Intelligent Understanding of Handwritten Geometry Theorem Proving

Yingying Jiang[1] *Feng Tian*[1] *Hongan Wang*[1,2]
[1] Intelligence Engineering Lab,
Institute of Software,
Chinese Academy of Sciences
Beijing, China
{jyy,tf,wha}@iel.iscas.ac.cn

Xiaolong Zhang[3] *Xugang Wang*[1] *Guozhong Dai*[1,2]
[2] State Key Laboratory of Computer Science,
Institute of Software, Chinese Academy of Sciences
{wxg,dgz}@iel.iscas.ac.cn
[3] The Pennsylvania State University
lzhang@ist.psu.edu

ABSTRACT
Computer-based geometry systems have been widely used for teaching and learning, but largely based on mouse-and-keyboard interaction, these systems usually require users to draw figures by following strict task structures defined by menus, buttons, and mouse and keyboard actions. Pen-based designs offer a more natural way to develop geometry theorem proofs with hand-drawn figures and scripts. This paper describes a pen-based geometry theorem proving system that can effectively recognize hand-drawn figures and hand-written proof scripts, and accurately establish the correspondence between geometric components and proof steps. Our system provides dynamic and intelligent visual assistance to help users understand the process of proving and allows users to manipulate geometric components and proof scripts based on structures rather than strokes. The results from evaluation study show that our system is well perceived and users have high satisfaction with the accuracy of sketch recognition, the effectiveness of visual hints, and the efficiency of structure-based manipulation.

Author Keywords
Geometry theorem proving, hand-drawn figures, hand-written proof scripts, recognition, structure based manipulation.

ACM Classification
H5.2. Information Interfaces and Presentation: User Interfaces. - Interaction styles; I.5.4. Pattern Recognition: Applications. - Signal processing.

General Terms Design, Algorithms, Human Factors

INTRODUCTION

Geometry theorem proving is one of the most challenging skills for students to learn in secondary school mathematics [5, 18]. Senk found that the results of geometry education in secondary school were disappointing, even in the second semester of learning geometry [26]. About a quarter of students gave up on problems of geometry proving, and only about 30% of students can complete 75% or more of geometry proofs correctly. Many students find it difficult to write down a formal proof because they do not understand the geometric properties involved in the proof [29].

Computer-assisted geometry proving has been studied by many researchers [2, 7, 9, 12, 19, 23]. Dynamic geometry systems are developed to assist users to create geometric constructions, explore geometry graphs, formulate conjectures, check facts [9, 12, 19, 23], and even build proofs [2, 7]. These tools are useful in helping users understand geometric properties, generate theorem proving ideas, and discover interesting geometry propositions. However, built upon the traditional WIMP interaction style, they impose pre-defined interaction styles and task structures that are defined by menus, buttons, mouse and keyboard actions, and so on. These interaction styles and task structures often do not match what students usually do in geometry proving in their real-life. Being forced to follow unfamiliar interaction styles and task structures, students may be distracted from the major task of exploring and understanding the relationships between proof steps and geometry figures, which is an important factor to improve the understanding of geometry proving [30].

Pen-based interaction offers an opportunity to enhance the learning of geometry proofs by leveraging advanced computational techniques and at the same time, by allowing students to follow the natural and traditional hand-written approach. With pen-based tools, students can write proof scripts and geometric figures on computer screens, just as what they do with pen and paper in real life. Computational tools can understand their hand-writing and offer intelligent help based on their actions and action contexts.

In this paper, we explore an approach to support pen-based geometry proving. A sketch recognition method is proposed to understand the hand-drawn geometry graphs and handwritten geometry proof scripts and build the

IUI'10, February 7–10, 2010, Hong Kong, China.

correspondence between geometric components and proof scripts. Based on the algorithm, we develop a system to provide dynamic and intelligent visual assistance to help users understand the process of proving and support structure-based manipulation of proof scripts. Figure 1 demonstrates some key features of our approach. Lines in Figure 1 are the recognition results of hand-written lines; when a geometry proof step, $AB \perp AC$, is selected, the corresponding geometry representation, the perpendicular lines AB and AC, are highlighted.

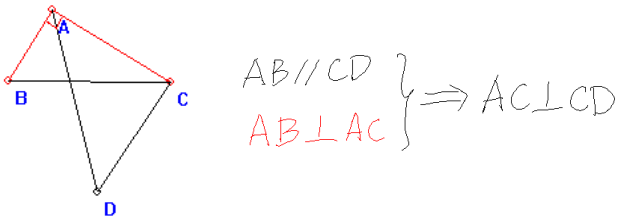


Figure 1. An example of intelligent visual hints that highlights the geometry representation of a selected proof step.

The paper is structured as the following. First, we review related research, and then outline the key factors in geometry proofs by analyzing a proving example. Next, we describe the detail of the recognition algorithm and intelligent tools to assist geometry proving. Furthermore, we present the evaluation of a prototype system of geometry proving. Finally, we discuss the results of our research and conclude the paper with future research directions.

RELATED WORK

Geometry Systems

Computer assisted teaching and learning systems are important to education. In geometry education, interactive geometry software environments [4, 9, 11, 12, 23] have offered new methods for teaching and learning geometry theorem proving [13]. For example, MMP/Geometer [9] and GeoProof [23] support automatic geometry theorem proving; GeoProof [23] used an automatic theorem prover to check facts. Although these systems are useful, they are based on the mouse-and-keyboard interaction, which is unnatural and imposes task structures that do not match what students do in real-life.

