IJDAR (2009) 12:65–81
DOI 10.1007/s10032-009-0081-0

ORIGINAL PAPER

Automated analysis of images in documents for intelligent document search

Xiaonan Lu · Saurabh Kataria · William J. Brouwer · James Z. Wang · Prasenjit Mitra · C. Lee Giles

Received: 14 January 2008 / Revised: 10 February 2009 / Accepted: 16 February 2009 / Published online: 16 April 2009

Abstract Authors use images to present a wide variety of important information in documents. For example, two-dimensional (2-D) plots display important data in scientific publications. Often, end-users seek to extract this data and convert it into a machine-processible form so that the data can be analyzed automatically or compared with other existing data. Existing document data extraction tools are semi-automatic and require users to provide metadata and interactively extract the data. In this paper, we describe a system that extracts data from documents fully automatically, completely eliminating the need for human intervention. The system uses a supervised learning-based algorithm to classify figures in digital documents into five classes: photographs, 2-D plots, 3-D plots, diagrams, and others. Then, an integrated algorithm is used to extract numerical data from data points and lines in the 2-D plot images along with the axes and their labels, the data symbols in the figure's legend and their associated labels. We demonstrate that the proposed system and its component algorithms are effective via an empirical evaluation. Our data extraction system has the potential to be a vital component in high volume digital libraries.

Keywords Image · Document search · Figure · 2-D plot · Data extraction · Text block extraction

X. Lu (✉) · J. Z. Wang · P. Mitra · C. L. Giles
Department of Computer Science and Engineering,
The Pennsylvania State University, University Park, USA
e-mail: xlu@cse.psu.edu

S. Kataria · J. Z. Wang · P. Mitra · C. L. Giles
College of Information Sciences and Technology,
The Pennsylvania State University, University Park, USA
e-mail: skataria@ist.psu.edu

J. Z. Wang
e-mail: jwang@ist.psu.edu

P. Mitra
e-mail: pmitra@ist.psu.edu

C. L. Giles
e-mail: giles@ist.psu.edu

W. J. Brouwer
Department of Chemistry, The Pennsylvania State University,
University Park, USA
e-mail: wjb19@psu.edu

1 Introduction

Authors frequently use images in documents to present important information. An image is commonly referred to as a figure within the embedding document. A figure can be created from a photographic picture, a graphics, a screen shot, a statistical plot, etc. The popularity of image search shows that end-users often seek to search for images and figures in documents.

Advanced information extraction, image processing, and indexing techniques that integrate image content with text can allow both keyword-based and image-based document queries. For example, if a user asks for documents with a specific topic and certain illustrations, documents that are most relevant in terms of both textual relevance and image relevance can be selected. Currently, almost all general-purpose search engines index textual information present in digital documents. They do not extract, analyze or index image content in such documents. With the increasing amount of non-textual information present in documents, it becomes important for search engines to utilize both text and image information so that they can better assisting end-users to find relevant documents.

Documents in the scientific domain often use images or figures presenting relations among data. Especially important to scientists is two-dimensional (2-D) plots that display the variation of a dependent variable, typically represented numerically along the Y-axis, as a function of another variable, typically represented numerically along the X-axis (although we have encountered special cases in environmental geochemistry where the opposite is true). Scientists, financial analysts, and other users would like the data published in documents and their associated metadata to be extracted and available in a form that is easily machine-processible. To enable better search of the data, the metadata is vital. End-users can run statistical processing tools to analyze the extracted data, identify past trends, forecast future behavior, simulate models using the extracted data, compare the extracted data with those obtained from their own experiments, etc.

Currently, end-users have to use semi-automatic tools to extract data from figures in digital documents. For example, there is anecdotal evidence to suggest that the National Institutes of Standards and Technology employs post-doctoral scholars to extract data from articles published in scholarly journals, using semi-automatic tools. The person using the tool points to one data point using a cursor and then the tool begins to identify similar points in its neighbourhood. Metadata about the X and Y axes and whether the data is a discrete variable (points) or a continuous variable (line curve) has to be entered manually. Ideally, we would like to avoid the manual labor involved in identifying the data points in 2-D plots and automate the process. To the best of our knowledge, no existing tool can both automate the process of identifying 2-D plots in digital documents and extract data and associated metadata from these plots. Semi-automatic methods do not scale when millions of documents in large digital libraries have to be processed and the information presented in figures in a large number of documents must be extracted.

