Hindawi Publishing Corporation
Advances in Human-Computer Interaction
Volume 2014, Article ID 908690, 13 pages
http://dx.doi.org/10.1155/2014/908690

Research Article

PaperCAD: A System for Interrogating CAD Drawings Using Small Mobile Computing Devices Combined with Interactive Paper

WeeSan Lee[1] and Thomas F. Stahovich[2]

[1] *Department of Computer Science & Engineering, University of California, Riverside, CA 92521, USA*
[2] *Department of Mechanical Engineering, University of California, Riverside, CA 92521, USA*

Correspondence should be addressed to WeeSan Lee; weesan@cs.ucr.edu

Received 31 July 2014; Revised 19 October 2014; Accepted 20 October 2014; Published 13 November 2014

Academic Editor: Kerstin S. Eklundh

Smartphones have become indispensable computational tools. However, some tasks can be difficult to perform on a smartphone because these devices have small displays. Here, we explore methods for augmenting the display of a smartphone, or other PDA, using interactive paper. Specifically, we present a prototype interface that enables a user to interactively interrogate technical drawings using an Anoto-based smartpen and a PDA. Our software system, called PaperCAD, enables users to query geometric information from CAD drawings printed on Anoto dot-patterned paper. For example, the user can measure a distance by drawing a dimension arrow. The system provides output to the user via a smartpen's audio speaker and the dynamic video display of a PDA. The user can select either verbose or concise audio feedback, and the PDA displays a video image of the portion of the drawing near the pen tip. The project entails advances in the interpretation of pen input, such as a method that uses contextual information to interpret ambiguous dimensions and a technique that uses a hidden Markov model to correct interpretation errors in handwritten equations. Results of a user study suggest that our user interface design and interpretation techniques are effective and that users are highly satisfied with the system.

1. Introduction

Smartphones, and the wide variety of software applications for them, have become indispensable computational tools. However, some tasks can be difficult to perform on a smartphone because these devices have small displays. Here, we explore methods for augmenting a smartphone display using interactive paper. Specifically, we present a prototype interface that enables a user to interactively interrogate technical drawings using an Anoto-based smartpen [1] and a PDA. Smartpens serve the same function as a traditional pen and also record the writing as time-stamped pen strokes. Smartpens are used with paper preprinted with a special dot pattern. A camera integrated into smartpen uses the dots to locate the pen tip on the page and digitize each pen stroke. Some smartpens process the digitized writing using application software running on a processor embedded in the device; other versions wirelessly transmit the digitize writing so that it can be processed by software running on a computer, smartphone, or another mobile computing device.

Our prototype software system is called PaperCAD. Figure 1 shows an example of a simple drawing formatted for use with the system. The drawing is printed on digital paper which also contains several printed buttons, such as a help button and an abort button, which are used to execute various software functions. The user can measure dimensions by drawing conventional dimension lines. In Figure 1, for example, the user has dimensioned the base and height of the triangle, the angle of one of the corners, and the radius of the circle. As each dimension is drawn, the system announces its value with synthesized speech. The values are obtained by querying a digital model of the drawing and are not scaled from the paper drawing. A symbolic label is associated with each dimension so that its value can be used in equations.

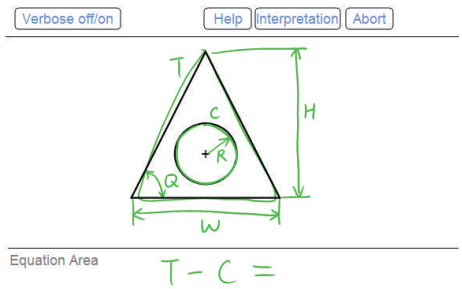


Figure 1: A drawing formatted for PaperCAD. The scale has been changed for clarity.

The user can also measure the area of a bounded region by simply tracing the boundary. In the current example, the triangle has been traced and its area has been associated with the label T. Likewise, the area of the circular hole has been associated with the label C. At the bottom of the page, the user can write algebraic equations. In this example, the user has written the equation "$T - C =$", which computes the area between the triangle and circle. The system interprets such equations and announces the result.

In this work, we explore methods for effectively combining the interaction modalities of smartpens and small mobile computing devices such as smartphones and tablet computers. For convenience we use the term personal digital assistant, or PDA, to refer to such mobile computing devices. Smartpens (e.g., the Livescribe Echo) are typically capable of providing audio output. Some smartpens can display small amounts of text on a character display embedded in the barrel of the pen. PaperCAD provides audio output and offers both verbose and concise feedback modes. In the verbose mode, the system uses synthesized speech to indicate the identity of recognized objects and to report the values of dimensions. In concise mode, the system uses tones to indicate whether or not objects have been recognized; synthesized speech is still used to report the values of dimensions. In addition to the audio output, PaperCAD provides graphical feedback using the dynamic video display of a PDA. After the user draws on the paper drawing, the beautified ink and CAD drawing in the neighborhood of the pen stroke are displayed on the PDA. To view a different part of the drawing, the user taps the smartpen at the desired location on the paper.

Beyond the user interface challenges, creating PaperCAD required us to solve several problems related to interpreting hand-drawn input. For example, we developed a recognizer for dimension arrows capable of recognizing arrows with a wide range of aspect ratios. We also developed a method that uses contextual information to interpret ambiguous dimensions. Similarly, we developed a technique that uses a hidden Markov model to automatically correct interpretation errors in handwritten equations.

The next section places this work in the context of related work. This is followed by an overview of the PaperCAD system and the details of its implementation. Results of a user study evaluating PaperCAD's usability are also presented.

2. Related Work

There has been extensive research on sketch-based user interfaces for tablet PCs and other similar pen-based devices with dynamic video displays. Examples include tools for simulating hand-drawn mechanical devices [2], a tool for sketching user interfaces [3], a tool for understanding military tactics [4], a circuit analysis tutor [5], a control system analysis tool [6], and a UML diagram tool [7]. Likewise, there has been significant progress in sketching 3D shapes [8, 9]. Our work differs in that it also addresses the challenges of user interface design for a visually static medium, namely, ink on paper.

Several recent research efforts have aimed at using smartpen technology to build pen-based applications. The PapierCraft [10] system enables users to edit documents by writing on paper printouts with an Anoto digital pen. Annotations and command gestures are captured by the digital pen and are then uploaded to a PC to be interpreted and executed on a digital version of the document. The NISMap system [11] is a military planning tool, also based on the Anoto digital pen. As the user annotates a paper map, digitized pen strokes are wirelessly transmitted to a PC for processing. ModelCraft [12] is used to apply annotations and edits to a 3D geometric model constructed from dot-patterned paper. Again, ink is captured with Anoto pens and is processed offline when the pen is docked. PaperCAD, by contrast, is an interactive paper-based interface.

Recent work has explored methods of integrating paper-based interfaces with digital devices. For example, ButterflyNet [13] enables field biologists to link digital photos to paper notebooks by drawing a gesture. The system relies on an Anoto pen wirelessly connected to a digital camera. Similarly, PocketPad [14] enables users to annotate digital documents displayed on a PDA by writing with an Anoto pen and paper. These systems do little recognition of the pen input, and the final document is produced only when the digital device and pen are docked with a computer.

Ariel [15] was an early attempt to introduce computational capabilities into paper drawings. It employed a WIMP interface projected onto the drawing and used a light pen only as a pointer. A-book [16] introduced the concept of an "interaction lens," a PDA placed over a paper document to provide a digital image of the contents. A GUI on the PDA is used to manipulate the digital model of the document. PaperCAD, by contrast, works directly from the paper.

Work in [17] has begun exploring methods of providing real-time user feedback directly from a smartpen using LEDs, voice coils, and audio speakers. Livescribe smartpens include small OLED displays.