Researchers have studied pen-based geometry systems to support more natural interactive activities. Liu et al [22] developed a pen-based geometry teaching system, which supported only geometry drawing but neglected geometry proving. GeoAssistor [10] is a pen-based geometry proving system, but the system cannot understand hand-written proof scripts.

Sketch Recognition

Sketch understanding systems have been used in education area. These systems made learning activities more interesting. MathPad2 [20] is a system to support the creation and exploration of mathematical sketches. Ouyang [24] studied the recognition of hand-drawn chemical diagrams and the generation of the 3D structure of the organic substance based on recognition results to support the interaction with and understanding of the substance. Alvarado [1] built a system to simulate physical phenomenon based on pen-based sketches. Sim-U-Sketch and VibroSketch [16] can assist the learning of the circuit and vibration knowledge by recognizing hand-drawn circuit and vibration diagrams. Research has also been done to recognize handwritten formulae, such as mathematical expression [20] and chemical expression [24,28]. However, these systems are domain-specific, and cannot be easily extended to other domains, such as plane geometry.

A geometry proof consists of hand-drawn geometry figures and handwritten proof scripts. Recognizing hand-drawn graphs has been researched. Paulson [25] studied the recognition of primitive sketches. Li [21] investigated synchronized recognition of graphs with real-time user feedback. However, little research has been done to recognize handwritten geometry proof scripts and build the correspondence between geometry figures and proof scripts.

GEOMETRY THEOREM PROOF IN PLANE GEOMETRY

To prove geometry theorem in plane geometry, a student usually draws the corresponding geometry graph, and writes the proof scripts step by step to prove the theorem. Sometimes, assistant lines are constructed to help the proof.

We collected ten geometry exercise books from junior-high students who were studying plane geometry, and analyzed these manuscripts to get students' typical handwritten proving styles on the paper. Figure 2 demonstrates a snippet from a geometry student's exercise book, which is the typical proving style in these exercise books.

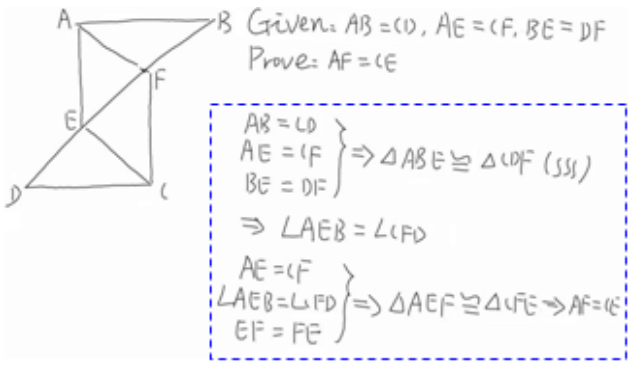


Figure 2. A snippet from a student's geometry exercise book.

As shown in the figure, the proving scripts are on the right of the hand-drawn graph and are within the blue dashed box. The proof scripts consist of a serial of preconditions and conclusions that are combined by the deduction symbols. We define each precondition or conclusion as a proof step. Deduction symbols, such as "}=>" and "=>", are between proof steps. The reason for a deduction is often behind the conclusion item and enclosed by a pair of brackets.

For the structure of a proof step, each proof step usually specifies two geometry objects and their relationship. The type of a geometry object can be "line", "circle", "triangle", "point", and so on. The relationships include "parallel", "perpendicular", "similar", "same", "equal", etc.

In this paper, we propose a computer-based approach to support such geometry proving. Our work focuses on an algorithm to understand geometry proving styles, shown in Figure 2, and to provide assistance to geometry proving by establishing the correspondence between geometry figures and geometry proof scripts. The recognition of handwritten deduction reason is not addressed in this research.

UNDERSTANDING HANDWRITTEN PROOFS

As shown in the preceding section, handwritten geometry proofs include hand-drawn geometry figures and handwritten geometry proof scripts. Both of them are composed of a serial of strokes. Our algorithm recognizes them respectively and then builds the correspondence between them.

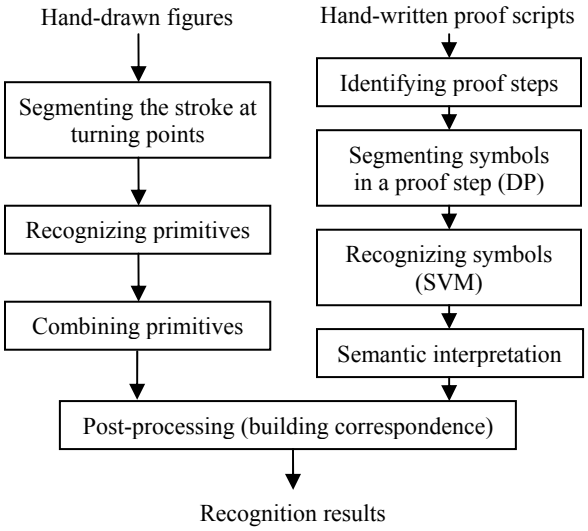


Figure 3. Architecture of hand-drawn geometry proof understanding.

