Neurocomputing 168 (2015) 35–46

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Lead curve detection in drawings with complex cross-points

Jia Chen [a], Min Li [c,1], Qin Jin [b,*], Shenghua Bao [d], Zhong Su [c], Yong Yu [a]

[a] Department of Computer Science and Engineering, Shanghai Jiao Tong University, China
[b] School of Information, Renmin University of China, China
[c] IBM China Research Laboratory, Beijing, China
[d] IBM Watson Group, San Jose, United States

ARTICLE INFO

Article history:
Received 27 November 2014
Received in revised form
15 April 2015
Accepted 10 June 2015
Communicated by Luming Zhang
Available online 20 June 2015

Keywords:
Lead curve
Curve-path
Cross-point graph
Structured learning

ABSTRACT

Lead curve detection in design drawings is a critical problem in a wide range of applications ranging from checking similar drawings in patent granting to constructing hyperlinks between image and text description in digitalization. The difficulty of the problem are two folds: unknown end point of lead curve and complex crossings. However, most previous curve detection algorithms are usually applied in simple or no crossing situations. We make four contributions in solving the problem: (1) we transform the problem into a new problem that finds an optimal path with the best score in the cross-point graph. We introduce the "cross-point graph" representation which captures the topology of cross-point connectedness. Based on the original drawing and the corresponding cross-point graph, we introduce the coupling concept "curve-path", which correlates the curve in the original drawing with the corresponding path in the cross-point graph. (2) we design a set of joint feature representations for curve-path which describes different characteristics of a curve and its corresponding path. (3) we define a task specific loss function for our customized structured SVM. We propose a mixed negative instance sampling strategy to learn the weights of different joint feature representations. We prune the search space effectively for fast lead curve detection. (4) we build a software to efficiently facilitate manual lead curve labeling. We release the patent drawing dataset with groundtruth to public for lead curve detection research. The extensive experimental results prove the effectiveness of our methods.

1. Introduction

Engineering design and drawings such as patent drawings, room designs and machine designs are very important types of drawings. Most of these drawings are publicly available as scanned images rather than its original Computer Aided Design (CAD) format. For example, patent drawings on the US patent site[1] are submitted in image format. The amount of such drawings increases rapidly. Take US patent for example, there were 36,034 design patent applications and 23,468 design patent grants in year 2013 [1]. This number is still increasing each year. Design patent alone has 501,790 grants during 1963 ~ 2013. Numerous room designs and machine designs are also available online as scanned images.

In these drawing images, there are not only curves describing the object but also "lead curves" linking the reference characters with a certain part of the object. An example is shown in Fig. 1: there are lead curves linking reference characters such as "110", "112", "112A" to different parts of the object. These reference characters will have further detailed explanations in the document. Based on statistics of the collected patent design images in our dataset, there are around 19 lead curves on average in a patent design drawing, which indicates that lead curves frequently appear in patent drawings.

Lead curves play an important role in many drawing related applications. For example, when a drawing image of a new patent is submitted, patent agency needs to check whether it is near duplicate or similar to any drawing images of granted patents. The existence of many lead curves as shown in Fig. 1 dramatically deteriorates patent image retrieval quality. Detecting these lead curves and removing them is beneficial for near duplicate/similar design patent detection. Another example is that in the current online patent database, there is no hyperlink between the drawing image and text description. That is, users have to manually switch between the object part in the drawing image and the detailed explanation in the text, which is tedious and inefficient. Once the lead curve in the patent image is automatically detected, a hyperlink can be constructed to ease the switching effort.

Standards for Drawings [2] only provides one hint for locating the start-point of a lead curve: it must originate in the immediate

* Corresponding author.
E-mail addresses: chenjia@apex.sjtu.edu.cn (J. Chen), mushi.lm@alibaba-inc.com (M. Li), qjin@ruc.edu.cn (Q. Jin), baoshhua@us.ibm.com (S. Bao), suzhong@cn.ibm.com (Z. Su), yyu@apex.sjtu.edu.cn (Y. Yu).
[1] url http://www.uspto.gov/

http://dx.doi.org/10.1016/j.neucom.2015.06.018
0925-2312/

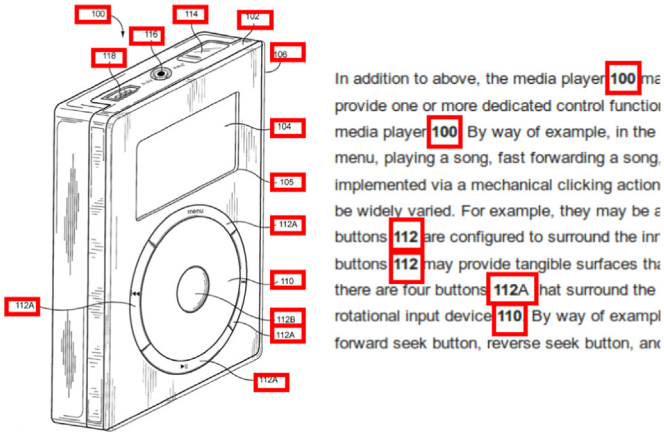


Fig. 1. An example of patent "Method and apparatus for use of rotational user inputs" (US 7345671 B2), one of Steve Jobs's top 5 favourite patents: reference characters are highlighted in red rectangles. In the drawing, lead curves link the reference characters to the object part. In the text description, there are detailed explanations for the corresponding reference characters. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this paper.)

proximity of the reference character. It also adds one constraint on lead curves that they cannot cross each other. The difficulty of lead curve detection lies in two challenges.

Unknown end-point of lead curve: the end-point of a lead curve is entirely unknown beforehand. A lead curve may end outside the object boundary, or on the object boundary, or at a certain position inside the object boundary.

Complex cross-points: a lead curve will cross many other curves that are misleading before it reaches the end. These cross-points can be the outer bound of the object, inner curves of the object and curves in the shaded area. Therefore, cross-points have different contexts.

Putting these two challenges together, complex cross-points are misleading for the detection process, especially when the end-point of the curve is unknown.

Previous curve detection algorithms [3–7] are usually applied in simple or no cross-point situation. In work [4], authors presented a novel method to detect curves with unknown end-points using minimal path techniques. Their method is evaluated on crack images, in which the cross-point situation is very simple. Authors in work [8] proposed a new and fast method for dominant point detection and polygonal representation of a discrete curve. They do not consider any cross-point situation.

We formulate the lead curve detection task as a problem of finding the best curve among all possible curves given the start-point. We make the following four contributions to tackle the challenges of unknown end-point of lead curve and complex cross-points.

- We introduce the "cross-point graph" representation which captures the topology of cross-point connectedness. We introduce a coupling concept "curve-path", which is a pair of the curve in the original image and the corresponding path in cross-point graph. Based on the cross-point graph representation and curve-path concept, we transform the lead curve detection task to a problem of searching in the cross-point graph for the path whose corresponding curve is most likely to be the lead curve.
- We design a set of joint feature representations for geometric plus context properties of a curve-path. The joint feature representations include smoothness at points, position relationship between the end node and object boundary, direction difference between the curve and the shaded area, angle difference between the curve parts inside and outside the object boundary and the length difference between the lead curve and other curves in the same image.
- We define a task specific loss function to measure the difference between two curve-paths. We also define a scoring function which gives higher score for the path of a groundtruth lead curve. We exploit max margin structured learning framework to learn the weightings of different joint feature representations. Furthermore, we design a mixed negative instance sampling strategy for the training stage and prune the search space significantly in the inference stage.
- We collect precise groundtruth by building a software to facilitate manual labeling and make the dataset publicly available.

The remaining of the paper is organized as follows. Section 2 introduces related work. Section 3 introduces the cross-point graph representation, describes the curve-path concept and transforms the lead curve detection task to a path searching problem on the cross-point graph. Section 4 gives the solution, which includes patent image parsing, joint feature representation design, inference and training. Section 5 reports extensive experimental results. Section 6 discusses the applications of lead curve detection in multimedia analysis. Section 7 draws some conclusions.

2. Related work

There have been many researches on contour detection in medical image analysis. A promising and vigorously researched technique is deformable model [9–13]. Priori knowledge about the location, size and shape can be embedded into the model. It has been widely applied in segmenting, matching and tracking anatomic structures. But the performance of deformable approach is usually affected by the initial condition and complex cross-points. In the case of patent image, when a lead curve goes into the internal of the object, there are many cross-points. Such situation is difficult for a deformable model to handle. The general particle filter (PF) framework is a powerful framework that has been applied in tracking and matching [14,15]. In work [15], authors relax the assumption of having ordered observations and extend the particle filter framework to estimate the posterior density by exploring different orders of observations. Another related field is contour detection, grouping and matching [14,16,17]. The general approach is to decompose a closed contour of a given model shape into a group of contour segments and match the resulting contour segments to edge segments in a given image.

In our work, we focus on detecting a particular kind of contour: lead curve. Generally speaking, our work solves the problem of contour detection in complex contour cross-point situation. Different from contour grouping in which edge broken is the major challenge, in our task cross-points are the major challenge. Cross-points will mislead the detection process. Our task is also different from contour matching where a template is given. It is difficult to give a lead curve template since both its length and cross-point status change dramatically among different images.

3. Problem formulation

In this section, we assume that the start-point is given and the image is a binary image where 1 s correspond to the drawing and 0 s correspond to the background. The details of extracting start-point and converting an image to a binary image will be described in Section 4.1. We formulate the problem by simulating the process that a human detects a lead curve.

