Analysis of Stroke Intersection for Overlapping PGF Elements

Yan Chen, Xiaoqing Lu
Institute of Computer Science & Technology
Peking University
State Key Laboratory of Digital Publishing Technology
Beijing, China
{cysmiling, lvxiaoqing}@pku.edu.cn

Jingwei Qu, Zhi Tang
Institute of Computer Science & Technology
Peking University
State Key Laboratory of Digital Publishing Technology
Beijing, China
{qujingwei, tangzhi}@pku.edu.cn

***Abstract*—Query-by-figure is an effective retrieval approach for educational documents. However, complex geometric diagrams in the field of mathematics education remain as obstacles in current retrieval systems. This study aims to explore a query method for plane geometric figures (PGFs) via sketched figures on smart mobile devices. We adopt an undirected graph model to describe PGFs and a divide-and-conquer strategy to analyze the relationships among strokes. Our main contribution is the detailed analysis of the stroke intersection that frequently occurs in PGFs. Numerous accurate elements obtained through overlapping analysis are then selected to construct strong descriptors for PGFs. Only the compressed query features, instead of a query figure, are transmitted to an image-based retrieval system located on a remote server, where the sketched PGF is finally recognized with low delay response. The experiments show that the proposed method achieves high efficiency and provides users with good interactive experience.**

Keywords—sketch recognition; plane geometric; mobile application; shape retrieval

I. INTRODUCTION

The advent of touch screen technology in many new equipment, such as smartphones, tablet PCs, smartwatches, and electronic whiteboards, brings numerous unprecedented opportunities for user interfaces, including directly inputting characters and graphs via sketching. Benefiting from this convenient input manner, sketch-based retrieval has become increasingly popular. Apart from clicking on keyboards or taking pictures, ordinary users can obtain information by drawing on touch screens, particularly for graph contents.

Plane geometric figures (PGFs) are widely used contents in mathematics education. As shown in Fig. 1, a plane geometric problem is consist of a PGF and a textural description of the PGF. In addition to keyword-based retrieval approaches, people have become increasingly interested in recognizing PGFs and in searching for related contents based on graph analysis. However, PGFs are mainly composed of points and lines, thus extracting the features of ordinary images, such as color, texture, and feature points, from PGFs is difficult. Consequently, existing image-based PGF retrieval systems frequently fail to obtain highly accurate results. However, the human visual system can easily perceive middle-level elements in PGFs, such as circles, triangles, and rectangles. This phenomenon has inspired us to explore an effective means to extract high-quality features for PGF recognition and retrieval, to detect elements or basic shapes, and to analyze topological properties and positional relationships in a layout, which provides a deep understanding of PGFs.

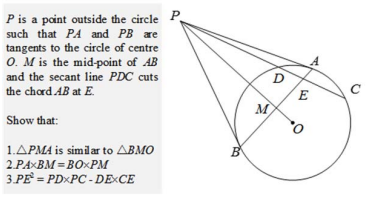

Fig. 1. Example of a plane geometric problem in mathematic textbooks.

Although numerous established methods for obtaining isolated geometric elements are already available, the sketch-based approach for PGF retrieval is still facing several formidable challenges. First, the relationships among visual or meaningful elements are more complicated than those in ordinary diagrams. For example, most symbols in a flowchart are adjoined, therefore traditional methods establish the relationships among symbols mainly depend on detecting connection points. However, nearly all PGF elements share strokes; that is, they overlap or are even embedded. Reasonably separating all the elements drawn consecutively stroke-by-stroke is a crucial problem. Second, a slight offset between the beautified elements and the original sketch position is acceptable in most existing systems. In PGFs, however, maintaining the accurate positions of overlapping elements is necessary to present and analyze the relationships among them. Third, the result of stroke intersection processing should be consistent with user intention. A tough balance should be considered to obtain effective PGF descriptors. The maximum number of elements is required for further recognition, including some elements generated with intersection strokes, while redundant elements should be screened.

The system proposed in this study comprises two subsystems: client-side and server-side subsystems. The client-side subsystem, which operates on mobile devices, records drawing lines, as well as performs beautification, shape reconstruction, and feature extraction. The server-side subsystem is an image-based retrieval system that maintains a PGF database and implements a retrieval workflow, including image preprocessing, detection of basic graph elements, description of element features and relationships, and matching of algorithms to evaluate similarity among PGFs. The comprehensive features of PGFs are extracted and described by the client-side subsystem,

978-1-5090-1792-8/16 $31.00 © 2016 IEEE
DOI 10.1109/DAS.2016.11

so that the server-side is relieved to focus on comparison and matching process.