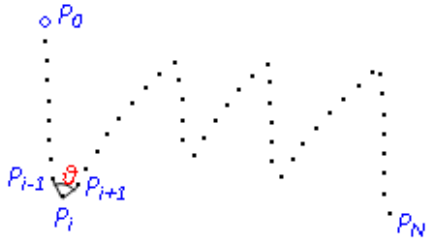
Figure 3 shows the architecture of our algorithm. The hand-drawn figure recognition method deals with each stroke separately. It consists of three steps: stroke segmentation, primitive recognition, and primitive combination. The recognition of hand-written proof scripts is based on proof step identification. For each proof step, DP-based (Dynamic Programming) symbol segmentation [8] and SVM-based symbol recognition [6] are performed to understand the handwritten proof step. After that, the meaning of the proof is created through semantic interpretation. A post-processing step builds the correspondence between the figures and the proofs.

Recognition of Hand-Drawn Geometry Figures

In plane geometry, a geometry figure is usually composed of points, lines, and circles. Our figure recognition method treats each stroke separately. For each stroke, our algorithm first segments it into sub-strokes at turning points. Then each sub-stroke is recognized separately to be a shape primitive (point, line, or circle). Furthermore, a hierarchy of the primitives is constructed. Combining all the recognition results of strokes leads to a geometry figure.

Stroke segmentation

A stroke contains a serial of points. For each stroke, it is re-sampled according to the equal between-point distance criterion. The re-sampled stroke can be represented as $\{P_0, P_1, \ldots, P_N\}$. $P_i(x, y)$ indicates the position of the *ith* point in the stroke. Figure 4 gives an example of a re-sampled stroke.



$$\theta = \arccos\left(\frac{|P_{i-1}P_i|^2 + |P_iP_{i+1}|^2 - |P_{i-1}P_{i+1}|^2}{2|P_{i-1}P_i||P_iP_{i+1}|}\right)$$

Figure 4. Re-sampled points of a stroke and an angle between two adjacent lines of this stroke.

To judge whether P_i is a turning point in a stroke, the algorithm calculates the angle θ between Line P_i P_{i-1} and Line P_i P_{i+1} using the law of cosines as shown in Figure 4. If the angle θ is smaller than a given threshold angle – *angleThres*, P_i is a turning point in the stroke; otherwise, P_i is not a turning point. In our research, *angleThres* is set to be $4\pi/5$.

After finding all turning points in a stroke, the stroke is divided into several sub-strokes separated by turning points. If the number of re-sampled points of a sub-stroke is less than a given threshold, *ptThres*, the sub-stroke is added to its preceding sub-stroke. After segmentation, each sub-

stroke constitutes a shape primitive. In our research, *ptThres* is set to be 5.

Primitive recognition

A shape primitive could be a point, a line, or a circle. The following formula defines how the type of a stroke is found.

$$type = \begin{cases} Point, & if\ (pointNum < ptThres) \\ Line, & if\ (density > lDenThres\ \&\ density < hDenThres) \\ Circle, & else \end{cases}$$

When the stroke in a shape primitive has less than *ptThres* points, the primitive is a point. Line recognition is based on stroke density. Here, we define the density of a stroke as the ratio of the stroke's length to its bounding box's diagonal length. When the density of a stroke in a primitive is between two given thresholds, *lDenThres* and *hDenThres*, the primitive is a line. If a stroke is not recognized as either a point or a line, it is taken as a circle. In our algorithm, *lDenThres* and *hDenThres* are set to be 0.7 and 1.3 respectively.

Letter labels are assigned to shape primitives. A point only needs one letter to be separated from other points. A line has two labels, each of which corresponds to an end of the line. A circle also has two labels: one for the center and the other for a point on the circle. Our algorithm keeps track of what letters have been assigned and guarantees the uniqueness of letter labels.

Primitive combination

After knowing shape primitives, the algorithm combines these primitives to form a high level geometry figure. For example, when the endpoint of a line is very close to the endpoint of another line, the two lines are linked by a common endpoint.

Figure 5 illustrates the recognition results of some hand-drawn geometry figures. Figure 5(a) is a hand-drawn curve, and Figure 5(b) shows its recognition result, a poly-line and letter lables on individual lines. The quadrangle in Figure 5(d) is the recognition result of the drawing in Figure 5(c). The hand-drawn graph in Figure 5(e) is recognized as a circle in Figure 5(f).

Recognition of Handwritten Geometry Proof Scripts

The basic unit of a geometry proof script is proof step. Our geometry proof script recognition method is based on proof step identification. The following sections describe proof step identification, symbol partition in a geometry item, symbol recognition, and semantic interpretation.

Proof step identification

A proof step is usually followed by a deduction symbol or by another step in a new line. We use this feature to identify individual proof steps.

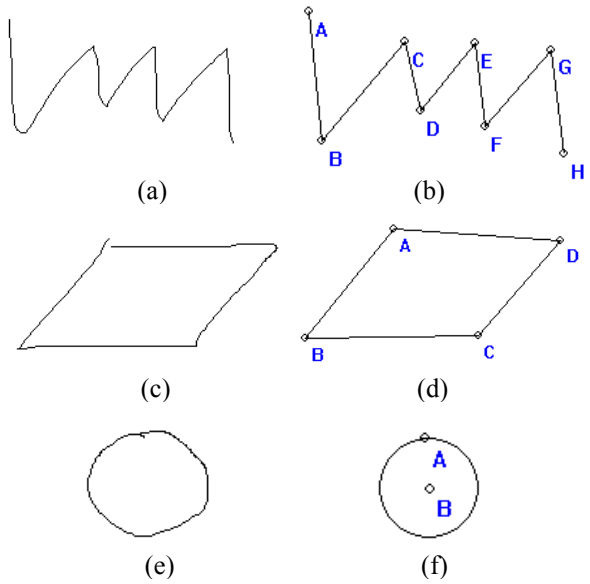


Figure 5. Hand-drawn geometry graphs and their recognition results.