Automated analysis of images within documents introduces important but challenging problems. First, the images must be extracted from the document. This process alone is difficult because documents are generated from a wide range of hardware and software platforms, and vary widely in quality. Second, we need to define a hierarchy of classes for images contained within documents with indexing and retrieval applications in mind. Third, because we are dealing with large digital libraries, images extracted from documents have to be classified into these pre-defined classes automatically. Images belonging to the same class may appear very different. Designing accurate classification algorithms is hard because of this diversity. Finally, designing an automated data and metadata extraction tool for 2-D plot images that segments mixed graphic and textual information, identifies data points, segregates overlapping data points, and traces intersecting curve lines, identifies legends, the symbols in the legends and the text associated with the symbols, identifies X and Y axes, their tics and their labels presents challenges because of the heterogeneity of these objects in different plots. Data extraction from figures becomes difficult because images in documents contain noise introduced either during image creation or during subsequent image processing. The nature of the data may also mean that extraction becomes challenging. For example, 2-D plots often present different types of data points represented by different shapes. When these data points are close to each other, in the figure, they may overlap. While the human eye can decipher the type of data point easily, segregating the different points from the amalgamated data point shown in the 2-D plot automatically is non-trivial. Similarly, the continuation of lines at intersection points introduce ambiguity that needs to be addressed using accurate automatic techniques. Dealing with skewed axes, noisy tics and overlapping text in legends and labels also becomes challenging.

In this paper, we address the following problems:

- Classify document images into predefined classes; and
- Extract data and metadata (numerical data and associated textual information) from 2-D plot images.

Different types of images present different types of information and require different image processing and extraction techniques. Therefore, the system first needs to classify the images and label them. We present a supervised-learning based method that classifies figures from digital documents into five classes, photographs, 2-D plots, 3-D plots, diagrams, and others, using texture and line features extracted from images. We also present a tool that extracts descriptive metadata and numerical data from 2-D plots. The tool extracts textual information, including text associated with the axes and the symbols present in the figure legend, numerical data associated with data points (commonly seen as small squares, circles, diamonds, etc.), and numerical data associated with line plots (for instance, solid line curves) automatically.

The 2-D plot data extraction component is an integral part of $Chem_X Seer$, a digital library for chemistry documents (Fig. 1). Data and metadata extracted from figures in the documents will be stored in a database. Our work is a first step that allows end-users to search for interesting (extracted) data and obtain the data by querying our database. Additional research is essential to improve the accuracy of our extraction, identify the precise semantics of the extracted data and metadata, and make the data fully machine-processible. Currently, the digital library is mainly used and tested by the chemistry and geosciences department within Penn State.

We performed experiments with digital documents from the CiteSeer digital library and the Royal Society of Chemistry journals. We also use 2-D plots obtained using the Google image search online tool. In addition, we generate synthetic 2-D plots, whose ground truth data values are known a priori, in order to evaluate our toolkit. Our experiments show that our methods achieve reasonable accuracy and can be used to

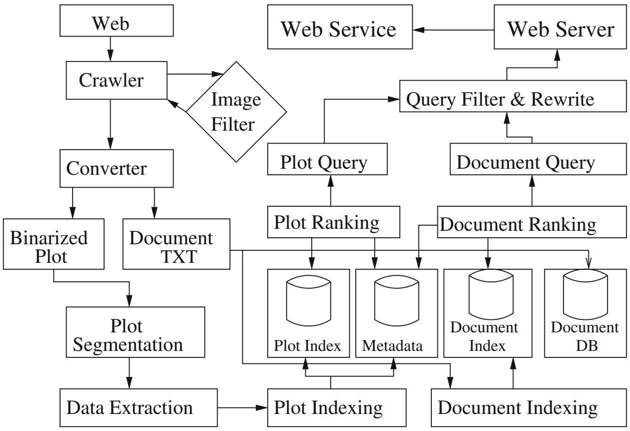


Fig. 1 Architecture of *ChemxSeer* plot search

design a data and metadata extraction tool for 2-D plots in digital documents.