The Newton's Pen [18] statics tutoring system also runs on a smartpen. Our work involves numerous advances beyond this, such as a robust arrow recognizer, an HMM technique for correcting interpretation errors in equations, and techniques for using context to disambiguate graphical objects. PaperCAD also introduces an interface for correcting recognition errors and provides new feedback modes, including concise audio and an LCD.

Researchers have developed numerous techniques for recognizing hand-drawn shapes and symbols. Rubine's method [19] is an example of early work in this area. This method employs an inductive learning method in which a model for each class of shapes is derived from a set of training examples. Each example is represented by a set of geometric and temporal features that are used to train a linear discriminator for classifying unknown shapes.

Many other recognizers also rely on inductive classifiers. For example, Lee et al.'s [18] graph-based recognizer represents a class of shapes in terms of the statistical distributions of a set of topological and geometric features. Gennari et al. [20] developed a technique that uses geometric and topological features to train a Naive Bayes classifier, while Fonseca et al.'s [21] method considers properties of the convex hull of a shape.

Hse and Newton [22] developed a particularly accurate recognizer based on Zernike moments, which provide a rotation invariant representation. Hammond and Davis [23] developed a recognizer that relies on hand-coded shape descriptions. More recently [24], they extended their approach to use hand-drawn and machine-generated examples to assist the developer in interactively creating shape descriptions. Shilman et al. [25] present a sketch recognition approach that requires a manually encoded visual grammar. A large corpus of training examples is used to learn the statistical distributions of the geometric parameters used in the grammar, resulting in a statistical model. Composite objects are defined hierarchically in terms of lower-level, single-stroke symbols, which are recognized using Rubine's method [19].

Many recognition approaches rely on template matching. Gross's [26] approach relies on a 3×3 grid inscribed in the symbol's bounding box. The sequence of grid cells visited by the pen distinguishes each symbol. Kara and Stahovich's image-based recognizer [27] represents symbols with 48×48 bitmap image templates. To recognize an unknown symbol, it is converted into a template and compared to all of the templates in the training set. The training template with the best match to the unknown symbol is used to classify it. Ouyang and Davis [28] developed a more efficient template-based method that encodes information such as shape and pen trajectory. To reduce recognition time, agglomerative hierarchical clustering is used. The hierarchy is used with branch and bound search to reduce recognition time.

The Dollar Recognizer [29] is a popular method for recognizing single-stroke gestures. It represents a shape with a template of equally-spaced points sampled from the original pen stroke. An unknown shape is compared to a definition template by computing the sum of the distances between corresponding points.

Kara and Stahovich [6] have developed a special-purpose recognizer for arrows. The corners defining an arrowhead are detected as points of minimum pen speed. These points are used to define a set of line segments. The angles between these segments are compared to a set of empirical thresholds to identify arrowheads. This technique was designed to identify arrows with arbitrary shafts. In their user studies, this recognizer achieved between 65% and 70% accuracy.

de Silva et al. [5] used inverse curvature to distinguish arrows from alphabetic characters. The pen stroke to be classified is sampled to 36 points, and the inverse curvature is computed for each point. These 36 values are passed to a neural network for classification. Because of the fixed number of sample points, the approach is designed for arrows with a consistent aspect ratio. In our experiments, we found this technique to be unsuitable for the wide range of aspect ratios encountered in our domain.

Our arrow recognizer builds upon the technique in [6] in that we also identify arrows by identifying their heads. However, we use inverse curvature to identify the location of a candidate arrowhead, rather than pen speed. Unlike the technique in [5], we use inverse curvature only to locate a candidate arrowhead. We then use a bitmap representation and a neural network, rather than angles, to classify candidate arrowheads. The techniques in [6] and [5] are limited to single-stroke arrows. Our technique can handle arrows drawn with multiple strokes. Additionally, while the method in [6] can handle arrows with arbitrary shafts, it is limited to arrows drawn from tail to head. Our technique can handle arrows which are drawn in either sense, head to tail or tail to head, and which may have a head at each end.

There has been extensive research in interpreting handwritten mathematical expressions [30]. For example, Matsakis [31] and LaViola Jr. [32] have both created interactive systems for interpreting such expressions. To achieve high accuracy, these systems rely on accurate character recognition. Matsakis's system uses Gaussian models constructed from a small set of examples symbols. LaViola's MathPad^2 system combines multiple recognizers to achieve high accuracy. In particular, the system combines Microsoft's handwriting recognizer [33] with a set of features computed from the pen strokes.

Despite advances in character recognizers, recognition errors cannot be completely eliminated. As a remedy, Matsakis's system provides a convenient interface for manually correcting errors. MathPad^2 uses a set of heuristics that can replace 5*in* with *sin*, for example. Kirchhoff's Pen [5] employs a similar approach based on domain knowledge for electrical circuit equations. Our HMM technique in PaperCAD provides a more principled approach to error correction in that it is based on a grammar and statistics about recognition accuracy. Our work is complementary to existing work on interpreting handwritten mathematical expressions [30–32] in that our automated error correction techniques could be integrated with other recognition approaches.

HMMs have been used for other sketch interpretation tasks. For example, Sezgin and Davis [34] use an HMM approach to recognize objects in hand-drawn sketches. A separate HMM is trained to recognize each kind of shape in the domain. A global optimization of the probability is used to determine the overall interpretation of a sketch. Kosmala et al. [35] describe a similar approach for recognizing mathematical expressions. Their approach, however, does not make use of a grammar to improve recognition accuracy.

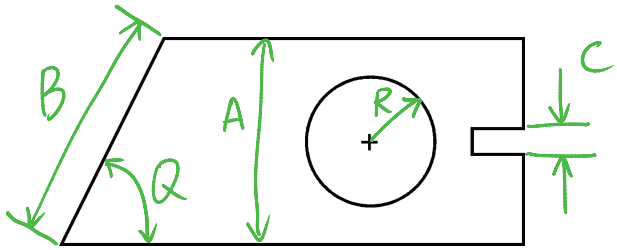


Figure 2: Typical dimensions.

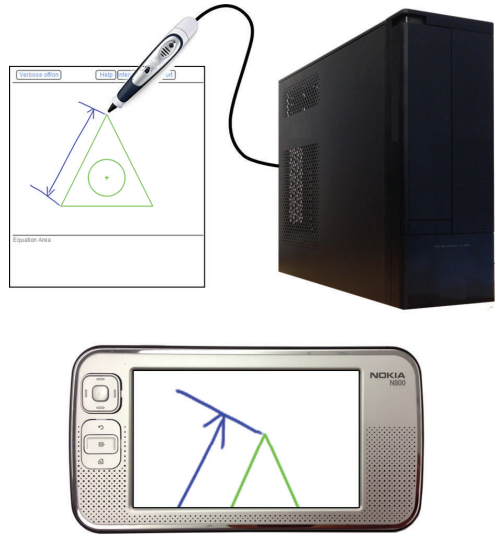


Figure 3: PaperCAD system including the FLY smartpen, paper drawing, Nokia N800, and PC. The latter is used to emulate smartpen functionality; users do not see the PC. (The devices are shown at different scales.)

3. System Overview

For a technical drawing to be complete, the size and location of every feature must be specified. However, even when a drawing is fully dimensioned, it is still frequently necessary to compute additional geometric information. For example, the diameter and location of a circular hole may be specified, but it may be necessary to determine the distance between the hole and the edge of the part.

