Graph-Based Locality-Sensitive Circuit Sketch Recognition

ABSTRACT
Pen-based interaction can visualize user’s input sketches, recognize and analyze handwritten characters and symbols with the help of computers, and return the result to user vividly. Thus, pen-based interaction is a favorable and convenient teaching approach in the field of education. Unfortunately, decided by the deficiency of visual nervous system in human brain, machines work differently from human in terms of vision. As a result, human handwriting is not apparent for machines as for people, especially when the global meaning of sketch’s content is complex. Typically, it is difficult for computers to divide users’ strokes into a series of components and detect or localize all the meaningful symbols. With the purpose of relieving this dilemma, we propose an effective approach in this work, which uses graph to model users’ input strokes for extracting graphic features of handwriting, takes advantage of cycles in the graph to detect some special symbols, and reasons the global implication of sketches through a variety of local areas. Afterwards, we apply our model to a practical problem in the area of education, i.e., recognition and analysis of hand-drawn circuit graph and build a complete system for sketch-based circuit graph learning. During this process, to tackle the problem of lacking training sample for building a reliable circuit graph elements classification module and make up for this insufficiency in sketch recognition field, we produce, collect, and publish a data set on hand drawn circuit components. Generally speaking, theoretical proof, extensive experimental study, and user survey have shown that our algorithm and system are efficient as well as stable with high value for application compared with the state-of-the-art method.

KEYWORDS
Pen Interaction, Graph, Circuit Sketch

1 INTRODUCTION
Pen interaction technology can visualize user’s hand-drawing, recognize symbols and characters on the tablet, and give a certain analysis to the sketch, and return the results back to users vividly with the assistance of computers. Thus, it allows users to simulate the real world with a stylus. For this advantage, pen interaction technology is a popular teaching method in the areas of education. In recent years, due to the development of the field of computer and human-computer interaction, more and more outstanding pen-interaction scientific research achievements have also emerged, including [12] and [14]. However, because of the lack of complex visual nervous system, human handwriting is not apparent for machines as for people. Therefore, it is difficult for computers to fully understand what human write or draw. In order to overcome this difficulty, a series of works have concentrated on presenting more optimized models to equip computers with artificial intelligence so as to handle human hand-drawing more appropriately. For example, Liu et al. [16] proposed Sketch-GAN for sketch coloring. Zhang et al. [28] proposed SketchNet to classify images on the Internet through sketches, so as to find a match between sketches and real images. There are also works focusing on recognition and analysis of sketches, e.g., processing the task with convolutional neural network (CNN) [21] [27].

Although there have been many achievements in the field of pen interaction, due to the architecture of computer itself, there is a fundamental difference between the way humans and machines work. Specifically, it is still challenging for computers to divide a series of input strokes into several components according to semantics correctly, detect the appearances of all the meaningful symbols, and localize them. Recently, with the rapid development of computer vision, humans have made great breakthroughs in the fields of image semantic segmentation [9] and object detection [11]. However, due to the great difference in image features and pixel distribution, models for real-world images cannot be applied to sketches directly. Besides, it is unrealistic to retrain these models without tons of training samples. Therefore, a lot of work need to be done in the area of sketch recognition. Most of the existing methods have serious restrictions on users’ handwriting, or cannot analyze users’ input strokes online, some of which are computationally expensive. Under this background, motivated by Chen et al. [4], who used graph to model users’ sketches so as to infer the global meaning of a image, we propose a graph-based locality-sensitive algorithm for detecting some special symbols in sketches. As shown in Fig. 1, the basic idea of our algorithm is to use vertex in graphs to model users’ input strokes, i.e., each vertex is corresponding to a stroke. And then, it utilizes cycles in local areas to detect target symbols. From our analysis and experiment, this strategy allows us to reason global meaning of the sketch efficiently.
In order to prove the effectiveness and reliability of our algorithm, we apply our algorithm into practice: pen-interaction circuit diagram recognition and analysis system, so as to facilitate electrical teaching and learning in the areas of education.

Circuit diagram is a basic and difficult point of electricity in physics. A qualified circuit simulation system should be user-friendly, i.e., users can input to the system conveniently and it should be vivid enough for students to learn electrical principle. At present, most of public circuit simulation systems require users to drag components to proper positions with mouse [2], which is relatively complex and tedious for users. Therefore, different from these systems, we build a pen-interaction circuit diagram recognition and analysis system with our graph-based locality-sensitive algorithm, which allows users to draw circuit diagrams on a tablet and is as convenient as drawing on paper.

Our pen-interaction circuit diagram recognition and analysis system consists with three modules: circuit component classifier module, sketch analysis module, and circuit state and attribute calculation module. Sketch analysis module is a direct application of our algorithm mentioned before. Circuit state and attribute calculation module takes advantage of the results in sketch analysis module to make a list of equations about unknown attributes and solve it to get the results. Note that although this calculation module is not directly to do with our main algorithm, aimed at helping students understand electrical laws better and equip the algorithm with higher practical value in the educational field, we also integrate this module into our system. As for building the circuit component classifier module, in order to address the problem of lacking training samples, we produce, collect, and publish a set of sketch circuit components by ourselves. Sufficient theoretical proof, intensive experiments, and user study have shown that our algorithm and system are stable and efficient. Compared with start-of-the-art methods, our algorithm has higher practical value.