When the pen is up, the algorithm judges whether the current stroke is the deduction symbol '}' by the SVM-based recognition method, which will be introduced below, or whether it is a stroke in a new line by the spatial relations to the current proof step. If the stroke is '}', the current proof step is finished. If the stroke is in a new line, the current proof step is finished and a new proof step is created. Otherwise, the DP and SVM based symbol segmentation and recognition process are performed to get the recognition result of the current handwritten proof step. If the last symbol is '=>', a proof step has been finished and the user is going to start a new proof step. Otherwise, the user is still working on the current proof step.

Symbol partition in a proof step

Dynamic programming is typically applied to optimization problems [8]. It is used to segment a handwritten proof step to several symbols in our algorithm.

Before symbol partitioning, the strokes in the proof step are sorted according to the left borders of their bounding boxes. Thus, the writing order of the symbols in a proof step is not restricted. After stroke sorting, overlapped strokes are merged to stroke blocks. The merged stroke blocks can be represented as $\{b_1, b_2 \ldots b_N\}$.

The following formula describes the approach to find optimal symbol segmentations by dynamic programming.

$$D(i,j) = \min\left\{D(i,k) + D(k+1,j), d(i,j)\right\}$$

$$i,j,k \in [1,N], i \le j, i \le k < j$$

where $D(i, j)$ is the reliability corresponding to the optimal symbol segmentation of stroke blocks $\{b_i, ..., b_j\}$ and $d(i, j)$ is the reliability of the candidate symbol that is composed of stroke blocks $\{b_i, ..., b_j\}$.

The symbols in a proof step have the following characteristics: first, they have similar widths and heights; second, the distances between stroke blocks in a symbol are often smaller than the distances between stroke blocks in different symbols; third, the width of a symbol is often not too large. Thus, *d(i,j)* can be calculated by the following formula:

$$d(i,j) = a * intraDF(i,j) + b * \frac{1}{interDF(i,j)} + c * wF(i,j)$$

$$i, j \in [1, N], i \le j$$

where intraDF is the intra-distance factor between stroke blocks in a symbol. *interDF* describes the inter-distance between adjacent symbols; *wF* is the width of a candidate symbol; and *a*, *b*, *c* are weights of these factors in the overall distance(In our algorithm, these three factors are treated equally: a=0.33, b=0.33 and c=0.33).

Suppose $d'(i,j)$ is the normalized distance between the bounding boxes of the stroke block b_i and the stroke block b_j, $intraDF(i,j)$ and $interDF(i,j)$ can be calculated by the following formulae.

$$intraDF(i,j) = \sum_{k=i+1}^{j} d'(k-1,k)$$

$$interDF(i,j) = \begin{cases} d'(i-1,i) + d'(j,j+1) & if\ (i>0\ \&\ j<N) \\ d'(i-1,i) + \overline{d}' & if\ (i>0\ \&\ j=N) \\ d'(j,j+1) + \overline{d}' & if\ (i=0\ \&\ j<N) \end{cases}$$

The following formula describes the calculation of $wF(i,j)$. It is normalized by the average height and the maximum width of the stroke blocks.

$$wF(i,j) = \frac{width(i,j) - \frac{1}{N}\sum_{k=0}^{N} height(k)}{\max_{k\in[1,N]}\{width(k)\}}$$

Symbol recognition

SVM classifier is proved to be effective to recognize hand-drawn mathematical symbols [17]. As most symbols in geometry proof are mathematical symbols, this research also adopts a SVM classifier to recognize handwritten geometry symbols. In particular, the multi-class classification is accomplished with Libsvm [6] and the classifier uses the RBF kernel. A handwritten symbol can contain online features that include sequential information about points and strokes, and offline features that are based on the symbol's corresponding image. As online features and offline features could complement each other [17, 27], both online features and offline features are used by our classifier.

Before extracting features for the SVM input, normalization is performed for the handwritten symbols. The normalized symbols have fixed size and contain the same number of points. Moreover, the image of the symbol is also generated.

Our algorithm uses the angles between adjacent points in the strokes as the online features and uses the ratios of black pixels as the offline features. Suppose the strokes in a symbol have *pN* points and the image is partitioned into m*m sub-images. The features used by the classifier can be represented as follows:

$$F = \begin{Bmatrix} \sin_1, \cos_1, \sin_2, \cos_2 ... \sin_{pN-1}, \cos_{pN-1}, \\ density_1, density_2, ..., density_{m*m} \end{Bmatrix}$$

$$\sin_i = \frac{y_{i+1} - y_i}{\sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2}}$$

$$\cos_i = \frac{x_{i+1} - x_i}{\sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2}}$$

where $density_i$ is the ratio of black pixels in the *ith* sub-image.

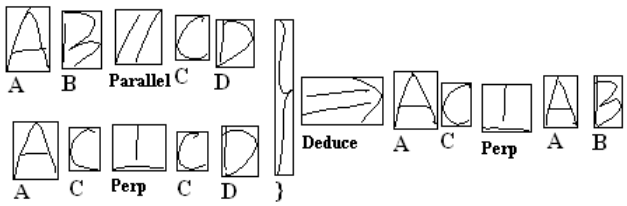
Currently, the geometry symbols that can be recognized by our algorithm include 10 digits, 26 English letters – both lower and upper cases, and some special symbols in geometry (∵ ∴ ∠ △ ▱ ≌ ⊥ ∽ + − ∟ = // } ⟹).

