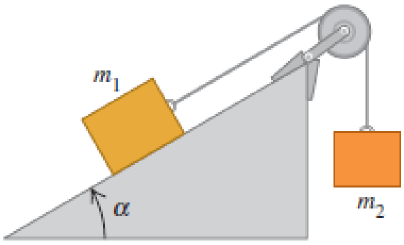
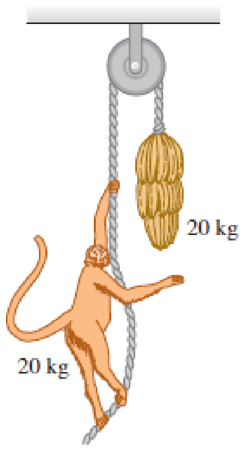
Continued on next page

ID (Page No)	Physics Problem	Animation Requirement
P-01 (163)	**5.2** • In Fig. E5.2 each of the suspended blocks has weight w. The pulleys are frictionless and the ropes have negligible weight. Calculate, in each case, the tension T in the rope in terms of the weight w. In each case, include the free-body diagram or diagrams you used to determine the answer. Figure **E5.2** (a) (b) (c)  	Open-Ended
P-02 (165)	**5.15** •• **Atwood's Machine.** A 15.0-kg load of bricks hangs from one end of a rope that passes over a small, frictionless pulley. A 28.0-kg counterweight is suspended from the other end of the rope, as shown in Fig. E5.15. The system is released from rest. (a) Draw two free-body diagrams, one for the load of bricks and one for the counterweight. (b) What is the magnitude of the upward acceleration of the load of bricks? (c) What is the tension in the rope while the load is moving? How does the tension compare to the weight of the load of bricks? To the weight of the counterweight? Figure **E5.15**  	Open-Ended

Continued on next page

ID (Page No)	Physics Problem	Animation Requirement
P-03 (165)	**5.19** •• CP A 750.0-kg boulder is raised from a quarry 125 m deep by a long uniform chain having a mass of 575 kg. This chain is of uniform strength, but at any point it can support a maximum tension no greater than 2.50 times its weight without breaking. (a) What is the maximum acceleration the boulder can have and still get out of the quarry, and (b) how long does it take to be lifted out at maximum acceleration if it started from rest?	Time-of-Interest, Point-of-Interest
P-04 (166)	**5.34** •• Consider the system shown in Fig. E5.34. Block A weighs 45.0 N and block B weighs 25.0 N. Once block B is set into downward motion, it descends at a constant speed. (a) Calculate the coefficient of kinetic friction between block A and the tabletop. (b) A cat, also of weight 45.0 N, falls asleep on top of block A. If block B is now set into downward motion, what is its acceleration (magnitude and direction)? Figure **E5.34** 	Open-Ended
P-05 (169)	**5.63** ••• CALC A 3.00-kg box that is several hundred meters above the surface of the earth is suspended from the end of a short vertical rope of negligible mass. A time-dependent upward force is applied to the upper end of the rope, and this results in a tension in the rope of $T(t) = (36.0\ \mathrm{N/s})t$. The box is at rest at $t = 0$. The only forces on the box are the tension in the rope and gravity. (a) What is the velocity of the box at (i) $t = 1.00$ s and (ii) $t = 3.00$ s? (b) What is the maximum distance that the box descends below its initial position? (c) At what value of t does the box return to its initial position?	Time-of-Interest, Point-of-Interest

Continued on next page

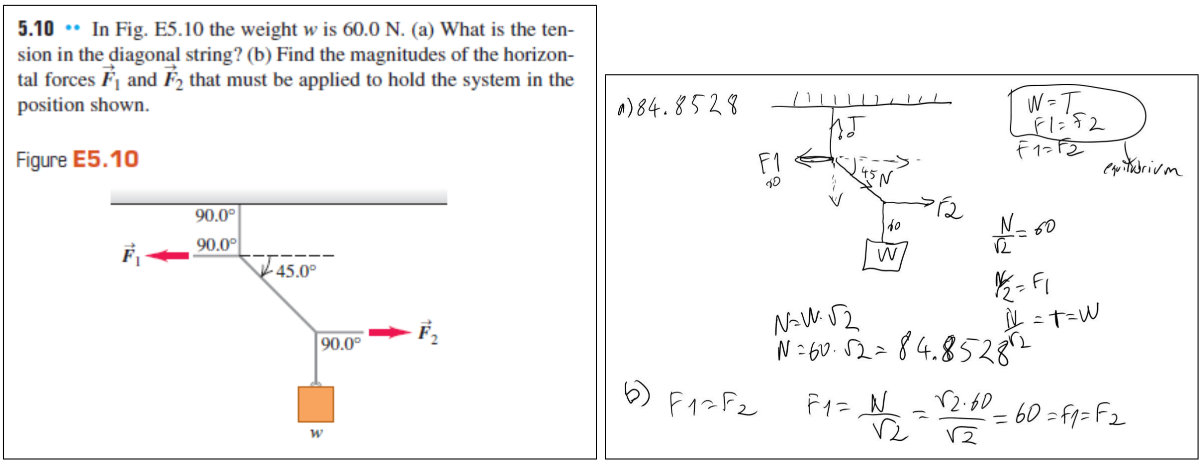
ID (Page No)	Physics Problem	Animation Requirement
P-06 (169)	**5.68** •• **CP** In Fig. P5.68 $m_1 = 20.0$ kg and $\alpha = 53.1°$. The coefficient of kinetic friction between the block and the incline is $\mu_k = 0.40$. What must be the mass m_2 of the hanging block if it is to descend 12.0 m in the first 3.00 s after the system is released from rest? Figure **P5.68**  	Time-of-Interest, Point-of-Interest
P-07 (172)	**5.105** • **The Monkey and Bananas Problem.** A 20-kg monkey has a firm hold on a light rope that passes over a frictionless pulley and is attached to a 20-kg bunch of bananas (Fig. P5.105). The monkey looks up, sees the bananas, and starts to climb the rope to get them. (a) As the monkey climbs, do the bananas move up, down, or remain at rest? (b) As the monkey climbs, does the distance between the monkey and the bananas decrease, increase, or remain constant? (c) The monkey releases her hold on the rope. What happens to the distance between the monkey and the bananas while she is falling? (d) Before reaching the ground, the monkey grabs the rope to stop her fall. What do the bananas do? Figure **P5.105**  	Open-Ended

9.3 Analysis of Student Solutions

As mentioned earlier in Section 9.1, we have collected 56 student solutions to a subset of the physics problems listed in Table 9.1:

- 4 solutions were collected for problems: E-01
- 3 solutions were collected for problems: K-03, K-04, W-07
- 2 solutions were collected for problems: K-02, K-05, K-06, K-07, K-08, K-11, K-12, M-03, M-05, W-06, W-09
- 1 solution was collected for problems: E-02, E-04, E-05, E-06, K-01, K-09, K-13, K-14, M-02, P-02, P-04, P-05, P-06, P-07, W-01, W-02, W-03, W-05, W-08, W-11, W-13
- no solutions were collected for remaining problems due to insufficient participation in the experiment. These are: E-03, E-07, E-08, K-10, M-01, M-04, M-06, M-07, M-08, P-01,P-03, W-04, W-10, W-12.

The challenges in sketch recognition for handwritten solutions have already been discussed in Sections 1.1 and 2.1. In this section, we will examine how the three categories of animation we identified earlier can be applied to the 56 solutions that we have acquired.



(a) E-02 Statement

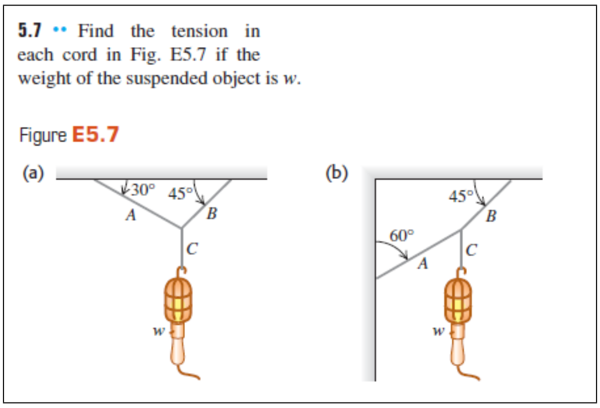
(b) E-02 Solution

Figure 9.1: Equilibrium Problem (E-02) and a corresponding student solution, showing an instance of Open-Ended Animation.

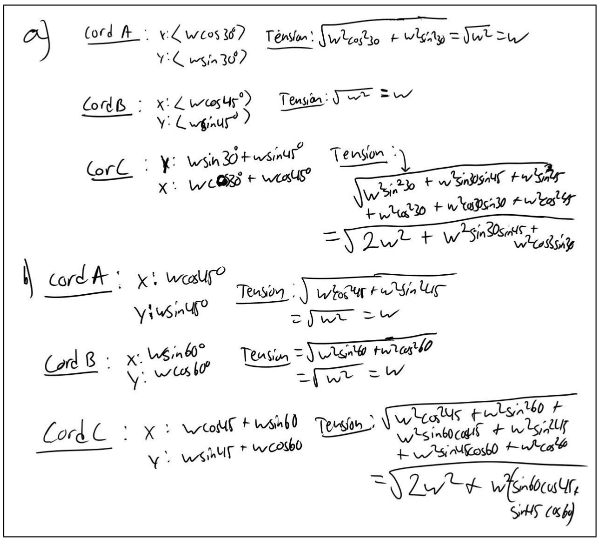
9.3.1 Analysis of Equilibrium Solutions

Figure 9.1(a) shows a multipart equilibrium problem, whose solution is depicted in Figure 9.1(b). The problem statement in this instance does not place any constraints on time or trajectory. The system is released from rest and is expected to remain in equilibrium, which is the central assumption for deriving one of the initial conditions. Thus, Figure 9.1(b) is an example of Open-Ended animation. Furthermore, in this instance, visual feedback can be provided in the form of inferred motion. If the computed answer is incorrect, then the equilibrium assumption will be nullified, and the box would move.

The solution for E-04, shown in Figure 9.2(b), is interesting because it does not contain a diagram. Such instances, by definition, cannot be animated. The solutions for both E-05 and E-06, shown in Figures 9.3 and 9.4, again depict systems released from rest, under the assumption of equilibrium. As with Figure 9.1, Open-Ended animation is the suitable approach here.



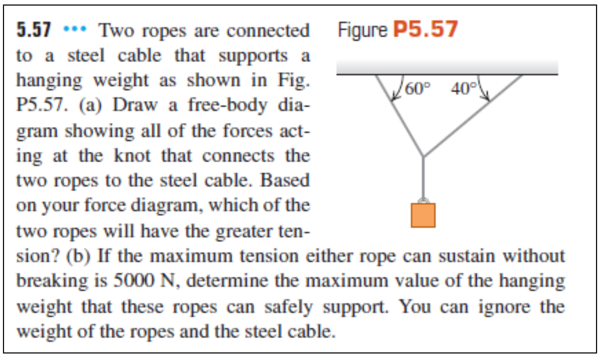
5.7 •• Find the tension in each cord in Fig. E5.7 if the weight of the suspended object is *w*.



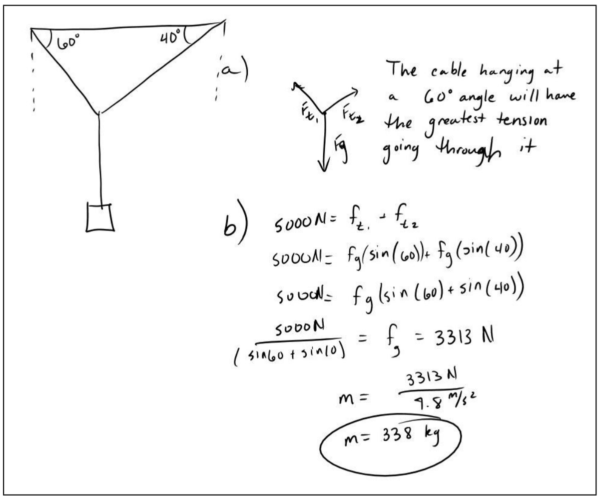
(a) E-04 Statement

(b) E-04 Solution

Figure 9.2: Equilibrium Problem (E-04) and a corresponding student solution, showing a scenario where no animation is possible.



5.57 ••• Two ropes are connected to a steel cable that supports a hanging weight as shown in Fig. P5.57. (a) Draw a free-body diagram showing all of the forces acting at the knot that connects the two ropes to the steel cable. Based on your force diagram, which of the two ropes will have the greater tension? (b) If the maximum tension either rope can sustain without breaking is 5000 N, determine the maximum value of the hanging weight that these ropes can safely support. You can ignore the weight of the ropes and the steel cable.



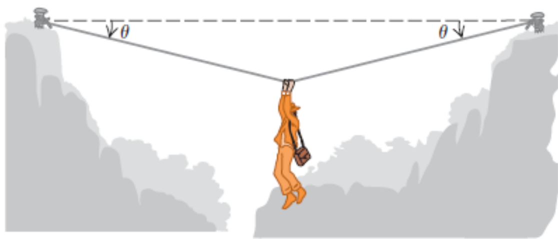
(a) E-05 Statement

(b) E-05 Solution

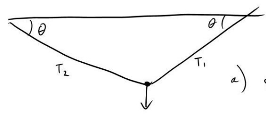
Figure 9.3: Equilibrium Problem (E-05) and a corresponding student solution, showing an instance of Open-Ended Animation.

5.56 •• An adventurous archaeologist crosses between two rock cliffs by slowly going hand over hand along a rope stretched between the cliffs. He stops to rest at the middle of the rope (Fig. P5.56). The rope will break if the tension in it exceeds 2.50×10^4 N, and our hero's mass is 90.0 kg. (a) If the angle θ is 10.0°, find the tension in the rope. (b) What is the smallest value the angle θ can have if the rope is not to break?

Figure **P5.56**



(a) E-06 Statement



$m = 90.0\ kg = 882\ N \downarrow$

rope breaks at tension $= 25{,}000$

$T_1 = T_2$

a) $882N = \sin T_1 + \sin T_2$

$882N = 2 \cdot \sin\theta \cdot T$

$882N = 2 \cdot \sin(10) \cdot T$

$\frac{882N}{2\sin(10)} = T = 2540\ N$

$2T = \text{total} = 2(2540N) = 5080N$

b) Tension $< 25{,}000\ N$

$T = \left(\frac{882\ N}{2\sin\theta}\right) \cdot 2 < 25000N$

$\frac{2\sin\theta}{2 \cdot 882N} < \frac{1}{25000N} \rightarrow \quad \sin\theta < \frac{882\ N}{25000N}$

$\theta < \sin^{-1}\left(\frac{882\ N}{25000N}\right)$

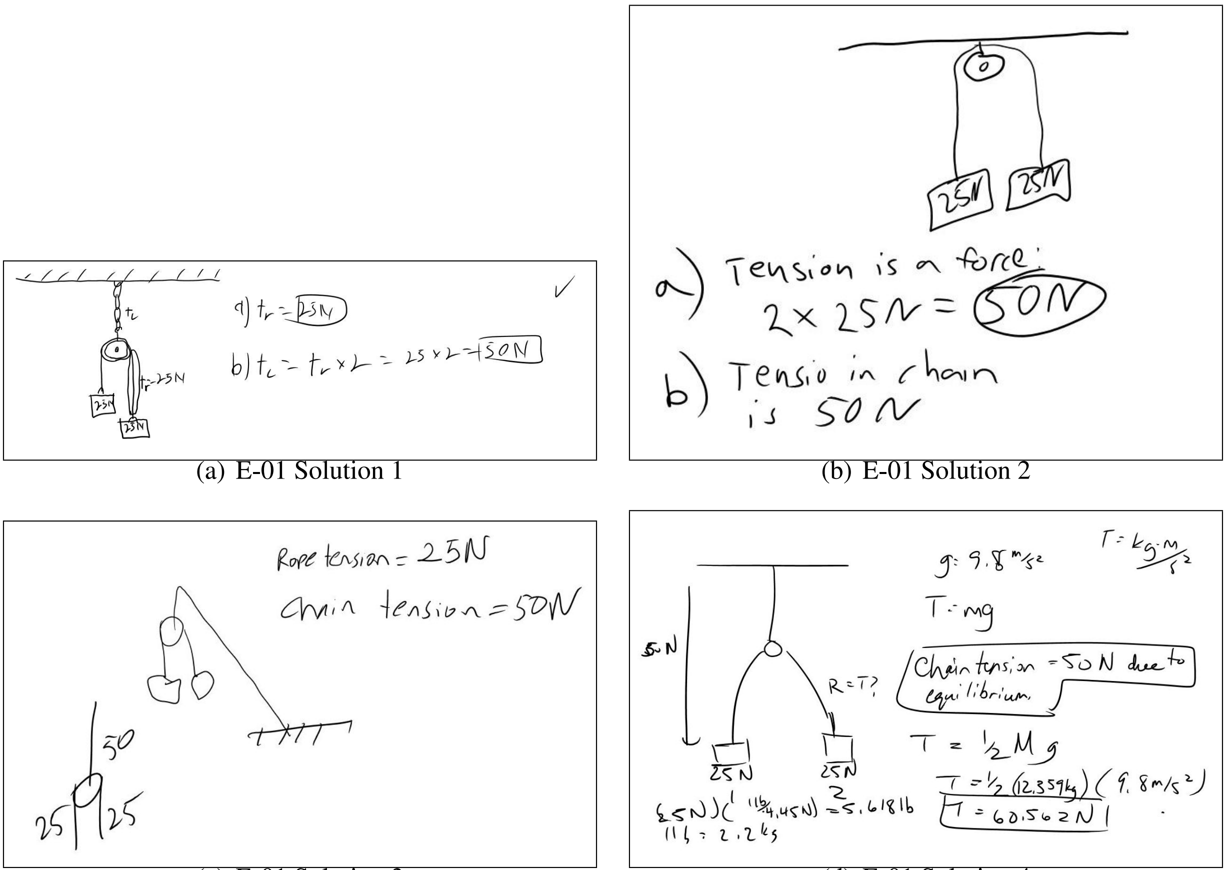
$\theta < 2°??$

(b) E-06 Solution

Figure 9.4: Equilibrium Problem (E-06) and a corresponding student solution, showing an instance of Open-Ended Animation.

5.1 • Two 25.0-N weights are suspended at opposite ends of a rope that passes over a light, frictionless pulley. The pulley is attached to a chain that goes to the ceiling. (a) What is the tension in the rope? (b) What is the tension in the chain?

Figure 9.5: Equilibrium Problem (E-01).



(a) E-01 Solution 1

(b) E-01 Solution 2

(c) E-01 Solution 3

(d) E-01 Solution 4

Figure 9.6: Four student solutions for E-01, shown in Figure 9.5.

We were able to correct four solutions for the equilibrium problem E-01 (Shown in Figure 9.5). The solutions are shown in Figure 9.6. All four of these solutions follow the same pattern as the other equilibrium solutions we have presented in this section. The student is required to work out some quantity, such that the system remains in equilibrium when it is released from rest. This scenario lends itself very well to open-ended animations. Figure 9.6(c) is an interesting case, because it contains two diagrams, both of which seem to be physically implausible. In such cases, it is not clear what animation should be used.

8.21 •• On a frictionless, horizontal air table, puck A (with mass 0.250 kg) is moving toward puck B (with mass 0.350 kg), which is initially at rest. After the collision, puck A has a velocity of 0.120 m/s to the left, and puck B has a velocity of 0.650 m/s to the right. (a) What was the speed of puck A before the collision? (b) Calculate the change in the total kinetic energy of the system that occurs during the collision.

(a) M-03 Statement

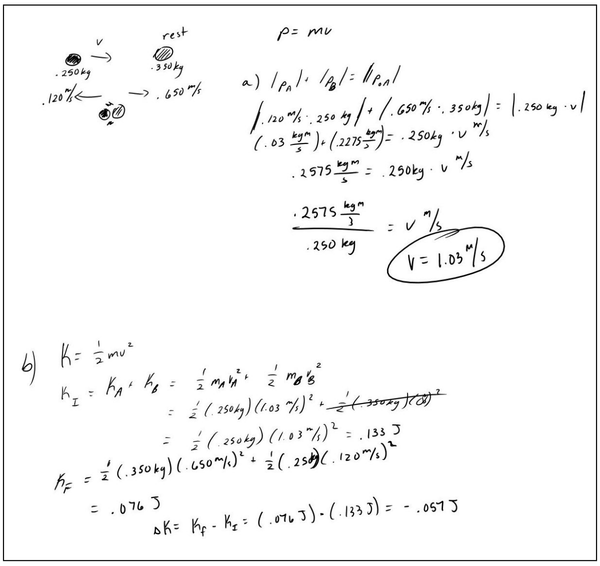
8.27 •• Two ice skaters, Daniel (mass 65.0 kg) and Rebecca (mass 45.0 kg), are practicing. Daniel stops to tie his shoelace and, while at rest, is struck by Rebecca, who is moving at 13.0 m/s before she collides with him. After the collision, Rebecca has a velocity of magnitude 8.00 m/s at an angle of 53.1° from her initial direction. Both skaters move on the frictionless, horizontal surface of the rink. (a) What are the magnitude and direction of Daniel's velocity after the collision? (b) What is the change in total kinetic energy of the two skaters as a result of the collision?