The remainder of this paper is organized as follows. Section 2 reviews prior work in closely related areas. In Sect. 3, we define several classes of images we are to handle and present our method for automatic image classification. In Sect. 4, we present an integrated method for extracting text blocks and numerical data from 2-D plot images. We describe the experimental setup and our results in Sect. 5. Finally, we conclude our work and suggest future research directions in Sect. 6.

2 Related prior work

Due to space limitations, we review only most relevant work in image classification methods, image analysis techniques, graphics recognition methods, text block segmentation, and image analysis for document search.

2.1 Image classification

Automatic image classification is often an important step in content-based image retrieval [7,36] and annotation. Prior efforts model the retrieval and annotation problems as automatic classification of images into classes corresponding to semantic concepts. Visual features and modeling techniques have attracted significant attention.

Textural features [14], color features [40], edge features, or combinations of these features [42] have been developed for classifying images. Chapelle et al. [6] used support vector machines to improve the histogram-based classification of images. Li et al. [22] utilized context information of image blocks, i.e., statistics about neighboring blocks, and modeled images using two-dimensional hidden Markov models to classify images. Maree et al. [28] proposed a generic image classification approach by extracting subwindows randomly and using supervised learning. Yang et al. [44] designed a method to learn the correspondence between image regions and keywords through Multiple-Instance Learning (MIL). Li et al. [23] profiled semantic concepts through clustering of image features extracted from training images and annotated new images by quantifying the probability for each word to be associated with the images and selecting words with highest probabilities.

2.2 Image analysis

Recognition and interpretation of graphics, such as engineering drawings, maps, schematic diagrams, and organization charts, are important steps for processing mostly graphics document images. Yu et al. [45] developed an engineering drawing understanding system for processing a variety of drawings. The system combines domain-independent algorithms, including segmentation and symbol classification algorithms, and domain-specific knowledge, for example a symbol library, in the processing of graphics. Okazaki et al. [30] proposed a loop-structure-based two-phase symbol recognition method for reading logic circuit diagrams. Blostein et al. [2] summarized various approaches to diagram recognition. Futrelle et al. [11] developed a system to extract and classify vector format diagrams in PDF documents. Shao et al. [33] designed a method for recognition and classification of figures in vector-based PDF documents.

2.3 Text block segmentation

In 2-D plot images, text information, including legend, axis labels, and caption, contain useful information for summarizing, indexing, and searching 2-D plots. Jung et al. [19] and Antani et al. [1] surveyed text information extraction in images and videos. They indicate that text extraction generally consists of three steps: text detection, localization, and extraction. Text detection refers to the determination of the presence of text in a given image; text localization is the process of determining the location of text in the image and generating bounding boxes around the text; and text extraction is the stage where the text components are segmented from the background. Thereafter, the extracted text images can be transformed into plain text using OCR technology.

Our work on extracting metadata from 2-D plot images is most related to text extraction from graphics, for instance, retrieving text lines from handwritten documents [34] using an adaptive local connectivity map, text extraction and segmentation on camera-captured document style images based on color clustering method [37], extraction of superimposed text from video frames [25], localization of text from video images based on Fast Hough Transform [3], stroke based text localization in video images [26,39], and text segmentation from complex background using sparse representations [31].

For the problem of extracting data from 2-D plot images, we do not consider text detection since text data is almost always present in 2-D plot images, though there may be very few of them in a given 2-D plot. The backgrounds of 2-D plot images are relatively uniform, i.e., there is small variance in background color. Thus, identifying text components from the background of an image is straightforward after bounding boxes of text are generated. Therefore, our main effort for extracting text data from 2-D plot images has been on text localization.

There are two categories of text localization methods: region-based and texture-based. Region-based methods [20, 29,46] work in a bottom-up fashion: first identifying substructure, such as connected components (CC) [9] or edges, and then merging these sub-structures to mark bounding boxes. Texture-based methods distinguish text in images from the background using textural features generated by various techniques including Gabor filters, Wavelet, FFT, and spatial variance [27,35]. Learning based methods have also been proposed to generate a filter set automatically [16,17,24].