The rest of this paper is organized into the following sections. Section II provides a brief review of related works. Section III explains the overall system architecture. Section IV describes the approach of the sketch process in detail. Section V reports the performance evaluation of our system based on a few test examples and comparisons with other similar systems. Finally, Section 6 discusses the conclusions drawn from the study and future perspectives.

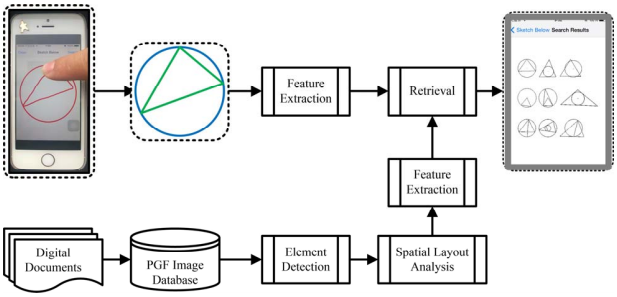

Fig. 2. Workflow of the proposed system. 1) The inputted sketch is beautyfied in real time. 2) Basic graph elements are detected and are converted into a feature vector. 3) The retrieval is performed on the server. 4) The results are returned for client-side.

II. Related Work

A. Graph Retrieval

The graph recognition and retrieval problem, such as that in electronic circuits [1], floor plans [2], and symbols [3], has gradually attracted considerable attention. PGFs are widely used in the field of mathematics education. Retrieving PGFs has recently caught the interests of many researchers. Feng et al. [4] proposed a structure analysis method to find distinct compound shapes and describe overlapping PGFs. Liu et al. [5] proposed a new shape descriptor, namely, the bag of shapes, to retrieve PGFs. Seo et al. [6] presented a diagram-understanding method that identifies the visual elements in a diagram while maximizing agreement between textual and visual data.

B. Basic Shape Detection

Detecting basic shapes, including triangle, regular quadrilateral, and circle, is the foundation for understanding PGFs. Most triangle detection methods are applied in traffic sign detection [7–9]. Circles detectors [10-12] are used for natural images and document graphs. Rectangle detection methods can be classified into parallel line-based [13], primitive-based [14], and Hough/Radon transform-based techniques [15].

C. Sketch Recognition

Various approaches for sketch recognition have been recently presented. These approaches differed mostly in their levels of domain specificity, extensibility, and hierarchical quality of recognition. Sciascio et al.[16] regarded a sketch as an image and proposed a spatial layout representation combining image features with a description of spatial layout information. Ding et al.[17] proposed an on-line sketch recognition algorithm for composite shapes, which separated a stroke into several shapes based on key point detection. Sousa et al.[18] adopted a graph-based technique to describe the spatial arrangement of drawing components and to encode topological relationships for vector drawings. Jayaraman et al. [19] proposed an interactive method for recognizing and vectorizing hand-drawn strokes in front of a webcam. Bresler et al. [20] presented a system utilizing a text/non-text separation approach to classify strokes and then separately recognized uniform symbols and arrows. Numerous methods that focus on sketched symbol recognition have also been proposed. Deufemia et al. [21] considered the geometric relationships among strokes and presented a two-stage methodology for sketched symbol recognition using latent-dynamic conditional random field and distance-based clustering. For sketched symbol recognition on mobile devices, Herold and Stahovich [22] presented a machine-learning approach called ClassySeg for automatic stroke segmentation. This approach selected points with curvature maxima and their neighboring points as candidate windows. Jorge and Fonseca [23] presented a method to recognize a simple vocabulary of geometry shapes with gestures for selection and deletion.