(b) M-05 Statement

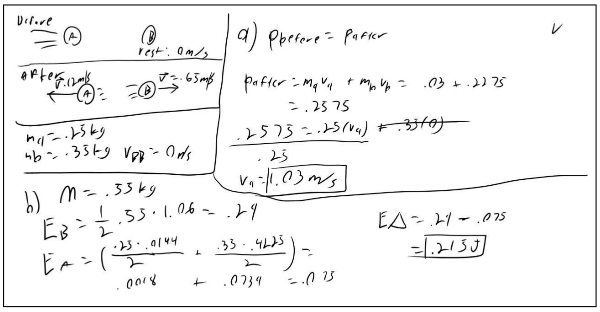
Figure 9.7: Problem statements for M-03 and M-05.

9.3.2 Analysis of Momentum Solutions

We were able to collect two solutions each for two momentum problems (M-03 and M-05). The statements for these problems are shown in Figure 9.7. M-03 is a multipart problem. In terms of animation requirements, the first solution, shown in Figure 9.8(a), can be modeled using Open-Ended animation. The student can compute the initial velocity from the given information, which can be used to set up the simulation. After the collision happens, the resulting velocities can be visually confirmed. The second solution, shown in Figure 9.8(b) can be animated in a similar manner. The student solutions for M-05 are shown in Figure 9.9, and can also be categorized as Open-Ended animation. However, these two solutions serve to reinforce an interesting point. In general, collision animations can be split into two parts: before-collision and after-collision. Sometimes, students draw the states of both these parts separately, as in Figure 9.8(a), 9.8(b) and 9.9(a). However, the before- and after- collision states can also be combined together into a single diagram, as shown in Figure 9.9(b), and in examples from Chapter 8. This variance directly impacts the challenge of understanding momentum diagrams. Recognition systems need to be able to make sense of both types of diagrams. However, as collision problems are usually simplified to focus on the aspects of the collision itself, and do not model complex effects like friction, they can still generally be categorized as falling the Open-Ended animation.

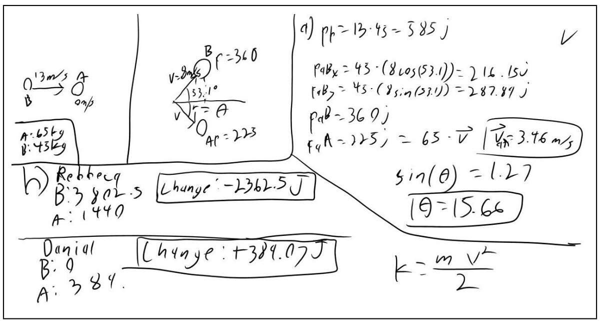


(a) M-03 Solution 1

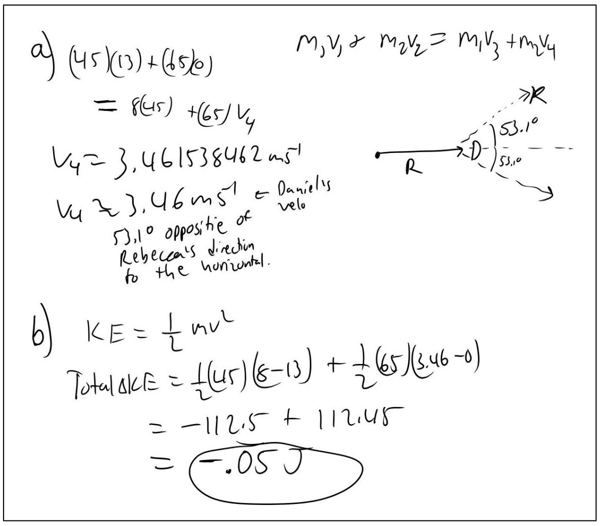


(b) M-03 Solution 2

Figure 9.8: Two solutions for M-03, each depicting instances that can be modeled using Open-Ended Animation.



(a) M-05 Solution 1

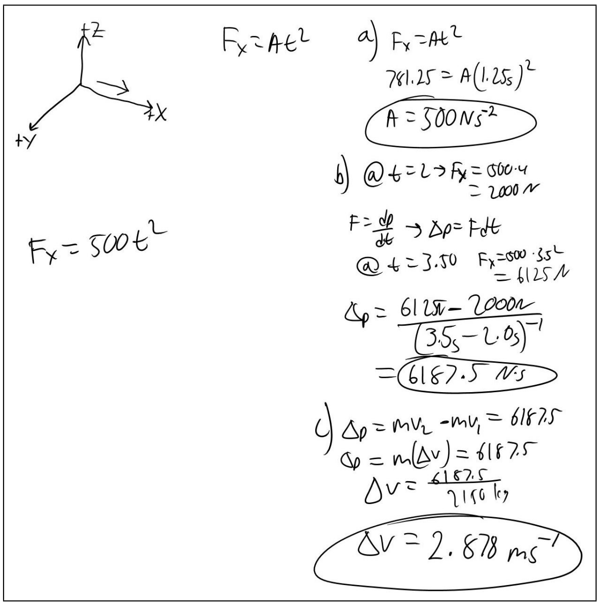


(b) M-05 Solution 2

Figure 9.9: Two solutions for M-05, each depicting instances that can be modeled using Open-Ended Animation.

8.11 • **CALC** At time $t = 0$, a 2150-kg rocket in outer space fires an engine that exerts an increasing force on it in the $+x$-direction. This force obeys the equation $F_x = At^2$, where t is time, and has a magnitude of 781.25 N when $t = 1.25$ s. (a) Find the SI value of the constant A, including its units. (b) What impulse does the engine exert on the rocket during the 1.50-s interval starting 2.00 s after the engine is fired? (c) By how much does the rocket's velocity change during this interval?

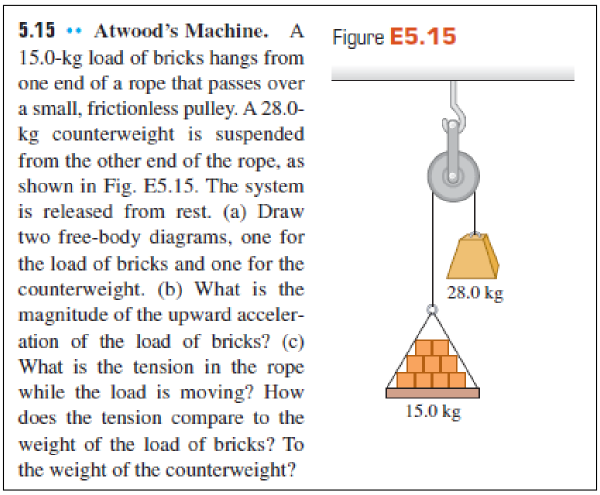
(a) M-02 Statement



(b) M-02 Solution

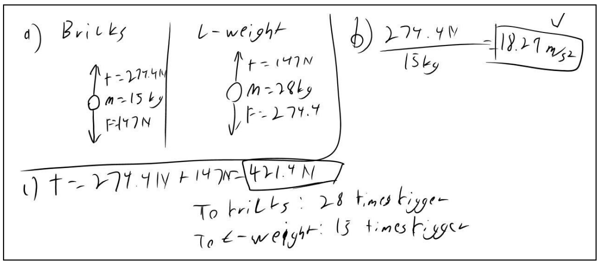
Figure 9.10: Momentum Problem (M-02) and a corresponding student solution, showing an instance of Time-of-Interest Animation.

The final example of a momentum solution is shown in Figure 9.10(b). This problem is different from the momentum problems examined in this work so far, because it does not involve collisions. Instead, this is an impulse problem and requires the student to calculate the value of impulse applied by a force over a period of time. As a time parameter is defined, this instance can be modeled using Time-of-Interest animation.



5.15 •• **Atwood's Machine.** A 15.0-kg load of bricks hangs from one end of a rope that passes over a small, frictionless pulley. A 28.0-kg counterweight is suspended from the other end of the rope, as shown in Fig. E5.15. The system is released from rest. (a) Draw two free-body diagrams, one for the load of bricks and one for the counterweight. (b) What is the magnitude of the upward acceleration of the load of bricks? (c) What is the tension in the rope while the load is moving? How does the tension compare to the weight of the load of bricks? To the weight of the counterweight?

(a) P-02 Statement

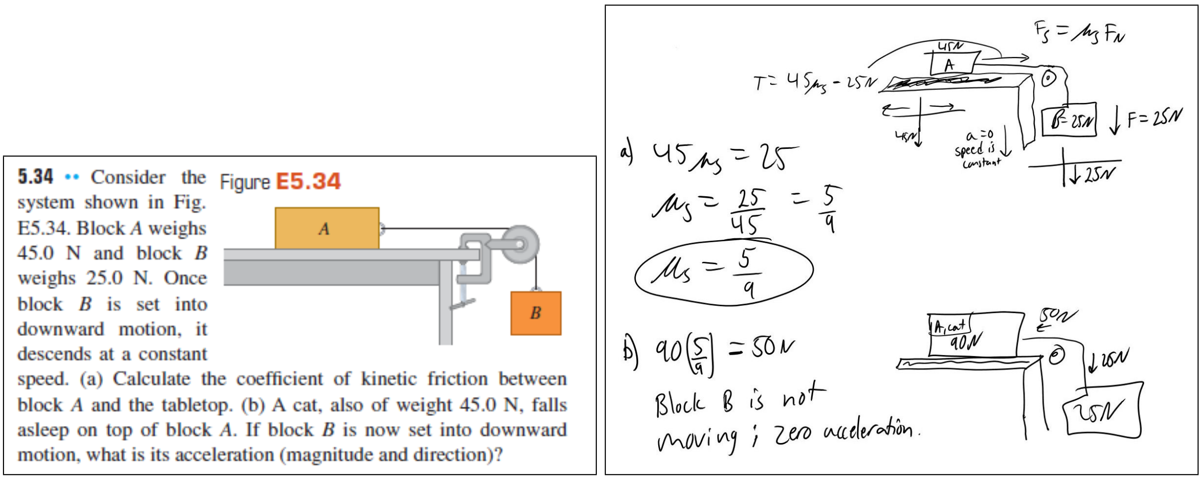


(b) P-02 Solution

Figure 9.11: Pulley Problem (P-02) and a corresponding student solution, showing an instance of Open-Ended Animation.

9.3.3 Analysis of Pulley Solutions

Pulley problems are interesting because they can be span more than one categories of animation, depending on how the question is framed. For example, pulley systems can be released from rest, which can be modeled as an Open-Ended animation. Alternatively, pulley problems may be framed in a piecemeal way. Figure 9.11(a) depicts a pulley problem that is explicitly framed in a piecemeal manner. The student is asked to draw separate free-body diagrams for both sides of the pulley. The computation for each side is also framed separately. with the final part asking for a decision-type statement. For the student solution in this case (See Figure 9.11(b)), Open-Ended animation is the only possible way to do animation. Yet, both diagrams will need to be considered, even if only one is to be animated. This is required because the two freebody diagrams in Figure 9.11(b) form a coherent whole, even if they are being considered in isolation. We cannot use Time-of-Interest or Point-of-Interest animation here because we have no hints about either of those parameters.



(a) P-04 Statement

(b) P-04 Solution

Figure 9.12: Pulley Problem (P-04) and a corresponding student solution, showing an instance of Open-Ended Animation.

Figure 9.12 is another instance where the pulley system is released from rest, and no hints are given for time or distance constraints. This problem has already been discussed in Section 8.4, and is a case where Open-Ended animation is a suitable strategy.

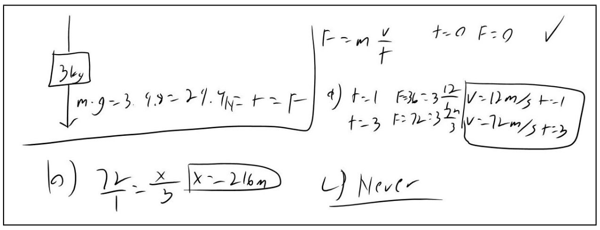
P-05 is an interesting multipart problem, that requires the student to compute both time and distance values in different parts (See Figure 9.13) The student solution can therefore be categorized as both Time-of-Interest or Point-of-Interest animation, depending on which part of the question is being considered.

P-06 (Shown in Figure 9.14) is similar to P-05 (Figure 9.13), because it can be considered an example of either Point-of-Interest or Time-of-Interest animation, as both constraints are provided in the statement (See Figure 9.14(a)). However, unlike P-05, both animation categories are not required for this solution.

The solution for P-07 (Shown in Figure 9.15(b)) consists mostly of text statements, making this a difficult case for animation. There is a partial figure drawn but it is not very clear.

5.63 ••• **CALC** A 3.00-kg box that is several hundred meters above the surface of the earth is suspended from the end of a short vertical rope of negligible mass. A time-dependent upward force is applied to the upper end of the rope, and this results in a tension in the rope of $T(t) = (36.0\ \mathrm{N/s})t$. The box is at rest at $t = 0$. The only forces on the box are the tension in the rope and gravity. (a) What is the velocity of the box at (i) $t = 1.00$ s and (ii) $t = 3.00$ s? (b) What is the maximum distance that the box descends below its initial position? (c) At what value of t does the box return to its initial position?

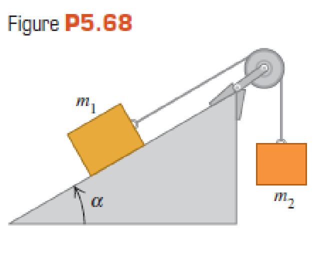
(a) P-05 Statement



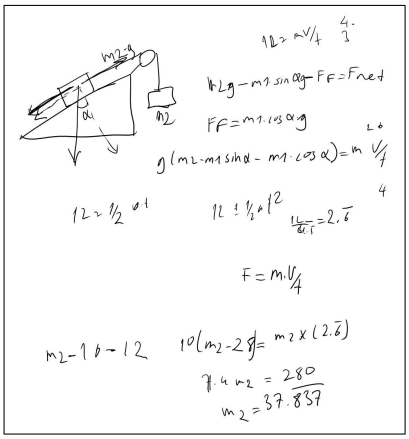
(b) P-05 Solution

Figure 9.13: Pulley Problem (P-05) and a corresponding student solution, showing an instance where both Time-of-Interest and Point-of-Interest animations are required.

5.68 •• **CP** In Fig. P5.68 $m_1 = 20.0$ kg and $\alpha = 53.1°$. The coefficient of kinetic friction between the block and the incline is $\mu_k = 0.40$. What must be the mass m_2 of the hanging block if it is to descend 12.0 m in the first 3.00 s after the system is released from rest?

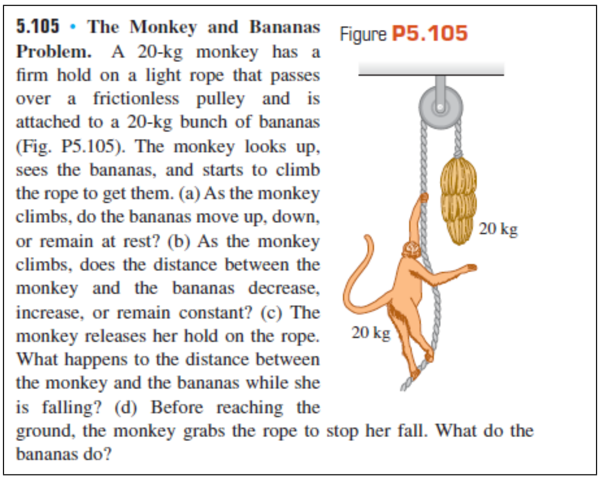


(a) P-06 Statement

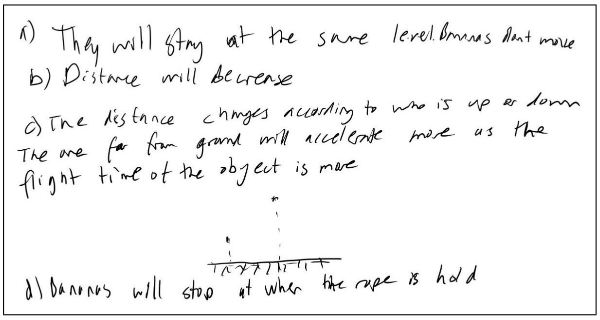


(b) P-06 Solution

Figure 9.14: Pulley Problem (P-06) and a corresponding student solution, which can be categorized as either Time-of-Interest or Point-of-Interest Animation.


5.105 • **The Monkey and Bananas Problem.** A 20-kg monkey has a firm hold on a light rope that passes over a frictionless pulley and is attached to a 20-kg bunch of bananas (Fig. P5.105). The monkey looks up, sees the bananas, and starts to climb the rope to get them. (a) As the monkey climbs, do the bananas move up, down, or remain at rest? (b) As the monkey climbs, does the distance between the monkey and the bananas decrease, increase, or remain constant? (c) The monkey releases her hold on the rope. What happens to the distance between the monkey and the bananas while she is falling? (d) Before reaching the ground, the monkey grabs the rope to stop her fall. What do the bananas do?

(a) P-07 Statement



(b) P-07 Solution

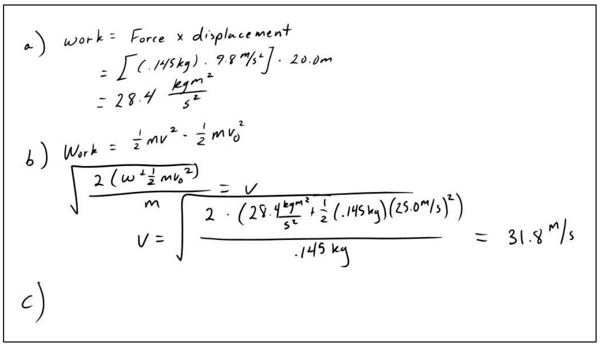
Figure 9.15: Pulley Problem (P-07) and a corresponding student solution, showing an instance of Open-Ended Animation.

9.3.4 Analysis of Work and Energy Solutions

For W-7 (Shown in Figure 9.16(a), the problem statement defines a distance limit, which enables us to categorize this particular problem as Point-of-Interest animation. However, an examination of three corresponding student solutions shown in Figure 9.17 presents some difficulties. Solution 1 (Figure 9.16(b)) is incomplete. The first two solutions (Figures 9.16(b) and 9.17(a)) both do not contain any diagrams. Solution 3 (Shown in Figure 9.17(b)) contains a sketched diagram, yet the student has not marked any distance limits for the diagram. This is an interesting case, because even though the parameters defined by the problem statement suggest a Point-of-Interest animation scenario, the student's diagram can only be modeled using Open-Ended animation, as it does not contain any annotations depicting a point of interest in the trajectory.

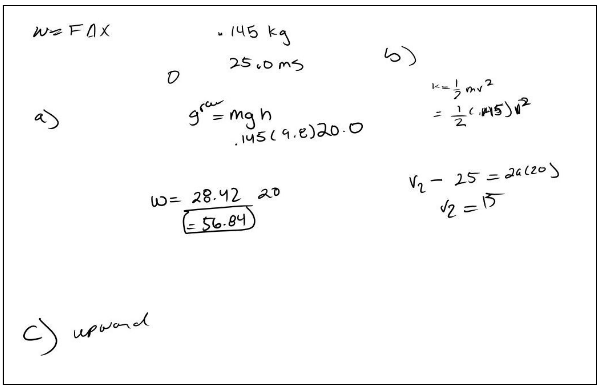
6.26 • A batter hits a baseball with mass 0.145 kg straight upward with an initial speed of 25.0 m/s. (a) How much work has gravity done on the baseball when it reaches a height of 20.0 m above the bat? (b) Use the work–energy theorem to calculate the speed of the baseball at a height of 20.0 m above the bat. You can ignore air resistance. (c) Does the answer to part (b) depend on whether the baseball is moving upward or downward at a height of 20.0 m? Explain.

(a) W-07 Statement

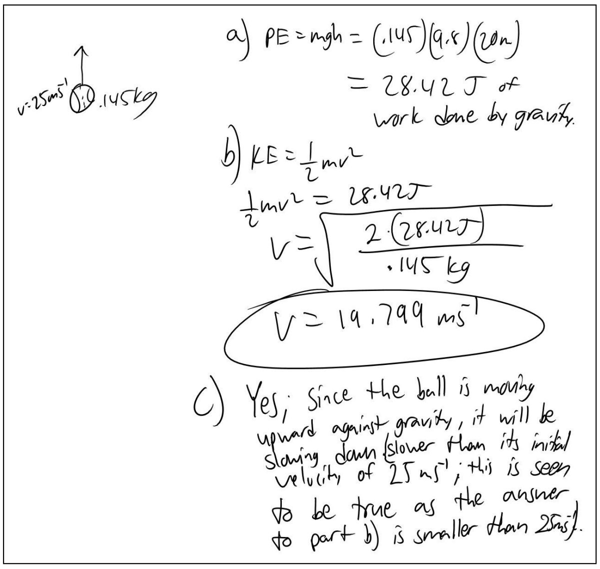


(b) W-07 Solution 1

Figure 9.16: Work and Energy Problem (W-07) and an incomplete student solution.