Semantic interpretation

After getting the proof recognition result, semantic meanings of a proof step can be extracted. For example, "△*ABC*" are not just four symbols, but have the meaning of a triangle consisting of three linked lines.

Our algorithm uses a semantic interpretation table to look up semantic meanings. The semantic interpretation table mainly includes two parts. One part defines the semantic meaning of symbols that indicate geometry shapes, such as '∠', '∟', '△', and '▱'. The other part explains the relationship symbols that describe relationships between geometry components, such as '∵', '∴', '≌', '⊥', '∽', '+', '-', '=', '//', and '⟹'.

Figure 6 shows the recognition result of a handwritten geometry proof.



Item1: Parallel<Line(A,B), Line(C,D)>

Item2: Perpendicular<Line(A,C), Line(C,D)>

Item3: Perpendicular<Line(A,C), Line(A,B)>

Deduce<<Item1,Item2>,Item3>

Figure 6. Recognition result of handwritten geometry proof.

Post-processing

With the recognition results, our algorithm builds the correspondence between the geometry graphs and the geometry proof scripts. The recognized figures are represented as a serial of graph primitives and their relationships.

After recognizing handwritten proof scripts, our algorithm connects a proof step with geometry objects. The correspondence between them is established by matching the recognized letters in proof step with the labels of graph primitives.

Such correspondence can also help to correct some errors in the recognition of handwritten proof scripts. As the proof scripts should be consistent with the graph, our algorithm uses the knowledge extracted from the graph to identify potential text recognition errors. For example, when "CD" is incorrectly recognized as "CP", the result can be corrected by knowing that there's a line "CD" in the graph and there's no line "CP" in the graph.

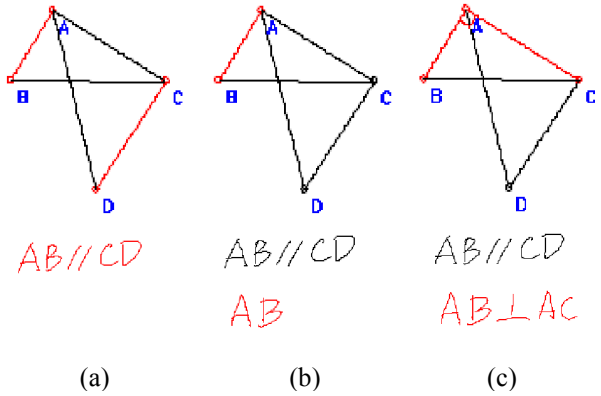
ASSISTANCE TO GEOMETRY PROOF

If a student does not understand the conditions of a proposition, she cannot really judge the correctness of the proposition [13, 14]. Moreover, the student may fail to relate each proof step in a text book proof to the accompanying figure [30]. To help students understand geometry proof, we provide intelligent visual hints and structure-based manipulation technique.

Intelligent hints for geometry proof

By using the correspondence between geometry figures and geometry proof scripts, we provide intelligent hints to help users understand geometry proving processes. The hints update when the pen is up (Figure 7) or object manipulation happens (Figure 1).

Figure 7 shows how intelligent hints work. When a user writes the geometry proof scripts, the current proof step as well as its corresponding figure is highlighted. The figures at the top of Figure 7 are the formal geometry figures recognized from hand-drawn figures. The bottom shows the geometry proof scripts. The recognition result of the handwritten geometry proof scripts is used implicitly and not shown in the figure. In Figure 7(a), when the step of *AB*//*CD* is written and recognized, Lines *AB* and CD in the geometry figure are highlighted. In Figure 7(b), handwritten *AB* in a new proof step is recognized, and Line *AB* in the geometry figure is highlighted. In Figure 7(c), the proof step of $AB \perp AC$ is recognized, and the perpendicular lines of *AB* and *AC* in the geometry figure are highlighted.



(a) (b) (c)

Figure 7. Highlighting the proof step the user is writing and its corresponding figure.

Intelligent hints also appear when the user manipulates the geometry proofs. For example, as shown in Figure 1, when the user selects a proof step, the step and its corresponding figure are highlighted.

With the intelligent hint technique, a proving process can be replayed by highlighting each geometry proof step with its corresponding figure. This approach can help to overcome the difficulty in relating the proof step in a text book proof to the accompanying figure, a big barrier in learning geometry proof [30].

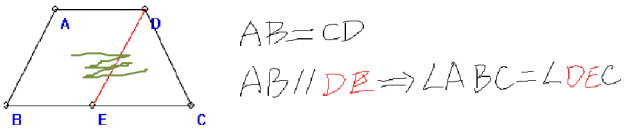
Efficient manipulation of geometry proofs

In geometry proving, some geometry proof steps may be used several times and the user need to write the same steps several times. Sometimes, the user may find errors in proofs and want to delete some proof steps. Thus, it is important to provide efficient methods to manipulate geometry proofs.