PaperCAD enables a user to measure arbitrary dimensions from paper drawings by simply dimensioning them. Figure 2 shows typical examples of dimensions. Dimensions can span directly between edges of the geometric model (A) or can span between extension lines (B). (An extension line is a short line segment that is parallel or perpendicular to a coordinate axis or model edge and is drawn near a model vertex.) Linear dimensions can be aligned with a model edge (B) or a coordinate axis (A). A pair of arrows can be used to construct gap dimensions (C). Curved arrows are used to dimension angles (Q). Radii are dimensioned with single-headed arrows (R) and diameters with double-headed arrows. Areas are measured by tracing their boundary.

The user must write a symbolic label, specifically an uppercase letter, for each dimension. Labels serve as a handle for retrieving dimensions; tapping on a label causes the system to announce the value of the dimension. Labels can also be used in equations. PaperCAD has the ability to interpret and evaluate handwritten algebraic equations containing labels, numbers, mathematical operators, and the sine and cosine functions.

Drawings for use with PaperCAD must be printed on dot-patterned pages. Each page has an equation area at the bottom and a row of buttons at the top (Figure 1). The "Verbose Off/On" button allows the user to toggle between concise and verbose audio feedback. The system defaults to verbose mode, in which synthesized speech is used to provide interpretive feedback for all objects. In concise mode, the system plays "success" and "failure" tones to indicate whether or not objects have been recognized. Even in concise mode, however, synthesized speech is still used to report the values of dimensions. The "Help" button provides guidance for using the system and assistance with errors. The "Abort" button is used to cancel operations.

PaperCAD provides several methods for correcting interpretation errors. The "Interpretation" button allows the user to select an alternative interpretation for misrecognized graphical objects. For example, if an extension line is misrecognized as a dimension arrow, the user can tap the "Interpretation" button until the extension line interpretation is selected. Interpretation errors for labels and equations can be corrected by drawing a cross-out gesture ("\") through the symbol or symbols and rewriting them. Additional terms can be added to an equation by drawing a caret ("∧") and writing the new terms.

PaperCAD relies on a draw-and-pause paradigm in which the user pauses after drawing each object to receive interpretive audio feedback. For example, once the user has completed drawing a dimension arrow, the system announces the type of dimension, such as model-aligned, and its value. It is important that interpretive feedback be provided after each object is drawn, because otherwise it would be difficult to associate the feedback with individual objects.

In addition to audio feedback, PaperCAD also provides visual feedback via a PDA, although the latter is not required for use of the system. After the user draws on the paper, the beautified ink and CAD drawing in the neighborhood of the pen stroke are displayed on the PDA. To view a specific part of the drawing on the PDA, the user taps it with the smartpen.

To facilitate efficient development, we implemented our prototype software system using a combination of actual hardware and simulated hardware capabilities. The hardware used (Figure 3) includes a LeapFrog FLY smartpen, a Nokia N800 PDA, and a traditional PC. The smartpen streams ink data to the PC via a debugging cable. The data is then processed by our prototype application running on the host PC, *which the user does not see*. The application produces audio feedback via a text to speech converter; the audio is played through the PC's speakers. The N800 communicates with the host PC via a simple web server integrated into our prototype application. The application dynamically generates

a web page comprised of the text of the most recent audio message and an image of the appropriate region of the digital paper. The web page is regenerated each time a drawing event occurs or an audio message is played. The N800's web browser refreshes its view of the page at a frequency of one Hz.

This development environment enabled us to focus on issues related to user interface design and the recognition of hand-drawn input, rather than issues related to developing applications for small mobile computing devices. However, as work in [18] has shown, it would be possible to implement this software directly on a smartpen.

4. System Details

Each time the user pauses, PaperCAD attempts to recognize the ink, which can represent extension lines, dimension arrows, text labels, editing gestures, and equations. If ink drawn in the drawing area of the page represents a recognizable object, the system reports its identity. Alternatively, if the ink is unrecognizable, the system announces that the input is invalid. In the equation area, if the ink can be interpreted as an editing gesture, it is processed immediately. Otherwise, the system delays processing until an equal sign is detected, at which time the ink is interpreted as an equation. The following sections describe the various interpretation algorithms in detail.

4.1. Extension Line Recognizer. An extension line is a straight line segment that is parallel or perpendicular to a coordinate axis or model edge and that is drawn near a model vertex. Extension lines serve as datums for measuring dimensions. A linear least squares line fit is used to determine if a candidate pen stroke is a straight line. Specifically, if the average perpendicular distance from the stroke's data points to the least squares line is less than or equal to 1.5% of the stroke length, the stroke is considered to be a straight line. An extension line is considered to be near a model vertex if one of its endpoints is within a threshold distance to the vertex. We use a tolerance equal to 5% of the average of the height and width of the model's bounding box.

The intended orientation of an extension line is determined by considering the local context. If an extension line is within 14° (i.e., 4% of 360°) of being parallel or perpendicular to a model edge or coordinate direction, then the program hypothesizes that user may have intended it to be parallel or perpendicular, respectively. If an extension line is nearly parallel or perpendicular to multiple model edges and/or coordinate directions, multiple hypotheses are generated. For example, in Figure 4, three hypotheses are generated: the extension line could be perpendicular to edge AB, perpendicular to edge BC, or parallel to the y-axis. The program will maintain all three hypotheses until the extension line is used as a datum, at which time additional contextual information is used to determine the intended orientation. For example, if a dimension line is drawn between this extension line and another that is vertical, this extension line will be interpreted as a vertical one, and the dimension will be reported as a horizontal distance.

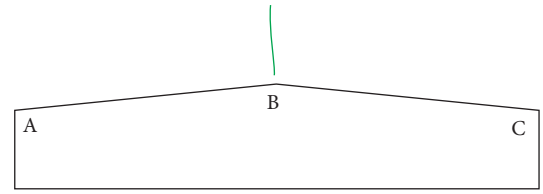


Figure 4: Ambiguous extension line.

4.2. Arrow Recognizer. Dimension arrows are challenging to recognize because the aspect ratio varies greatly. For short arrows, the arrowhead width may be comparable to the shaft length, whereas for long arrows the width may be comparatively insignificant. In fact, for long arrows, the arrowhead can be confused with the kind of hooking that often occurs at the ends of digitized pen strokes. To avoid these difficulties, we have developed an arrow recognizer that first decomposes a candidate arrow into a shaft region and head regions. In so doing, the program is able to examine the arrowhead at an appropriate scale regardless of the length of the shaft.

Arrows can be drawn with multiple pen strokes. When this occurs, the strokes are combined into a single equivalent pen stroke prior to recognition. To begin, the stroke that was drawn second is added to the first stroke by joining them at their closest ends with a straight line segment, thus producing a combined stroke. Subsequent strokes are added to the evolving combined stroke until a single combined stroke is produced. We have found that people often draw the shaft of an arrow first. Thus, this procedure effectively attaches the arrowheads to the shaft. Interestingly, however, the approach has proven to work even when the shaft is not drawn first.

Once all of the pen strokes have been combined, the equivalent single pen stroke is resampled to produce 100 evenly spaced points. A line segment is then constructed between each pair of consecutive points. Next, the cosine of the angle between adjacent segments is computed. The cosine is inversely related to the curvature. For example, if two consecutive segments are nearly collinear, the cosine is close to 1.0. If there is a large discontinuity, such as a 90° bend, the cosine is close to 0.0. For this reason, the cosine of the angle between adjacent segments is called "inverse curvature."

The shaft of an arrow is identified as the largest set of consecutive data points for which the inverse curvature is greater than zero. For example, Figure 5 shows a double-headed arrow and its inverse curvature representation. In this case, the shaft extends from data point 28 to 81. Data points that precede and follow the shaft are considered to be potential arrowheads. To facilitate recognition of the arrowheads, it is desirable for the candidate arrowheads to contain a small piece of the shaft. Thus, candidate arrowheads are augmented with eight data points from the appropriate end of the shaft. Augmenting the arrowheads in this fashion also ensures that each candidate arrow has two candidate arrowheads.