(a) W-07 Solution 2

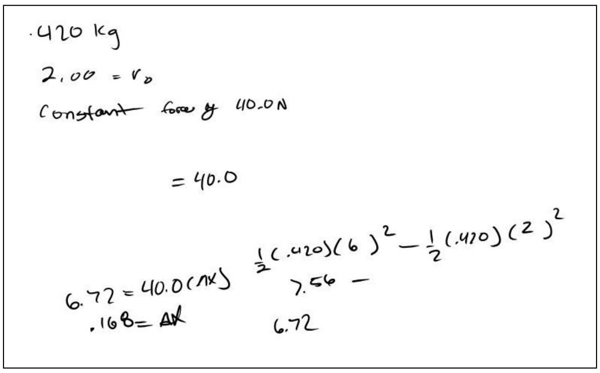


(b) W-07 Solution 3

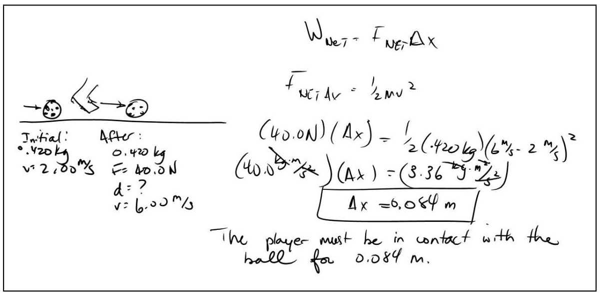
Figure 9.17: Two additional student solutions for the problem in Figure 9.16.

6.24 •• A soccer ball with mass 0.420 kg is initially moving with speed 2.00 m/s. A soccer player kicks the ball, exerting a constant force of magnitude 40.0 N in the same direction as the ball's motion. Over what distance must the player's foot be in contact with the ball to increase the ball's speed to 6.00 m/s?

Figure 9.18: Work and Energy Problem (W-06).



(a) W-06 Solution 1



(b) W-06 Solution 2

Figure 9.19: Two student solutions for the problem in Figure 9.18.

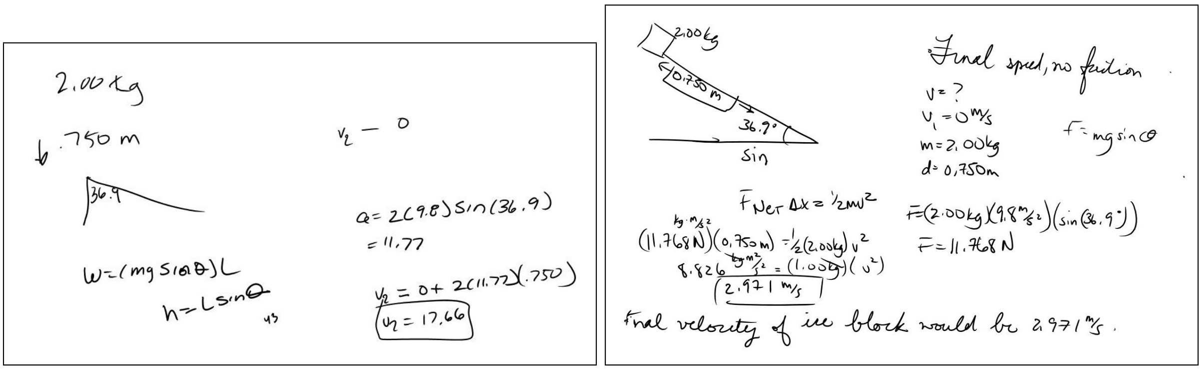
W-06 (Shown in Figure 9.18) represents a momentum problem (scenario describes an impulse calculation), that is to be solved using work and energy concepts. As a distance parameter is clearly specified, this problem can be categorized as Point-of-Interest animation. However, when we examine the student solutions in Figure 9.19, we again encounter a non-existent diagram in Figure 9.19(a), for which we can't generate an animation. The second solution (Figure 9.19(b)) contains a diagram but again, the student has not clearly marked the distance range, instead representing the scenario as a before- and after- collision diagram. This sort of diagram is difficult to model using Point-of-Interest diagram, therefore Open-Ended animation must be employed for the diagram in Figure 9.19(b).

Like W-06, W-09 (Figure 9.20) is another case where a distance parameter is specified by the problem statement, leading us to conclude that the problem may belong to the category of Point-of-Interest animation. However, when we look at the student solutions in Figure 9.21, we see that for one solution (Figure 9.21(a), the diagram is very poorly specified for animation purposes[3]. However, for the second solution 9.21(b), the diagram is much clearer, and can be animated using Point-of-Interest animation techniques.

[3] This does not mean that the student's version has flaws. "Poor Specification" in this respect merely implies that it is difficult to construct an animation for this diagram.

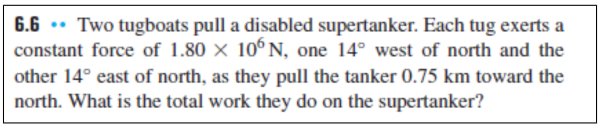
6.28 •• A block of ice with mass 2.00 kg slides 0.750 m down an inclined plane that slopes downward at an angle of 36.9° below the horizontal. If the block of ice starts from rest, what is its final speed? You can ignore friction.

Figure 9.20: Work and Energy Problem (W-09).



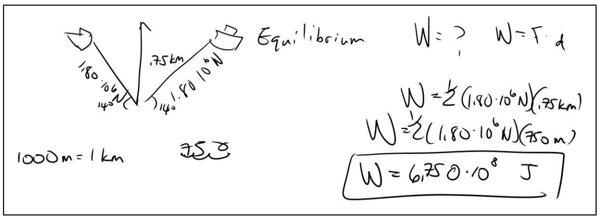
(a) W-09 Solution 1 (b) W-09 Solution 2

Figure 9.21: Two student solutions for the problem in Figure 9.20.



6.6 •• Two tugboats pull a disabled supertanker. Each tug exerts a constant force of 1.80×10^6 N, one 14° west of north and the other 14° east of north, as they pull the tanker 0.75 km toward the north. What is the total work they do on the supertanker?

(a) W-01 Statement



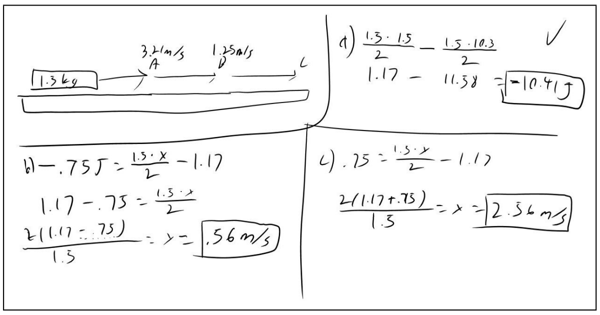
(b) W-01 Solution

Figure 9.22: Work and Energy Problem (W-01) and a corresponding student solution, showing an instance of Point-of-Interest Animation.

Animation categories for the remaining examples in this section are fairly self-evident. For problems that explicitly define a distance or time parameter, the animation category can be deduced as Point-of-Interest or Time-of-Interest animation. If no such parameters (or equivalent) are defined in the problem statement, then the animation category may be deduced as Open-Ended animation. However, analysis of earlier examples allows us to surmise that sometimes, the deduced categories may not be used directly. If the student solution does not contain a diagram, animation is not possible. For Point-of-Interest or Time-of-Interest animation categories, the diagram must contain a distance or time variable (indicated by an appropriate diagram element or annotation). If no such parameter can be inferred from the student's solution, then the only recourse is to use Open-Ended animation.

6.14 •• A 1.50-kg book is sliding along a rough horizontal surface. At point *A* it is moving at 3.21 m/s, and at point *B* it has slowed to 1.25 m/s. (a) How much work was done on the book between *A* and *B*? (b) If −0.750 J of work is done on the book from *B* to *C*, how fast is it moving at point *C*? (c) How fast would it be moving at *C* if +0.750 J of work were done on it from *B* to *C*?

(a) W-02 Statement

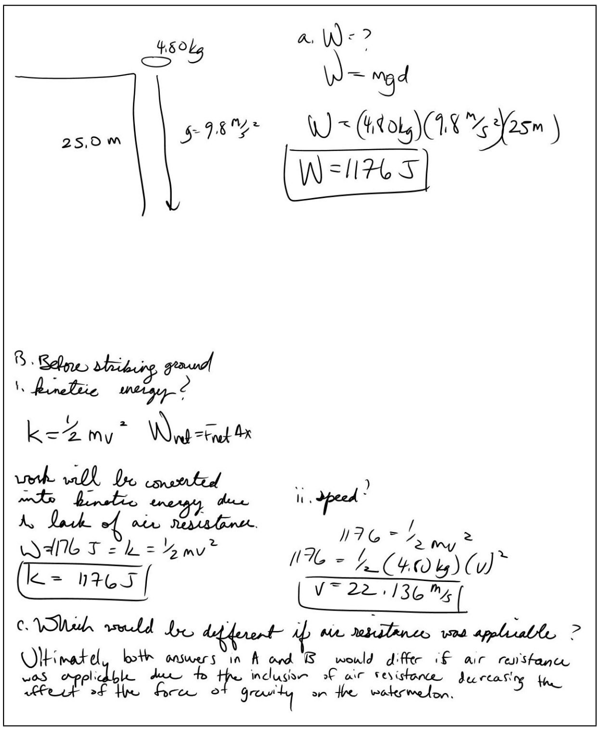


(b) W-02 Solution

Figure 9.23: Work and Energy Problem (W-02) and a corresponding student solution, showing an instance of Point-of-Interest Animation.

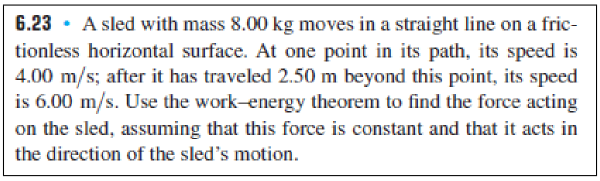
6.18 • A 4.80-kg watermelon is dropped from rest from the roof of a 25.0-m-tall building and feels no appreciable air resistance. (a) Calculate the work done by gravity on the watermelon during its displacement from the roof to the ground. (b) Just before it strikes the ground, what is the watermelon's (i) kinetic energy and (ii) speed? (c) Which of the answers in parts (a) and (b) would be *different* if there were appreciable air resistance?

(a) W-03 Statement



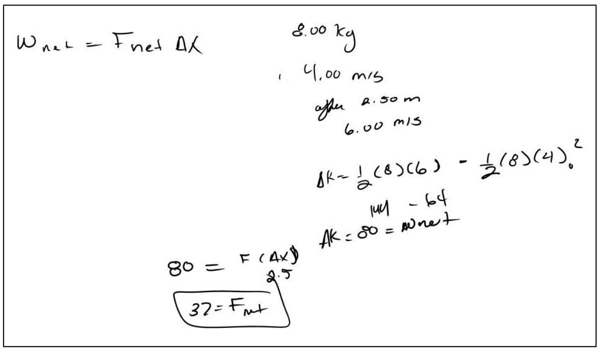
(b) W-03 Solution

Figure 9.24: Work and Energy Problem (W-03) and a corresponding student solution, showing an instance of Point-of-Interest Animation.



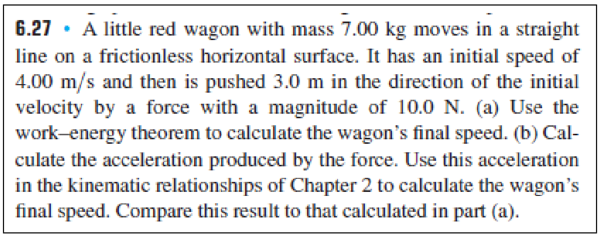
6.23 • A sled with mass 8.00 kg moves in a straight line on a frictionless horizontal surface. At one point in its path, its speed is 4.00 m/s; after it has traveled 2.50 m beyond this point, its speed is 6.00 m/s. Use the work–energy theorem to find the force acting on the sled, assuming that this force is constant and that it acts in the direction of the sled's motion.

(a) W-05 Statement



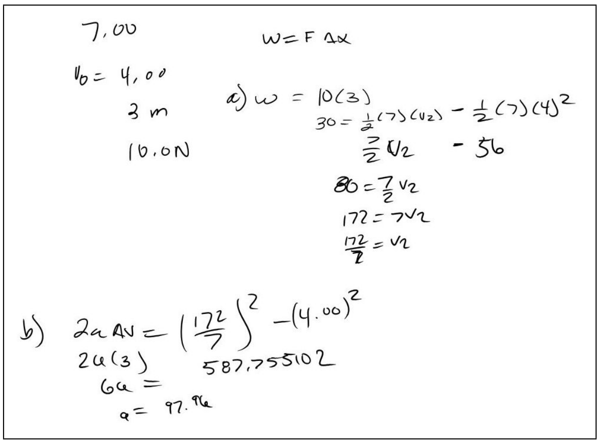
(b) W-05 Solution

Figure 9.25: Work and Energy Problem (W-05) and a corresponding student solution. In this instance, the problem statement indicates a Point-of-Interest animation category but the solution does not contain a diagram. Such instances cannot be animated.



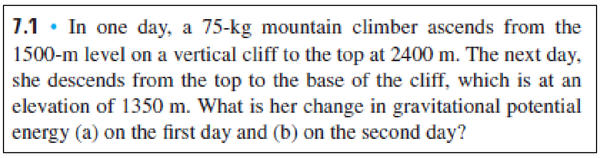
6.27 • A little red wagon with mass 7.00 kg moves in a straight line on a frictionless horizontal surface. It has an initial speed of 4.00 m/s and then is pushed 3.0 m in the direction of the initial velocity by a force with a magnitude of 10.0 N. (a) Use the work–energy theorem to calculate the wagon's final speed. (b) Calculate the acceleration produced by the force. Use this acceleration in the kinematic relationships of Chapter 2 to calculate the wagon's final speed. Compare this result to that calculated in part (a).

(a) W-08 Statement

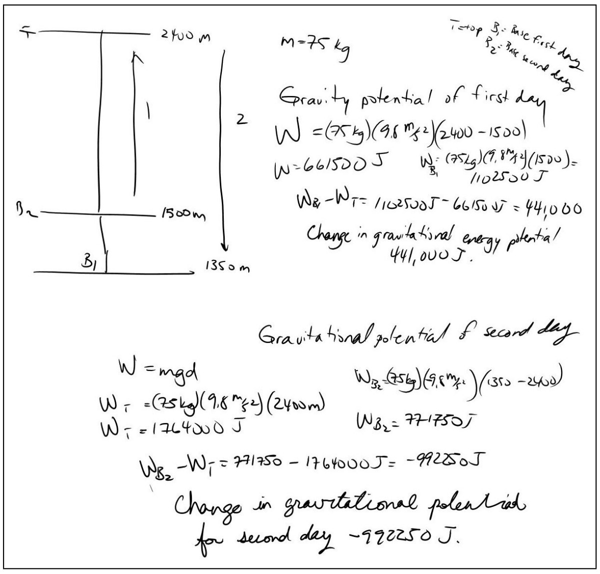


(b) W-08 Solution

Figure 9.26: Work and Energy Problem (W-08). As with Figure 9.25, the statement indicates a Point-of-Interest animation but the student solution does not contain a diagram.

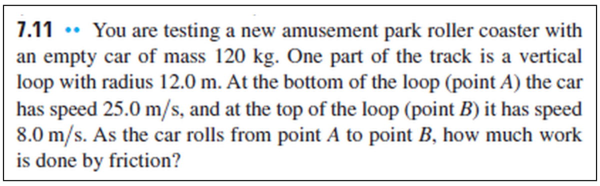

7.1 • In one day, a 75-kg mountain climber ascends from the 1500-m level on a vertical cliff to the top at 2400 m. The next day, she descends from the top to the base of the cliff, which is at an elevation of 1350 m. What is her change in gravitational potential energy (a) on the first day and (b) on the second day?

(a) W-11 Statement

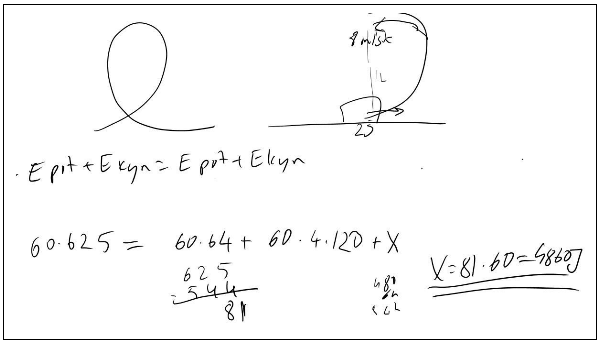


(b) W-11 Solution

Figure 9.27: Work and Energy Problem (W-11) and a corresponding student solution, showing an instance of Point-of-Interest Animation.

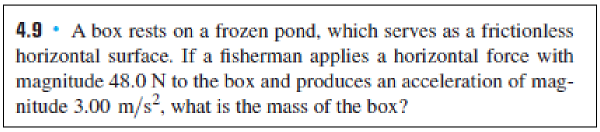

7.11 •• You are testing a new amusement park roller coaster with an empty car of mass 120 kg. One part of the track is a vertical loop with radius 12.0 m. At the bottom of the loop (point A) the car has speed 25.0 m/s, and at the top of the loop (point B) it has speed 8.0 m/s. As the car rolls from point A to point B, how much work is done by friction?

(a) W-13 Statement



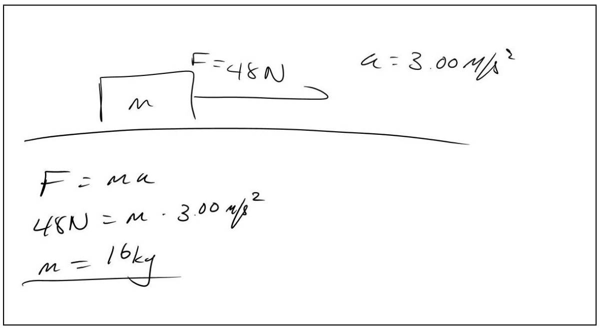
(b) W-13 Solution

Figure 9.28: Work and Energy Problem (W-13) and a corresponding student solution, showing an instance of Point-of-Interest Animation.



4.9 • A box rests on a frozen pond, which serves as a frictionless horizontal surface. If a fisherman applies a horizontal force with magnitude 48.0 N to the box and produces an acceleration of magnitude 3.00 m/s^2, what is the mass of the box?

(a) K-03 Statement



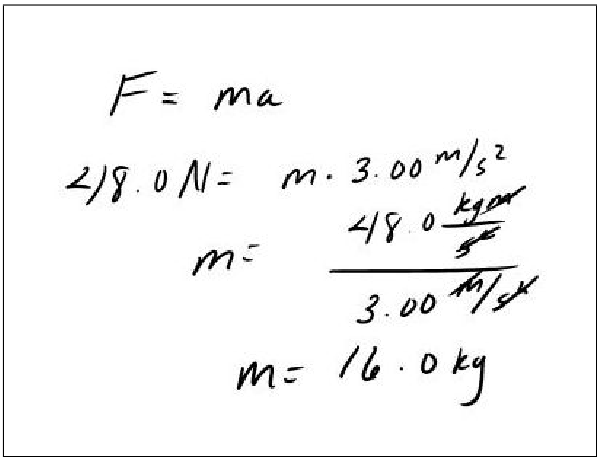
(b) K-03 Solution 1

Figure 9.29: Kinematics Problem (K-03), and a student solution. This example belongs to the Open-Ended animation category.

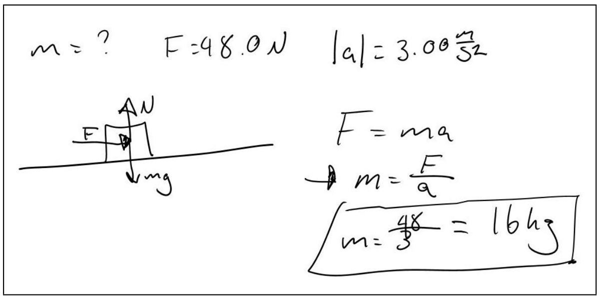
9.3.5 Analysis of Kinematics Solutions

K-03 and its solutions (Figures 9.29 and 9.30) have already been discussed in Section 1.1.1. As the statement does not define a time or distance parameter, this problem can be categorized as Open-Ended animation. This categorization can be applied to solutions 1 and 3 (Figures 9.29(b) and 9.30(b)), but not to solution 2 (Figure 9.30(a)) which does not contain a diagram.

K-04 (Figure 9.31(a)) is an interesting problem because it defines both a distance and a time parameter. Consequently, this problem can be modeled as either a Point-of-Interest or a Time-of-Interest animation. However, if we look at the three student solutions, only one contains a diagram (See Figure 9.31(b)). The other two (Figures 9.32(a) and 9.32(b)) cannot be animated as no diagram is available.

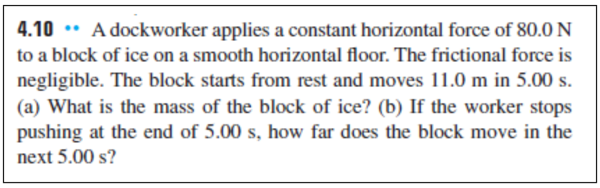


(a) K-03 Solution 2



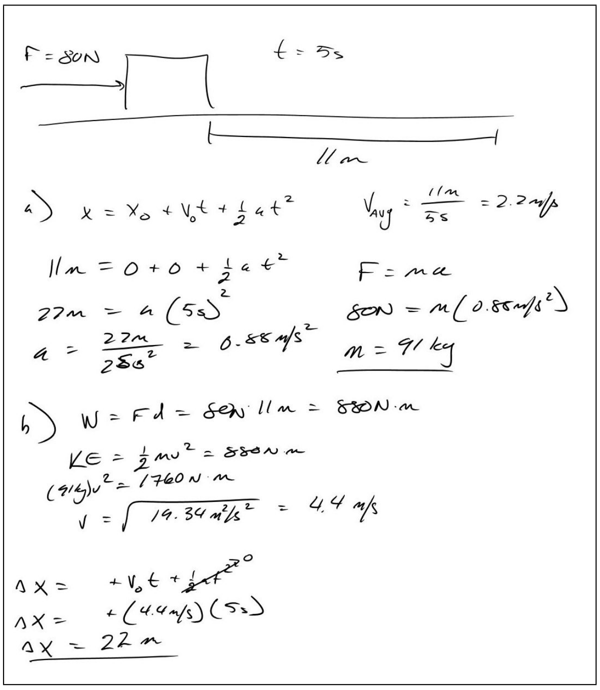
(b) K-03 Solution 3

Figure 9.30: Two additional student solutions for the problem in Figure 9.29, one of which does not contain a diagram.