2.4 Analyzing images for document search

Although much attention has been devoted to the summarization and retrieval of text information within documents, relatively little attention has been given to utilizing images within documents for searching digital libraries. Carberry et al. [5] studied a corpus to determine how information graphics, bar charts, line graphics, etc., are used in multi-modal documents and proposed a method for recognizing the message conveyed by those graphics for information retrieval purposes. The work illustrated the popular usage of information graphics in general documents and the effectiveness of deriving simple messages conveyed by bar charts and line graphics using textual hints.

Srihari et al. [38] presented a general model for multimodal information retrieval which includes: analyzing users' information needs, expressing the needs through composite query fields, and determining the most appropriate weighted combination of query fields. Futrelle [11] explored critical issues related to automated summarization of diagrams within documents and proposed an approach for parsing vector format diagrams.

Our work aims at extracting information from 2-D plot images and focused on automated analysis and information extraction. The extracted information may then be combined with other sources of information, using the model proposed by Srihari et al.

3 Classification of images

In this section, we first define several important classes of images contained within documents. Then, we present the image features designed to discriminate between different classes and the supervised learning based approach for automatic classification of images.

3.1 Classification of images

We take the functionalities of images in documents and their visual characteristics into consideration when we define image classes. Some visual features need to be used by computers, as well as by human beings, to distinguish among different classes. The functionalities of images may determine the techniques used for extracting further information from images.

Based on a manual study of approximately five thousand images extracted from randomly selected documents in CiteSeer [12] scientific literature digital library and observation of document search results from Internet-search engines, we define an initial hierarchy of image classes within documents. At the top level, there are two classes: photographic images vs. non-photographic images. The non-photograph category is further divided into four classes: 2-D plot, 3-D plot, diagram, and other. Functionalities and visual appearances of images in every class are presented as follows.

Photograph. A continuous-tone image recorded by a camera or created by photo processing software. This is consistent with the definition given by Li and Gray [21]. Images of pathological tissue taken under a microscope or computed tomography images are considered photographs. Similarly, pictures generated by computer graphics techniques, e.g., dinosaur shots from the movies, also fall into this class. Further classification and analysis of photographs is becoming a main problem in the image retrieval field [23].

Non-photograph. A non-continuous-tone image, as defined by Li and Gray [21]. That is, a figure is either classified as a photograph or a non-photograph due to the exclusive nature of these definitions. In practice, however, there can be cases of a composite figure with a part of it from a photograph and the rest with no continuous tones. Another example can be a photograph of some man-made objects that has no or minimal continuous tones. The classification of these images can depend on the propotions of continuous tones in them. Examples of photographs and non-photographic images contained within documents are presented in Fig. 2.

2-D plot. A non-photographic image that contains a two-dimensional coordinate system (i.e., horizontal and vertical axes) and a series of points, lines, curves, or areas that represent the variation of a variable in comparison with that of another variable. Examples of 2-D plot include scatter plots, curves, and histogram bar charts.

3-D Plot. A non-photographic image that contains a three-dimensional coordinate system and a series of points, lines, curves, or areas that represent the variation of a variable in comparison with that of two other variables.