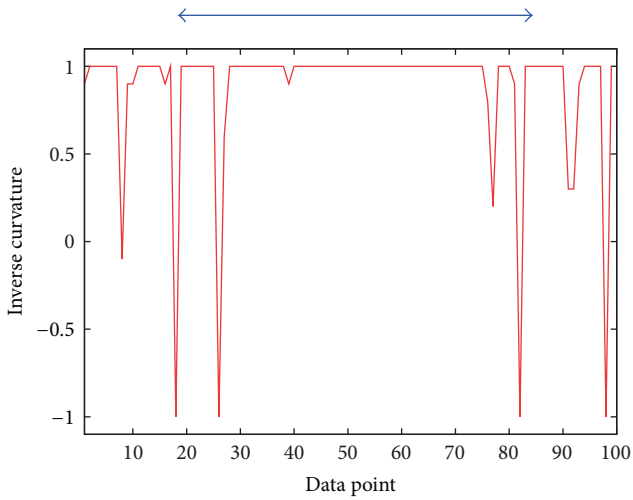


FIGURE 5: The inverse curvature of a double-headed arrow.

Once the two candidate arrowheads have been located, we use a bitmap representation and a neural network classifier to determine if the candidate arrowheads actually are arrowheads. To begin, a candidate arrowhead is uniformly scaled and sampled to produce a 17 × 17 binary bitmap. The bitmap is then transformed into a sort of distance map. Black pixels in the binary bitmap are represented by a +1 in the distance map. Nonblack pixels that neighbor black pixels are represented by a 0 in the map. All other pixels are represented by a −1. The map is essentially a gray-scale image in which three shades of gray (black, gray, and white) are used to blur the original image. We have found that using this form of gray-scale image results in higher recognition accuracy than the binary bitmaps.

The gray-scale bitmaps are classified with a neural network consisting of an input layer with 289 units, a hidden layer with 20 units, and an output layer with 12 units. Each input unit corresponds to a pixel in the 17 × 17 gray-scale image. Each output unit corresponds to a particular arrowhead orientation or to a particular type of nonarrowhead. The network classifies arrowheads as having an orientation of 0°, 45°, 90°, 135°, 180°, 225°, 270°, or 315°. The nonarrowhead classifications include horizontal line, vertical line, diagonal line with positive slope, and diagonal line with negative slope.

If both candidate arrowheads from a candidate arrow are classified as arrowheads, the candidate arrow is a double-headed arrow. Likewise, if only one candidate arrowhead is classified as such, the candidate arrow is a single-headed arrow. Otherwise, the candidate arrow is not an arrow. Double-headed arrows are further classified as curved arrows for measuring angles and straight arrows for measuring linear dimensions. The distinction is based on the straightness of the shaft. If the shaft is a straight line according to the definition of straight used for extension lines (see Section 4.1), the arrow is assumed to be straight, and otherwise it is curved.

To train the neural network, we collected sample arrows from seven engineering students. Each subject was asked to provide 211 arrows that varied in length and orientation. To ensure systematic data collection, subjects were provided with "targets" for the arrows. For example, subjects were provided with boxes with various lengths and orientations and were asked to draw arrows that spanned them. In this fashion, subjects provided double-headed arrows with lengths of 2, 4, 8, and 16 cm and orientations that varied in 45° increments. Using a similar approach, subjects provided examples of single-headed arrows, radial dimensions (single-headed arrow), diametral dimensions (double-headed arrows), and angular dimensions (curved double-headed arrows). Finally, to obtain additional variety in the data, each test subject was asked to draw 10 arbitrary linear dimensions, each comprised of a pair of extension lines and a double-headed arrow.

As the neural network classifies only the arrowhead portion of an arrow, it was necessary to extract the arrowheads from the sample arrows prior to training the network. This was done using the inverse curvature approach described above. As part of the training process, we used a cross validation approach to determine the optimal number of nodes in the hidden layer of the network. The network was trained on data from six subjects and was tested on data from the seventh. (The curved arrows were not used for training but were used for testing.) We determined that 20 is the optimal number of nodes. For this network topology, the average classification accuracy for the seven iterations of cross validation was 96.8%. Here, accuracy is defined in terms of the number of correctly classified arrows. For an arrow to be correctly classified, both of its candidate arrowheads must be correctly classified.

4.3. Interpreting Dimensions. Once a dimension arrow has been recognized, PaperCAD must determine which dimension it represents. The program searches for suitable datums near the ends of the arrow. Datums include extension lines, model edges, and crosses at the centers of arcs. If the arrow spans between compatible datums, the program reports the appropriate dimension. For example, if the tail of a single-headed arrow is at the cross of an arc, and the head is on the arc itself, the program reports the radial dimension. Likewise, if a double-headed arrow connects two parallel extension lines, the program reports the distance between them. Recall that the program may maintain multiple interpretations for the orientation of an extension line. When an ambiguous extension line is related to another datum via an arrow, the program uses the additional context to identify the intended orientation of the extension line. The program selects an orientation that is consistent with the orientation of both the other datum and the arrow.

Context is also used to correct errors from the arrow recognizer. For example, if both ends of a single-headed arrow touch parallel datums, such as two extension lines, the program assumes that the arrow was misrecognized and was intended to be a double-headed arrow. Similarly, if the two ends of a straight double-headed arrow touch datums that intersect one another, the program assumes that the arrow was intended to be an angular dimension (curved arrow). Similar reasoning is used to correct other interpretation errors.

4.4. Tracing Gesture Recognizer. A user can query the area of a bounded region by tracing its perimeter with one or more pen strokes. PaperCAD identifies tracing gestures by first identifying the set of model edges that are near the ink. To facilitate this, the model edges are sampled. Each line is sampled with 100 equally spaced points, and each arc is sampled with 180 equally spaced points. If 80% of the sample points of an edge are near ink data points, we assume the edge was traced. The threshold for "nearness" is the same as used with extension lines (i.e., 5% of the average of the height and width of the model's bounding box). The value of 80% was selected as a compromise to allow some amount of sloppiness, without a high risk of false positives. If the set of traced edges forms a closed polygon, the gesture is assumed to be a tracing gesture.

4.5. Text Label and Edit Gesture Recognizer. PaperCAD requires the user to provide a symbolic label for each dimension arrow and traced area. To increase recognition accuracy, labels are restricted to single uppercase letters. Labels are recognized using Microsoft's handwriting recognizer [33]. This recognizer is also used to recognized the cross-out gesture ("\") and the caret gesture ("∧") used for inserting new terms into an equation.

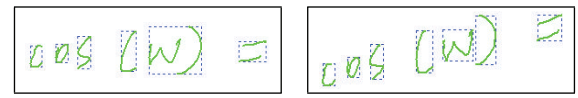
5. Equation Interpretation

Equations are interpreted using a three-step process. First, the ink is segmented into individual characters. Then, the Microsoft handwriting recognizer is used to classify each of the characters. The resulting interpretation is often unreliable. Thus, the final step is an error correction process in which a hidden Markov model (HMM) is used to correct interpretation errors.

A simple approach to segmenting an equation into characters is to group stokes with overlapping bounding boxes as shown in Figure 6(a). However, if characters are drawing too close together or are slanted, this simple approach can fail. For example, in Figure 6(a), "W" and ")" are incorrectly clustered together. We have found that we can obtain more accurate clustering results by rotating the characters a small amount prior to the bounding box calculation. This minimizes the effect of slanting. Empirically, we have found that rotating the entire equation 11° counter-clockwise produces the optimal result. For example, when the equation in Figure 6(a) is rotated by 11°, "W" and ")" are correctly clustered as shown in Figure 6(b). After the clustering is computed, the equation is rotated 11° clockwise so that the characters are not distorted when they are sent to the character recognizer. Note that because the two strokes of an equal sign are disconnected, they are treated as a special case during clustering.