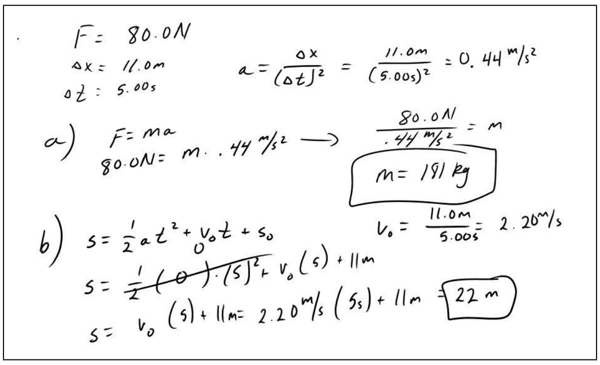
4.10 •• A dockworker applies a constant horizontal force of 80.0 N to a block of ice on a smooth horizontal floor. The frictional force is negligible. The block starts from rest and moves 11.0 m in 5.00 s. (a) What is the mass of the block of ice? (b) If the worker stops pushing at the end of 5.00 s, how far does the block move in the next 5.00 s?

(a) K-04 Statement

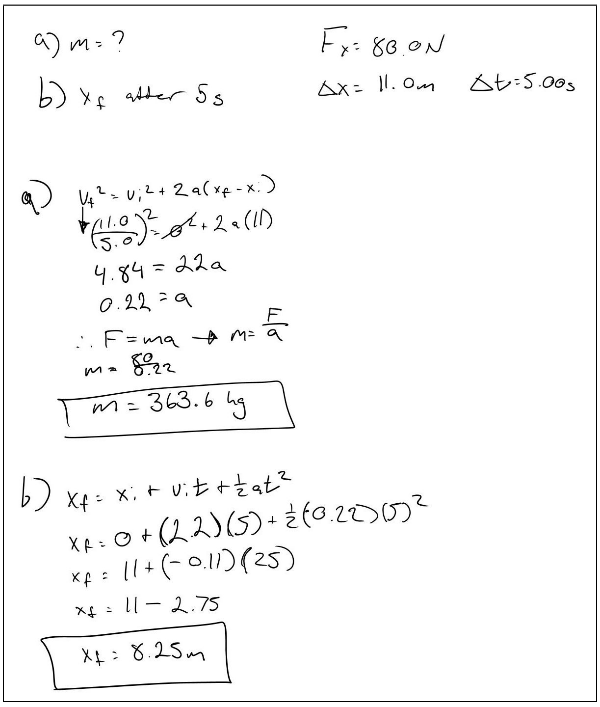


(b) K-04 Solution 1

Figure 9.31: Kinematics Problem (K-04), and a student solution. This example belongs to the Open-Ended animation category.



(a) K-04 Solution 2

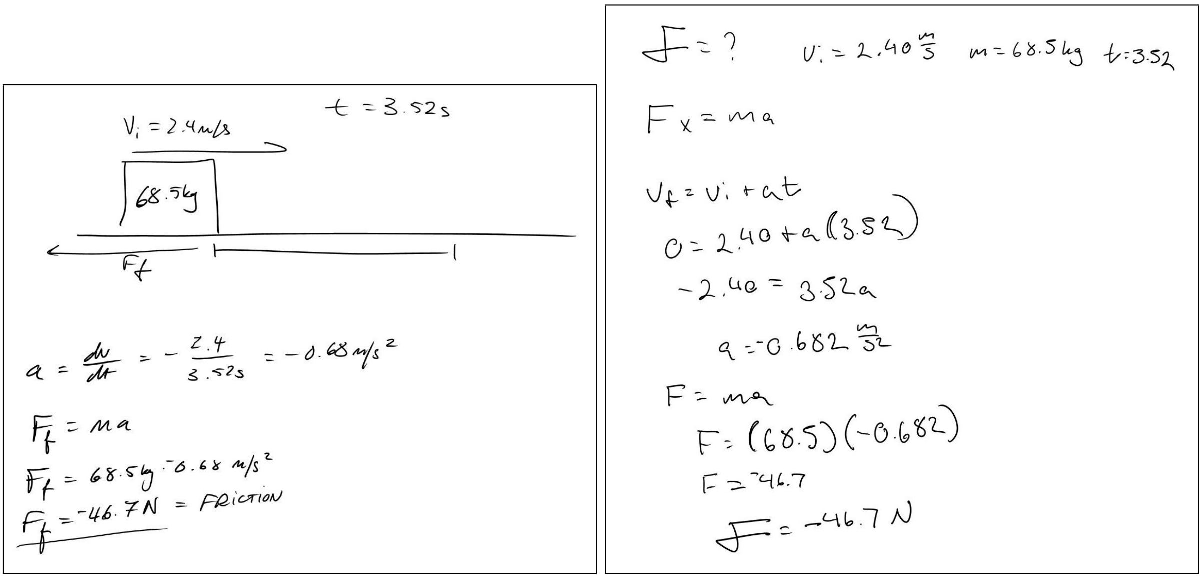


(b) K-04 Solution 3

Figure 9.32: Two additional student solutions for the problem in Figure 9.29, one of which is incomplete.

4.7 •• A 68.5-kg skater moving initially at 2.40 m/s on rough horizontal ice comes to rest uniformly in 3.52 s due to friction from the ice. What force does friction exert on the skater?

Figure 9.33: Kinematics Problem (K-02), representing an instance of Open-Ended Animation.



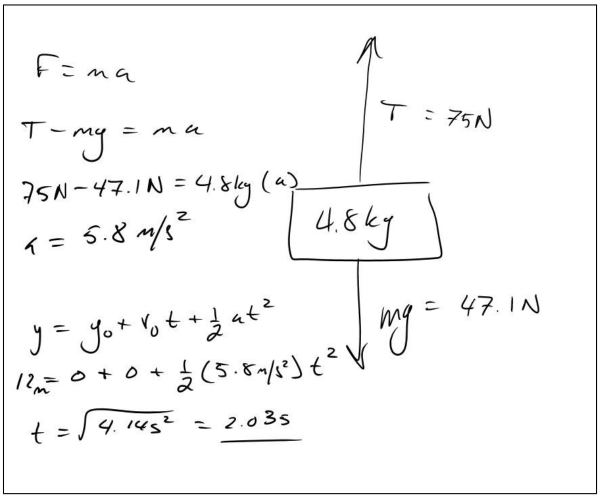
(a) K-02 Solution 1 (b) K-02 Solution 2

Figure 9.34: Two student solutions for the problem in Figure 9.33, one of which does not contain a diagram.

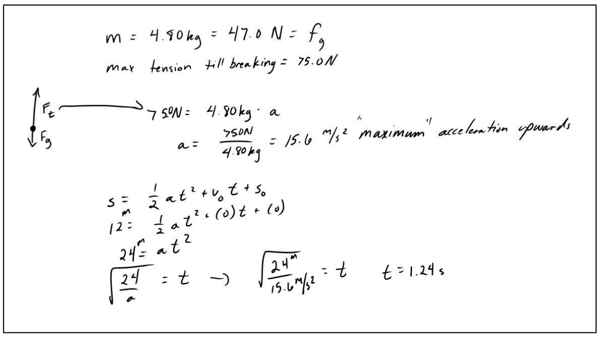
Animation categories for the remaining student solutions for kinematics problems are self-evident, and we have adopted an approach similar to the one used for reporting animation categories for work and energy problems. Problems that explicitly define a distance or time parameter have been marked as Point-of-Interest or Time-of-Interest category. If no such parameters (or equivalent) are defined in the problem statement, then the animation category is reported as Open-Ended animation.

4.33 **CP** A 4.80-kg bucket of water is accelerated upward by a cord of negligible mass whose breaking strength is 75.0 N. If the bucket starts from rest, what is the minimum time required to raise the bucket a vertical distance of 12.0 m without breaking the cord?

Figure 9.35: Kinematics Problem (K-05), representing an instance where either Point-of-Interest or Time-of-Interest animation may be used.



(a) K-05 Solution 1



(b) K-05 Solution 2

Figure 9.36: Two student solutions for the problem in Figure 9.35

4.38 • **CP** An oil tanker's engines have broken down, and the wind is blowing the tanker straight toward a reef at a constant speed of 1.5 m/s (Fig. P4.38). When the tanker is 500 m from the reef, the wind dies down just as the engineer gets the engines going again. The rudder is stuck, so the only choice is to try to accelerate straight backward away from the reef. The mass of the tanker and cargo is 3.6×10^7 kg, and the engines produce a net horizontal force of 8.0×10^4 N on the tanker. Will the ship hit the reef? If it does, will the oil be safe? The hull can withstand an impact at a speed of 0.2 m/s or less. You can ignore the retarding force of the water on the tanker's hull.

Figure **P4.38**

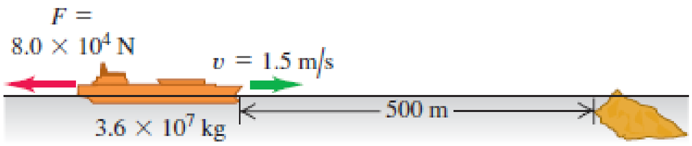
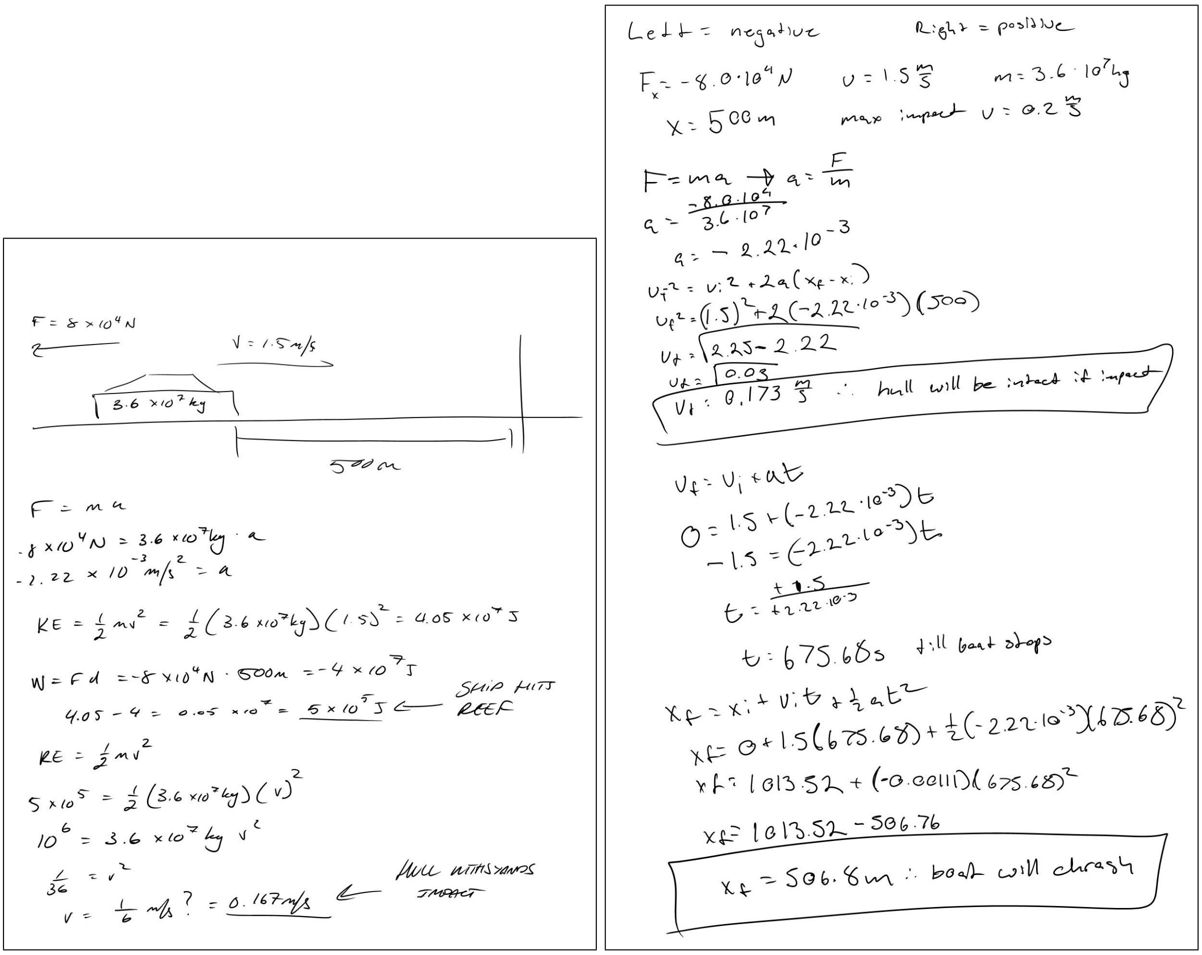


Figure 9.37: Kinematics Problem (K-06), representing an instance of Open-Ended Animation.

K-11 (Figure 9.43 is an interesting case. This is the only problem in our dataset where the answer is not a number or a function, but is instead a sketched diagram. So far, we have only considered graphing as a feedback mechanism. Figures 9.44(a) and 9.44(b) suggest the need to support sketched diagrams as animation inputs rather than animation specifications. However, this complexity can be considered in another way. The last part of the question (See Figure 9.43 that asks for the graph flows from the previous part. Thus, animating this particular class of student solutions is equivalent to finding out the link between the graph quantity and some part of the scenario, which can be modeled using one of the three animation categories (Open-Ended, Time-of-Interest, and Point-of-Interest).



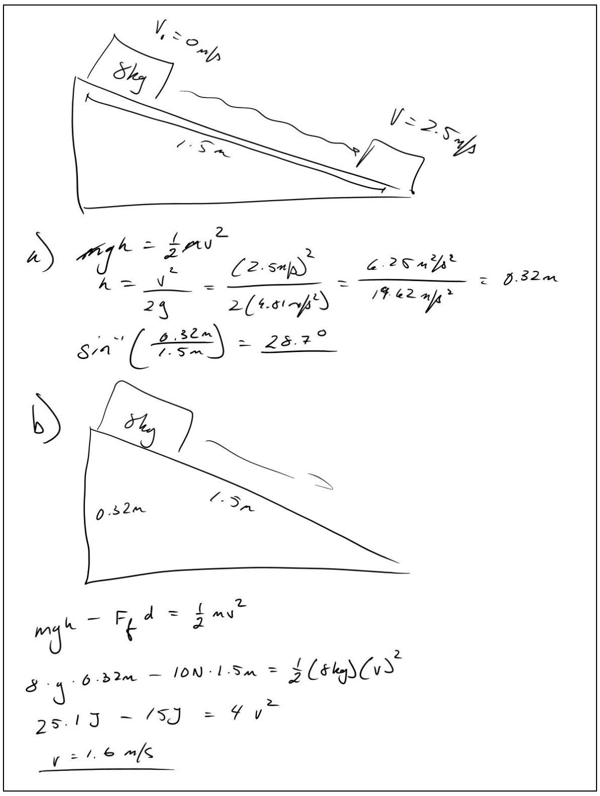
(a) K-06 Solution 1

(b) K-06 Solution 2

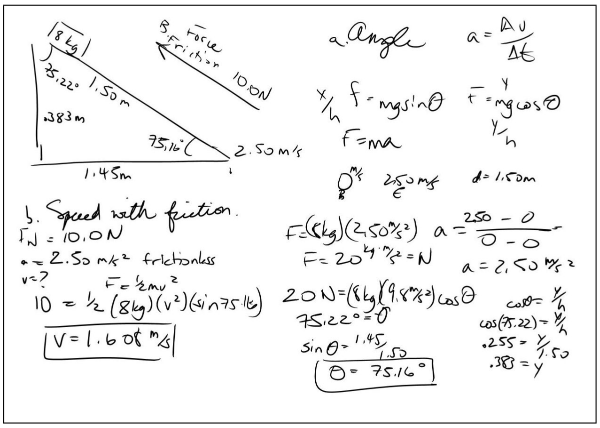
Figure 9.38: Two student solutions for the problem in Figure 9.37

5.16 •• **CP** A 8.00-kg block of ice, released from rest at the top of a 1.50-m-long frictionless ramp, slides downhill, reaching a speed of 2.50 m/s at the bottom. (a) What is the angle between the ramp and the horizontal? (b) What would be the speed of the ice at the bottom if the motion were opposed by a constant friction force of 10.0 N parallel to the surface of the ramp?

Figure 9.39: Kinematics Problem (K-07), representing an instance of Point-of-Interest Animation.

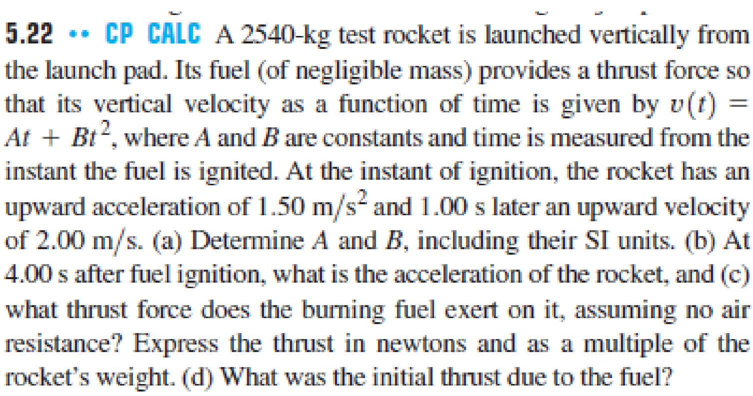


(a) K-07 Solution 1



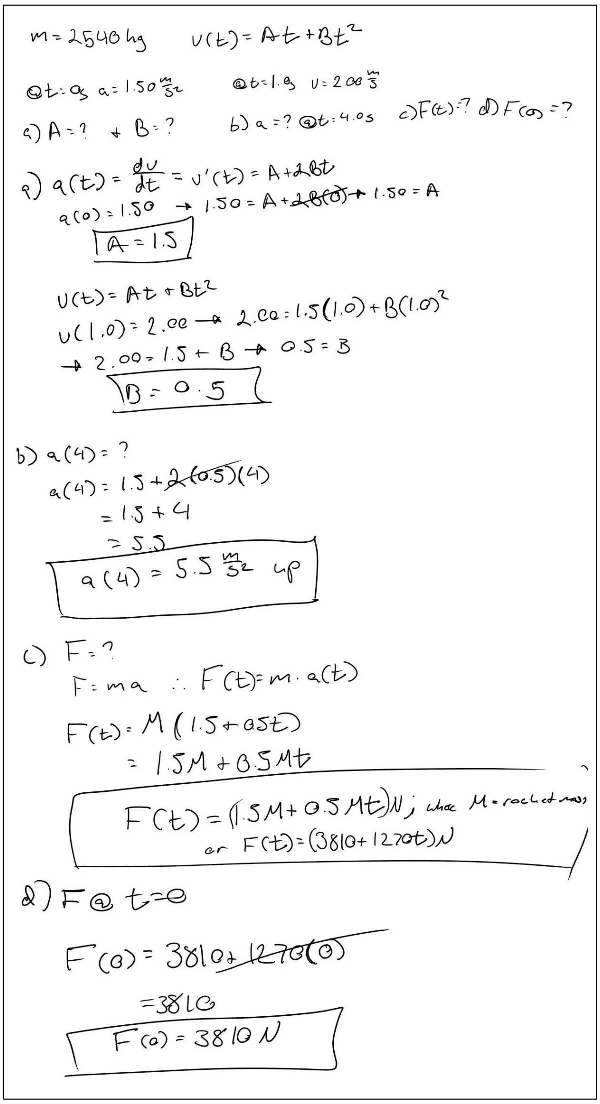
(b) K-07 Solution 2

Figure 9.40: Two student solutions for the problem in Figure 9.39

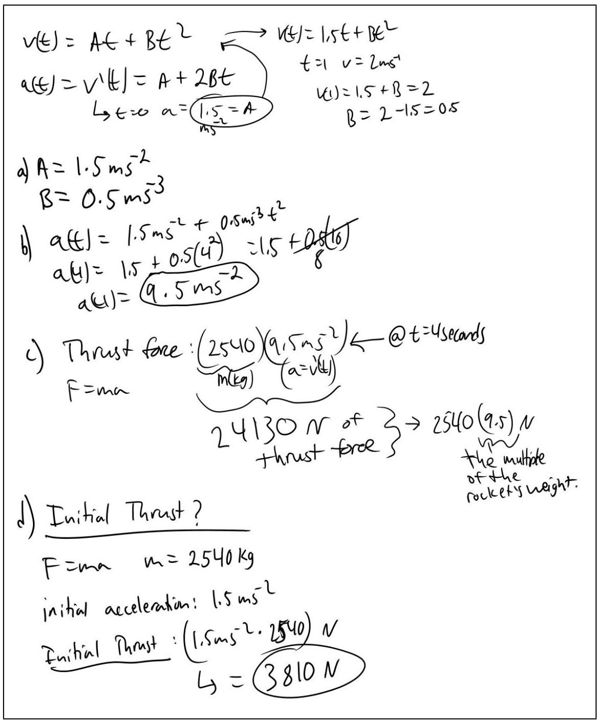


5.22 •• **CP CALC** A 2540-kg test rocket is launched vertically from the launch pad. Its fuel (of negligible mass) provides a thrust force so that its vertical velocity as a function of time is given by $v(t) = At + Bt^2$, where A and B are constants and time is measured from the instant the fuel is ignited. At the instant of ignition, the rocket has an upward acceleration of 1.50 m/s^2 and 1.00 s later an upward velocity of 2.00 m/s. (a) Determine A and B, including their SI units. (b) At 4.00 s after fuel ignition, what is the acceleration of the rocket, and (c) what thrust force does the burning fuel exert on it, assuming no air resistance? Express the thrust in newtons and as a multiple of the rocket's weight. (d) What was the initial thrust due to the fuel?