Simulation of human's lead curve detection process: take Fig. 2 for example, from the start-point, we follow the curve without difficulty before we reach cross-point *A*. After looking around, we decide to follow curve segment a_1 since curve segment a_2 and a_3 belong to the boundary of the object. From cross-point *B*, the curve segment will enter the shaded area. However humans can still pick the right curve to follow on each cross-point until reach the end-point. How do humans know they reach the end-point? In most cases a cross-point is considered as an end-point when all the candidate curve segments have very different directions from the curve we are following. Other cases will be discussed in detail in Section 4.2.

Abstraction of the process: in the above process, the difficulty of lead curve detection mainly happens on cross-points. A cross-point is the intersection of two or more curves or the end-point of a curve. On the contrary, a trivial point is a point in the middle of a curve segment. If we abstract cross-points as nodes, curve segments between cross-points are edges between nodes. We get a graph which preserves the simplified structure of the connectedness between cross-points by removing trivial points and using edges instead. We denote such a graph as *cross-point graph*: $G=\{\mathbf{N},\mathbf{E}\}$. The cross-point graph of the drawing image in Fig. 3(a) is shown in Fig. 3(b). Considering the cross-point graph alone, the human's detection process can be formulated as finding the path $\mathbf{y}=e_i\cdots e_j$ whose corresponding curve is most likely to be the lead curve given the start node *n*, where $n\in\mathbf{N}$ and $e_i\in\mathbf{E}$.

To formulate the problem, we need to first introduce three concepts: curve-path, joint feature representations and scoring function. Since a path in the cross-point graph loses much geometric and context information, we introduce the curve-path concept $(\mathbf{x},\mathbf{y})$, which is a pair of the path and its corresponding curve, where $\mathbf{x}$ is a set of pixels in the binary image and $\mathbf{y}$ is the path in the cross-point graph. Geometric information includes smoothness at points while context information includes object boundary, shaded area and average length of the lead curves in the same image. Then, these geometric plus context properties of lead curve-path are quantitatively represented by a set of joint feature representations. With joint feature representations, we finally define a scoring function which measures the likelihood of a curve-path being a lead curve. See following for details.

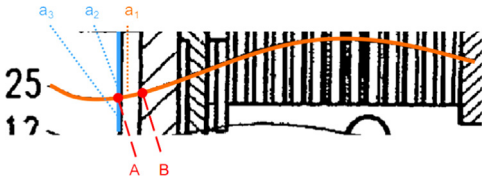


Fig. 2. Illustration of human detection process.

Curve-path: to associate a path in cross-point graph with a curve in binary image, we construct a mapping between the edges and the curve segments in the simplification process. That is, each edge $e_i\in\mathbf{E}$ in the cross-point graph is mapped to a curve segment $\mathbf{x}_i$ in the drawing image as illustrated in Fig. 4. We denote such mapping as *edge-to-curve mapping*: $\mathcal{M}(e_i)=\mathbf{x}_i$. It is a one-to-one mapping. Given a path $\mathbf{y}=e_i\cdots e_j$ in the cross-point graph, we get the corresponding curve $\mathbf{x}=\mathbf{x}_i\cup\cdots\cup\mathbf{x}_j$ by applying edge-to-curve mapping iteratively.

Joint feature representation: on the curve-path, we construct a corresponding joint feature representation $\Psi(\mathbf{x},\mathbf{y})$ to characterize its geometric and context properties. In particular we design six joint features based on six heuristics:

1. smoothness at inner node points: a path usually goes along the direction of the most smooth edge.
2. smoothness at end node point: a path usually ends at a crossing point under certain smoothness condition.
3. position between end node and object boundary: people usually start to label the object part (end-point) from a closer side of the boundary.
4. direction difference from the shaded area: when a lead curve crosses a shaded area, the directions of the curve at cross nodes are significantly different from those of line segments in the shaded area.
5. angle difference between the curve parts inside and outside the object boundary: the part of a lead curve inside the object boundary does not deviate significantly from the part outside the object boundary.
6. difference from average length: the length of a single lead curve does not deviate significantly from the average length of all the lead curves in the same image.

The mathematical form of joint feature representation is defined as an *n*-dimensional vector and the detail of each element in the vector will be given in Section 4.2.

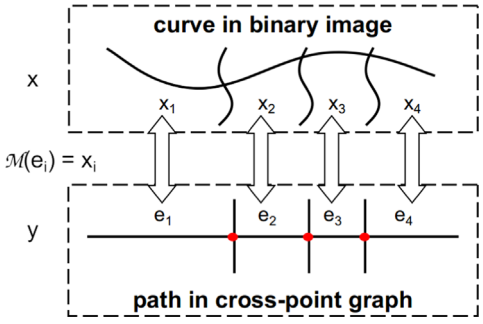


Fig. 4. Illustration of edge-to-curve mapping in curve-path.

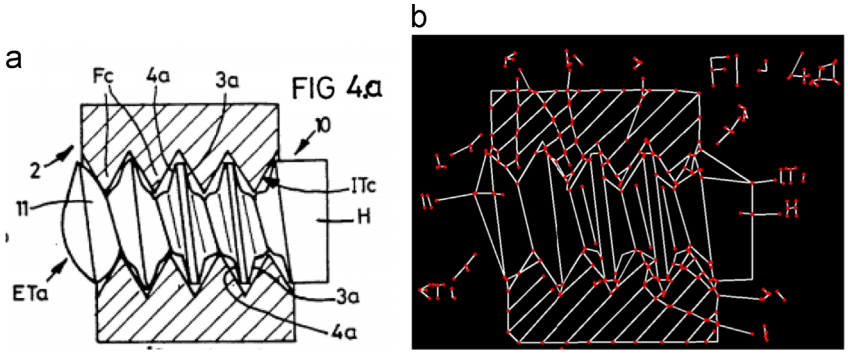


Fig. 3. A drawing and its cross-point graph representation: in the cross-point graph, nodes are shown in red color; edges are shown in white color. (a)Drawing (b) Cross-point graph

Scoring function: given the joint feature representations $\Psi(\mathbf{x},\mathbf{y})$, we define a scoring function $F(\mathbf{x},\mathbf{y})$ which gives higher score to path $\mathbf{y}$ whose corresponding curve $\mathbf{x}$ is more likely to be a lead curve. We define the scoring function as a linear combination of the joint feature representations.

$$F(\mathbf{x},\mathbf{y}) = \mathbf{w}^\top \Psi(\mathbf{x},\mathbf{y}) \tag{1}$$

Now we get the strict formulation of our main problem. *Main problem: Given the start-point, lead curve detection is to find a path with the maximum score in the cross-point graph. The scoring function is defined on both the path in the cross-point graph and the pixels in the binary image.*

$$\hat{\mathbf{y}} = \underset{\mathbf{y}\in\mathcal{Y}}{\operatorname{argmax}} \mathbf{w}^\top \Psi(\mathbf{x},\mathbf{y}) \tag{2}$$

All the concepts and notations introduced in this section are listed in Table 1.

4. Solution

Given the patent image, we parse the image to get graph related information and binary image related information. Graph related information includes cross-point graph and start node. Binary image related information includes object boundary, shaded area and smoothness at nodes. Given the graph related information, we prune the search space in the inference stage and sample negative instances in the train stage. We design joint feature representations based on both graph related information and binary image related information. Finally, we utilize max-margin structured SVM to learn the weightings of joint feature representations. The flowchart of the solution is shown in Fig. 5.

4.1. Patent image parsing

We first convert an image to a binary one by thresholding. Since most drawing images are exported from software, the simple thresholding technique performs well with threshold set to 100. On the binary image, the foreground pixels are 1 s and background pixels are 0 s.

4.1.1. Cross-point graph

The curves in the binary image vary in thickness, which makes directly contructing the cross-point graph difficult. To address this issue, we extract skeleton of the foreground. The thickness of curves on the skeleton is 1 pixel, which provides an ideal start to build the cross-point graph. Based on pixel connectedness in 8-neighborhood, we construct the skeleton graph on skeleton pixels. Now curves are simplified to edges in the skeleton graph. Given the definition that the cross-point is the intersection of two or more curves or the end of the curve, we induce that nodes of degree 2 on skeleton graph must not be cross-points. Thus we simplify the skeleton graph by iteratively applying node contraction operations on those nodes of degree 2. On the simplified skeleton graph, the remaining nodes are strong candidates for cross-points. However, the performance of most skeleton extraction algorithms is affected by changes in line thickness more or less, which leads to generating unnecessary small branches. To suppress the influence of skeleton extraction algorithm, we further apply Harris corner detection on the original drawing to verify whether a node in the simplified skeleton graph is a cross-point. The cross-point graph is generated by applying node contraction operations on nodes that fail to pass the Harris corner detection test. We further simplify the cross-point graph by contracting nodes of degree 2 and remove small edges that end at nodes of degree 1 with curve length smaller than 5 pixels.

4.1.2. Start node

According to *Standards for Drawings* [2], the start node of a lead curve must be in the immediate proximity of the reference characters. We locate start node by finding reference characters. Reference characters are usually small connected components of foreground pixels in the binary image. We extract small connected components from foreground pixels and apply classical OCR techniques to refine candidates. Given the candidate reference characters, we search for start nodes on nearby nodes in cross-point graph. Considering the fact that the degree of start node is 1, we can further narrow down the search scope. Finally, nodes of degree 1 close to reference character candidates are extracted as start nodes.