Once an equation has been segmented, each cluster is classified with the Microsoft handwriting recognizer. To improve accuracy, the recognizer is biased to return one of 44 legal characters which include capital letters, numerical digits, "(", ")", "+", "−", "∗", "/", "·", and "=".

The Microsoft handwriting recognizer is intended for cursive rather than printed characters. Consequently



(a) (b)

Figure 6: (a) Simple bounding box approach to clustering. (b) Rotating the equation 11° counter-clockwise before constructing bounding boxes results in more accurate clustering.

the recognition results are somewhat unreliable. Some common errors can be easily identified and repaired. For example, equal signs are often interpreted as a pair of minus signs. Thus, our program reinterprets a pair of consecutive minus signs as an equal sign. However, most errors are more difficult to identify. For example, the term "COS" may be interpreted as "C05", "(OS", and so forth. We use an HMM to correct these sorts of errors. With this approach, the output of the handwriting recognizer constitutes a sequence of observations (e.g., "C" - "0" - "5"), and the correct interpretation of the clusters constitutes a sequence of hidden states (e.g., "C" - "O" - "S"). We use the Viterbi algorithm [36] to find the most probable sequence of states that could have produced the sequence of observations. This is an instance of the most probable path problem.

An HMM can be represented as a three-tuple: $\lambda(A_{ij}, B_j(k), \pi_i)$. Here, A_{ij} is the state transition probability distribution, describing the probability of transitioning from state i to state j; $B_j(k)$ is the observation probability distribution, describing the probability of observing symbol k in state j; π_i is the initial state probability distribution, describing the probability of state i being the first state in a sequence. (For a detailed overview of HMMs, see [36].)

Our HMM has 49 states which include the 44 legal characters (see above) as well as 5 special states. The latter include the space character and four "compound" states that include "SI", "SIN", "CO", and "COS". The compound states help the system to identify "SIN" and "COS". For example, if the sequence of observations is "S" - "I" - "N", the most probable sequence of states is "S" - "SI" - "SIN". Our HMM has 49 observation symbols corresponding to the 49 states. The most likely observation for most states is the same as the state itself. The compound states are the exception. For example, the most likely observation for state "SIN" is "N".

We computed the A_{ij} matrix based on a simple grammar for legal equations (Table 1). Specifically, equations may contain single-letter variables, decimal numbers, "SIN", and "COS". Furthermore, multiplication must be explicit (e.g., $X * Y$, rather than XY), the arguments to "SIN" and "COS" must be enclosed in parentheses, and equations must end with an equal sign. From the complete grammar, we generated the set of legal state transitions. We computed the complete A_{ij} matrix by assuming that all legal transitions from a state were equally likely, with the total probability of such transition summing to 99%. Similarly, we assumed that all illegal transitions from a state were equally likely, with the total probability of such transitions summing to 1%. For example, the probability that any particular digit follows

Table 1: Legal state transitions for equation HMM. *digit* = digits from "0" to "9"; *alpha* = all uppercase letters; *alpha_cs* = all uppercase letters except "C" and "S"; *op* = "+", "−", "∗", and "/".

S_i	S_{i+1}
" ", "="	" "
op	*digit*, *alpha*
"("	*digit*, *alpha*, "+", "−", "."
")"	*op*, "="
"."	*digit*
digit	*digit*, *op*, "=", ")", "."
alpha_cs	*op*, "=", ")"
"C"	*op*, "=", ")", "CO"
"S"	*op*, "=", ")", "SI"
"CO"	"COS"
"SI"	"SIN"
"SIN", "COS"	"("

a decimal point is 0.99 ∗ (1/10), whereas the probability that any particular nondigit follows a decimal point is 0.01 ∗ (1/39). The pi_i matrix was derived from the grammar in a similar fashion.

The $B_j(k)$ matrix was obtained from experimental data. Several subjects provided examples of both individual symbols and complete equations. These were processed with the Microsoft handwriting recognizer to produce a confusion matrix. This was then used to estimate the $B_j(k)$. Because the confusion matrix was relatively sparse, the $B_j(k)$ matrix contained many zero probability entries. To make the HMM more robust, we added 10% to every value in the $B_j(k)$ matrix and then renormalized each row to 100%. This enabled the system to tolerate recognition errors not observed in the training data.

We have found that our HMM significantly improves recognition accuracy. For example, in one experiment, we evaluated the system's performance on the set of equations used to train the $B_j(k)$ matrix. We compared the interpretation accuracy both with and without the HMM. Here, accuracy is defined in terms of the edit distance, which is the minimum number of character changes needed to transform the interpretation result into the correct string. Without the HMM, the average edit distance was 2.81; with the HMM, it was 1.58. The equations contained, on average, 13.17 characters. Thus, without the HMM, on average only 78.6% of each equation was correctly recognized, while, with the HMM, 88.0% was correct. While this 43% reduction in errors is a significant increase in accuracy, there is still room for improvement.

6. User Study

We conducted a user study to evaluate the performance and usability of PaperCAD. The study included 10 volunteer subjects, none of whom had provided any training data for the system. All of the subjects were mechanical engineering students at our university. Subjects were compensated with a $15 gift certificate. Each session involved a single student and lasted about an hour. Sessions were videotaped with an overhead camera, and all digital data, such as pen strokes and button clicks, was logged to a hard disk.

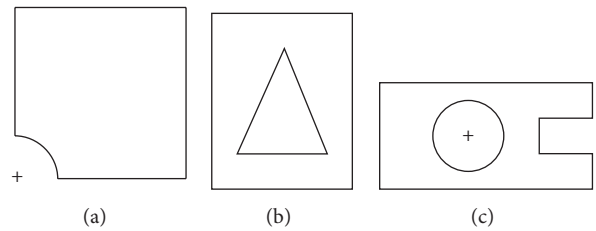


Figure 7: Drawings from user study.

Each session began with a warm-up exercise in which the subject practiced using the smartpen to operate a calculator application. This provided experience with the feel of the device and the draw-and-pause user interaction paradigm. Next, the subject viewed a brief slide presentation illustrating how the system is used to make measurements on a drawing and evaluate equations. Finally, each subject was provided with a one-sheet reference guide for the system. The guide provided examples of typical types of dimensions, a list of editing gestures, and tips for writing equations. With the guide in hand, each subject was asked to perform a set of tasks. In some cases, the subjects were asked to complete tasks using either concise or verbose audio, without the LCD display on the PDA. In other cases, they were asked to use the LCD display combined with audio feedback. In the remainder of the cases, the subjects were allowed to use whichever feedback modalities they preferred. The complete experimental protocol is as follows.

(1) Using verbose audio without the LCD: measure the height and width of a rectangular plate with a quarter-circle cutout at one corner (Figure 7(a)). Measure the angle of one corner of the plate. Measure the area of the plate with a tracing gesture. Write an equation to determine the area of the quarter-circle cutout.

(2) Using verbose audio without the LCD: measure the linear and angular dimensions of a triangle and its area. Construct equations to compute the perimeter and area.

(3) Using concise audio without the LCD: repeat step (2) for another triangle.

(4) Using the LCD and audio: repeat step (2) for a parallelogram.

(5) Using desired feedback methods: for a rectangle with a triangular hole (Figure 7(b)), measure the distance from the triangle's vertices to the edges of the rectangle. Compute the area bounded by the rectangle and triangle.

(6) Using desired feedback methods: compute the area and perimeter of a hexagon. Measure the area of the hexagon.