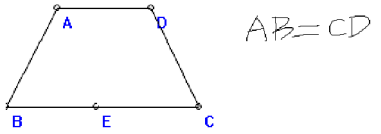
Structure-based manipulation methods of sketches are proved to be more efficient than stroke-based approaches [3, 15]. In plane geometry, a proof consists of geometry figures and geometry proof scripts. By recognizing hand-drawn geometry figures and extracting the structure of handwritten proof scripts, structured manipulation can be supported. We designed pen gestures for structure-based selecting, deleting, moving, cutting, and pasting operations in handwritten geometry proofs. These structure-based methods allow the

operations of handwritten geometry proof scripts at the granularity of proof step.

For example, with structure-based manipulation, deleting a geometric component in geometry figures can also lead to the deletion of related proof scripts. Figure 8 demonstrates an example of deleting a line in geometry figures. Figure 8(a) shows the selected line (in red color), its corresponding geometry proof scripts, *DE* (in red color), and the deleting gesture (in green color). Figure 8(b) is the effect after the operation. Both the selected line and the related geometry proof steps are deleted.



(a) Deleting a line in the geometry figure



(b) Geometry figure and proof scripts after deleting

Figure 8. An example of deleting a line in geometry figure.

EVALUATION

With the proposed sketch understanding algorithm and the assistance technique in plane geometry proof, we built a pen-based geometry proving tool – PenProof (Figure 9).

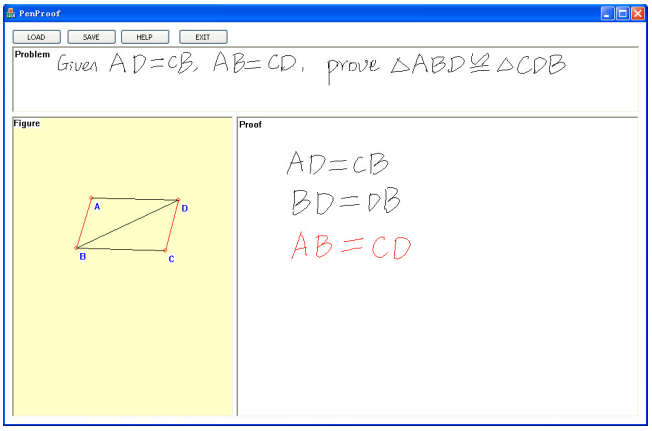


Figure 9. The user interface of PenProof.

In PenProof, there are three separate areas for defining geometry problems, drawing geometry figures, and writing geometry proofs. In the figure area, the hand-drawn figures are recognized into formal figures and dynamic geometry is supported based on MMP/Geometer [9]. In the proof area, handwritten proof scripts are recognized in the background to provide assistance to geometry proving.

We conducted a study to evaluate the PenProof system. The study had two goals: to test the recognition accuracy of our sketch understanding algorithm, and to obtain user feedback on the system.

Task, Subjects, Apparatus, and Procedure

The test task was to write down geometry proofs provided by us and did not require any knowledge beyond secondary schools, so we recruited twelve graduate students who possessed the required geometry knowledge. The test was on a machine equipped with a 2.4GH CPU, 4G memory and a Wacom screen.

To test the recognition accuracy of our sketch understanding algorithm, we provided each subject 10 figures and 10 proofs, and each subject was asked to select and finish 4 figures and 4 proofs in PenProof. The provided figures consisted of points, lines, and circles. The provided proofs consisted of proof steps and deduction symbols. However, the provided proofs did not include all the geometry symbols supported in this paper. An additional evaluation of the symbol recognition accuracy was conducted by asking each subject to draw all the geometry symbols twice.

After trying the PenProof tool, each subject was asked to answer a post-test questionnaire to grade the graph recognition accuracy, the structured interaction, the visual hints, the comfort, and the enjoyment of the tool, all in a 7-level Likert scale (1-vary bad, 7-very good).

Recognition accuracy

The evaluation results show that accuracy of hand-drawn figure recognition is 92.1%. Most errors were at the juncture of the graphs due to the warp of the points. Users could correct the errors either by manipulating the formal figures or by deleting and redrawing the figures. As for the proof recognition, the total recognition accuracy is 87.3%. The errors include the proof step identification errors, symbol segmentation errors, and symbol recognition errors. The errors of proof step identification were mainly caused by the misrecognition of the deduction symbols. Segmentation errors occurred because some users wrote adjacent symbols too close. Symbol recognition errors were due to irregularly written symbols.

As for the geometry symbol recognition, our SVM-based geometry symbol recognizer used 20 samples for each symbol to train the classifier. These samples were collected from one user and were written regularly. The evaluation results show that the recognition accuracy for the same person achieves 96.4%. When using the classifier to

recognize symbols collected from subjects, the average accuracy is 90.1%. The errors are caused by symbols written with irregular stroke orders or irregular symbol shapes.

User Feedback

Figure 10 exhibits the results of subjective evaluation. As shown, the recognition accuracy, the structured interaction, and the dynamic visual hints all received good feedback from subjects. In addition, the subjects thought it was comfortable and enjoyable to use the tool.

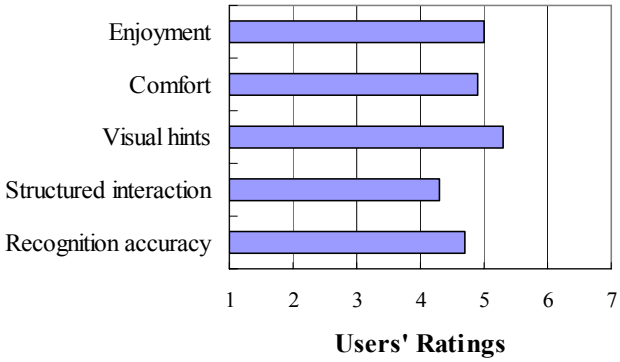


Figure 10. Users' ratings

While all subjects thought the tool was useful, they also offered some constructive suggestions. Currently, the letter labels for shape primitives are generated automatically by the PenProof system. Some users suggested that the tool allow a user-controlled labeling design. Some users pointed out that the tool should support more proving styles, such as proofs written in a two-column form.