4.1.3. Boundary

Boundary is the outer contour of an object in the binary image. We fill holes in the binary image to get objects using the morphological reconstruction algorithm [18]. Then we apply the erosion operation with a disk of 10-pixel radius to remove lead curves and apply the dilation operation with the same disk radius to restore the filled object. At last we trace the boundary of filled objects. An example of the detected boundary is shown in Fig. 6(b).

4.1.4. Shaded area

Shaded area is a group of parallel line segments with uniform distance. We detect basic line segments in the binary image using a line segment detection algorithm [19]. We merge these basic

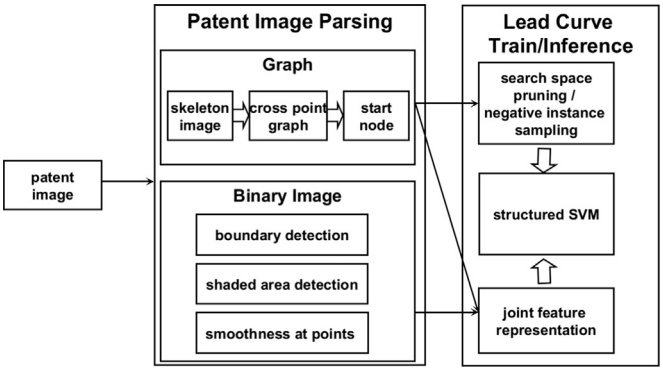


Fig. 5. Flowchart of the solution.

Table 1
Concept and notation table

$G=(V,E)$	Cross-point graph, which is an undirected graph, where V is the vertex set and E is the edge set
$\mathbf{y}$	A path in the cross-point graph, $\mathbf{y}=e_i\cdots e_j$, where $e_i\in E$
$\mathbf{x}$	A curve in the binary image, $\mathbf{x}=\mathbf{x}_i\cdots\mathbf{x}_j$
$\mathcal{M}(e_i)=\mathbf{x}_i$	Edge-to-curve mapping
$(\mathbf{x},\mathbf{y})$	Curve-path, which is a pair of a path and its corresponding curve
$\Psi(\mathbf{x},\mathbf{y})$	Joint feature representation
$\mathbf{w}$	Weighting vector of joint feature representation
$F(\mathbf{x},\mathbf{y})$	Scoring function of a curve-path, $F(\mathbf{x},\mathbf{y})=\mathbf{w}^\top\Psi(\mathbf{x},\mathbf{y})$

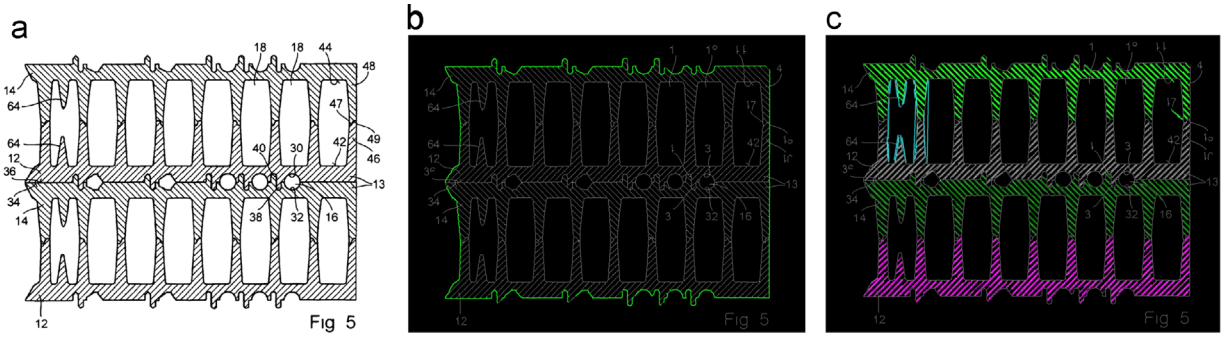


Fig. 6. An example of boundary detection and shaded area detection. (a) Patent image. (b)Detected boundary. (c) Detected shaded area.

line segments into one line segment if they lie on the same line and the distance is smaller than 20 pixels. To find a group of line segments of the same direction, we construct a line segment graph, in which each line segment is a node and an edge connects two nodes if the direction difference between these two line segments is smaller than $\pi/18$ and the distance between these two line segments is smaller than 50 pixels. On this graph, we extract connected components and preserve those that have more than 10 nodes as shaded areas. An example of the shaded area is shown in Fig. 6(c).

4.1.5. Smoothness at points/nodes

We compute the smoothness at point p on curve x by the direction difference between two curve segments. As shown in Fig. 7, there are two neighboring curve segments $x_{forward}$ and $x_{backward}$ at point p. We name the curve segment closer to the start-point as *backward edge* and the curve segment farther to the start-point as *forward edge*. Within the curve segment, directions change from point to point. We sample a point p_b on the backward edge at a fixed step from point p and also sample a point p_f on the forward edge at the same fixed step[2] from point p. The direction difference is the angle between vector $\overrightarrow{p_b p}$ and vector $\overrightarrow{pp_f}$. Since a node in the cross-point graph corresponds to a point in the image, we will use smoothness at points and smoothness at nodes interchangeably in the following description.

4.2. Joint feature representation design

In general, a geometric context property includes geometric property such as smoothness at inner nodes and the end node while context information includes object boundary, shaded area, other lead curves in the same image. The joint feature representation is defined on the curve-path to describe both the information in the cross-point graph and binary image. On the cross-point graph side, the joint feature representation will describe all the nodes in the path $\mathbf{y}$ including the inner node $\mathbf{y}_{inner}$ and the end node $\mathbf{y}_{end}$. On the binary image side, the joint feature representation will describe curve pixels at nodes $\mathbf{x}_n$, shaded area pixels $\mathbf{x}_s$ and object pixels $\mathbf{x}_o$. Shaded area pixels $\mathbf{x}_s$ are pixels of a particular shaded area region. Object pixels $\mathbf{x}_o$ are pixels on the boundary of an object. Putting both sides together, we get 6 meaningful types of joint feature representations in Table 2.

4.2.1. Smoothness at inner nodes $\Psi(\mathbf{x}_n, \mathbf{y}_{inner})$

It is designed to characterize the smoothness on the inner nodes of a curve-path. As shown in Fig. 8(a), a cross-point is node B. Suppose that the lead curve is $A \rightarrow B \rightarrow C$ and the backward edge at B is $A \rightarrow B$. We design a heuristic about the smoothness of inner nodes: *A lead curve usually goes along the direction of the most smooth forward edge.*(here, forward edge $B \rightarrow C$ connects to $A \rightarrow B$ more smoothly than forward edge $B \rightarrow D$).

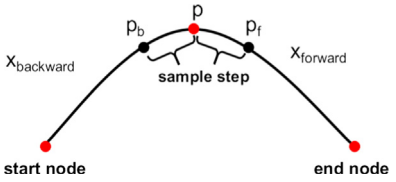


Fig. 7. Calculation of smoothness

We calculate the percentage of inner nodes that comply with this heuristic. Using percentage makes the feature invariant to the length of the path. However, directly using the percentage form for $\Psi(\mathbf{x}_n, \mathbf{y}_{inner})$ does not make sense since the importance of the feature does not grow linearly with its value. Consider the following situation: suppose the percentage value 50% is important while the percentage value 90% is not important. Weighting this percentage in training becomes a problem. If it assigns a large weight to this feature, then value 90% results in high score and vice versa. We need to slice the percentage equally in its range (i.e. $[0, 100]$) to different levels (e.g. 10 levels) and assign the percentage value to the corresponding level. Then we get a 10-dimensional binary feature vector, each dimension corresponding to whether the percentage value is assigned to the level. The slicing technique for percentage value helps us to characterize which level of the percentage value is important and it will be also used in other types of joint feature representation.

4.2.2. Smoothness at the end node $\Psi(\mathbf{x}_n, \mathbf{y}_{end})$

It is designed to characterize the smoothness at the end node of a curve-path. A lead curve either ends at a node of degree 1 or a node of degree larger than 2. The situation is relatively easy when it ends at a node with degree 1 since the path cannot be extended from this node. However, the situation is a little complex if it ends at a node of degree larger than 2. Similar to $\Psi(\mathbf{x}_n, \mathbf{y}_{inner})$, we use smoothness as well between forward edge and backward edge to characterize an end node. As shown in Fig. 8(b), the cross-point is node B. Suppose the current path ends at B, but there is a misleading forward edge at node B: $B \rightarrow D$. We design a heuristic about smoothness of the end node: *A lead curve usually ends at a cross-point where any one of its forward edge candidates does not belong to it according to smoothness evaluation on backward edges.* (see in Fig. 8(b), for the forward edge candidate $B \rightarrow D$, the backward edge $C \rightarrow B$ connects to $B \rightarrow D$ more smoothly than $A \rightarrow B$. Thus $A \rightarrow B$ should stop at B.)

If the end node of the current path complies with the above heuristic, the feature value is 1 and 0 vice versa. Thus $\Psi(\mathbf{x}_n, \mathbf{y}_{end})$ is a 1-dimensional binary vector.

[2] The step is set to 10 in our implementation.

Table 2
Joint feature representations.