D. Plane Geometry Applications

There are a lot of interactive applications focus on graph recognition, such as Lucidchart[24], AppSeed[25] and Live Trace. Lucidechart provides a method for building diagrams on various devices. AppSeed makes use of vision to parse graphical components drawn on a white board. Live Trace in Adobe Illustrator transforms sketches into vector graphics. Also there are several popular plane geometry applications, such as Sketchometry [26], GeoGebra [27], Live Geometry [28], Cinderella [29], Dr. Geo [30], and Geometry Expressions [31]. Sketchometry is an interactive construction and exploration tool for plane geometry, which instantly converts the hand drawings of users into geometric constructions that can be modified and moved around. GeoGebra [24] is an interactive geometry, algebra, statistics, and calculus application that is intended for learning and teaching mathematics. Constructions can be made using points, vectors, segments, lines, and polygons.

III. System Architecture and Workflow

The proposed system comprises two parts: client-side and server-side systems. As shown in Fig. 2, the client-side subsystem aims to obtain input information including tracking user input with point coordinates and time stamps, beautifying strokes in real time, decomposing intersecting strokes, and detecting basic graph elements. After extracting the important features of a sketch, a feature descriptor is built and sent to the server-side subsystem.

In the server-side subsystem, we analyze all the PGFs in our database, compute for their corresponding feature vectors in advance, and address the following problems to respond to the client side request.

A. Basic element detection

Identifying the basic elements in a figure as many as possible is important to recognize the entire PGF further. Triangles are detected via key points and edge tracking based on a geometric restriction [5]. For circle detection, we use the randomized circle detection method presented by Chung et al. [11]. We detect regular quadrilaterals through the parallelism analysis of their critical components, namely, parallel line pairs. [13]

B. Analyzing spatial relationships

Many PGFs contain similar or even the same type or number of elements; however, the various layouts of these elements make them appear considerably different. Therefore, exploring and describing the spatial relationships among elements are also important.

C. Computing similarity

Numerous features can be extracted from the PGFs in the database, such as single geometric primitive histograms, dual-element structure binary features, main primitive features, global features, and boundary curvature (scaled). A bag-of-shapes PGF descriptor is adopted to build feature vectors. Then, cosine similarity is used to measure the relevance between a PGF query and a candidate PGF based on their feature vectors and to output the most similar PGFs.

The aforementioned problems only occur in the tasks in the server-side subsystem. Additional problems and details of the solutions can be found in our previous studies [4,5].

IV. Sketch Analysis

To recognize sketch-based PGFs, the client-side subsystem should detect all isolated and combined elements and construct feature vectors for further retrieval. We adopt an effective divide-and-conquer strategy to analyze each stroke immediately after it is inputted rather than until all strokes of a PGF have been completed. In this section, we first introduce the basic model for sketch processing. We then explain the solutions to the special problems of PGF recognition. Finally, we briefly describe the beautification method and the construction of feature vectors.

A. Graph Model

An undirected graph model G = (V, E) is established to record stroke information. In this model, the vertices denote the multipurpose snap points (MS-Points), and the edges denote all existing strokes. MS-Points denote the starting, ending, and junction points of a stroke. Strokes can be classified into two categories: associated and isolated strokes. An associated stroke connects or intersects with at least one other stroke, whereas an isolated stroke does not. Each stroke is assigned with an unique edge ID, whereas, the IDs of MS-Points may change in our model because they are dynamically generated, merged, or deleted when new strokes snap to or cross over existing strokes.

B. Adaptive Template for Recognizing Partly Constrained Elements

The recognition of isolated elements drawn with one stroke has been studied comprehensively by many researchers. However, two obstacles lead to the failure of current methods in recognizing sketched PGFs.

First, an element is frequently completed with more than one stroke. We define two states, namely, opening and completion, to distinguish whether an element is completed. In the opening state, although one or more strokes are inputted, the element that the user intends to draw is still incomplete and its type cannot be identified based on existing strokes. By contrast, a closed graph is formed in the completion state. The completion state can be determined through the connection of MS-Points. In the completion state, we use a depth-first search algorithm to obtain possible closed sub-graphs among all possible combinations of existing strokes. Then a template-based recognizer is adopted to determine whether a closed graph is an element, as shown in Fig. 3(a). The element recognizer evaluates the similarity between all templates and the point set of candidate strokes. If one matched template is found, then a new element is established.

The second problem involves the relationship between a newly added element and existing elements. Establish this relationship correctly is not easy, because ordinary people cannot draw every element at an accurate location (shown in the first column of Fig. 3(b)). If this relationship is ignored, as most popular methods do, then the input results (shown in the second column of Fig. 3(b)) may inaccurately reflect the intention of the user. In summary, apart from identifying the correct type of a newly added element, the correct positional relationships between this element and existing elements must also be established.