Generally speaking, this paper has mainly three contributions:

1 proposes an efficient detection algorithm for special sketch symbols;
2 builds a complete, stable, and satisfying system for circuit diagram teaching and learning;
3 makes up for the deficiency of datasets in the areas of sketch recognition and circuit component classification.

The following parts will be organized as follows: section II introduces some related works about this paper; section III elaborates the models for our algorithm and system; section IV explains algorithms proposed in this paper in detail; section V gives a theoretical analysis to the correctness and time complexity of our algorithm; section VI shows our experimental study for evaluating the performance of our algorithm and system in practical.

2 RELATED WORK

We introduce some related works concentrating in symbol detection, application of graphs in computer vision, and sketch recognition areas.

In terms of symbol detection, Girshick et al. [9] combined method of region proposal with CNN to improve traditional object detection algorithm. Besides, there are a number of previous works that focused on object detection, e.g. [11] and [20]. However, their target symbols and scenes are both real-world images. A lot of work needs to be done if we want to retrain the models and apply them to the area of sketch. In addition, Xu et al. [26] proposed to use random Hough transfer to detect lines and circles in an image. Unfortunately, the effectiveness of this algorithm is limited when applied to sketches, where different users have different drawing habits and the noise or destabilization of sketch symbols are uncertain.

In terms of graphs, Lee et al. [15] proposed a graph-based hand-drawn symbol recognition method that modeled similarities between hand-drawn symbols and known symbols by calculating graph matching; Chen et al. [4] proposed a graph-based algorithms for inferring global information of images; I.Burak Özer et al. [17] used graphs to establish descriptions for digital images and videos. These works show that graphs have important applications in the field of computer vision, including hand-drawn recognition area. Inspired by this, the basic idea of this paper is similar to these works, which completes the detection and recognition of hand-drawn targets based on graphs.

In terms of sketch recognition, Chu et al. [5] enriched assessment feedback through sketch recognition and made it support Chinese character educational interfaces. There are also works directly concerning with circuit sketch recognition. Valois et al. [23] completed the identification and beautification of hand-drawn circuit diagrams by extracting the structure and topological relationship of the images; Gennari L et al. [8] combined the geometric features of the image and domain knowledge to explain circuit sketches. There are more or less user restrictions on hand drawing, e.g., users are not allowed to draw other parts of a circuit diagram before finishing the current component. Feng et al. [7] used two-dimensional dynamic programming to store the information of each state of each stage of the user’s drawing of the circuit diagram to identify the circuit diagram and relaxed this restriction for users. However, it puts high demands on time and memory since the algorithm needs to handle state transitions on two dimensions. Therefore, this
paper is dedicated to propose a new algorithm that can efficiently and accurately detect and localize sketch circuit components and infer global connect relationships of circuit diagrams.

3 MODEL PROPOSAL

For later discussion about our model, we list the notations used in this paper in Table 1.

### Basic Model

**Sketch:** Let us consider following online scene: users make their sketches on a digital canvas, and system reads the input in strokes, *i.e.*, each time when the stylus falls on the sketchpad and leaves it, we take the movement track of the nib during this process as an input unit. Every time when the system receives an input unit, it will send the corresponding stroke into a working queue, which is an independent thread. Considering that the output generated by the analysis process can disturb users’ drawing more or less, users are also free to set appropriate delay between the finish time of an input stroke and trigger time of the analysis process. Specially, if the delay is long enough, our system will be converted to offline mode. In other words, the system allows users to freely switch between online and offline modes. It totally depends on their drawing habits and whether they want to get feedback results of the system in real time.

After extracting users’ input strokes, we use graph to model them and their inner relationship. We adopt the following methods: Every input stroke will be viewed as a vertex in the graph. There will be an edge between two vertexes if and only if their corresponding strokes come into contact with each other, *i.e.*, there is a common pixel passed by both strokes on the screen. As shown in Fig. 1, every stroke in Fig. 1(a) corresponds to its own unique vertex in Fig. 1(b). In this way, the vertex set, edge, and edge set of the graph can be defined as:

\[ V \triangleq \{i|\forall s_i \in S\}, \]  
\[ e_{i,j} \triangleq t_{i,j}, \forall s_i, s_j \in S, \]  
\[ E \triangleq \{(i, j)|t_{i,j} > 0\}. \]

It is worth noting that \( E \) should be a multi-set since there can be more than one contacting point between two strokes, as shown in strokes with number 6 and 7 in Fig. 1(a). In addition, we do not have the constraint \( i \neq j \) in definition formulas 2 and 3, which means that we allow a stroke contacts itself to form a selfloop. As shown in stroke with number 4 in Fig. 1(a), the user sketches the border of the ammeter with only one stroke, with adjacent start point and end point forming a selfloop. Therefore, simple graph is not powerful enough to model our problem, so we use a more general graph with multiple edges and selfloops allowed.