DISCUSSION

As seen, the error rates of our current approach are relative low. The recognition accuracies of hand-drawn figures and hand-written proofs are 92.1% and 87.3%, respectively.

The most serious errors were related to incorrect deduction symbols. This may frustrate subjects when they used structured manipulation techniques and lead to relatively low ratings on structured manipulation compared with other ratings.

Figure 11 shows one error caused by incorrect recognition of deduction symbols. Here, a subject tried to select the item "AC//BD" and reuse it in a proof that followed. However, the structured selection that operates at the granularity of proof step selected "AC//BD}", instead of "AC//BD". This error was because the deduction symbol "}" was incorrectly recognized as "1" and consequently, led to incorrect identification of proof items.

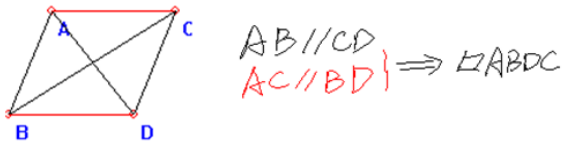


Figure 11. An error caused by incorrect recognition of deduction symbol.

In addition to these errors related to recognition algorithms, we also observed some errors produced by subjects when they wrote proofs inconsistent with the figures. The mismatch between drawn figures and written proofs can cause conflict interpretations. In this situation, subjects received no hint or even wrong hints. Figure 12 is an example of errors caused by conflicting figure and proof. Here, the proof step of "∠ABD=∠CDB" were written down, but "∠ABD" did not exist in the geometry figure. Thus, no visual hint in the geometry figure can be provided.

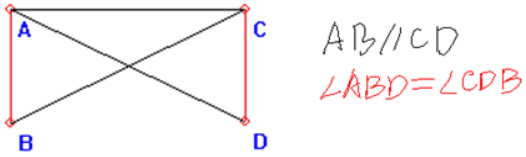


Figure 12. An error due to conflicting figure and proof.

To address these errors, we need to enhance our algorithms in several ways. First, the accuracy of deduction symbols should be further improved. Second, new algorithms are required to detect conflicting figures and proofs.

It should be noted that in our evaluation, we did not measure the influence of errors on performance quantitatively. This is because of the lack of comparable systems. We have not found any system that offers the same level of support for hand-written geometry proof. Thus, we focused on the evaluation of the design concept first, rather than quantified user performance.

CONCLUSION

This paper presented a pen-based geometry proving system. We proposed a sketch recognition algorithm to recognize and correlate hand-drawn geometry figures and hand-written geometry proof scripts. While the hand-drawn figure recognition is based on the detection of shape primitives using turning points, the hand-written geometry proof scripts recognition is based on the identification of proof steps. Based on the recognition results, we designed intelligent visual hints and structured manipulation technique to assist the understanding of geometry proving. Results from our evaluation study show that the algorithm

is effective and the assistance is useful. This pen-based geometry theorem proving approach has the potential to enhance the learning of geometry proofs by following the natural and traditional proving approach and meanwhile provide intelligent help to users.

The contribution of this research lies in two aspects. Technically, we have developed algorithms to establish the connections between hand-drawing and hand-written structures. While we applied our techniques in a geometry proving system, our algorithms can be expanded into other areas that have similar tasks.

Cognitively, the proposed "intelligent hint" design suggests a way to reduce cognitive loads in pen-based user interfaces by leveraging intelligent methods. In conventional user interfaces, such as WIMP-based designs, techniques to help reducing cognitive loads (e.g., highlighting relevant objects) are mature. However, objects in pen-based UI are usually not well-structured and well-recognized, so it is a challenge to use these techniques to assist users. Our approach shows that based on user action contexts and object correspondence, we can provide useful cues to reduce cognitive loads.

In the future, we are interested in extending our research in the following directions. First, we will enhance our recognition algorithms to address the limitation of our current algorithms. The current method could only recognize a limited set of symbols, and only support limited graph types and proving styles. In the future, we will try to recognize more characters besides current geometry symbols.

Second, we will explore other error correction strategies for PenProof. Currently, users are allowed to correct errors by erasing and rewriting the geometry proof. In the future, other error correction strategies, such as the multimodal methods, would be explored.

Third, we will extend our research into other areas. Although this research focuses on pen-based geometry theorem proving in plane geometry, pen-based proving techniques could be used in solid geometry theorem proving as well. In the future, we will support proving in solid geometry by recognizing and understanding hand-drawn solid geometry figures.

ACKNOWLEDGMENTS

This research is supported by National Key Basic Research and Development Program of China under Grant No. 2009CB320804, the National Natural Science Foundation of China under Grant No. U0735004, No.60603073, and the National High Technology Development Program of China under Grant No. 2007AA01Z158, No.2009AA01Z337.