$\Psi(\mathbf{x}_n, \mathbf{y}_{inner})$	Smoothness at inner nodes
$\Psi(\mathbf{x}_n, \mathbf{y}_{end})$	Smoothness at the end node
$\Psi(\mathbf{x}_o, \mathbf{y}_{end})$	Position between the end node and the object boundary
$\Psi(\mathbf{x}_s, \mathbf{y})$	Direction difference from the shaded area
$\Psi(\mathbf{x}_o, \mathbf{y})$	Angle difference between the curve parts inside and outside the object boundary
$\Psi(\mathbf{x}, \cdot)$	Difference from average length

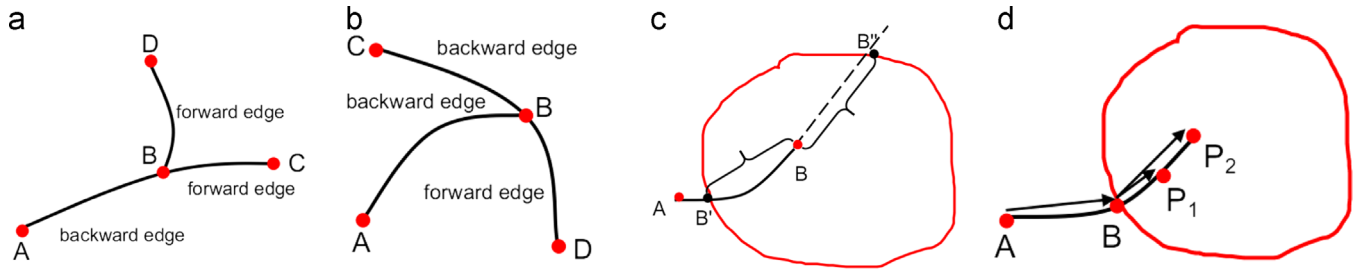


Fig. 8. Illustration of joint feature representation (a) $\Psi(\mathbf{x}_n, \mathbf{y}_{inner})$, (b) $\Psi(\mathbf{x}_n, \mathbf{y}_{end})$, (c) $\Psi(\mathbf{x}_o, \mathbf{y}_{end})$, and (d) $\Psi(\mathbf{x}_o, \mathbf{y})$. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this paper.)

4.2.3. Position between the end node and the object boundary $\Psi(\mathbf{x}_o, \mathbf{y}_{end})$

It is designed to characterize the position relationship between the end node and the object's boundary. Usually people choose to draw the lead curve from the side of the drawing which is closer to its referenced object part. As shown in Fig. 8(c), the red curve is the boundary of the object. Lead curve $A \rightarrow B$ crosses the object boundary at B'. If we extend the lead curve $A \rightarrow B$, to cross the other side of the boundary at B'', the length of BB' should be smaller than BB''. Otherwise, people can label the detail part B from another side of the drawing. Based on this observation, we design a heuristic about the position of the end node:

A lead curve usually ends at an inner point inside the object boundary which is nearer to the crossed boundary.

The ratio of BB' to BB'' characterizes this heuristic. If it is greater than 1, lead curve should start from the other side of the drawing. We get a binary feature value based on whether the ratio is greater than 1. As a result, $\Psi(\mathbf{x}_o, \mathbf{y}_{end})$ is a 1-dimensional binary feature vector.

4.2.4. Direction difference from the shaded area $\Psi(\mathbf{x}_s, \mathbf{y})$

It is designed to characterize the direction difference between the curve and the line segments in the shaded area. When a lead curve crosses a shaded area, the direction of the curve is usually very different from the direction of line segments in the shaded area, which makes it distinguishable from the shaded area. For each node that belongs to the shaded area, we calculate the direction difference between the curve and the line segment. The angle between the curve and the shaded area is calculated by the minimum of the direction difference on all these nodes. Based on this observation, we design a heuristic about the direction difference between the lead curve and the shaded area:

When a lead curve crosses a shaded area, the directions of the curve at cross-points are significantly different from those of line segments in the shaded area.

If the curve does not cross the shaded area, the angle is set to $\pi/2$. Otherwise, the calculated angle ranges from 0 to $\pi/2$. Note that we are computing the angle between an undirected line and a curve, thus the maximum of the angle is $\pi/2$ rather than π. We tune the angle on the validation set and get a binary feature by thresholding it at $\pi/6$. As a result, $\Psi(\mathbf{x}_s, \mathbf{y})$ is a 1-dimensional binary feature vector. The value is 1 when the angle is larger than $\pi/6$ and vice versa.

4.2.5. Angle difference between the curve parts inside and outside the object boundary $\Psi(\mathbf{x}_o, \mathbf{y})$

Although the situation becomes complex and more misleading after the lead curve enters the object boundary, the lead curve part within the object boundary does not deviate a lot from the part outside the object boundary. As shown in Fig. 8(d), the start-point is A and the cross boundary point is B. We get $\overrightarrow{AB}$ as the direction of the lead curve part outside the object boundary. For the lead curve part inside the object boundary, we sample points P_i by 20-pixel step and calculate the angle difference between $\overrightarrow{AB}$ and $\overrightarrow{BP_i}$. The deviation between the part within the object boundary and that outside the object boundary is calculated by the maximum of these angles. Based on this observation, we design a heuristic about the deviation of the lead curve part within the object boundary from the part outside the object boundary:

The lead curve part within the object boundary does not deviate significantly from the part outside the object boundary.

If the curve does not cross the object boundary, the angle is set to 0. Otherwise, the calculated angle ranges from 0 to π. We tune the angle on the validation set and get a binary feature by thresholding it at $\pi/6$. As a result, $\Psi(\mathbf{x}_o, \mathbf{y})$ is a 1-dimensional binary feature vector.

4.2.6. Difference from average length $\Psi(\mathbf{x}, \cdot)$

The lead curves in the same image have similar length. If we have a good estimation of most lead curves' length in the image, this estimation can be used to help improve predictions on lead curves that are difficult[3]. Thus, we use the average length of baseline predictions within an image as an estimation of the standard length of a lead curve in this image. Based on this observation, we design a heuristic about the length of a lead curve:

The length of a single lead curve does not deviate significantly from the average length of lead curves in the same image.

We calculate the ratio between the current lead curve length and the standard length of a lead curve. We slice the ratio value to three levels: $[0, 0.25)$, $[0.25, 2)$, $(2, \infty)$ after tuning on the validation set. As a result, $\Psi(\mathbf{x}, \cdot)$ is a 3-dimensional binary feature vector.

[3] As shown in the experimental section, the baseline method has a decent performance.

4.3. Inference

In the inference stage, we start from the given start-node to find a path that maximizes $F(\mathbf{x},\mathbf{y})$. The computational cost of a brute-force solution is extremely high for two reasons. First, the number of possible paths grows exponentially with the number of nodes in the cross-point graph. Second, the computational cost of $F(\mathbf{x},\mathbf{y})$ cannot be neglected if it is called many times.

For the issue of possible exponentially growth of paths, we solve it by pruning the search space. There are three hints that help us to prune the majority part of the search space and simultaneously ensure that the correct path is preserved in the search space.

Hint 1: the length of a lead curve is smaller than half of the image diameter. Thus, we can neglect nodes whose distance is larger than half of the image diamater. This helps to remove around 3/4 nodes in the cross-point graph.

Hint 2: for nodes on the groundtruth lead curve path, their shortest paths to the start-node are usually subpaths of the groundtruth path. Thus, we can only consider edges on the shortest path from the start-node to the remaining nodes. This greatly reduces the amount of edges to a number around the number of nodes. Furthermore, the remaining edges form a tree rather than a graph with circles, which reduces the search space of possible paths greatly. We denote this space as dijkstra space since all the edges are found by the single source shortest path dijkstra algorithm. An example is shown in Fig. 9(a).

Hint 3: the only exception of hint 2 is the situation shown in Fig. 9(b). In this situation, the shortest path is not a subpath of the groundtruth path. Thus, we utilize greedy forward search of a path from the start-node. In each step of the greedy forward search, we choose the most smooth edge on each cross-point node. We add edges in this path to the search space and denote this space as the augmented dijkstra space. An example is shown in the circle area of Fig. 9(c).

For the computation cost of repeated calling $F(\mathbf{x},\mathbf{y})$, we solve it by dynamic programming, which reduces the average cost of $F(\mathbf{x},\mathbf{y})$ to $O(1)$.

4.4. Training

Assume that we are given a collection of binary images of lead curves $\{\mathbf{x}^{(i)}\}$ and corresponding paths $\{\mathbf{y}^{(i)}\}$ in cross-point graph. We want to learn the weighting $\mathbf{w}$ from the $(\mathbf{x}^{(i)},\mathbf{y}^{(i)})_{i=1,\ldots,n}$ pairs. An image can have more than one lead curves and we let the superscript i run on the lead curves. First, we define the loss function to quantify the loss associated with a path candidate $\mathbf{y}$ given the groundtruth path $\mathbf{y}^{(i)}$.

$$\Delta(\mathbf{y}^{(i)},\mathbf{y}) = 1 - \frac{|\mathcal{M}(\mathbf{y}^{(i)}) \cap \mathcal{M}(\mathbf{y})|}{|\mathcal{M}(\mathbf{y}^{(i)}) \cup \mathcal{M}(\mathbf{y})|} \tag{3}$$

We measure the loss function by the relative difference between curve pixel sets of two paths, where the numerator means the intersection of curve pixels of $\mathbf{y}^{(i)}$ and $\mathbf{y}$ while the denominator is the union of pixels of $\mathbf{y}^{(i)}$ and $\mathbf{y}$. When $\Delta(\mathbf{y}^{(i)},\mathbf{y})$ is 0, it means that path $\mathbf{y}$ is same as the ground truth path $\mathbf{y}^{(i)}$ measured by curve pixels.