**Architecture for Circuit Sketch Recognition System**

We take an practical case: pen-based circuit sketch teaching and learning system as an example, to deploy our special sketch symbol detection algorithm for realistic problems and
Circuit
we make up for the lack of sketch recognition data sets, es-
wire, power supply, light bulb, resistor, ammeter, and volt-
and publish a set of data for this task on our own, including
is even rarer. To address this problem, we produce, collect,
Specifically, the data set for hand drawn circuit components
sets for traditional and real-world image classification tasks.
sets published on the Internet are small, compared with data
the area of sketch recognition, the number of training data
lem for us is to collect a large enough training data set. In
train a reliable classifier for our classification task upon a
eral strokes passed from the sketch analysis module as input
lay, which can be set by the user. The circuit state and at-
tribute calculation module is called after the whole circuit
sketch is completed and it will return the calculation results
to the user. Without loss of generality, we uniformly stipu-
late that the sketch analysis module will be called immedi-
ately after the system receives an input stroke.

Circuit Component Classifier Module : The classi-
fier module is a CNN (Convolutional Neural Network) de-
coder. It takes the circuit component image formed by sev-
eral strokes passed from the sketch analysis module as input
and outputs its category. To build this module, we need to
train a reliable classifier for our classification task upon a
variety kinds of circuit components. One challenging prob-
lem for us is to collect a large enough training data set. In
the area of sketch recognition, the number of training data
sets published on the Internet are small, compared with data
sets for traditional and real-world image classification tasks.
Specifically, the data set for hand drawn circuit components
is even rarer. To address this problem, we produce, collect,
and publish a set of data for this task on our own, including
wire, power supply, light bulb, resistor, ammeter, and volt-
meter. Their sample drawings can be found in table II. Thus,
we make up for the lack of sketch recognition data sets, es-
pecially hand drawn circuit component data sets. We will
show the details of the data set in the experimental study
section.

We use a CNN as our classifier. Since strokes and textures
of sketch circuit components are relatively simple, they have
fewer features compared to real-world images. And for re-
cent large and deep CNNs have numbers of parameters, there
is a risk of over-fitting when the features of training sam-
pies are less [10]. Therefore, instead of using these large
etworks, we adopt LeNet, which is a relatively lightweight
CNN, with some fine tuning. In the end, our model achieves
95.8% accuracy on the test data set and meet the reliability
requirements. The details of our classifier module, training,
and testing are shown in the experimental study section.

Sketch Analysis Module : In this module, we put our
graph model and algorithm into practical application. We
use above modeling method to model users’ input stroke
into graphs. For each new stroke, the algorithm used in this
module will only consider the relationship between the new
stroke and nearby strokes, which means that the algorithm
is locality sensitive. Upon this locality-sensitive measure,
we extract a subgraph of our graph model and examine if
there is a cycle. According to our analysis, the cycle can pro-
vide important clues for the segmentation of user stroke se-
quence and the detection of sketch circuit components. All
the algorithm details are shown in the algorithm design sec-
tion.

After finding a cycle in the subgraph, we send the image
composed by corresponding strokes of vertexes in the cy-
cle and receive the classification result. Afterwards, we use
this result to mark the strokes sent into the classifier mod-
ule before. Thus, every stroke which is sent into the classi-
fier module will get its own mark. For example, if a stroke is
part of a resistor, it will get a resistor mark after the result is
returned from the classifier. When the user’s input finishes,
all the strokes without mark will be viewed as wires. At this
time, we can traverse the whole graph model to infer the
relationships between all wires and circuit components and
get the global information of the input circuit sketch. In the
algorithm analysis section, we will give a detailed proof of
the feasibility of only using local information to achieve a
global state. Finally, we take extracted global relationships
between wires and circuit components as input, to call cir-
cuit state and attribute calculation module, and obtain the
simulation result of the circuit, which is the output expected
by the user.

Circuit State and Attribute Calculation Module : When
we get all the relationships between all kinds of circuit com-
ponents, the calculation of circuit state and attribute can be
solved easily. In detail, we find the solution through the fol-
lowing Kirchhoff’s current law (KCL) and Kirchhoff’s volt-
age law (KVL) [24].
Theorem 1. **KCL:** The sum of all the current entering a node is equal to the sum of all the current leaving the node:

\[ \sum_{k=1}^{n} i_k = 0, \quad (4) \]

where \( n \) is the degree of the node, i.e., the number of circuit branches connected to the node, \( i_k \) is the current of the \( k \)-th branch. The sign of \( i_k \) reflects the relationship between the reference current direction and the actual current direction in the branch. The positive sign is the same, and the negative sign is the opposite.

Theorem 2. **KVL:** The algebraic sum of the potential difference (voltage) of all components along the closed circuit is equal to zero:

\[ \sum_{k=1}^{m} v_k = 0, \quad (5) \]

where \( m \) is the number of circuit components along the closed circuit, \( v_k \) is the potential difference at ends of the element. The sign of \( v_k \)'s meaning is similar to the corresponding explanation in Theorem 1.