Figure 9.41: Kinematics Problem (K-08), representing an instance of Time-of-Interest Animation.



(a) K-08 Solution 1



(b) K-08 Solution 2

Figure 9.42: Two student solutions for the problem in Figure 9.41, neither of which contains a diagram.

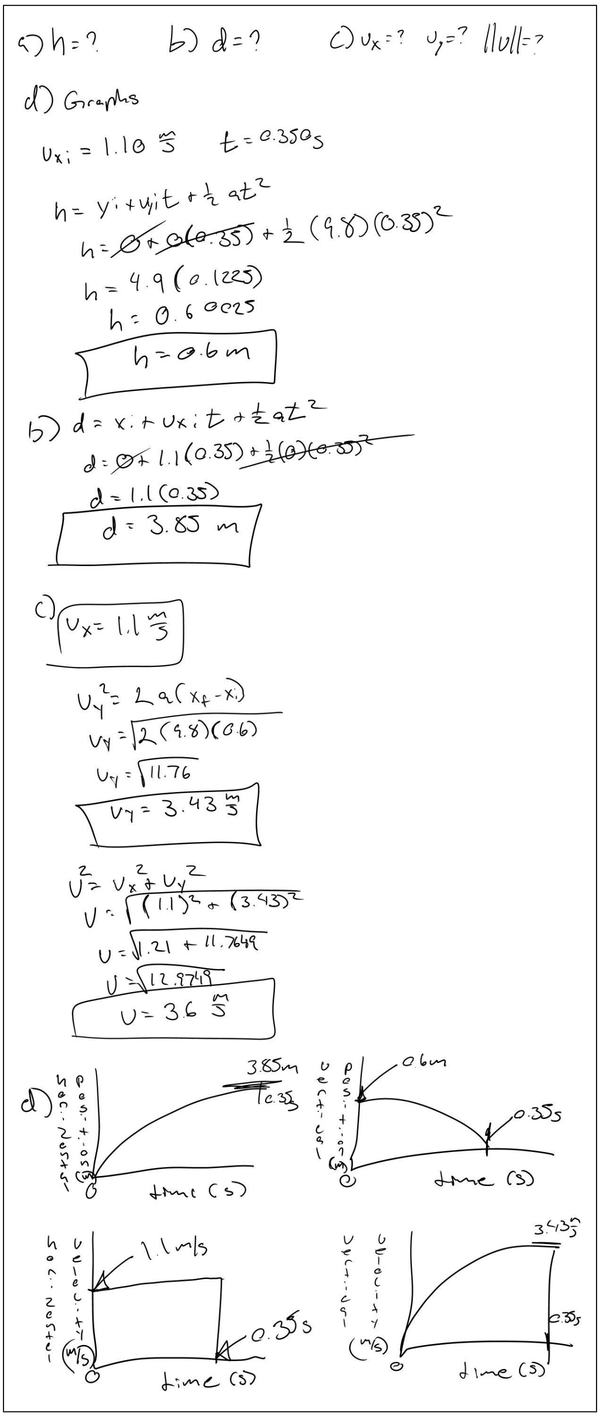
3.9 • A physics book slides off a horizontal tabletop with a speed of 1.10 m/s. It strikes the floor in 0.350 s. Ignore air resistance. Find (a) the height of the tabletop above the floor; (b) the horizontal distance from the edge of the table to the point where the book strikes the floor; (c) the horizontal and vertical components of the book's velocity, and the magnitude and direction of its velocity, just before the book reaches the floor. (d) Draw x-t, y-t, v_x-t, and v_y-t graphs for the motion.

Figure 9.43: Kinematics Problem (K-11), representing an instance of Time-of-Interest Animation.

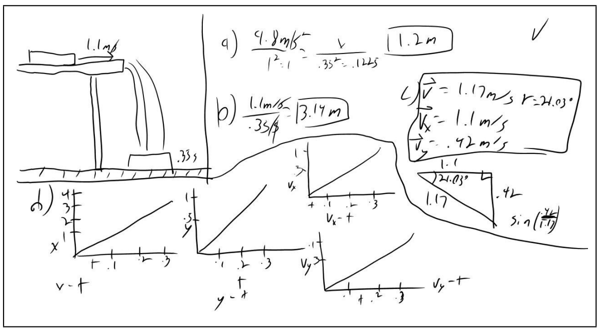
9.4 Summary

We have conducted a user study to collect real-world samples of physics solutions to better understand the animation requirements for diagrams sketched by students. The previous chapter examined some real world physics problems and described how they could be modeled using our prototype system. In doing so, it became apparent that by and large, the animation could be categorized in three distinct ways: Open-Ended, Time-of-Interest and Point-of-Interest.

In this chapter, we first described a set of 50 physics problems selected from Young's University Physics 13th Edition [226], and listed the animation category, based on the scenario and parameters described in the problem statement. These results are presented in Table 9.1. We then described our experimental procedure for collecting student solutions for the chosen physics problems. We were not able to collect the number of solutions that we had initially hoped for. However, the collected dataset is varied enough to provide insight into the requirements for animation of sketched physics diagrams. We have listed all 56 solutions collected during our experiment, and provided a discussion of which animation category each diagram belongs to. For some solutions, we were unable to identify the animation category because the student had not sketched a diagram. However, whenever a diagram was sketched, it could almost always be described using the three categories suggested by our earlier modeling of physics problems.



(a) K-11 Solution 1



(b) K-11 Solution 2

Figure 9.44: Two student solutions for the problem in Figure 9.43, one of which does not contain a diagram.

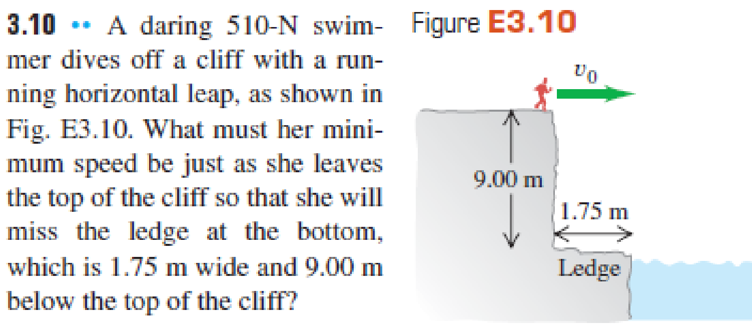
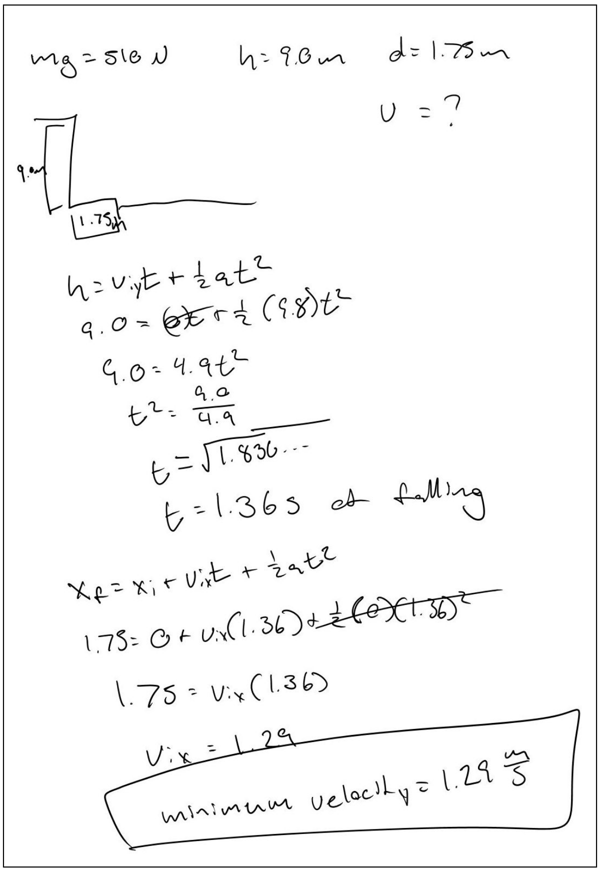
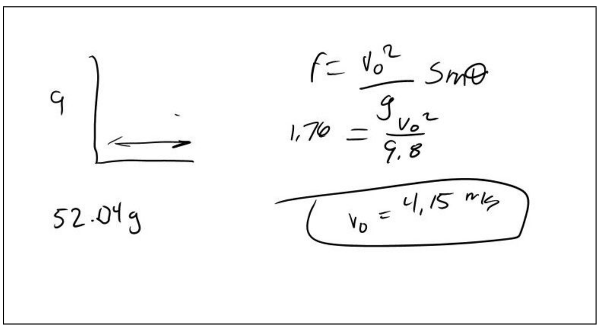

3.10 •• A daring 510-N swimmer dives off a cliff with a running horizontal leap, as shown in Fig. E3.10. What must her minimum speed be just as she leaves the top of the cliff so that she will miss the ledge at the bottom, which is 1.75 m wide and 9.00 m below the top of the cliff?

Figure 9.45: Kinematics Problem (K-12), representing an instance of Point-of-Interest Animation.



(a) K-12 Solution 1

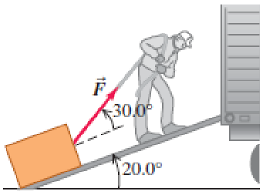


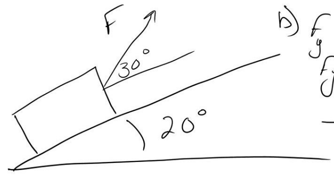
(b) K-12 Solution 2

Figure 9.46: Two student solutions for the problem in Figure 9.45

4.4 • A man is dragging a trunk up the loading ramp of a mover's truck. The ramp has a slope angle of 20.0°, and the man pulls upward with a force $\vec{F}$ whose direction makes an angle of 30.0° with the ramp (Fig. E4.4). (a) How large a force $\vec{F}$ is necessary for the component F_x parallel to the ramp to be 60.0 N? (b) How large will the component F_y perpendicular to the ramp then be?

Figure **E4.4**





b) $F_y = F \sin\theta$

$F_y = 69.3N \cdot \sin 30°$

$F_y = 34.65$

c) $F_{x\parallel} = F\cos\theta$

$60N = F\cos 30°$

$F = 69.3N$

(a) K-01 Statement

(b) K-01 Solution

Figure 9.47: Kinematics Problem (K-01), and a student solution. This example belongs to the Open-Ended animation category.

5.28 •• A box of bananas weighing 40.0 N rests on a horizontal surface. The coefficient of static friction between the box and the surface is 0.40, and the coefficient of kinetic friction is 0.20. (a) If no horizontal force is applied to the box and the box is at rest, how large is the friction force exerted on the box? (b) What is the magnitude of the friction force if a monkey applies a horizontal force of 6.0 N to the box and the box is initially at rest? (c) What minimum horizontal force must the monkey apply to start the box in motion? (d) What minimum horizontal force must the monkey apply to keep the box moving at constant velocity once it has been started? (e) If the monkey applies a horizontal force of 18.0 N, what is the magnitude of the friction force and what is the box's acceleration?

a) $F_s = (40)(.4)$

$F_s \leq 16$

b) 6.0

c) 16

D) $fk = (.20)(40)$

$= 8$

E) $\frac{10}{4.08} = 2.45$

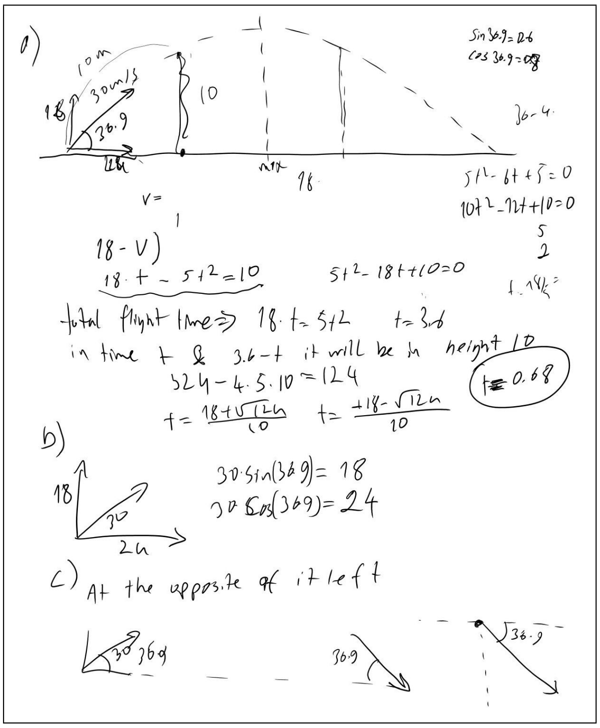
(a) K-09 Statement

(b) K-09 Solution

Figure 9.48: Kinematics Problem (K-09), and a student solution. This example belongs to the Open-Ended animation category. However, the student solution does not contain a diagram, and hence cannot be animated.

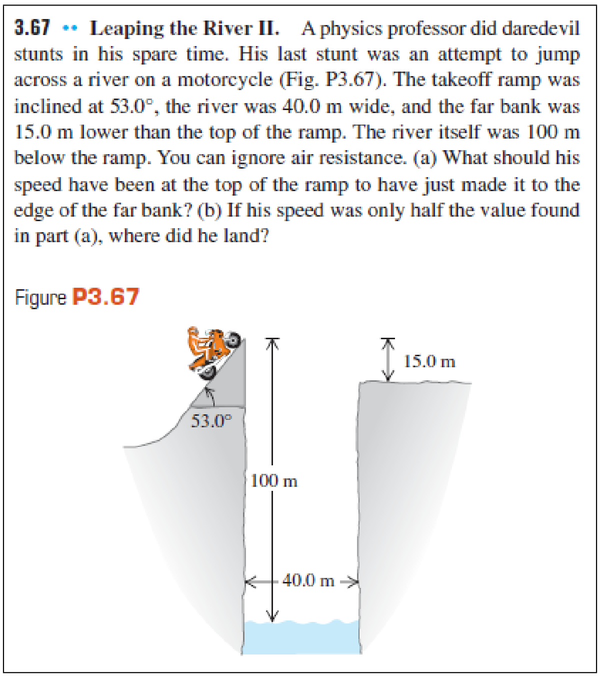
3.17 • A major leaguer hits a baseball so that it leaves the bat at a speed of 30.0 m/s and at an angle of 36.9° above the horizontal. You can ignore air resistance. (a) At what *two* times is the baseball at a height of 10.0 m above the point at which it left the bat? (b) Calculate the horizontal and vertical components of the baseball's velocity at each of the two times calculated in part (a). (c) What are the magnitude and direction of the baseball's velocity when it returns to the level at which it left the bat?

(a) K-13 Statement



(b) K-13 Solution

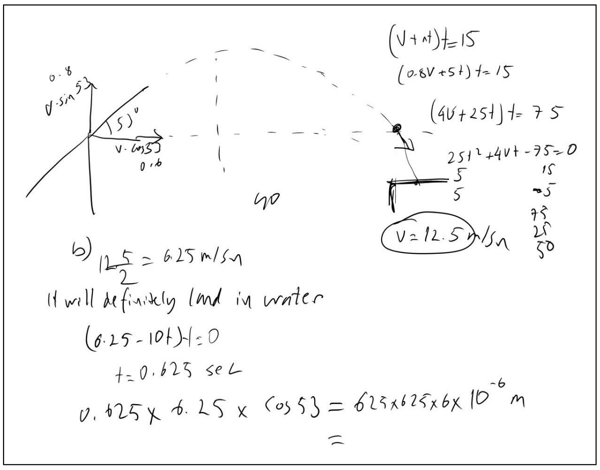
Figure 9.49: Kinematics Problem (K-13), and a student solution. This example belongs to the Point-of-Interest animation category.



3.67 •• **Leaping the River II.** A physics professor did daredevil stunts in his spare time. His last stunt was an attempt to jump across a river on a motorcycle (Fig. P3.67). The takeoff ramp was inclined at 53.0°, the river was 40.0 m wide, and the far bank was 15.0 m lower than the top of the ramp. The river itself was 100 m below the ramp. You can ignore air resistance. (a) What should his speed have been at the top of the ramp to have just made it to the edge of the far bank? (b) If his speed was only half the value found in part (a), where did he land?

Figure **P3.67**

(a) K-14 Statement



(b) K-14 Solution

Figure 9.50: Kinematics Problem (K-14), and a student solution. This example belongs to the Point-of-Interest animation category.

CHAPTER 10
CONCLUSION

10.1 Contributions

Section 1 states that our overarching research goal was to:

Investigate methods and techniques to enhance the state-of-the-art for pen-based intelligent tutoring systems in the domain of physics, with an emphasis on supporting natural workflow and providing animation support for sketched diagrams.

We have made the following important contributions toward this research goal:

1. Students can work on physics problems using our prototype ITS in a natural manner. We place three constraints on student input a) Only one diagram is permitted per solution b) the allowed diagram elements are limited to set described in Section 4.3.4 and c) chained mathematical steps are not permitted. All of these are soft limits. If the student sketches more than one diagram, unsupported elements, or writes chained expressions, our prototype would either not recognize them or would behave in an unpredictable manner. However, in order to support natural interaction, we still allow students to write them down.

2. Another goal was to adapt pen-based methods for intelligent tutoring purposes. We have made significant contributions toward this goal. Our system supports the use of two basic shapes (circles and polygons) to represent objects, two different types of surfaces (poly-lines and line segments) to model constrained motion, 3 different types of connectors to link shapes (wires, pulleys, and springs), and four different kinds of annotations (dotted lines,

arrows, intervals, and mathematical expressions) to construct simulations from diagram elements. Low-level recognition heuristics for some of these elements are well known (for example, PaleoSketch [161] describes heuristics for recognizing circles, polygons, helixes, lines, polylines). For others, we have designed new recognition methods (dotted lines, intervals, pulleys). These three primitives are not very common in sketch recognition literature. Further experimentation is required to investigate the accuracy of our recognition heuristics for these diagram elements. Similarly, we have devised high-level recognition and beautification methods tailored to deal with the sub-domains of physics that can be modeled using our prototype ITS. Of these, the QuickDraw [39] system is quite powerful and can be used to beautify certain aspects of physics diagrams with a high degree of accuracy.

3. Animation of student diagrams is an key research goal for us. We had initially used a single monolithic physics engine in our earlier prototypes (See Appendix A). This approach was useful, allowing us to support quick animation, without the user having to provide a full mathematical description for how objects should move in a particular scenario. Additionally, a physics engine encapsulates a subset of domain knowledge necessary for its operation. We demonstrated that realtime data transformations can be used to extend the animation support provided by a single physics engine. However, this approach was difficult to extend to new problem domains. In this work, we have described an animation framework that uses multiple simulators, each tailored to a particular class of physics problems. This is an extensible and modular design that will enable us to easily extend the capabilities of our system in the future.

4. We have analyzed of 50 real world physics problems and 56 student solutions, in order to identify the animation requirements for real-world physics solutions. Our findings indicate that, within the domain of kinematics, animations can be categorized broadly in three ways: Open-Ended, Time-of-Interest, and Point-of-Interest. Interestingly, we found that even if the

text of a problem indicates a particular category of animation, it may not be possible to use the said category with a student's sketched diagram. The chief hurdle is the definition of time and distance/point parameters for animation. If the student has not marked these clearly, then Open-Ended animation remains the only viable option. Additionally, if the student elects not to sketch a diagram, we cannot do animation.

5. We have also described a solution checking method based on deductive reasoning principles. Our method verifies a solution by finding a path through a graph structure. At this stage, we do not have a full implementation of the solution checker. In particular, we plan to integrate a mathematics package (such as Mathematica or Matlab) to be able to check the soundness of algebraic manipulations. We have described the limitations and strengths of our solution checker, which is still in its first iteration and can possibly yield significant improvements with more sophisticated methods.

6. A discussion of the range of physics problems that can be modeled using our prototype is presented in Chapter 8. Similarly, Chapter 8.4 presents an analysis of the limitations of our overall method and architecture, and also highlights some missing behaviors which need to be supported to extend the capabilities of our system.

10.2 Future Work

10.2.1 Improvement in Diagram Recognition

Until now, our goal has primarily been to model various types of physics problems and to try to devise methods for animating them. This has resulted in less focus on improving recognition accuracy and system stability and more focus on getting animations correct, even if several tries are required to write the symbols properly. An important benefit of this approach is that we now have a stable sketch recognition pipeline in place. Currently, the recognizers plugged into the pipeline are based on heuristic methods and are somewhat brittle. However, as our architecture is now stable, it is a simple matter to plug in more sophisticated recognition systems. Recognition heuristics allow for rapid development and testing but may not be as reliable as machine learning methods. Using machine learning methods poses additional challenges. Large amounts of labeled physics diagram and solution data are required to train machine learning algorithms, in order to yield good recognition accuracy.