(7) Using desired feedback methods: measure the radius, diameter, and area of two circles. Determine

the difference between the diameters of two concentric circles.

(8) Using desired feedback methods:measure the height and width of the notch in the part in Figure 7(c), and determine the area of the part excluding the hole.

7. Results

The log files that the PaperCAD system captured during the user study enabled us to perform a detailed analysis of the system's performance and usability. Table 2 presents accuracy data for the interpretation of extension lines. Of the 450 extension lines drawn during the study, 375 (83%) were correctly interpreted by the system. Forty-five of the 75 interpretation errors were caused by the system. Another 28 errors were a result of user error. For example, in nine cases, the user drew the intended extension line in the wrong location, while in five cases the user failed to pause before drawing the next extension line. Two of the interpretation errors were caused by the digital pen failing to capture all of the ink that was written.

Tables 3, 4, and 5 present accuracy data for the interpretation of arrows. The system correctly interpreted 83% of the single-headed arrows. The misinterpreted single-headed arrows were confused with extension lines. The system correctly interpreted 77% of the double-headed arrows. About half of the interpretation errors were caused by the system. The remaining errors were a result of user error. For example, 13 of the interpretation errors resulted from the user failing to complete the arrow before the timeout, while 11 were a result of failing to draw a necessary extension line before drawing the arrow. The system correctly interpreted 85% of the curved arrows. Again, about half the interpretation errors were caused by the system, while the rest were caused by user error. Most of the user errors were due to a failure to complete drawing before the system timeout or a failure to pause before drawing another object.

The PaperCAD system successfully used contextual information to overcome some of the errors in interpreting arrows. The results in Tables 3, 4, and 5 reflect this. Table 6 provides a more detailed view of the effectiveness of our context-based correction process. Contextual information was used to change the initial interpretation of 26 arrows. Eighty-one percent of these changes resulted in the correct interpretation.

The system performed well at interpreting tracing gestures (Table 7). The system correctly interpreted 94% of the 108 tracing gestures. Only one of the six errors was caused by the system, while five were caused by user error.

Interpretation of text proved to be the biggest challenge for the system. The system correctly interpreted *all* of the characters in 56% of the equations (Table 8). The equations contained a total of 925 characters, with an average of 6.4 characters per equation. The handwriting recognizers (the Microsoft handwriting recognizer and our "=" recognizer) incorrectly interpreted 114 of the 925 characters. Our HMM reduced this to 88 misinterpretations, which is a 23% reduction in errors. Using the HMM, the system correctly interpreted 90% of the characters. However, as equations have multiple characters, and thus multiple opportunities for interpretation errors, only 56% of the equations were interpreted

Table 2: Interpretation of extension lines.

Event	Correct	Incorrect
Correctly interpreted	375	
Ink capture failure (H)		2
Drawn in wrong direction, not parallel/perpendicular to anything (U)		3
Drawn in wrong location (U)		9
Drawn incorrectly, over-stroked (U)		2
Drawn incorrectly, tapped first (U)		3
Drawn on the perimeter of a circle (U)		1
Drawn too far from the drawing (U)		5
No pause before drawing another (U)		5
Not recognized (S)		29
Angle misinterpreted (S)		1
Location misinterpreted (S)		8
Misinterpreted as a single arrow (S)		3
Misinterpreted as a tap (S)		4
Total	**375**	**75**

H: hardware error; U: user error; S: system error.

Table 3: Interpretation of single-headed arrows.

Event	Correct	Incorrect
Correctly interpreted	24	
Misinterpreted as an extension line (S)		5
Total	**24**	**5**

S: system error.

Table 4: Interpretation of double-headed arrows.

Event	Correct	Incorrect
Correctly interpreted	233	
Did not complete before timeout (U)		13
Did not pause before writing label (U)		1
Missing one arrowhead (U)		1
Not perpendicular to the datums (U)		1
Extension lines not parallel (U)		3
Did not draw extension line (U)		11
Did not realize that previously drawn extension line was not valid (U)		2
Datums not perpendicular (U)		5
Not recognized (S)		1
Extension line misinterpreted (S)		3
Misinterpreted as a single arrow (S)		15
Misinterpreted as missing a datum (S)		1
Misinterpreted as not perpendicular to datums (S)		14
Total	**233**	**71**

U: user error; S: system error.

Table 5: Interpretation of curved arrows.

Event	Correct	Incorrect
Correctly interpreted	68	
Did not complete drawing the arrow before timeout (U)		3
Forgot to pause and attempted to draw another arrow (U)		1
Missing one arrowhead (U)		1
Misinterpreted as a single arrow (S)		6
Misinterpreted as missing a datum (S)		1
Total	**68**	**12**

U: user error; S: system error.

Table 6: Context-based error correction.

Event	Correct	Incorrect
Single arrow corrected to a curved arrow	4	
Single arrow corrected to a double arrow	17	
Single arrow incorrectly changed to curved arrow		3
Double incorrectly changed to curved arrow		2
Total	**21**	**5**

Table 7: Interpretation of traces.

Event	Correct	Incorrect
Correctly interpreted	102	
Did not complete before timeout (U)		3
Drawn too far from the drawing (U)		3
Misinterpreted as not forming a closed contour (S)		1
Total	**102**	**6**

U: user error; S: system error.

Table 8: Interpretation of equations.

Event	Correct	Incorrect
Correctly interpreted	93	
Ink capture failure (H)		3
Not written in the equation area (U)		1
Forgot to tap abort before writing new equation (U)		4
Incorrect division operator (U)		1
Incorrect multiplication operator (U)		4
Misinterpreted because of timeout (U)		9
Written correctly but at least one character misrecognized (S)		52
Total	**93**	**74**

H: hardware error; U: user error; S: system error.

perfectly. While most interpretation errors were caused by the system, there were some user errors. For example, some users drew the wrong symbols for multiplication or division (e.g., "x" instead of "$*$"). The PaperCAD system enables the user to edit equations. As shown in Table 9, just over two-thirds of the editing operations were successful. Most of the problems were again due to errors in character recognition.

Table 9: Equation Editing.

Event	Correct	Incorrect
New term correctly interpreted	38	
Did not complete before timeout (U)		1
Did not realize cross-out failure and continued to write a new term (U)		1
Misinterpreted (S)		15
Total	**38**	**17**

U: user error; S: system error.

Table 10: Interpretation of labels.

Event	Correct	Incorrect
Correctly interpreted	443	
Ink capture failure (H)		4
Did not complete before timeout (U)		4
Did not realize cross-out failure and continued to write label (U)		3
Invalid Label (U)		5
Misinterpreted (S)		42
Total	**443**	**58**

H: hardware error; U: user error; S: system error.

Table 11: Interpretation of cross-outs.

Event	Correct	Incorrect
Correctly interpreted	83	
Ink capture failure (H)		1
Drew "/" instead of "\" (U)		4
Did not pause before writing new label or term (U)		6
Did not realize the ink was not a valid label or term (U)		8
Terms were not yet processed by the system (U)		2
Not recognized (S)		5
Location misinterpreted (S)		4
Total	**83**	**26**

H: hardware error; U: user error; S: system error.

Character recognition errors also occurred when recognizing text labels (Table 10). The character recognizer achieved 88% accuracy on labels. Because the labels comprise single-characters, the HMM could not improve this result.

When the characters in equations and labels were misinterpreted, the users crossed them out and rewrote them. As shown in Table 11, 76% of the cross-out gestures were successful. The majority of errors were user errors. These included, for example, failing to pause before writing the new character and crossing-out text before the system had processed it.

Table 12: Perceived ease of drawing correctly recognized objects and equations. 10 = easy; 1 = hard.