Based on Theorem 1 and Theorem 2, we can find the current and voltage relationships between different circuit components. Plus, we have the following Ohm’s Law for each component:

\[ U = IR, \quad (6) \]

where \( I \) is the current flowing through the component, \( U \) is the voltage at ends of the element, and \( R \) is its resistance. By solving the equations constructed by combining Equation (4), Equation (5), and Equation (6) together, we can get the state and attribute of the whole circuit graph, including direction and magnitude of current and voltage drop. Finally, it will show the result through the graphic interface. Some details of this module are shown in the algorithm design section.

4 ALGORITHM DESIGN

In this section, we firstly propose our main algorithm: Locality-Sensitive Special Sketch Symbol Detection (L₄S₄D). And then, we apply our main algorithm to the problem of sketch circuit graph recognition with some minor adjustments, and give an algorithm for extracting the connection relationships between circuit components. At last, we introduce our algorithm for circuit state and attribute calculation module to finish a completed system.

**Basic L₄S₄D**

We use graph to model users’ input strokes, and utilize cycles to detect some special sketch symbols. It is worth noting that when we select strokes and send them into the classifier, we only take those that have close local relationships with the new stroke into consideration, since sketch strokes usually have good local influence. The main algorithm is shown in Algorithm 1 (1).

In general, the algorithm takes advantage of a feature of users’ hand drawn symbols: a large number of hand drawn symbols are made up of a circular or rectangular frame plus the inner-frame content. In other words, the algorithm is good at detecting symbols with a closed border. Thus, we name our algorithm Locality-Sensitive Special Sketch Symbol Detection (L₄S₄D).

Firstly, we adopt a stroke segmentation step to process the new stroke in line 1 of Algorithm 1 (1). In this step, we use the degree on our graph model as an indicator to divide the stroke. To be specific, as soon as the user paints the first pixel, we will allocate a new vertex for this new stroke. Therefore, the degree of the new stroke will increase by 1. When this degree is 2, we will start a new sub-stroke with a new vertex. That is to say, the degree of vertex corresponding to a new sub-stroke is at most 2. This step makes sure that a single stroke does not participate in forming multiple cycles, which is helpful for us to extract and analyze all the cycles one by one. Afterwards, we will focus on each sub-stroke.

---

**Algorithm 1 Basic L₄S₄D**

**Require:** New stroke \( s \); Graph model \( G < V, E > \); Previous stroke set \( S \); Previous cycle set \( C \).

1. Segment \( s \) into sub-strokes \( s_{n+1}, s_{n+2}, \ldots, s_{n+m} \);
2. for \( p = 1 \) to \( m \) do
   3. mark \( s_{n+p} \) = \( -1 \);
   4. Update \( S \) and \( G \) with above modeling methods;
   5. if \( \exists c \in C \), satisfies \( s_{n+p} \subseteq c \) then
      6. \( c = c \cup \{ s_{n+p} \} \);
   7. else
      8. \( S_L \triangleq \{ s_i | d_{i,n+p} \leq \text{loc} \} \);
      9. \( S_L = S_L - \{ s_i | \text{marked}(s_i) \neq -1 \} \);
     10. \( V_L = \{ i | s_i \in S_L \} \);
     11. \( G_L = G[V_L] \);
     12. \( c = \{ s_i | \exists j \in V_L, e_{l,i} \notin \text{cut}[G_L] \} \);
     13. if \( c \neq \emptyset \) then
         14. \( c = c \cup \{ s_i | s_i \subseteq c \} \);
         15. \( C = C \cup \{ c \} \);
     16. end if
   17. end if
18. \( r \triangleq \text{classifier}(c \text{'s corresponding image}) \);
19. mark \( s_i \) = \( r, \forall s_i \in C \);
20. end for
Based on the common feature of our target symbols, i.e., a closed border, a natural idea is to divide the algorithm into two main branches to deal with the two cases respectively: the new stroke falls in a previously detected cycle and it is not in any cycle, corresponding to the condition statement in line 5 of Algorithm 1 (1). Note that users may not draw an inner-frame stroke completely in the outside frame, i.e., some small parts of the stroke may extend out of the border. Therefore, we use notation $\subseteq$, which means that the stroke will be viewed as an inner-frame stroke as long as most parts of it fall in the outside border. In this way, we relax the constraint on user hand drawing and users do not need to deliberately control the input stroke to place it in the frame completely. Obviously, if a new stroke is in a cycle, we need to send the cycle and all the strokes in it into the classifier. Although the cycle’s strokes have been marked previously, since the content in it has changed, we have to send some marked strokes into classifier again to ensure the final classification results. This step corresponds to line 6 of Algorithm 1 (1).