Our learning scenario is supervised: given the loss function, our goal is to find a hypothesis function $f(\mathbf{y})$ to minimize the risk over the distribution $P(\mathbf{x},\mathbf{y})$ of the labeled pair $(\mathbf{x}^{(i)},\mathbf{y}^{(i)})$.

$$\mathcal{R}_P^{\Delta}(f) = \int_{\mathcal{X}\times\mathcal{Y}} \Delta(\mathbf{y}, f(\mathbf{y}))\mathrm{d}P(\mathbf{x},\mathbf{y}) \tag{4}$$

In our problem, the form of hypothesis function $f(\mathbf{y})$ is closely related to the score function $F(\mathbf{x},\mathbf{y})$ on curve-path.

$$\begin{aligned} f(\mathbf{y}) &= \underset{\mathbf{y}\in\mathcal{Y}}{\operatorname{argmax}} \mathbf{w}^{\top}\Psi(\mathbf{x},\mathbf{y}) \\ &= \underset{\mathbf{y}\in\mathcal{Y}}{\operatorname{argmax}} F(\mathbf{x},\mathbf{y}) \end{aligned} \tag{5}$$

For the learned score function $F(\mathbf{x},\mathbf{y})$, we require that it should output higher score on the groundtruth path $\mathbf{y}^{(i)}$ than any other path $\mathbf{y}\in\mathcal{Y}$.

$$\forall i \in \{1,\ldots,n\}, \forall \mathbf{y}\in\mathcal{Y} : \mathbf{w}^{\top}(\Psi(\mathbf{x}^{(i)},\mathbf{y}^{(i)}) - \Psi(\mathbf{x}^{(i)},\mathbf{y})) \geq 0 \tag{6}$$

This constraint is the separable case in hard max-margin structured SVM. After applying similar techniques in loss-sensitive slack margin structured SVM [20], we get the following optimization problem for the training stage:

$$\begin{aligned} &\underset{\mathbf{w},\xi_i\geq\mathbf{0}}{\operatorname{argmin}} \mathbf{w}^{\top}\mathbf{w} + C\sum_i \xi_i \\ &\quad s.t. \forall i \in \{1,\ldots,n\}, \forall \mathbf{y}\in\mathcal{Y}, \\ &\qquad \mathbf{w}^{\top}\delta\Psi(\mathbf{x}^{(i)},\mathbf{y}^{(i)},\mathbf{y}) \geq \Delta(\mathbf{y}^{(i)},\mathbf{y}) - \xi_i \end{aligned} \tag{7}$$

where $\delta\Psi(\mathbf{x}^{(i)},\mathbf{y}^{(i)},\mathbf{y}) = \Psi(\mathbf{x}^{(i)},\mathbf{y}^{(i)}) - \Psi(\mathbf{x}^{(i)},\mathbf{y})$. On the right side of the constraint, we introduce the slack variable ξ_i and the loss function $\Delta(\mathbf{y}^{(i)},\mathbf{y})$.

We customize the cutting plane algorithm in work [20] to solve the optimization problem in Eq. (7). As listed in Algorithm 1, we need to customize the $\hat{\mathbf{y}} = \operatorname{argmax}_{\mathbf{y}\in\mathcal{Y}} H(\mathbf{y})$ step. This step is actually mining the most contradicting negative instance. Similar to the inference stage, the space of possible negative instance grows exponentially with the number of nodes in the cross-point graph. We can sample the

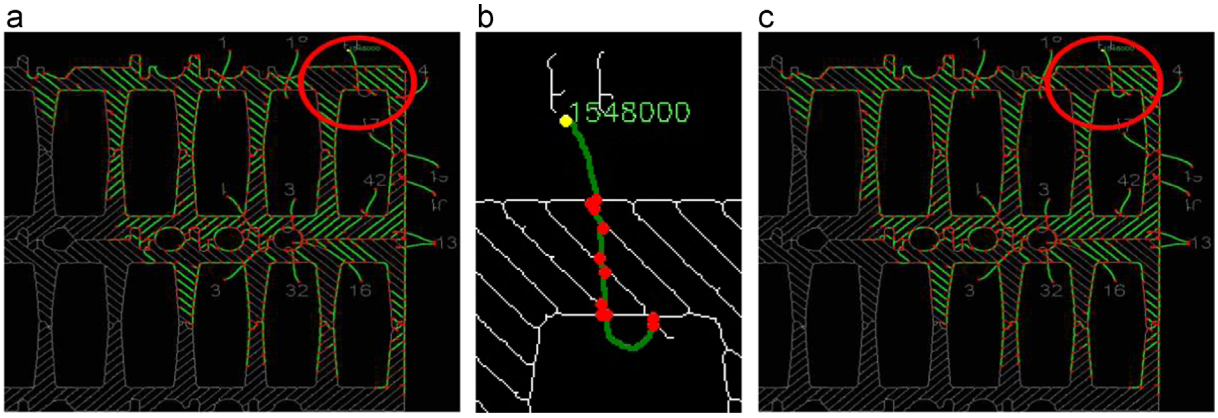


Fig. 9. Illustration of search space pruning: the start node is highlighted in yellow color; edges and nodes in the search space are highlighted in green and red color respectively. (a) Dijkstra space. (b) An exception of hint 2. (c) Augmented dijkstra space. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this paper.)

negative instance from this space. In particular, we focus on sampling hard negative instances. A strategy is to sample on the augmented dijkstra space introduced in the inference stage. This strategy is more focused on negative instances that are hard to be distinguished from the groundtruth on inner node. We denote this sampling strategy as *dijkstra sampling strategy*. We introduce another sampling strategy that is focused on negative instances which are hard to be distinguished from the groundtruth on end node. In this strategy, we extend the groundtruth lead curve by following the most smooth edge. Then we only sample different paths from this extended path. We denote this sampling strategy as *extended path sampling strategy*. These two sampling strategies are randomly mixed in line 6 of Algorithm 1. That is, in each call of line 6, there is 50% chance to run dijkstra sampling strategy and 50% chance to run extended path sampling strategy. We mix these two strategies randomly rather than merge the underlying search spaces directly because in the merged space there are a number of circles which lead to too many candidate paths.

Algorithm 1. Cutting plane algorithm for learning weightings $\mathbf{w}$ of joint feature representation.

input : $(\mathbf{x}^{(1)}, \mathbf{y}^{(1)}), \cdots (\mathbf{x}^{(n)}, \mathbf{y}^{(n)}), C, \epsilon$
output: weightings $\mathbf{w}$
1 $S_i \leftarrow \phi$ for all $i = 1, \cdots, n$;
2 **repeat**
3 **for** $i = \ , \cdots, n$ **do**
4 $H(\mathbf{y}) = \Delta(\mathbf{y}^{(i)}, \mathbf{y}) - \mathbf{w}^\top \delta\Psi(\mathbf{x}^{(i)}, \mathbf{y}^{(i)}, \mathbf{y})$;
5 where $\mathbf{w} = \sum_j \sum_{\mathbf{y}' \in S_j} \alpha_{(j\mathbf{y}')} \delta\Psi(\mathbf{x}^{(i)}, \mathbf{y}^{(i)}, \mathbf{y}')$;
6 compute $\hat{\mathbf{y}} = \arg\max_{\mathbf{y} \in \mathcal{Y}} H(\mathbf{y})$;
7 compute $\xi_i = \max\{0, \max_{\mathbf{y} \in S_i} H(\mathbf{y})\}$;
8 **if** $H(\hat{\mathbf{y}}) > \xi_i + \epsilon$ **then**
9 $S_i \leftarrow S_i \cup \{\hat{\mathbf{y}}\}$;
10 $\alpha_S \leftarrow$ optimize the dual of SVM over $S, S = \cup_i S_i$;
11 **until** *no* S_i *has changed during iteration*;

5. Experiment

5.1. Data collection and labeling

To the best of our knowledge, there is no public dataset available for lead curve evaluation. We collect patent images from American Patent Agency website[4]. On the one hand, patent images have complex cross-points, which makes lead curve detection a very challenge task. On the other hand, lead curve detection in the patent image has real-world impact. In our dataset, there are 175 images and 1890 lead curves in total. We make it publicly available at url http://lm.apexlab.org/patent-lead-curve/, including the patent images, extracted cross-point graph and labeled lead curves.

We categorize the difficulty of lead curve detection to three difficulty levels: easy, medium and hard. In the easy level, the lead curve goes through less than 3 cross-points and it does not enter the inner part of the object. In the medium level, the lead curve goes through more than 3 cross-points and enters the inner part of the object, but it does not enter the shaded area. In the hard level, the lead curve goes through more than 3 cross-points and enters not only the inner part of the object but also the shaded area. Three examples belonging to different difficult levels are shown in Fig. 10. In our dataset, the easy, medium and hard level proportions p_l are 44%, 37%, 19% respectively. That is, more than half of lead curves belong to medium and hard levels. We split the image

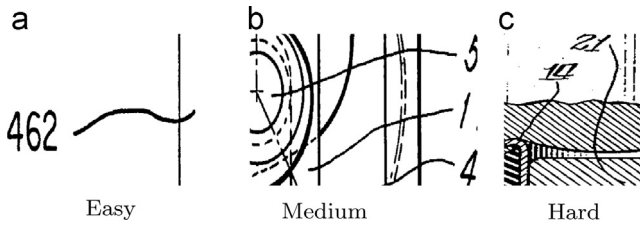


Fig. 10. Illustration of three difficulty levels. (a) Easy. (b) Medium. (c) Hard.

dataset equally into four subsets. In each subset the proportion of the three levels is kept similar to p_l. We randomly select one subset as validation set. On the remaining three subsets, we do 3-fold cross-validation with two subsets as training sets and one subset as test set in each fold.