In this vein, we have constructed a dataset of physics problems and their solutions, comprising 50 problems and 56 student solutions to a subset of chosen problems, acquired from 8 students. Chapter 9 describes our experiment for collecting solutions. In this work, we have used this dataset to identify common animation categories for a range of physics problems. In the future, we plan to use this dataset to test the effectiveness of existing state-of-the-art algorithms for ink stroke segmentation, text/non-text division, and sketch recognition. In summary, the sketch recognition pipeline and overall architecture for our prototype ITS is firmly in place. We have identified weak points in our pipeline and have begun work on improve it. Additionally, focusing on the animation and feedback mechanisms, rather than improving recognition, has also allowed us to identify common patterns which can be supported and expanded by future iterations of our pen-based ITS.

10.2.2 Improvement in Mathematics Recognition

Accurate recognition of handwritten mathematics is a key aspect of any pen-based tutoring system for STEM disciplines. The math recognizer in our prototype system uses the StarPad [191] framework. However, StarPad does not report actual accuracy numbers, instead reporting user satisfaction metrics. In practice, we have found that StarPad's recognition performance is good but errors occur often enough to be problematic.[1]

Mathematics recognition performance needs to improve drastically to be viable in tutoring systems. We plan to immediately start work on a new cross-platform, mathematics and handwriting recognition engine to address this concern. Mathematics and handwriting recognition is a key element in a wide variety of pen-based systems, and should to be more tractable than general physics diagram recognition. This notion allows us to hope that the development of a new math and handwriting recognition system should not be very difficult.

10.2.3 Support for New Types of Diagram Elements and Annotations

Our prototype uses only two basic shapes (circles and polygons) to model elements in a sketched diagram. Support for arbitrary shapes might be needed for some domains of physics. Additionally, it may be useful to allow users to define their own shapes as groupings of stroke objects (e.g., a stick figure described using multiple strokes[2])

10.2.4 Multimodal Interaction Methods

For this work, pen-based input and the text of a physics problem were used as inputs. Given the recent developments in sensing technologies, we foresee that eventually, other sources of input

[1] The following symbols are often problematic to write:

$$\{A, , , ., 5, S, R, N\}$$

.

[2] This functionality is supported in earlier systems such as MathPad^2. However, in MathPad^2, excessive lassoing is required to group such ink strokes together.

may also become necessary. These may include facial expression recognition, sound input, gaze detection and touch input. With these types of inputs available for an intelligent tutor, the interaction metaphor is not limited sketch understanding, but becomes closer to a perceptual user interface [203, 158] which must then deal with issues of multimodal input fusion [202, 157, 113].

10.2.5 Improvements in Animation Capabilities

Section 8.4 provides a detailed discussion of the limitations of our animation methodology. One important area of future work is to expand the animation capabilities for existing domains, and to construct new simulation engines, in order to support the animation of new types of physics problems.

Another area of future work is to provide more comprehensive support for the three categories of animation identified in Chapter 9. Specifically, this task can be broken down into identifying easy-to-use interaction metaphors that enable a user to easily indicate a large number of animation parameters.

10.2.6 Usability Testing

Intelligent tutoring systems require extensive testing to ascertain their impact upon student learning. Our prototype ITS is not ready for user testing at this stage, primarily due to the variation in recognition performance. Several things need to happen before our ITS is ready for a full fledged usability evaluation. The following list is a condensed version of other points already discussed in this chapter:

- Mathematics recognition performance must improve significantly.
- Our sketch recognition pipeline can benefit from more sophisticated classification methods, yielding improvements in stability and recognition performance.
- Deeper analysis of hand-written solutions from physics students to help identify the usability issues for our prototype system.

APPENDIX A
EARLIER PROTOTYPES

We have constructed three prototype systems prior to the system described in this work. This section briefly outlines the progress from each prototype to the next and describes the differences between them and our current system.

A.1 Prototype 1: Proof of Concept

The primary goal of our initial prototype was to investigate how the answer to a physics problem could be used to animate a sketched diagram. To this end, we constructed a proof-of-concept system that fused mathematical sketching [119] with an underlying physics engine to allow creation of dynamic illustrations for selected physics concepts. This fusion provided us with a mechanism to infer how to make a proper animation given different levels of granularity of user input, from diagram only to complete behavioral specification. Through the use of a customized physics engine [140], we were able to encode the relationship between acceleration, velocity, and position/orientation into the system. Figure A.1 shows a spring system that showcases how the encoded relationship enables a user to specify the minimum information needed to animate the diagram.

A.2 Prototype 2: Sketch Beautification

While the first prototype proved useful in constructing animations, it had several shortcomings. First, there was no mechanism for correcting sketched drawings. Our initial prototype was also very simple in its design. Second, it didn't support the use of equations with dependencies on other mathematical steps in the solution. For example, in Figure A.2, the net force on the box can be computed as $\sum F = mg \sin \alpha$, which depends on the values of α and m defined by the user.

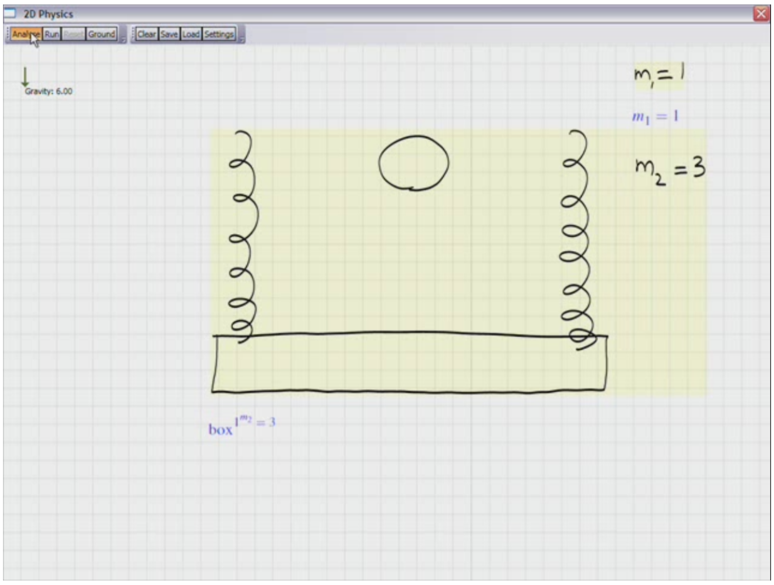


Figure A.1: A spring system sketched in our proof-of-concept system. The user writes down only the masses of the ball and the board. All the remaining mathematical description necessary for animation is provided transparently by our system.

Our initial system also didn't include any feedback mechanisms e.g., graphing, or viewing how a particular quantity associated with a shape changed during the course of animation.

The most major contribution of our second prototype was its sketch-correction mechanism. Figure A.2 shows an inclined plane problem that highlights the approximate nature of hand-drawn sketches. The triangle drawn by the user is approximately right-angled. Likewise, the inscribed angle α is not exactly $\pi/4$. Such approximations are acceptable with pen and paper diagrams because the user relies on his imagination to see concepts in action. However, these inaccuracies caused problems for the physics engine in our first prototype due to ambiguity between the precise mathematical specifications and the imprecise drawings. We, therefore developed techniques to mitigate these inaccuracies.

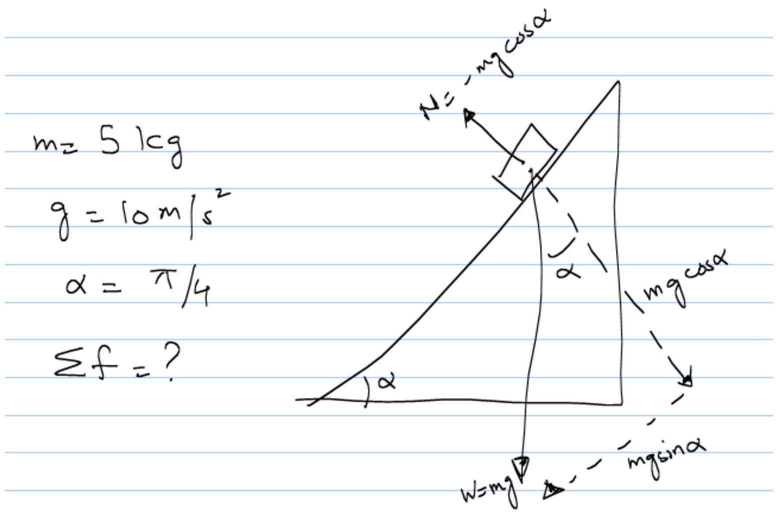


Figure A.2: A typical inclined plane diagram drawn by a student.

A.3 Prototype 3: PhysicsBook

Our third prototype was called PhysicsBook [40]. PhysicsBook shared the use of a customized physics engine for encoding domain knowledge with previous iterations. This enabled it to develop an understanding of physics problems that require a student to work out either force(s), acceleration, velocity, displacement, position, or mass. As before, users can associate their own equations with the diagram by using a simple gesture set. With PhysicsBook, users could also use annotations such as arrows, dotted lines and angles to provide more information for the animation subsystem. PhysicsBook included optimizations to cut down on extraneous associations. Lastly, It extended the capabilities of previous iterations to branches of physics related to $f = ma$. This was achieved by designing a framework that could perform the necessary data transformations required to convert given variables in an equation to one of the acceptable inputs.

APPENDIX B
IRB DOCUMENTATION

University of Central Florida Institutional Review Board
Office of Research & Commercialization
12201 Research Parkway, Suite 501
Orlando, Florida 32826-3246
Telephone: 407-823-2901 or 407-882-2276
www.research.ucf.edu/compliance/irb.html

Approval of Human Research

From: **UCF Institutional Review Board #1**
FWA00000351, IRB00001138

To: **Salman S. Cheema** and Co-PI: **Joseph J. LaViola II**

Date: **August 23, 2013**

Dear Researcher:

On 8/23/2013, the IRB approved the following human participant research until 8/22/2014 inclusive:

Type of Review:	UCF Initial Review Submission Form
Project Title:	PhysicsBook: A Sketch-based Intelligent Tutoring System
Investigator:	Salman S Cheema
IRB Number:	SBE-13-09570
Funding Agency:	
Grant Title:	
Research ID:	N/A

The scientific merit of the research was considered during the IRB review. The Continuing Review Application must be submitted 30days prior to the expiration date for studies that were previously expedited, and 60 days prior to the expiration date for research that was previously reviewed at a convened meeting. Do not make changes to the study (i.e., protocol, methodology, consent form, personnel, site, etc.) before obtaining IRB approval. A Modification Form **cannot** be used to extend the approval period of a study. All forms may be completed and submitted online at https://iris.research.ucf.edu .

If continuing review approval is not granted before the expiration date of 8/22/2014, approval of this research expires on that date. When you have completed your research, please submit a Study Closure request in iRIS so that IRB records will be accurate.

Use of the approved, stamped consent document(s) is required. The new form supersedes all previous versions, which are now invalid for further use. Only approved investigators (or other approved key study personnel) may solicit consent for research participation. Participants or their representatives must receive a copy of the consent form(s).

In the conduct of this research, you are responsible to follow the requirements of the Investigator Manual.

On behalf of Sophia Dziegielewski, Ph.D., L.C.S.W., UCF IRB Chair, this letter is signed by:

Signature applied by Joanne Muratori on 08/23/2013 01:47:00 PM EDT

Joanne Muratori

IRB Coordinator

Page 1 of 1

University of Central Florida

University of Central Florida Institutional Review Board
Office of Research & Commercialization
12201 Research Parkway, Suite 501
Orlando, Florida 32826-3246
Telephone: 407-823-2901, 407-882-2012 or 407-882-2276
www.research.ucf.edu/compliance/irb.html

Acknowledgment of Study Closure

From : **UCF Institutional Review Board #1**
FWA00000351, IRB00001138

To : **Salman S. Cheema** and Co-PI: **Joseph J. LaViola II**

Date : **August 18, 2014**

Dear Researcher:

On 8/18/2014 the IRB conducted an administrative review of the FORM: Study Closure Request that you submitted in iRIS. The study has been closed within the system.

This report is in regards to:

Type of Review:	Study Closure
Project Title:	PhysicsBook: A Sketch-based Intelligent Tutoring System
Investigator:	Salman S. Cheema
IRB Number:	SBE-13-09570
Funding Agency:	
Grant Title:	
Research ID:	N/A

As part of this action:

- The research is permanently closed to enrollment.
- All participants have completed all research-related interventions.
- Collection of private identifiable information is completed.
- Analysis of private identifiable information is completed.

Thank you for notifying the IRB of this modification.

On behalf of Sophia Dziegielewski, Ph.D., L.C.S.W., UCF IRB Chair, this letter is signed by:

Kanielle Chap

IRB Coordinator

Submission Reference Number: 021667

LIST OF REFERENCES

[1] Gregory D. Abowd. Software engineering issues for ubiquitous computing. In *Proceedings of the 21st International Conference on Software Engineering*, ICSE '99, pages 75–84, New York, NY, USA, 1999. ACM.

[2] M. Agrawal, A Zotov, Ming Ye, and S. Raghupathy. Context aware on-line diagramming recognition. In *Frontiers in Handwriting Recognition (ICFHR), 2010 International Conference on*, pages 682–687, Nov 2010.

[3] Shaaron Ainsworth, Vaughan Prain, and Russell Tytler. Drawing to Learn in Science. *Science*, 333(6046):1096–1097, August 2011.

[4] Algodoo, 2011. `http://www.algodoo.com/wiki/Home`.

[5] Christine Alvarado. A natural sketching environmant: Bringing the computer into early stages of mechanical design. Master's thesis, MIT, 2000.

[6] Christine Alvarado and Randall Davis. Sketchread: A multi-domain sketch recognition engine. In *Proceedings of the 17th Annual ACM Symposium on User Interface Software and Technology*, UIST '04, pages 23–32, New York, NY, USA, 2004. ACM.

[7] Christine Alvarado and Michael Lazzareschi. Properties of real-world digital logic diagrams. In *Pen-Based Learning Technologies, 2007. PLT 2007. First International Workshop on*, pages 1–6. IEEE, 2007.

[8] Derek Anderson, Craig Bailey, and Marjorie Skubic. Hidden markov model symbol recognition for sketch-based interfaces. In *AAAI Fall Symposium*, pages 15–21, 2004.

[9] John R. Anderson, Albert T. Corbett, Kenneth R. Koedinger, and Ray. Pelletier. Cognitive tutors: Lessons learned. *Journal of the Learning Sciences*, 4(2):167–207, 1995.

[10] Lisa Anthony and Jacob O. Wobbrock. A lightweight multistroke recognizer for user interface prototypes. In *Proceedings of Graphics Interface 2010*, GI '10, pages 245–252, Toronto, Ont., Canada, Canada, 2010. Canadian Information Processing Society.

[11] Lisa Anthony and Jacob O. Wobbrock. $n-protractor: A fast and accurate multistroke recognizer. In *Proceedings of Graphics Interface 2012*, GI '12, pages 117–120, Toronto, Ont., Canada, Canada, 2012. Canadian Information Processing Society.

[12] Lisa Anthony, Jie Yang, and Kenneth R. Koedinger. Evaluation of multimodal input for entering mathematical equations on the computer. In *CHI '05 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '05, pages 1184–1187, New York, NY, USA, 2005. ACM.

[13] Lisa Anthony, Jie Yang, and Kenneth R. Koedinger. Benefits of handwritten input for students learning algebra equation solving. In *Proceedings of the 2007 Conference on Artificial Intelligence in Education: Building Technology Rich Learning Contexts That Work*, pages 521–523, Amsterdam, The Netherlands, The Netherlands, 2007. IOS Press.

[14] Lisa Anthony, Jie Yang, and Kenneth R. Koedinger. A paradigm for handwriting-based intelligent tutors. *Int. J. Hum.-Comput. Stud.*, 70(11):866–887, November 2012.

[15] Relja Arandjelovic and Tevfik Metin Sezgin. Sketch recognition by fusion of temporal and image-based features. *Pattern Recognition*, 44(6):1225 – 1234, 2011.

[16] T. Artieres, S. Marukatat, and P. Gallinari. Online handwritten shape recognition using segmental hidden markov models. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 29(2):205–217, Feb 2007.

[17] Olufunmilola Atilola, Stephanie Valentine, Hong-Hoe Kim, David Turner, Erin McTigue, Tracy Hammond, and Julie Linsey. Mechanix: A natural sketch interface tool for teaching truss analysis and free-body diagrams. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, 28:169–192, 5 2014.

[18] Seok-Hyung Bae, Ravin Balakrishnan, and Karan Singh. Ilovesketch: As-natural-as-possible sketching system for creating 3d curve models. In *Proceedings of the 21st Annual ACM Symposium on User Interface Software and Technology*, UIST '08, pages 151–160, New York, NY, USA, 2008. ACM.

[19] Seok-Hyung Bae, Ravin Balakrishnan, and Karan Singh. Everybodylovessketch: 3d sketching for a broader audience. In *Proceedings of the 22Nd Annual ACM Symposium on User Interface Software and Technology*, UIST '09, pages 59–68, New York, NY, USA, 2009. ACM.

[20] Henry S. Baird. Background structure in document images. *International Journal of Pattern Recognition and Artificial Intelligence*, 08(05):1013–1030, 1994.

[21] David Baraff. An introduction to physically based modeling: rigid body simulation i-unconstrained rigid body dynamics. *SIGGRAPH Course Notes*, 1997.

[22] David Baraff. An introduction to physically based modeling: Rigid body simulation ii-nonpenetration constraints. *SIGGRAPH course notes*, pages D31–D68, 1997.

[23] Adrien Bernhardt, Adeline Pihuit, Marie-Paule Cani, and Loic Barthe. Matisse: Painting 2d regions for modeling free-form shapes. In *SBIM '08*, pages 57–64, 2008.

[24] Akshay Bhat and Tracy Hammond. Using entropy to distinguish shape versus text in hand-drawn diagrams. In *IJCAI*, volume 9, pages 1395–1400, 2009.

[25] Christopher M Bishop, Markus Svensen, and Goeffrey E Hinton. Distinguishing text from graphics in on-line handwritten ink. In *IWFHR*, volume 4, pages 142–147, 2004.

[26] Rachel Blagojevic, Samuel Hsiao-Heng Chang, and Beryl Plimmer. The power of automatic feature selection: Rubine on steroids. In *Proceedings of the Seventh Sketch-Based Interfaces and Modeling Symposium*, SBIM '10, pages 79–86, Aire-la-Ville, Switzerland, Switzerland, 2010. Eurographics Association.

[27] Rachel Blagojevic, Beryl Plimmer, John Grundy, and Yong Wang. Using data mining for digital ink recognition: Dividing text and shapes in sketched diagrams. *Computers & Graphics*, 35(5):976–991, 2011.

[28] Julien Blanchard and Thierry Artieres. On-line handwritten documents segmentation. In *Frontiers in Handwriting Recognition, 2004. IWFHR-9 2004. Ninth International Workshop on*, pages 148–153. IEEE, 2004.

[29] Lars Bollen and Wouter R van Joolingen. Simsketch: Multi-agent simulations based on learner-created sketches for early science education. *IEEE transactions on learning technologies*, page 1, 2013.

[30] Jared N. Bott, Daniel Gabriele, and Joseph J. LaViola, Jr. Now or later: An initial exploration into user perception of mathematical expression recognition feedback. In *Proceedings of the Eighth Eurographics Symposium on Sketch-Based Interfaces and Modeling*, SBIM '11, pages 125–132, New York, NY, USA, 2011. ACM.

[31] Jared N. Bott and Joseph J. LaViola, Jr. A pen-based tool for visualizing vector mathematics. In *Proceedings of the Seventh Sketch-Based Interfaces and Modeling Symposium*, SBIM '10, pages 103–110, Aire-la-Ville, Switzerland, Switzerland, 2010. Eurographics Association.

[32] Martin Bresler, Daniel Pruša, and Václav Hlavác. Simultaneous segmentation and recognition of graphical symbols using a composite descriptor. *CVWW*, 13:16–23, 2013.

[33] Sarah Buchanan and Joseph J. Laviola, Jr. Cstutor: A sketch-based tool for visualizing data structures. *Trans. Comput. Educ.*, 14(1):3:1–3:28, March 2014.

[34] Chris Calhoun, Thomas F Stahovich, Tolga Kurtoglu, and Levent Burak Kara. Recognizing multi-stroke symbols. In *AAAI Spring Symposium on Sketch Understanding*, pages 15–23, 2002.

[35] Giovanni Casella, Vincenzo Deufemia, Viviana Mascardi, Gennaro Costagliola, and Maurizio Martelli. An agent-based framework for sketched symbol interpretation. *Journal of Visual Languages & Computing*, 19(2):225–257, 2008.

[36] Sonya Cates. *Combining Representations for Improved Sketch Recognition*. PhD thesis, Massachusetts Institute of Technology, September 2009.

[37] R Cattoni, T Coianiz, S Messelodi, and CM Modena. Geometric layout analysis techniques for document image understanding: a review. *ITC-irst Technical Report*, 9703(09), 1998.