On the other hand, if the new stroke is not in any cycle, we will find its nearby strokes, whose corresponding vertices have a short distance to the new vertex in our graph model, as shown in line 8 of Algorithm 1 (1). Here we introduce a hyper parameter $loc$ to indicate the locality sensitivity of the algorithm. We only consider those vertices with a distance no more than $loc$ to the new one. Therefore, a higher $loc$ means a weaker locality sensitivity. If there is a need to consider those strokes that are relatively far from the new one, $loc$ should be higher. At the same time, the algorithm will consider less local information and more global information.

Since this branch satisfies the condition that the new stroke does not in any existing cycle, those marked strokes should not influence the segmentation and analysis of following strokes. Therefore, we do not take marked strokes into consideration and delete them from $S_L$ in line 9 of Algorithm 1 (1). We then extract the induced subgraph $G_L$ for $S_L$’s corresponding vertex set $V_L$ in the current graph model $G$. After this step, we will detect our target symbols only in this subgraph $G_L$ instead of the global graph, which reflects the local sensitivity of our algorithm. Subsequently, taking advantage of the closed border, we only need to examine whether there is a cycle in $G_L$. When we find a cycle, we can separate out the border strokes of our target sketch symbols, and thus complete the detection, location, and recognition tasks. In line 12 of Algorithm 1 (1), we find the cut-edge of $G_L$, which is equal to finding a cycle as shown by the following theorem[1].

**Theorem 3.1.** An edge is a cut-edge if and only if it is not contained in any cycle.

Consider that some users may draw the inner-frame content of a symbol at first and the border afterwards, in line 14 of Algorithm 1 (1), we add those strokes which are enclosed in the extracted border, i.e., the corresponding strokes of the cycle. Here we also relax the constrain: a stroke will be accepted as an inner-frame stroke if most parts of it falls in the border, for the same reason mentioned above.

Finally, we send the final settled stroke set $c$’s corresponding image into the classifier and mark these strokes with the classification result. At this time, the algorithm will process next sub-stroke. When all the sub-strokes have been addressed, the system will get ready to receive next input stroke.

**Algorithms for Circuit Sketch Analysis**

**Fine-tuned LS$^4$D:** With the foundation of Basic LS$^4$D, we can detect and locate all the symbols with a closed border efficiently. According to our observation and user study, users tend to use relatively less strokes to finish drawing a symbol, which is consistent with the properties of sketches: simple and convenient. From this point of view, we adjust the local sensitivity of the algorithm to the maximum, i.e., $set loc = 1$. In other words, we only take those strokes that directly adjacent to the new stroke into consideration. Since the degree of a new input sub-stroke’s corresponding vertex is 2 at most, the algorithm can only find cycles with length no more than 3. Unfortunately, through our survey, some users still prefer to draw a resistor with 4 strokes, corresponding to the left, up, right, and bottom borders respectively. In this case, the Basic LS$^4$D algorithm will fill to detect this kind of resistor, as shown in Fig. 2(a). Therefore, we introduce the concept of rectangular closure and utilize it to fine tune the Basic LS$^4$D algorithm to solve this problem. To calculate the rectangular closure of a set of strokes, firstly, we need to find the smallest rectangle which can enclose all the strokes in the set, i.e., the boundary coordinates of the rectangle is the maximum and minimum values of the strokes in the set on axes $x$ and $y$ ($x_{max}$, $x_{min}$, $y_{max}$, and $y_{min}$). Subsequently, we find all the strokes that are in the rectangle calculated in the last step. The set of these strokes is the rectangular closure of the original stroke set. In other words, we expand the original $S_L$ through a rectangular rule and we will use expanded $S_L$ instead of the original one for the following steps. The steps for calculating the rectangular closure of $S_L$ can be formulated as the following equations:

$$R \triangleq \min\{\text{Rect} | \forall s_i \in S_L, s_i \subseteq \text{Rect}\}. \quad (7)$$

$$S_L = \{s_i | s_i \in R\}. \quad (8)$$

We insert Equation (7) and Equation (8) after line 8 of Algorithm 1 (1) to obtain a fine-tuned algorithm of Basic LS$^4$D.
As shown in Fig. 2(b), the problem can be solved efficiently after we use the rectangular closure.

**Power Supply Detection**: The $LS^4D$ algorithm can detect those sketch symbols with closed border efficiently. However, in the scene of circuit sketch recognition, the symbol of power supply does not have this feature. Thus, $LS^4D$ fails to detect sketch power supplies and we need another mechanism to tackle this problem. In this paper, we continue using our graph model to find the symbol of power supply.

As shown in Table II, the power supply’s symbol consists of two unconnected strokes, indicating the positive and negative pole respectively. In addition, strokes belonging to the power supply usually have only one wire connected, which are unlike those belonging to the wires, since both ends are connects by components in a normal circuit graph. In other words, if a circuit graph is sketched standardly, a stroke belongs to the power supply if and only if the degree of its corresponding vertex is 1. A typical case is shown in Fig. 1(c). Note that this method requires users to draw their graph standardly and strictly. If there is a wire stroke which has only one end connected, the degree of the corresponding vertex is also 1. In this case, simply using vertexes with degree 1 does not work and some other discrimination mechanisms are needed to ensure the robustness of the system. The details of these methods are beyond the scope of this paper.