5.1.1. Data labeling

To label the groundtruth, we build a software whose interface is shown in Fig. 11(a). On the left is a list of patent drawings; on the right is the labeling interface. The basic labeling unit is a curve segment $\mathbf{x}_i$, which corresponds to an edge e_i in the cross-point graph. Users select curve segments that belong to lead curve. As shown in Fig. 11(a), the curve segment is highlighted in yellow color when users are selecting it and the curve segment is shown in green color after selection. Small groundtruth curve segments may be neglected if the user could only select curve segments one by one. This leads to the broken groundtruth lead curve issue. To solve this issue, we introduce the batch mode for selection/deselection operation. Fig. 11(b) and Fig. 11(c) shows an example of labeling in batch mode. First, users select curve segments including the groundtruth. Then they deselect the curve segments that do not belong to the groundtruth. The batch approach ensures that the selected curve segments are connected and accelerates the labeling process. The batch approach may introduce false groundtruth curve segments which are not removed by deselection operations. Thus, we refine the groundtruth by searching the longest labeled path from the start point. An example of refined groundtruth lead curves are shown in Fig. 11(d): each start point is shown in yellow color and each groundtruth lead curve is shown in green color. Using this software, we collect high quality groundtruth.

5.2. Evaluation metric and baseline

We evaluate the performance of our model based on curve pixels. Based on the notation introduced in Section 3, the curve pixel set of the groundtruth lead curve $\mathbf{y}$ is $\mathcal{M}(\mathbf{y})$ and that of the predicted lead curve $\bar{\mathbf{y}}$ is $\mathcal{M}(\bar{\mathbf{y}})$. We adopt three common metrics: precision $(p = |\mathcal{M}(\mathbf{y}) \cap \mathcal{M}(\bar{\mathbf{y}})| / |\mathcal{M}(\bar{\mathbf{y}})|)$, recall $(r = |\mathcal{M}(\mathbf{y}) \cap \mathcal{M}(\bar{\mathbf{y}})| / |\mathcal{M}(\mathbf{y})|)$ and F1 $(f1 = 2pr/p+r)$. Higher precision means that the predicted lead curve has smaller branch part from the lead curve path but it may branch/stop early on the groundtruth lead curve as shown in Fig. 12(a). Higher recall means that the predicted lead curve branches/stops later from the groundtruth lead curve but it may have an additional large branch part as shown in Fig. 12(b). Higher F1 score means that the predicted lead curve branches/stops later from the groundtruth lead curve with a smaller branch part as shown in Fig. 12(c).

To the best of our knowledge, we are the first to solve lead curve detection problem in complex cross-points situation. There is no state-of-the-art algorithms for comparison. Instead we exploit a heuristic style algorithm as the baseline for comparison. The baseline algorithm works as follows: it starts from the start node and it chooses to go on a forward edge that is the most smooth to the current backward edge on each cross-point. It stops either when the angle between the backward edge and the

[4] http://www.uspto.gov/

forward edge is larger than the threshold θ or when the pixel number of the curve is larger than the threshold l. θ and l are two parameters in the baseline. We tune them upon the validation dataset and use the best parameter in test. To be specific, we tune θ among $15^\circ, 30^\circ, 45^\circ$ and l among $250, 500, 750$ pixels on the validation set. The best configuration is $\theta = 30^\circ$ and $l = 500$ pixels.

5.3. Train and test on different settings

We design four training settings and four test settings: train/test on data of each difficulty level respectively and train/test on data of all difficulty levels. We denote the baseline as *BASELINE* and models trained on easy, medium, hard levels and all levels as $STRUCT_{easy}$, $STRUCT_{medium}$, $STRUCT_{hard}$, $STRUCT_{all}$, respectively. We evaluate these four methods on four test settings and list the result on the three metrics in Table 3. Each row corresponds to a method and each column corresponds to a metric. Columns are grouped by test settings. On all metrics and all the settings, our models trained on easy, medium, hard and all settings all outperform the baseline. On the all test setting, the best model is $STRUCT_{all}$, which outperforms the baseline by 7.2% on precision, 1.4% on recall and 6.2% on F1. On the easy test setting, the best model is $STRUCT_{all}$, which outperforms the baseline by 4.0% on precision, 0.5% on recall and 3.6% on F1. On the medium test setting, the best model is $STRUCT_{all}$, which outperforms the baseline by 7.8% on precision, 1.0% on recall and 6.1% on F1. On the hard test setting, the best model is $STRUCT_{all}$, which outperforms the baseline by 7.6% on precision, 2.5% on recall and 7.3% on F1. The improvement over the baseline on F1 metric increases as the difficulty level increases, which shows that our model can handle complex cross-points situation well. Overall, $STRUCT_{all}$ is the best model. Paired t-test shows that the improvement by $STRUCT_{all}$ is significant on all metrics and all the settings at $\alpha = 0.01$. We also show examples for the predicted results of lead curve on different difficulty levels in Fig. 13.

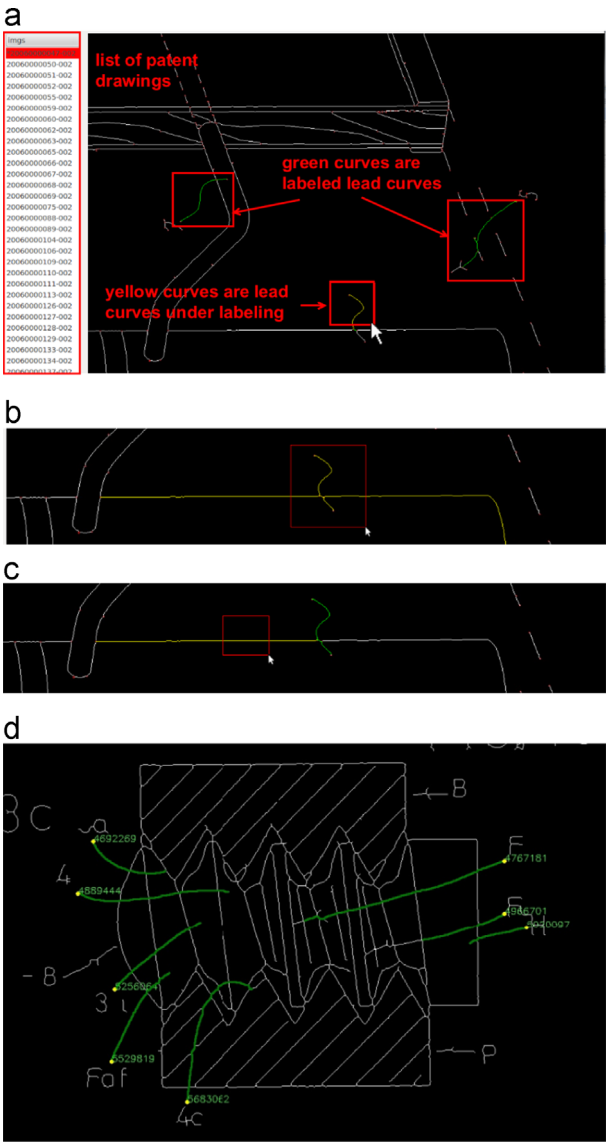


Fig. 11. Data labeling interface. (a) Interface for labeling lead curve. (b) Bat selection. (c) Deselection. (d) Refined groundtruth lead curve. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this paper.)

5.4. Study on joint feature representation

We study the influence of each type of joint feature on the final performance. To be specific, we start with the two basic joint features, smoothness at inner nodes $\Psi(\mathbf{x}_n, \mathbf{y}_{inner})$ and smoothness at the end node $\Psi(\mathbf{x}_n, \mathbf{y}_{end})$, and add other joint features such as position between the end node and the object boundary $\Psi(\mathbf{x}_o, \mathbf{y}_{end})$, direction difference from the shaded area $\Psi(\mathbf{x}_s, \mathbf{y})$, angle difference between the curve parts inside and outside the object boundary $\Psi(\mathbf{x}_o, \mathbf{y})$, difference from the average length $\Psi(\mathbf{x}, \cdot)$ sequentially. As shown in Table 4, with each feature added, the precision increases significantly while the recall drops slightly. The F1 metric increases steadily. Note that on the initial feature set $\Psi(\mathbf{x}_n, \mathbf{y}_{inner})$ and $\Psi(\mathbf{x}_n, \mathbf{y}_{end})$, the precision is 0.894, much lower than the recall 0.968. When all the features are added, the precision increases to 0.933 while the recall drops to 0.949. The significant improvement in precision shows that most parts of the predicted lead curve are in the groundtruth while the slight drop in the recall shows that the coverage of groundtruth is maintained. Additionally, we do case studies on the performance boost from adding each joint feature in Fig. 14. In Fig. 14(a), we see that the predicted lead curve stops properly with considering the position of boundary after the boundary feature is introduced. Fig. 14(b) shows that lines in the shaded area are not included in the predicted lead curve after the shaded area feature is introduced. The part of predicted lead curve inside the object will not deviate a lot in angle from the part outside the object as shown in Fig. 14(c). Fig. 14(d) shows that introducing the average estimated length of lead curves in the same image improves the prediction performance.