Therefore, we explore a new approach for adaptive templates to obtain accurate types and layouts of elements. To establish the appropriate relationship, we must first obtain the MS-Points of newly added strokes before any beautification or

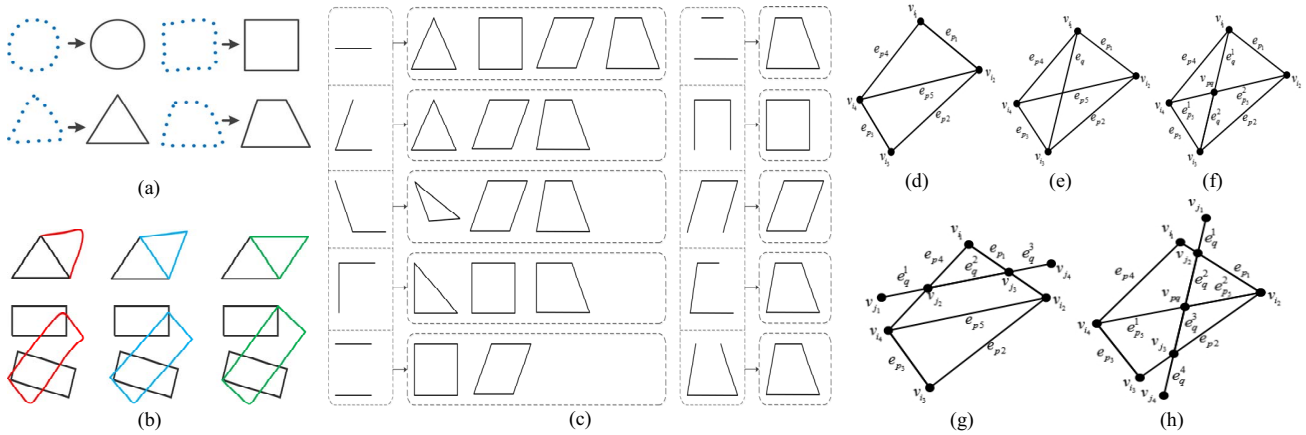

Fig. 3. (a) Template-based recognizer simply maps sketch points into uniform templates. (b) Ordinary people draw element at unaccurate location as shown in the first column. Current systems recognize them ignoring the relationship between a newly added element and existing elements as shown in the second column, while the satisfactory results are obtained with the proposed algorithm as shown in the third column.. (c) The reverse index for tempaltes. (d)-(h) Examples of intersection analysis.

recognition operation performed. In most PGFs that users intend to draw, elements exhibit strong relations, such as sharing a vertex, connecting at a crossing point, and intersecting with a specific angle. Identifying the nearest MS-Points is essential to understand positional relationships. If one or more nodes of new strokes are determined to share the same position with existing MS-Points within a predefined threshold, then we cannot use the template freely to recognize the isolated element. Moreover, some parts of the potential elements are restricted to their position or layout, and thus, we must use adaptive templates to recognize them.

The adaptive templates used in this study are constructed as follows. The original templates are divided into several components according to the number of edges. The analysis of the similarity among all components is performed to screen duplicate ones. A reverse component index is created for all templates, as shown in Fig. 3(c).

Therefore, the process of recognizing constrained elements is divided into two stages. The first stage aims to identify the components of a template for the parts of the new set of strokes that consist of sharing points. In the second stage, the number of candidate templates is narrowed down to a small scale based on the selection with the reverse component index. Template matching for the rest of the parts is performed to determine the final element type among the candidate templates. In this way, accurate results reflecting users' intention can be obtained, as shown in the third column of Fig. 3(b).

C. Intersection and Derived Elements

Unlike existing methods for recognizing an isolated element, the proposed model for PGF recognition can obtain the derived elements generated by stroke intersections.