**Connect Relationship Analysis**: With fine-tuned $LS^4D$ algorithm and a simple mechanism to locate the power supply, we can detect and locate all the circuit components as well as divide all the strokes according to the components they belong to. Now we can take advantage of these previous results to analyze their relationships between each other. The connect relationship analysis algorithm is shown in Algorithm 2 (2).

Algorithm 2 (2) plays a connecting role in the whole sketch recognition and analysis system, since we convert user’s input strokes into the format needed by future calculation on the circuit graph in line 16. In line 11, due to the flexibility of wires in the circuit graph, it is equivalent to directly use a stroke’s corresponding vertex in our graph model to represent the node in the actual circuit graph. We also introduce the concept of pin in line 13, which makes it convenient for us to extract physical relationships from connect relationships in graph model.

**Circuit State and Attribute Calculation**: After extracting the connect relationships between all the wires and components, the last step is to calculate the state and attribute of the circuit graph. The basic idea is to construct a set of equations according to the electrical laws, i.e., Equations (4), Equation (5), and Equation (6). Specifically, we use nodes converted from strokes with mark Wire to build Equation (4), and use detected circuit components to build Equation (6). As for Equation (5), we need to find some loops in the graph to meet the need of this equation. We can find all the loops from the positive pole of power supply to the negative pole in the same way as the above cycle finding step. And then we apply the voltage information of all the components on the loops to Equation (5). Due to the forms of these equations, the set of equations constructed above can be converted into a linear system of equation. Therefore, we can consider using Gauss elimination method([18]) to solve it. The details of the solving process is not the core aspects of this paper. As long as conditions are sufficient, the rank of the matrix is no less than the number of unknown quantities and we can obtain the solution as teachers and students can. Here, we have built a complete circuit sketch recognition and analysis system and implemented all the related algorithms.

5 **THEORETICAL ANALYSIS**

In this section, we demonstrate the correctness of our main algorithm, i.e., Graph-based $LS^4D$, from a theoretical point of view. And then, we analyze the time complexity of key steps in the algorithm. Thus, it is shown that our algorithm can complete the tasks efficiently while ensuring correctness.

**Proof of Correctness**

We utilize graph to model users’ input strokes and the classification results of cycles in local areas to infer the global information of the graph. The proof of correctness consists of two parts. Firstly, we show that the algorithm can detect all the cycles meeting the requirements without omission,
i.e., the recall rate is 100%. Secondly, we show that the algorithm can detect all the expected symbols without any error report, i.e., the precision is 100%. Taken together, the correctness of the algorithm can be proved.

**Proof for Recall Rate**: Theorem 4 is used to show that our algorithm can detect all the required cycles and make sure a high recall rate.

**Theorem 4.1.** $LS^4D$ can detect all the cycles with length no more than $2 \times \text{loc} + 1$.

To prove Theorem 4, we first introduce the following lemma:

**Lemma 1.** The contribution of a newly added sub-stroke to the number of rings that have not entered the classifier before is at most 1.

**Proof.** Suppose the algorithm is currently ready to process a sub-stroke $s$ (does not in any existing cycle), the corresponding vertex is $v$, and the graph model at present is $G < V, E >$. Since we only consider those strokes with $\text{mark} = -1$, we can temporarily delete those vertexes whose corresponding strokes do not satisfy this property and get the resulting graph $G'$. Determined by the operation of $LS^4D$, we send strokes whose corresponding vertexes are on a cycle to the classifier. Therefore, all the edges are cutting edges in graph $G'$, because if there is a non cutting edge, there must be a cycle in $G'$ according to Theorem 3, which is contradict to the operation of the algorithm. Consequently, $G'$ is an acyclic graph. And if there is a cycle in graph $G' + v$, the cycle must contain vertex $v$. Besides, due to our stroke segmentation method, the degree of a new sub-stroke’s corresponding vertex is at most 2, i.e., $\text{deg}(v) \leq 2$. Now suppose that $v$ contribute more than 1 to the number of cycles. Then $v$ should on more than 1 cycle. Obviously $\text{deg}(v) = 1$ or $\text{deg}(v) = 0$ is impossible, so we only analyze the case that $\text{deg}(v) = 2$ and let $v$ be adjacent to vertexes $a$ and $b$.

There should be two different sets of vertexes: $\{v_i\}$ and $\{v_j\}$, which form cycles: $v,a,v_i,v_j,...,v_k,b,v$ and $v,a,v_i,v_j,...,v_k,b,v$, respectively. Then, in the two groups of vertexes, from the last vertex counting from the front to the back, to the last vertex counting from the back to the front, a cycle without $v$ will be formed between these two vertexes, as shown in Fig. 4. This is contradict to our previous analysis result: $G'$ is acyclic. In this way, we have proved Lemma 1.