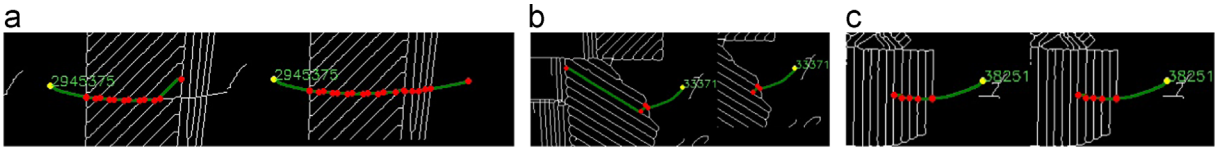


Fig. 12. Illustration of high precision, high recall and high f1: the left image is the predicted lead curve $\bar{\mathbf{y}}$ and the right image is the groundtruth lead curve $\mathbf{y}$. (a) High precision ($p = 0.829$, $r = 0.568$, $f1 = 0.674$). (b) High recall ($p = 0.351$, $r = 1.000$, $f1 = 0.519$). (c) High f1 ($p = 1.000$, $r = 1.000$, $f1 = 1.000$).

Table 3
Train and test on different settings: the best score on metric is emphasized in bold font.

train / test	all			easy			medium			hard		
	p	*r*	*f*1	*p*	*r*	*f*1	*p*	*r*	*f*1	*p*	*r*	*f*1
BASELINE	0.861	0.935	0.866	0.908	0.981	0.926	0.868	0.934	0.870	0.833	0.917	0.837
$STRUCT_{easy}$	**0.944**	0.916	0.912	**0.963**	0.952	0.948	0.955	0.905	0.911	**0.922**	0.916	0.899
$STRUCT_{medium}$	0.942	0.920	0.914	0.961	0.957	0.950	**0.956**	0.910	0.916	0.918	0.918	0.899
$STRUCT_{hard}$	0.943	0.919	0.914	0.962	0.957	0.951	**0.956**	0.909	0.915	0.918	0.918	0.899
$STRUCT_{all}$	0.933	**0.949**	**0.928**	0.948	**0.986**	**0.962**	0.946	**0.944**	**0.931**	0.909	**0.942**	**0.910**

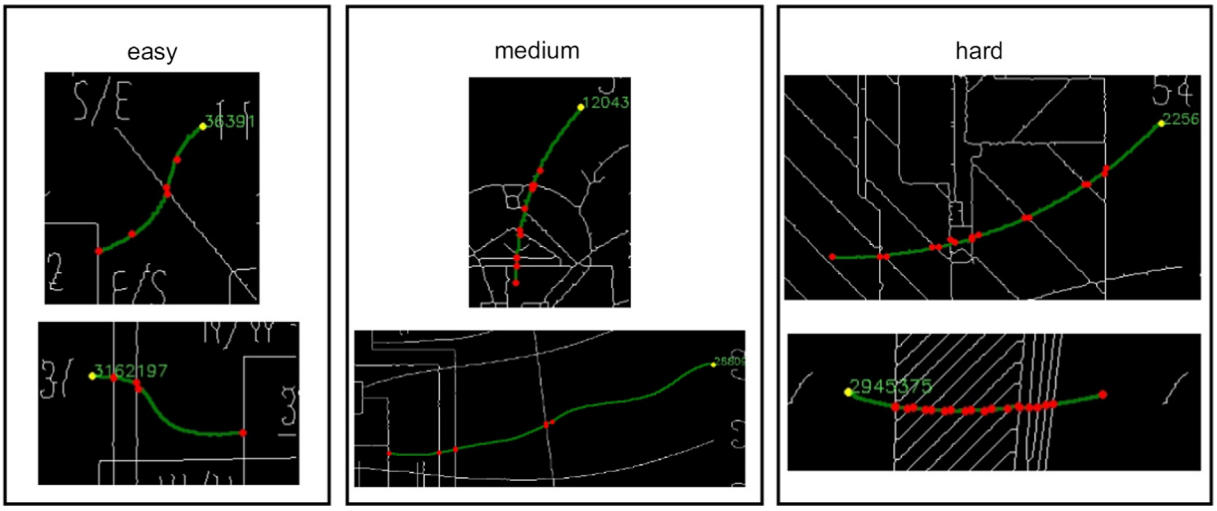


Fig. 13. Case study of predicted lead curves of different difficulty levels: the predicted curve is shown in green color; the start node is shown in yellow color; other nodes in the path is shown in red color. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this paper.)

Table 4
Study on joint feature representation by adding each feature sequentially.

Added features	*p*	*r*	*f*1
Smoothness at inner nodes and end node $\Psi(\mathbf{x}_n, \mathbf{y}_{inner})$, $\Psi(\mathbf{x}_n, \mathbf{y}_{end})$	0.878	0.97	0.903
Position between end node and object boundary $\Psi(\mathbf{x}_o, \mathbf{y}_{end})$	0.894	0.968	0.912
Direction difference from shaded area $\Psi(\mathbf{x}_s, \mathbf{y})$	0.904	0.967	0.919
Angle difference between the curve parts inside and outside the object boundary $\Psi(\mathbf{x}_o, \mathbf{y})$	0.924	0.953	0.922
Difference from average length $\Psi(\mathbf{x}, \cdot)$	0.933	0.949	0.928

5.5. Study on negative instance sampling strategy in training

As discussed in Section 4.4, we introduce three negative instance sampling strategies: dijkstra sampling, extended path sampling and randomly mixed sampling. As shown in Table 5, the randomly mixed sampling strategy performs better than the other two strategies. We attribute the improvement to the fact that the mixed sampling strategy brings different kinds of negative instances to the training process.

5.6. Robustness of the model

In our training algorithm, there is one parameter C, which is the tradeoff between the training error and the margin to the classification boundary. We tune C among 0.01, 0.1, 1, 10 and corresponding F1 scores are 0.928, 0.915, 0.919, 0.909. This shows the robustness of our model with respect to the parameter C.

6. Discussion

Lead curve detection technique has a wide range of applications. We categorize its applications to two types: cleaning image and parsing image. In the cleaning image type application, the lead curve detection technique can be used to remove the lead curve from the drawing to get the original clean design. Cleaned drawing itself is useful in many situations such as getting a cleaned drawing of a room design. Furthermore, cleaned drawings will significantly boost the performance of content-based drawing retrieval task ranging from duplication detection of design patent to finding similar room designs. In the parsing image type application, we give two examples of exploiting the two different ends of the detected lead curves. For the reference character end of the lead curve, detected lead curves can be used to ease the manual understanding process of the drawing by highlighting the corresponding lead curve when the user select the reference characters in the text. For the object part end of the lead curve, we can highlight the object parts as salient regions for users that start browsing the drawing. The parsed image type application makes the drawing more interactive based on the detected lead curves.

7. Conclusion

In this paper, we address the task of lead curve detection in drawings, which is a key problem in a wide range of applications,

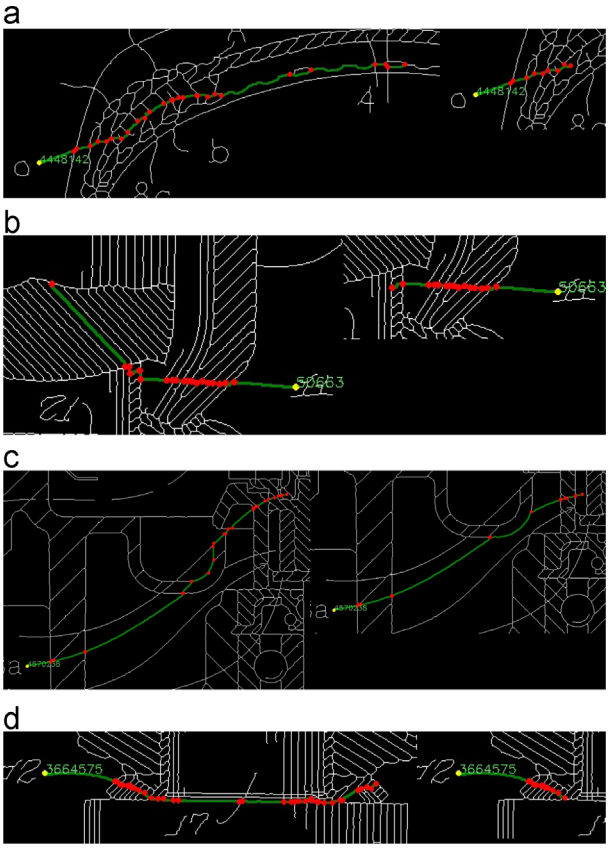


Fig. 14. The left image and the right image contrast the predicted lead curves *before* and *after* adding the feature. (a) Adding position between end node and object boundary $\Psi(\mathbf{x}_o, \mathbf{y}_{end})$. (b) Adding direction difference from shaded area $\Psi(\mathbf{x}_s, \mathbf{y})$. (c) Adding angle difference between the curve parts inside and outside the object boundary $\Psi(\mathbf{x}_o, \mathbf{y})$. (d) Adding difference from average length $\Psi(\mathbf{x}, \cdot)$.

Table 5
Study on the influence of negative instance sampling strategy.