REFERENCES

1. Alvarado, C. and Davis, R. (2001). Resolving ambiguities to create a natural sketch based interface. In *Proc. IJCAI 2001*, AAAI Press, 1365-1371.
2. Anderson, J.R., Boyle, C.F., Yost, G. (1985). The geometry tutor. In *Proc. IJCAI 1985*, 1–7.
3. Ao, X., Li, J.F., Wang, X.G. and Dai, G.Z. (2006). Structuralizing digital ink for efficient selection. In *Proc. IUI 2006*, ACM Press, 148-154.
4. Cabri Geometry. http://www.cabri.com/
5. Chazan, D. (1993). High school geometry students' justification for their views of empirical evidence and mathematical proof. *Educational Studies in Mathematics*, 24(4), 359-387.
6. Chang, C.C. and Lin, C.J. (2001). LIBSVM: a library for support vector machines, 2001. http://www.csie.ntu.edu.tw/_cjlin/libsvm.
7. Chou, S.C., Gao, X.S. and Zhang, J.Z. (1996). Automated Generation of Readable Proofs with Geometric Invariants. *J. Autom. Reasoning.* 17(3), 349-370.
8. Cormen, T.H., Leiserson, C.E., Rivest, R.L. and Stein, C. (2001). *Introduction to Algorithms, 2nd ed*. MIT Press, Cambridge, USA, 2001.
9. Gao, X.S. and Lin, Q. (2002). MMP/Geometer - a software package for automated geometry reasoning. In *Proceedings of ADG 2002*, Springer-Verlag, 44-46.
10. GeoAssistor: A Pen-based Geometry Learning Tool for Students. http://research.microsoft.com/en-us/um/beijing/projects/research/education/penbased.aspx
11. Geometer's Sketchpad. http://www.dynamicgeometry.com/
12. Geometry Explorer. http://homepages.gac.edu/~hvidsten/explorer/
13. Hanna, G. (1998). Proof as understanding in geometry. *Focus on Learning Problems in Mathematics*. 20(2&3), 4-13.
14. Hoyles, C. and Healy, L. (1999). Linking informal argumentation with formal proof through computer-integrated teaching experiences. In *Proceedings of the 23rd conference of the international group for the psychology of mathematics education*, 1999, 105-112.
15. Jiang, Y.Y., Tian, F., Wang, X.G., Zhang, X.L., Dai, G.Z. and Wang, H.A. (2009). Structuring and manipulating hand-drawn concept maps. In *Proc. IUI 2009*, ACM Press, 457-462.
16. Kara L.B. (2004). *Automatic parsing and recognition of hand-drawn sketches for pen-based computer interfaces*. Doctor's dissertation, Department of Mechanical Engineering, Carnegie Mellon University, Pittsburgh, PA, USA, 2004.
17. Keshari, B. and Watt, S.M. (2007). Hybrid mathematical symbol recognition using support vector machines. In

Proc. ICDAR 2007, IEEE Computer Society Press, 859-863.

18. Knuth, E.J. (2002). Teachers' conception of proof in the context of secondary school mathematics. *Journal of Mathematics Teacher Education*, 5(1), 61-88.
19. Kortenkamp, U. and Richter-Gebert, J. (2004). Using automatic theorem proving to improve the usability of geometry software. In *Proceedings of the Mathematical User Interfaces Workshop 2004*, 2004.
20. LaViola, J. and Zeleznik, R. (2004). MathPad2: A System for the Creation and Exploration of Mathematical Sketches. *ACM Transactions on Graphics*. 23(3), 432-440.
21. Li, J.F., Zhang, X.W., Ao, X. and Dai, G.Z. (2005). Sketch recognition with continuous feedback based on incremental intention extraction. In *Proc. IUI 2005*, ACM Press, 145-150.
22. Liu, Y.Y., Lin, Q. and Dai, G.Z. (2007). PIGP: A Pen-Based Intelligent Dynamic Lecture System for Geometry Teaching. In *Proc. Edutainment 2007*. Springer Berlin / Heidelberg, 381-390.
23. Narboux, J. (2007). A graphical user interface for formal proofs in geometry. *Journal of Automated Reasoning*. 39(2), 161-180.
24. Ouyang, T.Y. and Davis, R. (2007). Recognition of Hand Drawn Chemical Diagrams. In *Proc. AAAI 2007*, AAAI Press, 846-851.
25. Paulson, B. and Hammond, T. (2008). PaleoSketch: accurate primitive sketch recognition and beautification. In *Proc. IUI 2008*, ACM Press, 1-10.
26. Senk, S.L. (1985). How well do students write geometry proofs? *The mathematics teacher*. 78(6), 448-456.
27. Tanaka, H., Nakajima, K., Ishigaki, K., Akiyama, K. and Nakagawa, M. (1999). Hybrid pen-input character recognition system based on integration of online-offline recognition. In *Proc. ICDAR 1999*, ACM Press, 209–212.
28. Wang, X., Shi, G.S. and Yang, J.F. (2009). The understanding and structure analyzing for online handwritten chemical formulas, In *Proc. ICDAR 2009*, IEEE Computer Society Press, 1056-1060.
29. Wong, W.K., Chan, B.Y. and Yin, S.K. (2005). A Dynamic Geometry Environment for Learning Theorem Proving. In *Proceedings of the 5th IEEE International Conference on Advanced Learning Technologies (ICALT) 2005,* IEEE Computer Society Press, 15-17.
30. Yang, H.H., Wong, W.K. and Chan, B.Y. (2006) Using computer-assisted instruction for the visualization of proof tree to improve the reading comprehension of geometry proofs. In *International Workshop on Human-Computer Interaction and Learning Technologies 2006*, 1431-1436.