Without losing generality, Fig. 3(d)-(h) shows a typical intersection analysis scenario with the graph model G = (V, E). We suppose that v_{i_1}, v_{i_2}, v_{i_3},v_{i_4}, e_{p_1}, e_{p_2}, e_{p_3}, e_{p_4}, and e_{p_5}, as shown in Fig. 3(d), are existing vertices and edges, as well as one detected quadrilateral and two detected triangles, namely, $\triangle v_{i_1} v_{i_2} v_{i_4}$ and $\triangle v_{i_2} v_{i_3} v_{i_4}$. A new stroke is sketched from v_{i_1}, to v_{i_3} [Fig. 3(e)], and its starting and ending points are snapped to v_{i_1} and v_{i_3}, respectively. First, a new edge e_q is added to G, and two triangles, namely, $\triangle v_{i_1} v_{i_2} v_{i_3}$ and $\triangle v_{i_1} v_{i_3} v_{i_4}$, are detected. Second, given that e_q and e_{p_5} intersect at point v_{pq}, numerous elements are formed, including four separate strokes (i.e., $e_{p_5}^1$, $e_{p_5}^2$, e_q^1, and e_q^2) and four subtriangles i.e., $\triangle v_{i_1} v_{pq} v_{i_2}$, $\triangle v_{i_2} v_{pq} v_{i_3}$, $\triangle v_{i_3} v_{pq} v_{i_4}$, and $\triangle v_{i_4} v_{pq} v_{i_1}$) [Fig. 3(f)]. All these triangles are stored in the element list for the following analysis. Fig. 3(g) and 3(h) show more complicated cases wherein a stroke intersects with multiple existing strokes.

In our current system, when a stroke is recognized as an arc, this stroke will be treated as an isolated element, and it will be disregarded in analyzing overlapping and intersecting strokes.

D. Sketch Beautification

To support our algorithm for recognizing sketch-based PGFs, we process a real-time beautification for the input sketch. This procedure is achieved by synchronously replacing the original irregular point sets with the beautified elements. We define sketch beautification as the process of normalizing input shapes according to the recognition results from the original input points. We improve the approach presented by Vatavu et al. [32] by adding the information on the positions and sizes of the input point sets to obtain accurate beautification results.

E. Element Selection and Feature Vector Construction

By performing the aforementioned processing of sketched PGFs, we obtain numerous elements for further recognition. However, not all of these elements are used to represent a PGF. The following steps are adopted to select valuable elements. Large isolated elements or arcs are selected according to the rank of their relative areas. A group of large overlapping elements, which contain all the vertex points in a PGF and share the smallest overlapping areas, is selected. Subsequently, we extract the features from these elements and their spatial relationships to build feature vectors for the sketched PGF. The feature vector comprises three types of features: single geometric primitive histograms, main primitive features, and global features (more details can found in our previous study [5]).The client-side subsystem transfers the vector to the server-side subsystem for further recognition and provides users with the retrieval result.

V. Experiment

To demonstrate that our system provides good beautification and robustness, we sketch several PGFs using different methods and PGFs with overlapping elements. Then, several examples of sketched PGFs are processed for the retrieval test, including those with different PGFs in our database. Furthermore, we compare our system with several popular geometric applications in terms of certain important aspects, including supporting a sketch. We demonstrate that our system has several superior distinguishing features compared with other geometric systems. Moreover, our system exhibits satisfactory beautification performance and retrieval results for the PGFs.

A. Implementation

Our system is based on the server-side and client-side subsystems. In the server-side subsystem, we implement and run our retrieval system on a server with a 2.13 GHz Intel Xeon E5506 CPU and an 8 GB memory using MATLAB R2013a. In the client-side subsystem, we use a MacBook Air with a 2.00 GHz Intel Core i7 CPU and an 8 GB memory to develop our sketch system using Xcode 6.2 and then operate it on an iPhone with iOS 7.

Considering that no authorized database for PGF evaluation is available, we build a database with 267 black/white PGF images extracted from digital PDF documents. Size ranges from 60 × 96 to 400 × 96. Then, we apply the database to estimate the performance of our system.