	Mean	Std. dev.
Extension lines	9.2	0.8
Arrows	9.6	0.7
Area tracing	9.8	0.4
Labels	9.1	1.7
Equations	8.3	1.7

Table 13: Perceived usability.

	Mean	Std. dev.
Equation interpretation accuracy	8.4	1.2
Ease of correcting labels/equations	8.7	1.3
Ease of using reinterpretation	9.1	1.0
Clear feedback	9.9	0.3
Prefer concise audio mode	5.3	3.6
Usefulness of LCD	5.8	3.8
Ease of using LCD	9.7	0.7
Ease of learning the system	9.7	0.7
Overall usability	9.6	0.7
Similarity to paper and pencil	8.9	1.1
Prefer pen-based to WIMP	8.8	1.3

8. User's Perceptions

The participants in our study completed a survey evaluating the usability of the system. Nine of the users reported similar experiences, while the tenth subject was an outlier. Specifically, on most survey questions, the tenth subject's answers differed from the mean by several standard errors of the mean. For this reason, we excluded this subject from our statistics. Background questions revealed that this was the only subject who had not had a course in engineering drawing.

Table 12 summarizes survey results about the ease of drawing various objects, such as arrows and equations. The data suggests that study participants perceived little difficulty in drawing with PaperCAD. Ease of equation writing is rated highly, but slightly lower than the ease of drawing other objects.

Table 13 summarizes results related to usability. Participants found equation interpretation accuracy to be good. Likewise, they found it easy to correct equations and labels and to use the reinterpretation button. Participants perceived that the system provided clear instructions and interpretive feedback. Interestingly, however, there was no strong preference for concise audio feedback versus verbose audio. Similarly, while participants found the LCD to be easy to use, they perceived it to be only mildly useful. Participants found the system to be easy to learn and, overall, very usable. Finally, they found the system to be very similar to paper and pencil, and they strongly preferred this system to systems based on a traditional WIMP interface.

As described in Table 14, the study participants' overall reaction to PaperCAD was quite positive. Participants considered the system to be near "wonderful" on a scale of "wonderful" versus "terrible," near "easy" on a scale of "easy" versus "hard," near "satisfying" on a scale of "satisfying" versus "frustrating," and near "stimulating" on a scale of "stimulating" versus "dull."

Table 14: Overall reaction to PaperCAD.

	Mean	Std. dev.
Wonderful = 10, terrible = 1	9.2	0.7
Easy = 10, difficult = 1	9.4	0.7
Satisfying = 10, frustrating = 1	9.2	0.8
Stimulating = 10, dull = 1	9.7	0.7

9. Discussion and Future Work

Our arrow recognizer had an overall accuracy for all types of arrows of 79%. When user errors are excluded, the accuracy is 88%. This is substantially more accurate than the 65% to 70% accuracy of the arrow recognizer reported in [6]. Note that the latter technique cannot be applied to our data, as it is limited to single-stroke arrows drawn from tail to head.

Our results indicate that the Microsoft handwriting recognizer is not well-suited for our task. This recognizer is typically used to recognize words written in cursive handwriting, and thus it is not optimized for interpreting equations and labels. We may be able to improve recognition accuracy using an image-based symbol recognizer such as those in [27, 28]. Alternatively, we could employ the approach used in LaViola's MathPad^2 [32]. His system uses the output of Microsoft's handwriting recognizer as just one feature in a more powerful classifier.

Our HMM-based technique for correcting interpretation errors in handwritten equations has proven to be effective. On the dataset we used to develop our technique, the technique reduced interpretation errors by 43%. In our user study (Section 7), the technique reduced errors by 22%. While this is a substantial improvement, there is clearly a need for a more accurate handwriting recognizer. The high error rate of the current handwriting recognizer is difficult to completely overcome, even with an effective error correction technique.

The generality of our error correction technique makes it suitable for other equation interpretation problems. Because the technique relies on a grammar for legal equations, the technique would be particularly useful in domains in which conventions constrain the content of equations.

Many of the interpretation errors were the result of user errors. For the 1803 drawing events in our user study, there were 123 (6.8%) user errors. For example, users sometimes forgot to pause after drawing objects or failed to complete an object before the timeout. Additionally, sometimes users did not follow the system's conventions such as drawing needed extension lines before drawing a dimension arrow. While we refer to such errors as "user" errors, they imply the need for additional improvement to our user interface design. Our system needs additional flexibility to accommodate variations in the way users draw.

Despite these errors, the survey results from our user study suggest that users are quite satisfied with PaperCAD's interpretation accuracy and user interface design. However, additional studies are needed to fully evaluate the system. For example, it would be useful to consider more complex drawings typical of those used in industry. Future studies should involve subjects who ordinarily use paper technical drawings such as construction workers and machinists. Also, we used letter-size drawings in our user study because of the limited size of our printer. To evaluate the system for its intended application, it is necessary to use large-format drawings.

We were surprised that study participants did not find the LCD to be more useful. Further investigation is needed to understand why. This could be a result of the study design: all participants were asked to use the audio-only feedback mode prior to trying the LCD. In retrospect, it would have been useful for half of the participants to use LCD feedback first. It is also possible that the LCD was considered redundant because audio feedback was simultaneously provided. Future studies should consider the use of an LCD-only mode and an audio-only mode. Finally, it is possible that the slow 1 Hz refresh rate of the display may have been a hindrance. This issue should also be explored in future studies.

We were also surprised that there was no strong preference for concise versus verbose audio. It is possible that the concise audio was insufficiently concise. It is also possible that novice users prefer verbose audio feedback, and more experienced users prefer concise feedback. Again, additional studies will be needed to investigate these issues.

10. Conclusion

We have presented a prototype user interface for augmenting the display of a smartphone, or other PDA, using interactive paper. Specifically, we present a prototype interface that enables a user to interactively interrogate technical drawings using a PDA and an Anoto-based smartpen. Our software system, called PaperCAD, enables users to query geometric information from CAD drawings printed on Anoto dot-patterned paper.

In addition to advances in user interface design, PaperCAD also entails advances in techniques for interpreting pen input. We developed an arrow recognition technique suitable for recognizing arrows with widely varying aspect ratios. An inverse curvature representation is used to locate candidate arrowheads, and a gray-scale bitmap representation and neural network are used to classify them. We also developed techniques that use context to disambiguate sketched dimensions. Finally, we developed a technique, based on a hidden Markov model (HMM), to correct interpretation errors in handwritten equations. The HMM is based on a grammar for legal equations and the confusion matrix for the handwriting recognizer.

We conducted a user study to evaluate the performance and usability of our interface. Participants in the study were asked to measure various dimensions from a set of drawings and to write equations to compute various geometric properties. The study evaluated PaperCAD's two methods for providing feedback: audio feedback with an adjustable level of conciseness and a PDA that provides a video display of the portion of the drawing near the pen tip.

Our user study suggests that PaperCAD's interface and our techniques for interpreting pen input are effective, although there is a need for additional flexibility to accommodate variations in the way users draw. Survey results indicate that users are quite satisfied with the interface and find it easy to use. While PaperCAD is a limited prototype, and additional studies are needed to further evaluate its design, this work is an important step toward the ultimate goal of bridging between the paper and digital worlds.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgment

The authors gratefully acknowledge the National Science Foundation for its support of this work through Award 0735695.

References

[1] A. B. Anoto, ANOTO, 2014.

[2] C. Alvarado and R. Davis, "Resolving ambiguities to create a natural sketch based interface," in *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI '01)*, pp. 1365–1371, 2001.

[3] J. A. Landay and B. A. Myers, "Sketching interfaces: toward more human interface design," *IEEE Computer*, vol. 34, no. 3, pp. 56–64, 2001.