Now we take advantage of Lemma 1 to prove Theorem 4:

**Proof.** According to Lemma 1, there is at most 1 newly added cycle for each sub-stroke. Besides, if there is a new cycle, the corresponding vertex of the new sub-stroke is on it, and the degree is 2. Based on the operation of $LS^4D$, we take all the vertexes with a distance no more than $\text{loc}$ to the new one into consideration. Therefore, the number of vertexes on the cycle is at most $2 \times \text{loc} + 1$ and the algorithm can find this cycle successfully.

**Proof for Precision Rate**: Decided by the algorithm itself, our algorithm never sends a set of non-cycle strokes into the classifier. In other words, when we send a found cycle of the graph model into classifier, it is also a loop from the perspective of strokes, which can form the closed border of a symbol. Thus, the detection report of our algorithm is reliable.

Finally, in summary, $LS^4D$ will detect all the required cycles accurately without omission. Since the target objects of the algorithm is symbols with closed borders, it can detect all the satisfied symbols correspondingly.

**Analysis of Time Complexity**

Due to the uncertainty of the number of sub-strokes, we take a single sub-stroke, instead of a whole input stroke as the unit for our time complexity analysis. According to the above analysis, the degree of a new vertex is at most 2. And if we set $\text{loc} = 1$, the algorithm can detect cycles whose length is at most 3, i.e., the new vertex and its two adjacent vertexes. Therefore, the computational cost of the key step, which is finding cycles, is constant level $O(1)$. However, when we set $\text{loc} > 1$, in the worst case, the algorithm has to consider all the vertexes of unmarked strokes, since there is always a way of constructing the graph to push all the vertexes in the graph have the distance at most 2 to the new vertex. In this case, the time complexity is $O(|V| + |E|)$. Since there is no cycle in the previous graph, and the degree
We divide our experiment into two parts. In the first part, we build and test our sketch circuit component classifier; we collect about 200 hand-drawn samples for each type of circuit component, and the total number of samples is about 1200. The details of our data set are shown in Table II. We also test our classification accuracy on the training samples [19], as follows:

- for wires, power supplies, light bulbs, and resistors, carry out the horizontal and vertical symmetry transformations on the images, as well as 90, 180, and 270 degrees rotation transformations;
- for wires, power supplies, and resistors, carry out 1.2, 1.5, and 1.8 times compression transformations horizontally and vertically;
- for ammeters and voltmeters, carry out the horizontal symmetry transformation.

Our training set consists of all original images and transformed images.

**Data Preprocess**: Since the strokes of sketch circuit components are simpler and have relatively few features, there is no need to use high-resolution images. Besides, the classification of sketch circuit components in this paper only considers the shape characteristics of sketch strokes and is not sensitive to color, so we take a single-channel grayscale image with a grayscale value of 0 to 255. Therefore, we uniformly preprocess the input samples into (32 × 32 × 1) size, where 32 × 32 is the input image resolution, and 1 represents the image channel number.

Furthermore, in order to fully expand the number of training samples, and consider a variety of different sketch styles as well as the equivalence of horizontal and vertical drawing of some components, we perform data augmentation on the training samples [19], as follows:

- carry out the horizontal and vertical symmetry transformations on the image. In addition, we perform data augmentation on some components, we perform data augmentation on the images, as well as 90, 180, and 270 degrees rotation transformations.
- carry out 1.2, 1.5, and 1.8 times compression transformations horizontally and vertically;
- for ammeters and voltmeters, carry out the horizontal symmetry transformation.

**Training and Testing Details**: We use LeNet-5 [3] as our basic network architectures. Different from the original network, we use dropout with a probability of 0.5 for the first two fully connected layers to avoid overfitting [22] and use LeakyReLU [25] as the activation function, except the output layer. We adopt Adam algorithm [13] (\(\alpha = 10^{-3}, \beta_1 = 0.9, \beta_2 = 0.99\)) for batch gradient descent, with a batch normalization strategy (batch size is 3). After each epoch, the model is tested on the cross-validation set. After about 15 epochs, the model loss converges. In the end, our model achieves a 99.9% accuracy rate on the training set, a 99.7% accuracy rate on the cross-validation set, and an overall accuracy rate of 95.8% on the test set after the entire training. The accuracy rates of each kind of component are shown in Table II. We also test our classification accuracy in our study, and get an overall accuracy of 96.5%.

The details of user study are shown in the following part. For future researches, we publish our trained LeNet model at http://47.103.30.151/data/LeNet.pt.

**User Study**: We convey user study on 17 users to test our algorithm. The experiment on a single user consists of two phases. In first phase, the user only draws circuit components, without wires, to test the classification accuracy of our classifier.

<table>
<thead>
<tr>
<th>Component</th>
<th>Sketch Sample</th>
<th>Total # of Samples</th>
<th>Test Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wire</td>
<td></td>
<td>220</td>
<td>100.0%</td>
</tr>
<tr>
<td>Power Supply</td>
<td></td>
<td>206</td>
<td>92.6%</td>
</tr>
<tr>
<td>Light Bulb</td>
<td></td>
<td>174</td>
<td>92.3%</td>
</tr>
<tr>
<td>Resistor</td>
<td></td>
<td>205</td>
<td>98.8%</td>
</tr>
<tr>
<td>Ammeter</td>
<td></td>
<td>184</td>
<td>93.1%</td>
</tr>
<tr>
<td>Voltmeter</td>
<td></td>
<td>181</td>
<td>96.4%</td>
</tr>
</tbody>
</table>
Figure 6: If a stroke belongs to different components, a segment of the stroke will be mis-marked and the level of confidence of the classifier will decrease.