B. Evaluation: Sketch with one or multiple strokes

We evaluate the performance of our system by sketching several typical PGFs using different methods as shown in Fig. 4. We sketch a circle using only one stroke (Fig. 4(a)), and using two strokes, each for one semi-circle (Fig. 4(b)). We separately sketch rectangles with one, two, three, and four strokes, as the five methods shown in Fig. 4(c)-(g). Given that the two typical elements can be inputted using different sketching methods, our

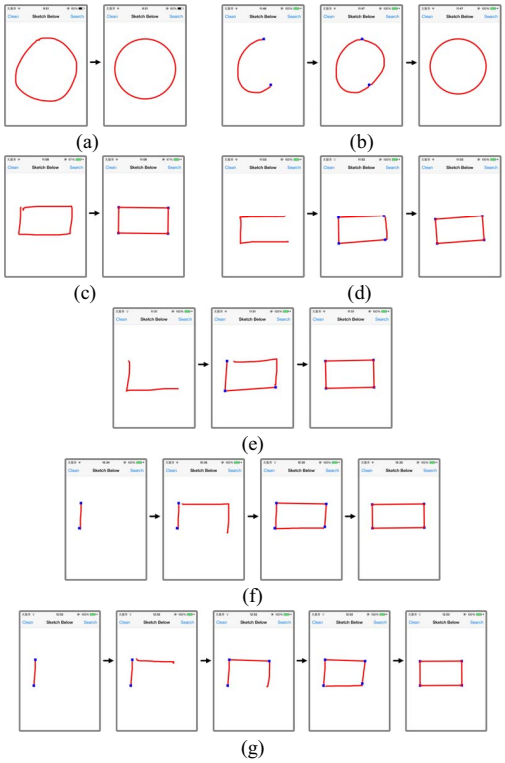

Fig. 4. Two typical examples, namely, a circle(a-b) and a rectangle(c-g) to evaluate the sketch method with one or multiple strokes.

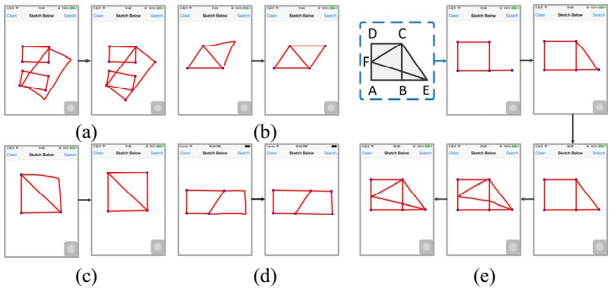

Fig. 5. Examples of snapping points(a), combination analysis(b-d), and recombination lines(e).

system can recognize a sketched PGF regardless of the number of strokes.

C. Evaluation：Adaptive beautification

In general, a PGF consists of several elements. Some of these elements are drawn with one or multiple strokes, whereas others are derived through intersections or combinations related to the previous strokes. A robust system explores all possible new elements based on each sketch step, as well as adjusts the relationship between the added element and existing elements. Such strategy does not only satisfy user intention but also enhance their experience. The example shown in Fig. 5 illustrates three benefits of the proposed adaptive beautification approach, which are discussed in the following paragraphs.

1) Accurate Point Snapping

As shown in Fig. 5(a), when a user attempts to create a third rectangle on a background that already contains two rectangles, he/she intends to reuse the vertices of the two existing rectangles. Ordinary users generally cannot draw the third rectangle at an accurate location. To solve this problem, our system first identifies the nearest vertices of the existing elements within the predefined threshold. Then, it uses the adaptive template to recognize elements and perform beautification.

2) Splicing a Large Element

After a user isolates an element, a large element can be established from the combination of the new element with one or more existing elements. Analyzing only one element for each new added stroke is insufficient to solve this problem. In our system, as shown in Fig. 5(b)-(c), after a second triangle is drawn, a regular quadrilateral is formed when the new triangle combines with an existing triangle. In Fig. 5(d), after a second trapezoid is drawn, a parallelogram is formed when the new trapezoid combines with an existing trapezoid.

3) Recombination Lines

According to the consistency of line direction, existing lines can combine to form a new line. As shown in Fig. 5(e), when line BE is added, a new line AE can have nearly the same direction as lines AB and BE. Then, line AE can be used to construct the new trapezoid AECD. Similarly, after adding the stoke CFE, the recombined line AE also becomes an edge of the derived triangle AEF.

D. Evaluation: Intersected Strokes and Retrieval Performance

Numerous sketched PGFs with intersecting, embedded, and overlapping strokes are tested. The results of the relationship analysis can be found in the retrieval performance of our system. Two classical types of PGF image are available in our experiment. Some of these images have significantly similar PGFs in our database, whereas others do not.