[4] K. D. Forbus, R. W. Ferguson, and J. M. Usher, "Towards a computational model of sketching," in *Proceedings of the International Conference on Intelligent User Interfaces (IUI '01)*, pp. 77–83, January 2001.

[5] R. de Silva, D. Tyler Bischel, W. Lee, E. J. Peterson, T. F. Stahovich, and R. Calfee, "Kirchhoff's pen: a pen-based circuit analysis tutor," in *Proceedings of the EUROGRAPHICS Workshop on Sketch-Based Interfaces and Modeling*, 2007.

[6] L. B. Kara and T. F. Stahovich, "Hierarchical parsing and recognition of hand-sketched diagrams," in *Proceedings of the Annual ACM Symposium on User Interface Software and Technology*, pp. 13–22, ACM, October 2004.

[7] T. Hammond and R. Davis, "Tahuti: a geometrical sketch recognition system for UML class diagrams," in *Proceedings of the AAAI Spring Symposium on Sketch Understanding*, pp. 59–68, 2002.

[8] R. C. Zeleznik, K. P. Herndon, and J. F. Hughes, "SKETCH: an interface for sketching 3D scenes," in *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '96)*, pp. 163–170, 1996.

[9] M. Masry, D. J. Kang, I. Susilo, and H. Lipson, "A freehand sketching interface for progressive construction and analysis of 3D objects," in *Proceedings of the AAAI Fall Symposium—Making Pen-Based Interaction Intelligent and Natural*, pp. 98–105, 2004.

[10] C. Liao, F. Guimbretière, and K. Hinckley, "PapierCraft: a command system for interactive paper," in *Proceedings of the 18th Annual ACM Symposium on User Interface Software and Technology (UIST '05)*, pp. 241–244, ACM, October 2005.

[11] P. R. Cohen and D. R. McGee, "Tangible multimodal interfaces for safety-critical applications," *Communications of the ACM*, vol. 47, no. 1, pp. 41–46, 2004.

[12] H. Song, F. Guimbretière, C. Hu, and H. Lipson, "ModelCraft: capturing freehand annotations and edits on physical 3D models," in *Proceedings of the 19th Annual ACM Symposium on User Interface Software and Technology (UIST '06)*, pp. 13–22, ACM, October 2006.

[13] R. B. Yeh, C. Liao, S. R. Klemmer et al., "ButterflyNet: a mobile capture and access system for field biology research," in *Proceedings of the Conference on Human Factors in Computing Systems (CHI '06)*, pp. 571–580, April 2006.

[14] E. Al-Imam and E. Lank, "Pocket-Pad: using ha ndhelds and digital pens to man age data in mobile contexts," in *Proceedings of the 1st International Conference on the Digital Society (ICDS '07)*, p. 13, 2007.

[15] W. E. Mackay, D. S. Pagani, L. Faber et al., "Ariel: augmenting paper engineering drawings," in *Proceedings of the Conference Companion on Human Factors in Computing Systems (CHI '95)*, pp. 421–422, 1995.

[16] W. E. Mackay, G. Pothier, C. Letondal, K. Bøegh, and H. Erik Sørensen, "The missing link: augmenting biology laboratory notebooks," in *Proceedings of the 15th Annual Symposium on User Interface Software and Technology (UIST '02)*, pp. 41–50, Paris, France, October 2002.

[17] C. Liao, F. Guimbretière, and C. E. Loeckenhoff, "Pen-top feedback for paper-based interfaces," in *Proceedings of the 19th Annual ACM Symposium on User Interface Software and Technology (UIST '06)*, pp. 201–210, October 2006.

[18] W. Lee, R. De Silva, E. J. Peterson, R. C. Calfee, and T. F. Stahovich, "Newton's Pen—a Pen-based tutoring system for statics," in *Proceedings of the 4th Eurographics Workshop on Sketch-Based Interfaces and Modeling (SBIM '07)*, pp. 59–66, August 2007.

[19] D. Rubine, "Specifying gestures by example," *Computer Graphics*, vol. 25, no. 4, pp. 329–337, 1991.

[20] L. Gennari, L. B. Kara, T. F. Stahovich, and K. Shimada, "Combining geometry and domain knowledge to interpret hand-drawn diagrams," *Computers and Graphics*, vol. 29, no. 4, pp. 547–562, 2005.

[21] M. J. Fonseca, C. Pimentel, and J. A. Jorge, "CALI: an online scribble recognizer for calligraphic interfaces," in *Proceedings of the AAAI Spring Symposium on Sketch Understanding*, pp. 51–58, 2002.

[22] H. H. Hse and A. R. Newton, "Recognition and beautification of multi-stroke symbols in digital ink," *Computers & Graphics*, vol. 29, no. 4, pp. 533–546, 2005.

[23] T. Hammond and R. Davis, "LADDER, a sketching language for user interface developers," *Computers & Graphics*, vol. 29, no. 4, pp. 518–532, 2005.

[24] T. Hammond and R. Davis, "Interactive learning of structural shape descriptions from automatically generated near-miss examples," in *Proceedings of the 11th International Conference on Intelligent User Interfaces (IUI '06)*, pp. 210–217, ACM, February 2005.

[25] M. Shilman, H. Pasula, S. Russel, and R. Newton, "Statistical visual lan guage models for ink parsing," in *Proceedings of the AAAI Spring Symposium on Sketch Understanding*, pp. 126–132, 2002.

[26] M. D. Gross, "Recognizing and interpreting diagrams in design," in *Proceedings of the Workshop on Advanced Visual Interfaces (AVI '94)*, pp. 88–94, Bari, Italy, June 1994.

[27] L. B. Kara and T. F. Stahovich, "An image-based, trainable symbol recognizer for hand-drawn sketches," *Computers and Graphics (Pergamon)*, vol. 29, no. 4, pp. 501–517, 2005.

[28] T. Y. Ouyang and R. Davis, "A visual approach to sketched symbol recognition," in *Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI '09)*, pp. 1463–1468, San Francisco, Calif, USA, July 2009.

[29] J. O. Wobbrock, A. D. Wilson, and Y. 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)*, pp. 159–168, ACM, New York, NY, USA, 2007.

[30] L. Zhang, D. Blostein, and R. Zanibbi, "Using fuzzy logic to analyze superscript and subscript relations in handwritten mathematical expressions," in *Proceedings of the 8th International Conference on Document Analysis and Recognition*, vol. 2, pp. 972–976, August 2005.

[31] N. Matsakis, *Recognition of handwritten mathematical expressions [M.S. thesis]*, Massachusetts Institute of Technology, Cambridge, Mass, USA, 1999.

[32] J. J. LaViola Jr., "An initial evaluation of MathPad2: a tool for creating dynamic mathematical illustrations," *Computers and Graphics (Pergamon)*, vol. 31, no. 4, pp. 540–553, 2007.

[33] J. A. Pittman, "Handwriting recognition: tablet PC text input," *Computer*, vol. 40, no. 9, pp. 49–54, 2007.

[34] T. M. Sezgin and R. Davis, "HMM-based efficient sketch recognition," in *Proceedings of the 10th International Conference on Intelligent User Interfaces (IUI '05)*, pp. 281–283, Island of Madeira, Portugal, January 2005.

[35] A. Kosmala, G. Rigoll, S. Lavirotte, and L. Pottier, "On-line handwritten formula recognition using statistical methods," in *Proceedings of the 14th International Conference on Pattern Recognition*, pp. 1306–1308, 1998.

[36] L. R. Rabiner, "Tutorial on hidden Markov models and selected applications in speech recognition," *Proceedings of the IEEE*, vol. 77, no. 2, pp. 257–286, 1989.