Figure 7: The algorithm tends to view looped wires as resistors.

Every user draws 10 components. Then in second phase, the user draws 5 circuit diagrams provided by us (part of users also drew some other samples designed by themselves), to test whether our algorithm can extract correct connect relationships.

Overall speaking, we collect 170 samples altogether for phase 1, with 6 misclassification and 145 samples for phase 2, with 14 anomalous analysis.

**Success Case**: Since our algorithm is sensitive to local cycles, coupled with the fact that most of the circuit components have closed borders, we can accurately detect and identify all the components drawn by users. Unlike those algorithms that are sensitive to time series and strictly require users to draw circuit components one after another, our algorithm is more flexible, since users can adopt more diverse stroke orders, e.g., finish all the borders of components at first and finally finish the inner-contents. This is why our locality-sensitive algorithm is superior to time-sensitivity algorithms. In the example in Fig. 5, which is a relative complex circuit problems in middle or high school, our algorithm can extract all the circuit components efficiently and give a correct analysis. From this perspective, our algorithm is stable and robust.

**Failure Case**: Here we list three kinds of failure cases found in our user study.

1. As shown in Fig. 6, since our algorithm divides and detects circuit components in units of strokes, when a user’s stroke belongs to two circuit component components at the same time, it may cause a misclassification or error connect relationships analysis.
2. As shown in Fig. 7, since our algorithm tends to send cycles into the classifier, it also easily causes misclassification of the looped wires, that is, redundant wires. We find from user study that our algorithm and classifier prefer to view looped wires as resistors.
3. As shown in Fig. 8, since our algorithm performs symbol detection based on cycles, if the user’s wire strokes and some circuit component strokes form a loop in advance, our algorithm will not be able to deal with such mixed cycle. In other words, although our algorithm allows users to draw other components before finishing the current one, it needs to be ensured that strokes drawn later do not form a cycle with the strokes of previously unfinished components.

Fortunately, from user study, it was found that few users have the drawing habits described in these three cases. The reasons are as follows.

1. For case 1, although there are users who like to draw some circuit parts at the same time by using “continuous strokes”, it is found from our observation that such mixed strokes usually belongs to a wire and a component, and that the classifier prefer to give the result of that component, instead of wire, as the picture shows. This is consistent with the situation described in [7]. According to our connect relationship extraction strategy, the component (or wire) connected to the other end of the wire will still be connected to this component. Therefore, this situation will not cause an abnormality to the entire system as a whole. Furthermore, if a more optimized stroke segmentation mechanism is adopted, e.g., [6], this problem can be solved fundamentally and thoroughly.
2. For case 2, loops formed by wires are extreme cases, which have no physical meaning and no educational value. Therefore, when the system is actually put into teaching, such a situation is considered to be irregular,
which is not our original intention to design this interactive circuit sketch analysis and recognition system.

3 For case 3, we find from user study that while drawing circuit diagrams, users tend to have coherent thinking, so they tend not to switch to the next circuit component before drawing a circuit component. Of course, there are still some users who return to the previous component to do some modify or supplement. Based on this point, our algorithm relaxes the restrictions and allows users to return to previously recognized or incomplete strokes. From our observation, this kind of "turning back" often happens immediately after the user finds that the drawing is incomplete. Thus, in practical applications, it usually does not happen that the user does not go back to do modify until unfinished strokes form cycles with subsequent strokes. In addition, according to our survey, since our system operates in online mode, which is the point that users also understand, due to users’ curiosity, they tend to finish target symbols, i.e., circuit elements at first, and expect real-time feedback from the system rather than waiting a long time to complete previous components.

In summary, intensive experimental study has proved that our algorithm and system is in accordance with most users’ sketch habits, which is also consistent with principles of user psychology, and has the ability to put into practical applications and the educational field.

7 CONCLUSION

This paper proposes a method to model users’ drawn strokes with vertexes in graphs, and proposes an efficient and reliable locality-sensitive algorithm to detect and localize symbols that have closed borders through information of cycles. Theoretical analysis proves that our algorithm is sufficiently robust and the time complexity is also lower than the current most advanced algorithms in general. After that, we apply it to the recognition and analysis system of circuit sketches. In this process, we publish the training and testing data set used by sketch circuit component classifier, which enriches the data set in the field of sketch recognition, especially for circuit sketch recognition. Finally, intensive user experiments have proved that our algorithm is of great value in the practical sense, which can be called start-of-the-art algorithm at present on this problem to our best knowledge. In the future, we would like to extend the algorithm to a wider range of applications, such as sketches of flow chart and UML diagram.

REFERENCES