1) Type I. First, we sketch a circle and a triangle on the screen of an iPhone. A line segment is then added in the triangle. Therefore, the sketch is composed of a circle and three triangles. As shown in Fig. 6(a), a PGF image that is nearly the same as the sketched figure is found in the database and returned as the best option(blue rectangle). Furthermore, the second image also exhibits high similarity with the query. Moreover, the rest of the reasults have the same number of elements.

2) Type II. We input a sketch that is composed of a circle and two triangles, as shown in Fig. 6(b). Given that this query has not completely similar PGFs in our database, the system returns the most possible similar results. Notably, most PGFs that consist of a circle and two triangles are included among the results (blue rectangle), which belong to the top two. Logically, the rest of the results also exhibit certain similarities with the query.

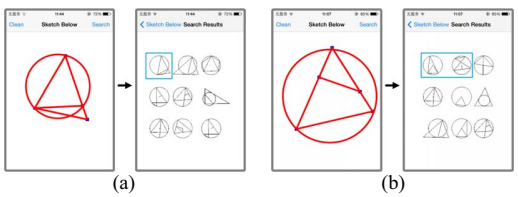

Fig. 6. Examples of intersected strokes and retrieval performance.

E. Evaluation: Comparison Results

We evaluate the performance of our system by comparing it with six popular geometric applications, as shown in TABLE I.

TABLE I. COMPARISON RESULTS WITH OTHER GEOMETRIC SYSTEMS

System	Sketch	Single-Stroke	Multi-Stroke
Sketchometetry	Y	Y	N
GeoGebra	Y	Y	N
Live Geometry	N	Y	N
Cinderella	N	Y	N
Dr. Geo	Y	Y	N
Geometry Expressions	N	Y	N
Proposed System	**Y**	**Y**	**Y**

VI. CONCLUSION AND FUTURE WORK

This study presents an interactive sketch-based retrieval system for PGFs that aims at document retrieval. The system consists of two subsystems: client-side and server-side subsystems. The client-side subsystem is implemented on an iPhone. With the beautification of sketched PGFs, we propose an approach to recognize PGFs based on the analysis of overlapping elements. With regard to the server-side system, we compare a query PGF with the candidate PGFs in our database through the feature vector provided by clients. The experiments conducted on an iPhone present satisfying beautification, robustness, and retrieval results. The comparison results with other popular plane geometric applications also demonstrate the good flexibility and user-friendliness of our system. In the future, we will optimize the selection algorithm of the candidate-derived elements, particularly for complicated PGFs. Furthermore, automatic adjustments will be considered to add elements to one PGF dynamically.

VII. ACKNOWLEDGEMENTS

This work is supported by the projects of National Natural Science Foundation of China (No. 61472014 and No. 61573028), the Natural Science Foundation of Beijing (No. 4142023) and the Beijing Nova Program(XX2015B010). We also thank the anonymous reviewers for their valuable comments.

REFERENCES

[1] P. De, S. Mandal, A. K. Das, and B. Chanda, "Detection of electrical circuit elements from documents images," Proc. SPIE 9402, Document Recognition and Retrieval XXII, 94020O, 2015.

[2] L.-P. D. L. Heras, O. R. Terrades, and J. Lladós, "Attributed Graph Grammar for Floor Plan Analysis," 2015 13th International Conference on Document Analysis and Recognition (ICDAR),2015.

[3] A. Boumaiza, and S. Tabbone, "Symbol Recognition Using a Galois Lattice of Frequent Graphical Patterns," 2012 10th IAPR International Workshop on Document Analysis Systems(DAS), pp.165-169, 2012.

[4] T. Feng, X. Lu, L. Liu, K. Li and Z. Tang, "Structure analysis for plane geometry figures," Proc. SPIE. 9021, Document Recognition and Retrieval XXI, 90210R, 2013.

[5] X. Lu, K. Li, J. Qu, L. Gao and Z. Tang, "Plane Geometry Figure Retrieval with Bag of Shapes," 2014 11th IAPR International Workshop on Document Analysis Systems (DAS), pp.1-5, April 2014.

[6] M. J. Seo, H. Hajishirzi, A. Farhadi and O. Etzioni, "Diagram Understanding in Geometry Questions," AAAI Conference on Artificial Intelligence, North America, January 2014.