[38] Salman Cheema, Sarah Buchanan, Sumit Gulwani, and Joseph J. LaViola, Jr. A practical framework for constructing structured drawings. In *Proceedings of the 19th International Conference on Intelligent User Interfaces*, IUI '14, pages 311–316, New York, NY, USA, 2014. ACM.

[39] Salman Cheema, Sumit Gulwani, and Joseph LaViola. Quickdraw: Improving drawing experience for geometric diagrams. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '12, pages 1037–1046, New York, NY, USA, 2012. ACM.

[40] Salman Cheema and Joseph LaViola. Physicsbook: A sketch-based interface for animating physics diagrams. In *Proceedings of the 2012 ACM International Conference on Intelligent User Interfaces*, IUI '12, pages 51–60, New York, NY, USA, 2012. ACM.

[41] Salman Cheema and Joseph J. LaViola, Jr. Applying mathematical sketching to sketch-based physics tutoring software. In *Proceedings of the 10th international conference on Smart graphics*, SG'10, pages 13–24, Berlin, Heidelberg, 2010. Springer-Verlag.

[42] Salman Cheema and Joseph J. LaViola, Jr. Towards intelligent motion inferencing in mathematical sketching. In *Proceedings of the 15th international conference on Intelligent user interfaces*, IUI '10, pages 289–292, New York, NY, USA, 2010. ACM.

[43] C.E. Cheong, Ho-Yon Kim, Jang-Won Suh, and J.-H. Kim. Handwritten numeral string recognition with stroke grouping. In *Document Analysis and Recognition, 1999. ICDAR '99. Proceedings of the Fifth International Conference on*, pages 745–748, Sept 1999.

[44] John Clement. Imagistic simulation and physical intuition in expert problem solving. In *Proceedings of the 16th annual conference of the Cognitive Science Society*, pages 201–206, 1994.

[45] Jonathan M. Cohen, John F. Hughes, and Robert C. Zeleznik. Harold: A world made of drawings. In *Proceedings of the 1st International Symposium on Non-photorealistic Animation and Rendering*, NPAR '00, pages 83–90, New York, NY, USA, 2000. ACM.

[46] Philip R. Cohen, Michael Johnston, David McGee, Sharon Oviatt, Jay Pittman, Ira Smith, Liang Chen, and Josh Clow. Quickset: Multimodal interaction for distributed applications. In *Proceedings of the Fifth ACM International Conference on Multimedia*, MULTIMEDIA '97, pages 31–40, New York, NY, USA, 1997. ACM.

[47] Cristina Conati, Abigail Gertner, and Kurt VanLehn. Using bayesian networks to manage uncertainty in student modeling. *User Modeling and User-Adapted Interaction*, 12(4):371–417, 2002.

[48] Travis J. Cossairt and Joseph J. LaViola, Jr. Setpad: A sketch-based tool for exploring discrete math set problems. In *Proceedings of the International Symposium on Sketch-Based Interfaces and Modeling*, SBIM '12, pages 47–56, Aire-la-Ville, Switzerland, Switzerland, 2012. Eurographics Association.

[49] Gennaro Costagliola, Mattia De Rosa, and Vittorio Fuccella. Recognition and autocompletion of partially drawn symbols by using polar histograms as spatial relation descriptors. *Computers & Graphics*, 39(0):101 – 116, 2014.

[50] Crayon physics, 2014. `http://www.crayonphysics.com/`.

[51] Richard C. Davis, Brien Colwell, and James A. Landay. K-sketch: a 'kinetic' sketch pad for novice animators. In *CHI '08*, pages 413–422, 2008.

[52] Mattia De Rosa. *New Methods, techniques and applications for sketch recognition*. PhD thesis, Universita degli studi di Salerno, 2014.

[53] Ruwanee de Silva, David Tyler Bischel, WeeSan Lee, Eric J. Peterson, Robert C. Calfee, and Thomas F. Stahovich. Kirchhoff's pen: A pen-based circuit analysis tutor. In *Proceedings of the 4th Eurographics Workshop on Sketch-based Interfaces and Modeling*, SBIM '07, pages 75–82, New York, NY, USA, 2007. ACM.

[54] Adrien Delaye and Eric Anquetil. {HBF49} feature set: A first unified baseline for online symbol recognition. *Pattern Recognition*, 46(1):117 – 130, 2013.

[55] Adrien Delaye and Cheng-Lin Liu. Contextual text/non-text stroke classification in online handwritten notes with conditional random fields. *Pattern Recognition*, 47(3):959 – 968, 2014. Handwriting Recognition and other {PR} Applications.

[56] Adrien Delaye and Cheng-Lin Liu. Multi-class segmentation of free-form online documents with tree conditional random fields. *International Journal on Document Analysis and Recognition (IJDAR)*, pages 1–17, 2014.

[57] Michel C. Desmarais and Xiaoming Pu. A bayesian student model without hidden nodes and its comparison with item response theory. *Int. J. Artif. Intell. Ed.*, 15(4):291–323, December 2005.

[58] Daniel Dixon, Manoj Prasad, and Tracy Hammond. icandraw: Using sketch recognition and corrective feedback to assist a user in drawing human faces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '10, pages 897–906, New York, NY, USA, 2010. ACM.

[59] N. Donmez and K. Singh. Concepture: A regular language based framework for recognizing gestures with varying and repetitive patterns. In *Proceedings of the International Symposium on Sketch-Based Interfaces and Modeling*, SBIM '12, pages 29–37, Aire-la-Ville, Switzerland, Switzerland, 2012. Eurographics Association.

[60] M.-P. Dubuisson and AK. Jain. A modified hausdorff distance for object matching. In *Pattern Recognition, 1994. Vol. 1 - Conference A: Computer Vision amp; Image Processing., Proceedings of the 12th IAPR International Conference on*, volume 1, pages 566–568 vol.1, Oct 1994.

[61] David H Eberly. *Game physics*. Elsevier, 2003.

[62] Christiane Fellbaum. *WordNet*. Wiley Online Library, 1998.

[63] D.G. Fernndez-Pacheco, F. Albert, N. Aleixos, and J. Conesa. A new paradigm based on agents applied to free-hand sketch recognition. *Expert Systems with Applications*, 39(8):7181 – 7195, 2012.

[64] Martin Field, Stephanie Valentine, Julie Linsey, and Tracy Hammond. Sketch recognition algorithms for comparing complex and unpredictable shapes. In *Proceedings of the Twenty-Second international joint conference on Artificial Intelligence-Volume Volume Three*, pages 2436–2441. AAAI Press, 2011.

[65] Michael Fligner, Joseph Verducci, Jeff Bjoraker, and Paul Blower. A new association coefficient for molecular dissimilarity. In *The Second Joint Sheffield Conference on Chemoinformatics*, 2001.

[66] Manuel J Fonseca, César Pimentel, and Joaquim A Jorge. Cali: An online scribble recognizer for calligraphic interfaces. In *AAAI spring symposium on sketch understanding*, pages 51–58, 2002.

[67] Kenneth Forbus, Jeffrey Usher, Andrew Lovett, Kate Lockwood, and Jon Wetzel. Cogsketch: Sketch understanding for cognitive science research and for education. *Topics in Cognitive Science*, 2011.

[68] Andrew Forsberg, Mark Dieterich, and Robert Zeleznik. The music notepad. In *Proceedings of the 11th annual ACM symposium on User interface software and technology*, UIST '98, pages 203–210, New York, NY, USA, 1998. ACM.

[69] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. Additive logistic regression: a statistical view of boosting (with discussion and a rejoinder by the authors). *The Annals of Statistics*, 28(2):337–407, 04 2000.

[70] Luoting Fu and Levent Burak Kara. From engineering diagrams to engineering models: Visual recognition and applications. *Computer-Aided Design*, 43(3):278 – 292, 2011.

[71] Luoting Fu and Levent Burak Kara. Neural network-based symbol recognition using a few labeled samples. *Computers & Graphics*, 35(5):955 – 966, 2011.

[72] Leslie Gennari, Levent Burak Kara, Thomas F Stahovich, and Kenji Shimada. Combining geometry and domain knowledge to interpret hand-drawn diagrams. *Computers & Graphics*, 29(4):547–562, 2005.

[73] Abigail S Gertner, Cristina Conati, and Kurt VanLehn. Procedural help in andes: Generating hints using a bayesian network student model. *AAAI/IAAI*, 1998:106–11, 1998.

[74] AC. Graesser, P. Chipman, B.C. Haynes, and A Olney. Autotutor: an intelligent tutoring system with mixed-initiative dialogue. *Education, IEEE Transactions on*, 48(4):612–618, Nov 2005.

[75] Arthur C Graesser, Katja Wiemer-Hastings, Peter Wiemer-Hastings, and Roger Kreuz. Autotutor: A simulation of a human tutor. *Cognitive Systems Research*, 1(1):35 – 51, 1999.

[76] ArthurC. Graesser, Shulan Lu, GeorgeTanner Jackson, HeatherHite Mitchell, Mathew Ventura, Andrew Olney, and MaxM. Louwerse. Autotutor: A tutor with dialogue in natural language. *Behavior Research Methods, Instruments, & Computers*, 36(2):180–192, 2004.

[77] Mark D. Gross. The electronic cocktail napkina computational environment for working with design diagrams. *Design Studies*, 17(1):53 – 69, 1996.

[78] Mark D. Gross and Ellen Yi-Luen Do. Ambiguous intentions: A paper-like interface for creative design. In *Proceedings of the 9th Annual ACM Symposium on User Interface Software and Technology*, UIST '96, pages 183–192, New York, NY, USA, 1996. ACM.

[79] John Grundy and J. Hosking. Supporting generic sketching-based input of diagrams in a domain-specific visual language meta-tool. In *Software Engineering, 2007. ICSE 2007. 29th International Conference on*, pages 282–291, May 2007.

[80] A. Hall, C. Pomm, and P. Widmayer. A combinatorial approach to multi-domain sketch recognition. In *Proceedings of the 4th Eurographics Workshop on Sketch-based Interfaces and Modeling*, SBIM '07, pages 7–14, New York, NY, USA, 2007. ACM.

[81] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. The weka data mining software: An update. *SIGKDD Explor. Newsl.*, 11(1):10–18, November 2009.

[82] Tracy Hammond and Randall Davis. Tahuti: A geometrical sketch recognition system for uml class diagrams. In *AAAI Spring Symposium on Sketch Understanding*, pages 59–68, Stanford, California, March 25-27 2002. AAAI Press.

[83] Tracy Hammond and Randall Davis. Ladder, a sketching language for user interface developers. *Computers & Graphics*, 29(4):518 – 532, 2005.

[84] Tracy Hammond and Brandon Paulson. Recognizing sketched multistroke primitives. *ACM Trans. Interact. Intell. Syst.*, 1(1):4:1–4:34, October 2011.

[85] Tracy A. Hammond and Randall Davis. Recognizing interspersed sketches quickly. In *Proceedings of Graphics Interface 2009*, GI '09, pages 157–166, Toronto, Ont., Canada, Canada, 2009. Canadian Information Processing Society.

[86] R.M. Haralick, Stanley R. Sternberg, and Xinhua Zhuang. Image analysis using mathematical morphology. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, PAMI-9(4):532–550, July 1987.

[87] Mary Hegarty. Mechanical reasoning by mental simulation. *Trends in Cognitive Sciences*, 8(6):280 – 285, 2004.

[88] J. Herold and T. F. Stahovich. Classyseg: A machine learning approach to automatic stroke segmentation. In *Proceedings of the Eighth Eurographics Symposium on Sketch-Based Interfaces and Modeling*, SBIM '11, pages 109–116, New York, NY, USA, 2011. ACM.

[89] James Herold and Thomas F. Stahovich. Speedseg: A technique for segmenting pen strokes using pen speed. *Computers and Graphics*, 35(2):250 – 264, 2011.

[90] Geoffrey Holmes, Bernhard Pfahringer, Richard Kirkby, Eibe Frank, and Mark Hall. Multiclass alternating decision trees. In Tapio Elomaa, Heikki Mannila, and Hannu Toivonen, editors, *Machine Learning: ECML 2002*, volume 2430 of *Lecture Notes in Computer Science*, pages 161–172. Springer Berlin Heidelberg, 2002.

[91] Jason I. Hong and James A. Landay. Satin: A toolkit for informal ink-based applications. In *Proceedings of the 13th Annual ACM Symposium on User Interface Software and Technology*, UIST '00, pages 63–72, New York, NY, USA, 2000. ACM.

[92] Heloise Hwawen Hse and A. Richard Newton. Recognition and beautification of multistroke symbols in digital ink. *Computers & Graphics*, 29(4):533 – 546, 2005.

[93] Takeo Igarashi, Satoshi Matsuoka, Sachiko Kawachiya, and Hidehiko Tanaka. Interactive beautification: a technique for rapid geometric design. In *UIST '97*, pages 105–114, 1997.

[94] Takeo Igarashi, Satoshi Matsuoka, and Hidehiko Tanaka. Teddy: a sketching interface for 3d freeform design. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '99, pages 409–416, New York, NY, USA, 1999. ACM Press/Addison-Wesley Publishing Co.

[95] Emanuel Indermühle, Horst Bunke, Faisal Shafait, and Thomas Breuel. Text versus non-text distinction in online handwritten documents. In *Proceedings of the 2010 ACM Symposium on Applied Computing*, SAC '10, pages 3–7, New York, NY, USA, 2010. ACM.

[96] Emanuel Indermühle, Volkmar Frinken, and Horst Bunke. Mode detection in online handwritten documents using blstm neural networks. *ICFHR*, 12:302–307, 2012.

[97] Anil K Jain, Anoop M Namboodiri, and Jayashree Subrahmonia. Structure in on-line documents. In *Document Analysis and Recognition, 2001. Proceedings. Sixth International Conference on*, pages 844–848. IEEE, 2001.

[98] Yingying Jiang, Feng Tian, Hongan Wang, Xiaolong Zhang, Xugang Wang, and Guozhong Dai. Intelligent understanding of handwritten geometry theorem proving. In *Proceedings of the 15th international conference on Intelligent user interfaces*, IUI '10, pages 119–128, New York, NY, USA, 2010. ACM.

[99] Gabe Johnson, Mark Gross, Ellen Yi-Luen Do, and Jason Hong. Sketch it, make it: Sketching precise drawings for laser cutting. In *CHI '12 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '12, pages 1079–1082, New York, NY, USA, 2012. ACM.

[100] W Lewis Johnson, Jeff Rickel, Randy Stiles, and Allen Munro. Integrating pedagogical agents into virtual environments. *Presence: Teleoperators and Virtual Environments*, 7(6):523–546, 1998.

[101] PN Johnson-Laird. Imagery, visualization, and thinking. *Perception and cognition at centurys end*, pages 441–467, 1998.

[102] David Johnston and Christine Alvarado. *Sketch Recognition of Digital Logical Circuits*. PhD thesis, MS Thesis, Dept. Comput. Sci. Eng., Univ. California, San Diego, CA, 2013.

[103] Bo Kang and Joseph LaViola. Logicpad: A pen-based application for visualization and verification of boolean algebra. In *Proceedings of the 2012 ACM International Conference on Intelligent User Interfaces*, IUI '12, pages 265–268, New York, NY, USA, 2012. ACM.

[104] Levent Burak Kara, Leslie Gennari, and Thomas F. Stahovich. A sketch-based tool for analyzing vibratory mechanical systems. *Journal of Mechanical Design*, 130(10):101101, 2008.

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

[106] Manolya Kavakli and John S Gero. Sketching as mental imagery processing. *Design Studies*, 22(4):347 – 364, 2001.

[107] Rubaiat Habib Kazi, Fanny Chevalier, Tovi Grossman, Shengdong Zhao, and George Fitzmaurice. Draco: Bringing life to illustrations. In *CHI '14 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '14, pages 579–582, New York, NY, USA, 2014. ACM.

[108] Rubaiat Habib Kazi, Takeo Igarashi, Shengdong Zhao, and Richard Davis. Vignette: interactive texture design and manipulation with freeform gestures for pen-and-ink illustration. In *CHI '12*, pages 1727–1736, 2012.

[109] Koichi Kise, Akinori Sato, and Motoi Iwata. Segmentation of page images using the area voronoi diagram. *Computer Vision and Image Understanding*, 70(3):370 – 382, 1998.

[110] J. Kittler, M. Hatef, R. P W Duin, and J. Matas. On combining classifiers. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 20(3):226–239, Mar 1998.

[111] Kenneth R Koedinger, Albert Corbett, et al. *Cognitive tutors: Technology bringing learning sciences to the classroom*. na, 2006.

[112] Tolga Kurtoglu and Thomas F Stahovich. Interpreting schematic sketches using physical reasoning. In *AAAI Spring Symposium on Sketch Understanding*, pages 78–85. AAAI Press Menlo Park, CA, 2002.

[113] Denis Lalanne, Laurence Nigay, philippe Palanque, Peter Robinson, Jean Vanderdonckt, and Jean-François Ladry. Fusion engines for multimodal input: A survey. In *Proceedings of the 2009 International Conference on Multimodal Interfaces*, ICMI-MLMI '09, pages 153–160, New York, NY, USA, 2009. ACM.

[114] Mary LaLomia. User acceptance of handwritten recognition accuracy. In *Conference Companion on Human Factors in Computing Systems*, CHI '94, pages 107–108, New York, NY, USA, 1994. ACM.

[115] Thomas K Landauer, Peter W. Foltz, and Darrell Laham. An introduction to latent semantic analysis. *Discourse Processes*, 25(2-3):259–284, 1998.

[116] James A. Landay and Brad A. Myers. Interactive sketching for the early stages of user interface design. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '95, pages 43–50, New York, NY, USA, 1995. ACM Press/Addison-Wesley Publishing Co.

[117] Jill H. Larkin and Herbert A. Simon. Why a diagram is (sometimes) worth ten thousand words. *Cognitive Science*, 11(1):65–100, 1987.

[118] J.J. LaViola and R.C. Zeleznik. A practical approach for writer-dependent symbol recognition using a writer-independent symbol recognizer. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 29(11):1917–1926, Nov 2007.

[119] Joseph J. Laviola, Jr. *Mathematical sketching: a new approach to creating and exploring dynamic illustrations*. PhD thesis, Brown University, Providence, RI, USA, 2005. AAI3174634.

[120] Joseph J. LaViola, Jr. and Robert C. Zeleznik. Mathpad2: a system for the creation and exploration of mathematical sketches. *ACM Trans. Graph.*, 23:432–440, August 2004.

[121] C. Lee, J. Jordan, T. F. Stahovich, and J. Herold. Newtons pen ii: An intelligent, sketch-based tutoring system and its sketch processing techniques. In *Proceedings of the International Symposium on Sketch-Based Interfaces and Modeling*, SBIM '12, pages 57–65, Aire-la-Ville, Switzerland, Switzerland, 2012. Eurographics Association.

[122] Seong-Whan Lee and Dae-Seok Ryu. Parameter-free geometric document layout analysis. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 23(11):1240–1256, 2001.

[123] WeeSan Lee, Ruwanee de Silva, Eric J. Peterson, Robert C. Calfee, and Thomas F. Stahovich. Newton's pen: a pen-based tutoring system for statics. In *Proceedings of the 4th Eurographics workshop on Sketch-based interfaces and modeling*, SBIM '07, pages 59–66, New York, NY, USA, 2007. ACM.

[124] WeeSan Lee, Levent Burak Kara, and Thomas F. Stahovich. An efficient graph-based recognizer for hand-drawn symbols. *Computers & Graphics*, 31(4):554 – 567, 2007.

[125] Chuanjun Li, Timothy S Miller, Robert C Zeleznik, and Joseph J LaViola Jr. Algosketch: Algorithm sketching and interactive computation. In *SBM*, pages 175–182. Citeseer, 2008.

[126] Yang Li. Protractor: A fast and accurate gesture recognizer. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '10, pages 2169–2172, New York, NY, USA, 2010. ACM.

[127] Yang Li, James A. Landay, Zhiwei Guan, Xiangshi Ren, and Guozhong Dai. Sketching informal presentations. In *ICMI '03*, pages 234–241, 2003.

[128] Shizhong Liao and Menghua Duan. Sketch recognition via string kernel. In *Natural Computation (ICNC), 2012 Eighth International Conference on*, pages 101–105, May 2012.

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

[130] Graham Mackenzie and Natasha Alechina. Classifying sketches of animals using an agent-based system. In Nicolai Petkov and MichelA. Westenberg, editors, *Computer Analysis of Images and Patterns*, volume 2756 of *Lecture Notes in Computer Science*, pages 521–529. Springer Berlin Heidelberg, 2003.

[131] James V Mahoney and Markus PJ Fromherz. Three main concerns in sketch recognition and an approach to addressing them. In *AAAI Spring Symposium on Sketch Understanding*, pages 105–112, 2002.

[132] JamesV. Mahoney and MarkusP.J. Fromherz. Interpreting sloppy stick figures by graph rectification and constraint-based matching. In Dorothea Blostein and Young-Bin Kwon, editors, *Graphics Recognition Algorithms and Applications*, volume 2390 of *Lecture Notes in Computer Science*, pages 222–235. Springer Berlin Heidelberg, 2002.

[133] Nicolas Mangano and Noi Sukaviriya. Inkus: A freehand method of creating business process models. In *Proceedings of the Seventh Sketch-Based Interfaces and Modeling Symposium*, SBIM '10, pages 143–150, Aire-la-Ville, Switzerland, Switzerland, 2010. Eurographics Association.