Sampling strategy	p	r	$f1$
Dijkstra	0.946	0.913	0.911
Extended path	0.565	0.782	0.622
Mixed	0.933	0.949	0.928

such as removing lead curves to improve patent image retrieval quality and constructing hyperlinks between the object part in the drawing image and the detailed explanation in the text. The difficulty of the problem lies in unknown end point of lead curve and complex cross-points while most previous curve detection algorithms only focus on simple or no cross-points situation. We transform the problem into a problem of finding the best path in a graph given the start node. In the transformation, we introduce the cross-point graph representation, which captures the topology of cross-point connectedness, and the coupling concept curve-path, which is a pair of the curve in the binary image and the corresponding path in the cross-point graph. Then, we design a set of joint feature representations for a curve-path, including smoothness at points, position relationship between the end node and object boundary, direction difference between the curve and the shaded area, angle deviation of the curve part inside the object from the curve part outside the object and the length difference between a lead curve and other lead curves in the same image. We further define a task specific loss function and customize structured SVM to learn the weightings of joint feature representations and to predict the lead curve. We design a mixed hard negative instance sampling strategy in the training stage and prune the search space significantly in the inference stage. Finally, we collect precise groundtruth by building a software to facilitate manual labeling and make the dataset publicly available. In experiments, we show that our model improves the baseline significantly. We study the effect of each type of joint feature representation and analyze the influence of negative instance sampling strategy. Our model achieves a performance of 0.928 at F1 metric, which is considered as meeting the need of real applications.

Acknowledgments

This research was partially undertaken on the Fundamental Research Funds for the Central Universities and the Research Funds of Renmin University of China (No. 14XNLQ01), the Beijing Natural Science Foundation (No. 4142029).

References

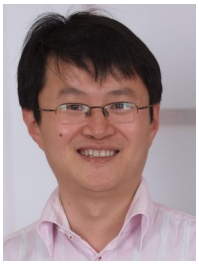
[1] U.S. patent statistics chart. ⟨http://www.uspto.gov/web/offices/ac/ido/oeip/taf/us_stat.htm⟩.
[2] Standard for Drawings. ⟨http://www.uspto.gov/patents/resources/general_info_concerning_patents.jsp⟩.
[3] M. Bozzini, M. Rossini, The detection and recovery of discontinuity curves from scattered data, J. Comput. Appl. Math. 240 (2013) 148–162.
[4] V. Kaul, A.J. Yezzi, Y.J. Tsai, Detecting curves with unknown endpoints and arbitrary topology using minimal paths, IEEE Trans. Pattern Anal. Mach. Intell. 34 (10) (2012) 1952–1965.
[5] Y.R. Huang, C.M. Kuo, C.-H. Hsieh, Head–shoulder moving object contour tracking using shape model, in: JCIS, 2006.
[6] X. Zhang, H. Wang, M. Hong, L. Xu, D. Yang, B.C. Lovell, Robust image corner detection based on scale evolution difference of planar curves, Pattern Recognit. Lett. 30 (4) (2009) 449–455.
[7] S. Lee, Y. Liu, Curved glide-reflection symmetry detection, IEEE Trans Pattern Anal. Mach. Intell. 34 (2) (2012) 266–278.
[8] T.P. Nguyen, I. Debled-Rennesson, Fast and robust dominant points detection on digital curves, in: ICIP, 2009, pp. 953–956.
[9] X. Bai, X. Wang, L.J. Latecki, W. Liu, Z. Tu, Active skeleton for non-rigid object detection, in: ICCV, 2009, pp. 575–582.
[10] M. Kass, A.P. Witkin, D. Terzopoulos, Snakes: active contour models, Int. J. Comput. Vis. 1 (4) (1988) 321–331.
[11] V. Caselles, R. Kimmel, G. Sapiro, Geodesic active contours, Int. J. Comput. Vis. 22 (1) (1997) 61–79.
[12] C. Xu, J.L. Prince, Snakes, shapes, and gradient vector ow, IEEE Trans. Image Process. 7 (3) (1998) 359–369.
[13] X. Han, C. Xu, J.L. Prince, A topology preserving level set method for geometric deformable models, IEEE Trans. Pattern Anal. Mach. Intell 25 (6) (2003) 755–768.
[14] C. Lu, L.J. Latecki, N. Adluru, X. Yang, H. Ling, Shape guided contour grouping with particle filters, in: ICCV, 2009, pp. 2288–2295.
[15] X. Yang, N. Adluru, L.J. Latecki, Particle filter with state permutations for solving image jigsaw puzzles, in: CVPR, 2011, pp. 2873–2880.
[16] L.J. Latecki, C. Lu, M. Sobel, X. Bai, Multiscale random fields with application to contour grouping, in: NIPS, 2008, pp. 913–920.
[17] W. Shen, Y. Wang, X. Bai, H. Wang, L.J. Latecki, Shape clustering: common structure discovery, Pattern Recognit. 46 (2) (2013) 539–550.
[18] P. Soille, Morphological Image Analysis: Principles and Applications, Springer-Verlag, New York, Inc. Secaucus, NJ, USA, 1999.
[19] R.G. von Gioi, J. Jakubowicz, J.-M. Morel, G. Randall, Lsd: a fast line segment detector with a false detection control, IEEE Trans. Pattern Anal. Mach. Intell. 32 (4) (2010) 722–732.
[20] I. Tsochantaridis, T. Joachims, T. Hofmann, Y. Altun, Large margin methods for structured and interdependent output variables, J. Mach. Learn. Res. 6 (2005) 1453–1484.

Jia Chen received double bachelor degrees on mathematics and computer science at Shanghai JiaoTong University in 2008. Now he is a Ph.D. candidate in department of computer science and engineering in Shanghai JiaoTong University. His research interests are in image annotation, content based image retrieval and machine learning.

Min Li received his B.Sc. degree in electronic information science and technology from University of Science and Technology of China (USTC) in 2004, and his M.sc. in electrical engineering from Institute of Electrical Engineering, Chinese Academy of Sciences (IEE, CAS) in 2007, and his Ph.D. degree in pattern recognition and intelligent system from Institute of Automation, Chinese Academy of Sciences (IA, CAS) in 2011. Dr. Li is a member of IEEE, and has served as reviewer of many major international conferences, such as IEEE Conference on Computer Vision and Pattern Recognition (CVPR), International Conference on Computer Vision (ICCV), IEEE International Conference on Image Processing (ICIP) and etc. He worked in IBM China Research from 2011 to early 2014, and then joined Alibaba Group in the mid of 2014. His research interests include visual surveillance, biometrics, image search and deep learning.

Dr. Qin Jin is currently an associate professor at Computer Science Department in School of Information at Renmin University of China (RUC) and is leading the Multimedia Computing Lab. Before joining RUC, Dr. Jin worked as a research scientist at IBM China Research Lab from 2012/04 to 2012/12 and a research faculty at Language Technologies Institute at Carnegie Mellon University (CMU) from 2007/02 to 2012/03. Dr. Jin received her Ph.D. in language and information technologies from Carnegie Mellon University in January 2007. Her main reserach interests include speech recognition and understanding, multimedia data analytics, natural language processing and machine learning in general. She is a member of APSIPA, IEEE, ISCA and ACM.

Shenghua Bao is currently manager of Knowledge Discovery and Analytics team in IBM Watson Group, where he leads the team to extract knowledge from a large variety of data sources and develop analytics services to enable cognitive literature understanding and discovery. Prior to that, Shenghua was co-lead of Global Technology Outlook 2014 at IBM T.J. Watson Research Center and research staff member at IBM Research-China. He joined IBM in 2008 after receiving his Ph.D. degree in Computer Science from Shanghai Jiao Tong University. His research interests lie primarily in web mining and text analytics. Shenghua has published more than 40 journal and conference papers and filed more than 30 US / international patent applications. Shenghua is an IBM Master Inventor.



Dr. Zhong Suis a Senior Technical Staff Member at the IBM Research - China (CRL) and Senior Manage of the Cognitive Understanding & Analytics department. He joined CRL after receiving his Ph.D. degree in computer science at Tsinghua University in 2002. He has been involved in many projects in CRL including text analytics, NLP, rich media analysis and information integration. He led a number of research projects including Chinese Knowledge Management (CKM), Market Intelligence Portal (MIP), Real Time Social Listening etc. His team has been awarded the Technical Accomplishment of IBM research for many times and Outstanding Technical Accomplishment of IBM research in 2008, 2010 and 2014. Dr. Su was awarded IBM Master Inventor in 2007 and 2013. He currently chairs the Invention and Disclosure Board in CRL. Dr. Su published more than 50 papers in top international conferences/journals, with more than 40 patents or patents pending. Dr. Su is adjunct professor of NanKai University, guest professor of APEX Lab, Shanghai JiaoTong University. He also takes the role of Vice Chairman of Technical Expert Council - IBM Greater China Group.

Yong Yugot his master degree at the Computer Science Department of East China Normal University. He began to work in Shanghai Jiao Tong University in 1986. Now he is the Ph.D. candidate tutor and the chairman of E-generation technology research center(SJTU-IBM-HKU). He was the teacher of the course “Computer Graphics and Human-machine Interface” and the course “Next Generation Web Infrastructure”. As the head coach of SJTU ACM-ICPC team, he and his team have won the 2002, 2005 and 2010 ACM ICPC Championships. His research interests include semantic Web, Web mining, information retrieval and computer vision.