[7] M. á. García-Garrido, M. á. Sotelo and E. Martín-Gorostiza, "Fast Road Sign Detection Using Hough Transform for Assisted Driving of Road Vehicles," Computer Aided Systems Theory - EUROCAST 2005, Springer Berlin Heidelberg, pp. 543-548, 2005.

[8] J. He and Y. Ma, "Triangle detection based on windowed Hough Transform," 2009 International Conference on Wavelet Analysis and Pattern Recognition (ICWAPR), pp. 95-100, 2009.

[9] W. Kuo and C. Lin, "Two-Stage Road Sign Detection and Recognition," 2007 IEEE International Conference on Multimedia and Expo, pp. 1427-1430, 2007.

[10] C. Akinlar and C. Topal, "EDCircles: A real-time circle detector with a false detection control," Pattern Recognition, vol. 46(3), 2013, pp. 725-740.

[11] K. Chung, Y. Huang, S. Shen, A. S. Krylov, D. V. Yurin and E. V. Semeikina, "Efficient sampling strategy and refinement strategy for randomized circle detection," Pattern Recognition, vol. 45(1), January 2012, pp. 252–263.

[12] D. Liu, Y. Wang, Z. Tang and X. Lu, "A robust circle detection algorithm based on top-down least-square fitting analysis," Computers and Electrical Engineering, vol. 40, 2014, pp. 1415–1428

[13] K. Li, X. Lu, H. Ling, L. Liu, T. Feng and Z. Tang ,"Detection of Overlapped Quadrangles in Plane Geometric Figures," 2013 12th International Conference on Document Analysis and Recognition (ICDAR), pp. 260-264, 2013.

[14] Y. Liu, T. Ikenaga and S. Goto, "An MRF model- based approach to the detection of rectangular shape objects in color images," Signal Processing, vol. 87(11), November 2007, pp. 2649–2658.

[15] H. Bhaskar, N. Werghi and S. Al-Mansoori, "Combined spatial and transform domain analysis for rectangle detection," 2010 13th Conference on Information Fusion (FUSION), IEEE,2010,pp. 26-29.

[16] E. D. Sciascio, F.M. Donini and M. Mogiello, "Spatial layout representation for query-by-sketch content-based image retrieval," Pattern Recognition Letters, vol. 23(13), 2002, pp. 1599-1612

[17] Z. Ding, Y. Zhang, W. Peng, X. Ye and H. Hu, "An On-line Sketch Recognition Algorithm for Composite Shape," Fuzzy Systems and Knowledge Discovery, 2005, pp. 120-129

[18] P. Sousa and M. J. Fonseca, "Sketch-based retrieval of drawings using spatial proximity," Journal of Visual Languages & Computing, vol. 21(2), April 2010, pp. 69–80

[19] P. K. Jayaraman and C. Fu, "Interactive Line Drawing Recognition and Vectorization with Commodity Camera," Proceedings of the ACM International Conference on Multimedia, ACM, 2014.

[20] M. Bresler, T. V. Phan, D. Prusa, M. Nakagawa and V. Hlavác, "Recognition System for On-Line Sketched Diagrams," 2014 14th International Conference on Frontiers in Handwriting Recognition (ICFHR), pp. 563-568, 2014.

[21] V. Deufemia, M. Risi and G. Tortora, "Sketched symbol recognition using Latent-Dynamic Conditional Random Fields and distance-based clustering," Pattern Recognition vol. 47(3), March 2014, pp. 1159–1171.

[22] J. Herold and T. F. Stahovich, "A machine learning approach to automatic stroke segmentation," Computers & Graphics, vol. 38, 2014, pp. 357–364.

[23] J. A. Jorge and M. J. Fonseca, "A Simple Approach to Recognise Geometric Shapes Interactively," Computer Technology & Development , 2001, pp. 266-274

[24] https://www.lucidchart.com/

[25] http://appseed.ca/

[26] http://en.sketchometry.org

[27] https://www.geogebra.org

[28] https://livegeometry.codeplex.com

[29] http://www.cinderella.de

[30] http://www.drgeo.eu

[31] http://www.geometryexpressions.com

[32] R. Vatavu, L. Anthony and J. O. Wobbrock, "Gestures as point clouds: a $P recognizer for user interface prototypes," Proceedings of the ACM International Conference on Multimodal Interfaces (ICMI), pp. 273-280, 2012.