[134] Song Mao, Azriel Rosenfeld, and Tapas Kanungo. Document structure analysis algorithms: a literature survey. In *Electronic Imaging 2003*, pages 197–207. International Society for Optics and Photonics, 2003.

[135] J. Mas, G. Sanchez, J. Llados, and B. Lamiroy. An incremental on-line parsing algorithm for recognizing sketching diagrams. In *Document Analysis and Recognition, 2007. ICDAR 2007. Ninth International Conference on*, volume 1, pages 452–456, Sept 2007.

[136] Noboru Matsuda and Kurt VanLehn. Gramy: A geometry theorem prover capable of construction. *Journal of Automated Reasoning*, 32(1):3–33, 2004.

[137] T. Matsushita, Cheng Cheng, Y. Murata, Bilan Zhu, and M. Nakagawa. Effect of text/non-text classification for ink search employing string recognition. In *Document Analysis Systems (DAS), 2012 10th IAPR International Workshop on*, pages 230–234, March 2012.

[138] Michael Mayo and Antonija Mitrovic. Optimising its behaviour with bayesian networks and decision theory. *International Journal of Artificial Intelligence in Education*, 12:124–153, 2001.

[139] George A. Miller. Wordnet: A lexical database for english. *Commun. ACM*, 38(11):39–41, November 1995.

[140] Ian Millington. *Game Physics Engine Development (The Morgan Kaufmann Series in Interactive 3D Technology)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2007.

[141] Antonija Mitrovic. An intelligent sql tutor on the web. *International Journal of Artificial Intelligence in Education*, 13(2):173–197, 2003.

[142] Antonija Mitrovic and Stellan Ohlsson. Evaluation of a constraint-based tutor for a database language. *International Journal on Artificial Intelligence in Education*, 10(3–4):238–56, 1999.

[143] Keisuke Mochida and Masaki Nakagawa. Separating figures, mathematical formulas and japanese text from free handwriting in mixed online documents. *International Journal of Pattern Recognition and Artificial Intelligence*, 18(07):1173–1187, 2004.

[144] Yuki Mori and Takeo Igarashi. Plushie: an interactive design system for plush toys. In *SIGGRAPH '07*, 2007.

[145] G. Nagy, S. Seth, and M. Viswanathan. A prototype document image analysis system for technical journals. *Computer*, 25(7):10–22, July 1992.

[146] Anoop M Namboodiri and Anil K Jain. Robust segmentation of unconstrained online handwritten documents. In *ICVGIP*, pages 165–170, 2004.

[147] Gabriele Nataneli and Petros Faloutsos. Robust classification of strokes with svm and grouping. In George Bebis, Richard Boyle, Bahram Parvin, Darko Koracin, Nikos Paragios, Syeda-Mahmood Tanveer, Tao Ju, Zicheng Liu, Sabine Coquillart, Carolina Cruz-Neira, Torsten Mller, and Tom Malzbender, editors, *Advances in Visual Computing*, volume 4841 of *Lecture Notes in Computer Science*, pages 76–87. Springer Berlin Heidelberg, 2007.

[148] Ulric Neisser and Paul Weene. A note on human recognition of hand-printed characters. *Information and Control*, 3(2):191 – 196, 1960.

[149] Mark W. Newman, James Lin, Jason I. Hong, and James A. Landay. Denim: an informal web site design tool inspired by observations of practice. *Hum.-Comput. Interact.*, 18:259–324, September 2003.

[150] Newton's playground, 2014. `http://www.educade.org/teaching_tools/newtons-playground`.

[151] L. O'Gorman. The document spectrum for page layout analysis. *IEEE Trans. Pattern Anal. Mach. Intell.*, 15(11):1162–1173, November 1993.

[152] Makoto Okabe, Shigeru Owada, and Takeo Igarash. Interactive design of botanical trees using freehand sketches and example-based editing. *Computer Graphics Forum*, 24(3):487–496, 2005.

[153] Michael Oltmans and Randall Davis. Naturally conveyed explanations of device behavior. In *Proceedings of the 2001 workshop on Perceptive user interfaces*, PUI '01, pages 1–8, New York, NY, USA, 2001. ACM.

[154] Tom Y. Ouyang and Randall Davis. Recognition of hand drawn chemical diagrams. In *Proceedings of the 22Nd National Conference on Artificial Intelligence - Volume 1*, AAAI'07, pages 846–851. AAAI Press, 2007.

[155] Tom Y Ouyang and Randall Davis. A visual approach to sketched symbol recognition. In *IJCAI*, volume 9, pages 1463–1468, 2009.

[156] Tom Y. Ouyang and Randall Davis. Chemink: a natural real-time recognition system for chemical drawings. In *Proceedings of the 16th international conference on Intelligent user interfaces*, IUI '11, pages 267–276, New York, NY, USA, 2011. ACM.

[157] Sharon Oviatt. Multimodal interfaces. *The human-computer interaction handbook: Fundamentals, evolving technologies and emerging applications*, pages 286–304, 2003.

[158] Sharon Oviatt and Philip Cohen. Perceptual user interfaces: Multimodal interfaces that process what comes naturally. *Commun. ACM*, 43(3):45–53, March 2000.

[159] Roy P. Pargas and Samuel Bryfczynski. Using ink to expose students' thought processes in cs2/cs7. *SIGCSE Bull.*, 41(1):168–172, March 2009.

[160] Rachel Patel, Beryl Plimmer, John Grundy, and Ross Ihaka. Ink features for diagram recognition. In *Proceedings of the 4th Eurographics workshop on Sketch-based interfaces and modeling*, SBIM '07, pages 131–138, New York, NY, USA, 2007. ACM.

[161] Brandon Paulson and Tracy Hammond. Paleosketch: accurate primitive sketch recognition and beautification. In *Proceedings of the 13th international conference on Intelligent user interfaces*, IUI '08, pages 1–10, New York, NY, USA, 2008. ACM.

[162] Brandon Chase Paulson. *Rethinking Pen Input Interaction: Enabling Freehand Sketching Through Improved Primitive Recognition*. PhD thesis, Texas A & M University, College Station, TX, USA, 2010. AAI3416265.

[163] João P. Pereira, Joaquim A. Jorge, Vasco A. Branco, Nelson F. Silva, Tiago D. Cardoso, and F. Nunes Ferreira. Cascading recognizers for ambiguous calligraphic interaction. In *Proceedings of the First Eurographics Conference on Sketch-Based Interfaces and Modeling*, SBM'04, pages 63–72, Aire-la-Ville, Switzerland, Switzerland, 2004. Eurographics Association.

[164] Adeline Pihuit, Marie-Paule Cani, and Olivier Palombi. Sketch-based modeling of vascular systems: A first step towards interactive teaching of anatomy. In *Proceedings of the Seventh Sketch-Based Interfaces and Modeling Symposium*, SBIM '10, pages 151–158, Aire-la-Ville, Switzerland, Switzerland, 2010. Eurographics Association.

[165] Y. Qi, M. Szummer, and T.P. Minka. Diagram structure recognition by bayesian conditional random fields. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 2, pages 191–196 vol. 2, June 2005.

[166] Lin Qiu. Sketchuml: The design of a sketch-based tool for uml class diagrams. In *World Conference on Educational Multimedia, Hypermedia and Telecommunications*, pages 986–994, 2007.

[167] Janet C. Read, Stuart MacFarlane, and Chris Casey. 'good enough for what?': Acceptance of handwriting recognition errors by child users. In *Proceedings of the 2003 Conference on Interaction Design and Children*, IDC '03, pages 155–155, New York, NY, USA, 2003. ACM.

[168] Jeff Rickel and W. Lewis Johnson. Animated agents for procedural training in virtual reality: Perception, cognition, and motor control. *Applied Artificial Intelligence*, 13(4-5):343–382, 1999.

[169] Dean Rubine. Specifying gestures by example. *SIGGRAPH Comput. Graph.*, 25(4):329–337, July 1991.

[170] William Rucklidge. *Efficient Visual Recognition Using the Hausdorff Distance*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1996.

[171] Eric Saund, David Fleet, Daniel Larner, and James Mahoney. Perceptually-supported image editing of text and graphics. In *Proceedings of the 16th Annual ACM Symposium on User Interface Software and Technology*, UIST '03, pages 183–192, New York, NY, USA, 2003. ACM.

[172] Eric Saund and Thomas P. Moran. A perceptually-supported sketch editor. In *Proceedings of the 7th Annual ACM Symposium on User Interface Software and Technology*, UIST '94, pages 175–184, New York, NY, USA, 1994. ACM.

[173] Richard Scheines and Wilfried Sieg. Computer environments for proof construction. *Interactive Learning Environments*, 4(2):159–169, 1994.

[174] Jeremy Scott and Randall Davis. Physink: Sketching physical behavior. In *Proceedings of the Adjunct Publication of the 26th Annual ACM Symposium on User Interface Software and Technology*, UIST '13 Adjunct, pages 9–10, New York, NY, USA, 2013. ACM.

[175] Tevfik Metin Sezgin and Randall Davis. Hmm-based efficient sketch recognition. In *Proceedings of the 10th International Conference on Intelligent User Interfaces*, IUI '05, pages 281–283, New York, NY, USA, 2005. ACM.

[176] Tevfik Metin Sezgin, Thomas Stahovich, and Randall Davis. Sketch based interfaces: Early processing for sketch understanding. In *ACM SIGGRAPH 2006 Courses*, SIGGRAPH '06, New York, NY, USA, 2006. ACM.

[177] T.M. Sezgin and R. Davis. Sketch interpretation using multiscale models of temporal patterns. *Computer Graphics and Applications, IEEE*, 27(1):28–37, Jan 2007.

[178] Faisal Shafait, Daniel Keysers, and Thomas M Breuel. Performance evaluation and benchmarking of six-page segmentation algorithms. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 30(6):941–954, 2008.

[179] D. Sharon and M. Van De Panne. Constellation models for sketch recognition. In *Proceedings of the Third Eurographics Conference on Sketch-Based Interfaces and Modeling*, SBM'06, pages 19–26, Aire-la-Ville, Switzerland, Switzerland, 2006. Eurographics Association.

[180] Sharpnlp project, 2014. `http://sharpnlp.codeplex.com/`.

[181] Roger N Shepard. Externalization of mental images and the act of creation. *Visual learning, thinking, and communication*, pages 133–189, 1978.

[182] Roger N Shepard. The mental image. *American psychologist*, 33(2):125–137, 1978.

[183] M. Shilman, P. Viola, and K. Chellapilla. Recognition and grouping of handwritten text in diagrams and equations. In *Frontiers in Handwriting Recognition, 2004. IWFHR-9 2004. Ninth International Workshop on*, pages 569–574, Oct 2004.

[184] M. Shilman, Zile Wei, S. Raghupathy, P. Simard, and D. Jones. Discerning structure from freeform handwritten notes. In *Document Analysis and Recognition, 2003. Proceedings. Seventh International Conference on*, pages 60–65 vol.1, Aug 2003.

[185] Michael Shilman, Hanna Pasula, Stuart Russell, and Richard Newton. Statistical visual language models for ink parsing. In *AAAI Spring Symposium on Sketch Understanding*, pages 126–132, 2002.

[186] Saul Simhon and Gregory Dudek. Sketch interpretation and refinement using statistical models. In *Proceedings of the Fifteenth Eurographics Conference on Rendering Techniques*, EGSR'04, pages 23–32, Aire-la-Ville, Switzerland, Switzerland, 2004. Eurographics Association.

[187] A Simon, J.-C. Pret, and AP. Johnson. A fast algorithm for bottom-up document layout analysis. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 19(3):273–277, Mar 1997.

[188] Mark K. Singley. The reification of goal structures in a calculus tutor: Effects on problem-solving performance. *Interactive Learning Environments*, 1(2):102–123, 1990.

[189] Steve Smithies, Kevin Novins, and James Arvo. Equation entry and editing via handwriting and gesture recognition. *Behaviour & Information Technology*, 20(1):53–67, 2001.

[190] Thomas F. Stahovich, Eric J. Peterson, and Hanlung Lin. An efficient, classification-based approach for grouping pen strokes into objects. *Computers & Graphics*, 42(0):14 – 30, 2014.

[191] Starpad, 2014. `http://pen.cs.brown.edu/starpad.html`.

[192] Anselm Strauss and Juliet M Corbin. *Basics of qualitative research: Grounded theory procedures and techniques.* Sage Publications, Inc, 1990.

[193] Ivan E. Sutherland. Sketch pad a man-machine graphical communication system. In *Proceedings of the SHARE Design Automation Workshop*, DAC '64, pages 6.329–6.346, New York, NY, USA, 1964. ACM.

[194] John Sweller. Cognitive load during problem solving: Effects on learning. *Cognitive Science*, 12(2):257–285, 1988.

[195] John Sweller. Cognitive load theory, learning difficulty, and instructional design. *Learning and Instruction*, 4(4):295 – 312, 1994.

[196] Paul Taele and Tracy Hammond. Lamps: A sketch recognition-based teaching tool for mandarin phonetic symbols i. *Journal of Visual Languages & Computing*, 21(2):109 – 120, 2010. Special Issue on Sketch Computation Special Issue on Sketch Computation.

[197] Robert Tarjan. Depth-first search and linear graph algorithms. *SIAM journal on computing*, 1(2):146–160, 1972.

[198] Dana Tenneson and Sascha Becker. Chempad: Generating 3d molecules from 2d sketches. In *ACM SIGGRAPH 2005 Posters*, SIGGRAPH '05, New York, NY, USA, 2005. ACM.

[199] Matthew Thorne, David Burke, and Michiel van de Panne. Motion doodles: an interface for sketching character motion. In *SIGGRAPH 2006*, 2006.

[200] J.D. Tubbs. A note on binary template matching. *Pattern Recognition*, 22(4):359 – 365, 1989.

[201] R. Sinan Tumen, M. Emre Acer, and T. Metin Sezgin. Feature extraction and classifier combination for image-based sketch recognition. In *Proceedings of the Seventh Sketch-Based Interfaces and Modeling Symposium*, SBIM '10, pages 63–70, Aire-la-Ville, Switzerland, Switzerland, 2010. Eurographics Association.

[202] Matthew Turk. Multimodal interaction: A review. *Pattern Recognition Letters*, 36:189–195, 2014.

[203] Matthew Turk and George Robertson. Perceptual user interfaces (introduction). *Commun. ACM*, 43(3):32–34, March 2000.

[204] Barbara Tversky. What does drawing reveal about thinking? In *IN*, pages 93–101, 1999.

[205] David G. Ullman, Stephen Wood, and David Craig. The importance of drawing in the mechanical design process. *Computers & Graphics*, 14(2):263 – 274, 1990.

[206] Kurt Vanlehn. The behavior of tutoring systems. *International journal of artificial intelligence in education*, 16(3):227–265, 2006.

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

[208] Jonathan Wai, David Lubinski, and Camilla P Benbow. Spatial ability for stem domains: Aligning over 50 years of cumulative psychological knowledge solidifies its importance. *Journal of Educational Psychology*, 101(4):817, 2009.

[209] Paul Wais, Aaron Wolin, and Christine Alvarado. Designing a sketch recognition front-end: user perception of interface elements. In *Proceedings of the 4th Eurographics workshop on Sketch-based interfaces and modeling*, SBIM '07, pages 99–106, New York, NY, USA, 2007. ACM.

[210] Benjamin Walther-Franks, Marc Herrlich, Thorsten Karrer, Moritz Wittenhagen, Roland Schröder-Kroll, Rainer Malaka, and Jan Borchers. Dragimation: Direct manipulation keyframe timing for performance-based animation. In *Proceedings of Graphics Interface 2012*, GI '12, pages 101–108, Toronto, Ont., Canada, Canada, 2012. Canadian Information Processing Society.

[211] Yalin Wang, Ihsin T Phillips, and Robert M Haralick. Document zone content classification and its performance evaluation. *Pattern Recognition*, 39(1):57–73, 2006.

[212] Rattapoom Waranusast, Peter Haddawy, and Matthew Dailey. Segmentation of text and non-text in on-line handwritten patient record based on spatio-temporal analysis. In Carlo Combi, Yuval Shahar, and Ameen Abu-Hanna, editors, *Artificial Intelligence in Medicine*, volume 5651 of *Lecture Notes in Computer Science*, pages 345–354. Springer Berlin Heidelberg, 2009.

[213] Andrew Witkin. An introduction to physically based modeling: Constrained dynamics. *Robotics Institute*, 1997.

[214] Jacob O. Wobbrock, Andrew D. Wilson, and Yang Li. Gestures without libraries, toolkits or training: A $1 recognizer for user interface prototypes. In *Proceedings of the 20th Annual ACM Symposium on User Interface Software and Technology*, UIST '07, pages 159–168, New York, NY, USA, 2007. ACM.

[215] A. Wolin, B. Eoff, and T. Hammond. Shortstraw: A simple and effective corner finder for polylines. In *Proceedings of the Fifth Eurographics Conference on Sketch-Based Interfaces and Modeling*, SBM'08, pages 33–40, Aire-la-Ville, Switzerland, Switzerland, 2008. Eurographics Association.

[216] A. Wolin, B. Paulson, and T. Hammond. Sort, merge, repeat: An algorithm for effectively finding corners in hand-sketched strokes. In *Proceedings of the 6th Eurographics Symposium on Sketch-Based Interfaces and Modeling*, SBIM '09, pages 93–99, New York, NY, USA, 2009. ACM.

[217] Beverly Park Woolf. *Building Intelligent Interactive Tutors*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2009.

[218] Working model 2d, 2011. `http://www.design-simulation.com/wm2d/index.php`.

[219] Yiyan Xiong and Joseph J. LaViola Jr. Technical section: A shortstraw-based algorithm for corner finding in sketch-based interfaces. *Computers and Graphics*, 34:513–527, October 2010.

[220] L. Xu, A Krzyzak, and C.Y. Suen. Methods of combining multiple classifiers and their applications to handwriting recognition. *Systems, Man and Cybernetics, IEEE Transactions on*, 22(3):418–435, May 1992.

[221] Chen Yang, Dana Sharon, and Michiel van de Panne. Sketch-based modeling of parameterized objects. In *SIGGRAPH 2005*, 2005.

[222] Kun Yang, Zhijun Li, and Jingwei Ye. Freely-drawn sketches interpretation using svms-chain modeling. *Engineering Applications of Artificial Intelligence*, 25(2):392 – 403, 2012. Special Section: Local Search Algorithms for Real-World Scheduling and Planning.

[223] Ming Ye and P. Viola. Learning to parse hierarchical lists and outlines using conditional random fields. In *Frontiers in Handwriting Recognition, 2004. IWFHR-9 2004. Ninth International Workshop on*, pages 154–159, Oct 2004.

[224] Ming Ye, Paul Viola, Sashi Raghupathy, Herry Sutanto, and Chengyang Li. Learning to group text lines and regions in freeform handwritten notes. In *Document Analysis and Recognition, 2007. ICDAR 2007. Ninth International Conference on*, volume 1, pages 28–32. IEEE, 2007.

[225] Jianfeng Yin and Zhengxing Sun. An online multi-stroke sketch recognition method integrated with stroke segmentation. In Jianhua Tao, Tieniu Tan, and RosalindW. Picard, editors, *Affective Computing and Intelligent Interaction*, volume 3784 of *Lecture Notes in Computer Science*, pages 803–810. Springer Berlin Heidelberg, 2005.

[226] Hugh D. Young, Roger A. Freedman, A.L. Ford, F.W. Sears, and M.W. Zemansky. *University Physics with Modern Physics*. Pearson Education, Limited, 2011.

[227] Bo Yu and Shijie Cai. A domain-independent system for sketch recognition. In *Proceedings of the 1st International Conference on Computer Graphics and Interactive Techniques in Australasia and South East Asia*, GRAPHITE '03, pages 141–146, New York, NY, USA, 2003. ACM.

[228] Zhenming Yuan, Hong Pan, and Liang Zhang. A novel pen-based flowchart recognition system for programming teaching. In *Advances in Blended Learning*, pages 55–64. Springer, 2008.

[229] Shane W Zamora and Eyrún A Eyjólfsdóttir. Circuitboard: Sketch-based circuit design and analysis. In *Proceedings of the International Conference on Intelligent User Interfaces (IUI) Workshop on Sketch Recognition*, 2009.

[230] Robert Zeleznik, Timothy Miller, Chuanjun Li, and Joseph J. Laviola, Jr. Mathpaper: Mathematical sketching with fluid support for interactive computation. In *Proceedings of the 9th international symposium on Smart Graphics*, SG '08, pages 20–32, Berlin, Heidelberg, 2008. Springer-Verlag.

[231] Robert C. Zeleznik, Andrew Bragdon, Chu-Chi Liu, and Andrew Forsberg. Lineogrammer: creating diagrams by drawing. In *UIST '08*, pages 161–170, 2008.

[232] Robert C. Zeleznik, Kenneth P. Herndon, and John F. Hughes. Sketch: An interface for sketching 3d scenes. In *SIGGRAPH '96*, pages 163–170, 1996.

[233] Xiang-Dong Zhou and Cheng-Lin Liu. Text/non-text ink stroke classification in japanese handwriting based on markov random fields. In *Document Analysis and Recognition, 2007. ICDAR 2007. Ninth International Conference on*, volume 1, pages 377–381, Sept 2007.

[234] Xiang-Dong Zhou, Da-Han Wang, and Cheng-Lin Liu. A robust approach to text line grouping in online handwritten japanese documents. *Pattern Recognition*, 42(9):2077 – 2088, 2009.