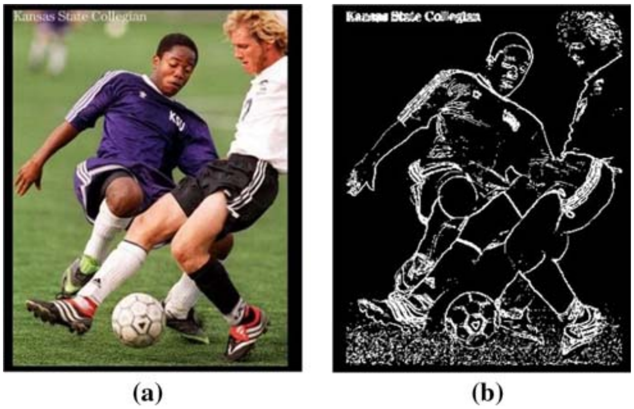


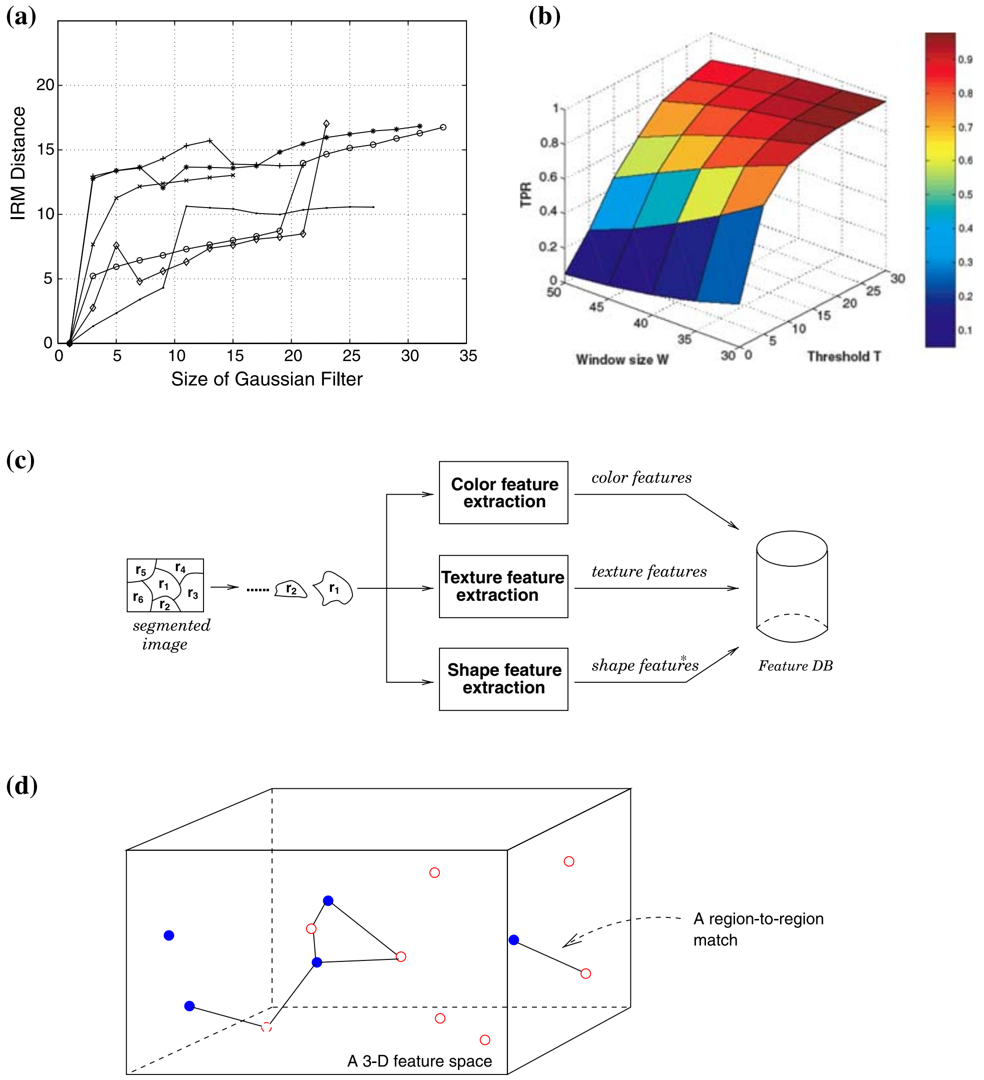
Fig. 2 **a** is classified as a "photograph" figure because it include continuous tones; **b** is a computer graphics, belonging to "non-photograph"

Diagram. A non-photographic image that shows arrangements and relational dependencies among a series of components. Components in the diagram are usually represented by closed contours such as: rectangles, ovals, diamonds, etc. Relational dependencies among components are represented by lines, arrows, etc. An Entity–Relationship diagram for modeling a database application is a typical example of diagram.

Other. A non-photographic image that does not belong to 2-D plot, 3-D plot, or diagram classes. Additional classes of figures will be created out of this class in the future. Some examples are pie charts and figures with sub-figures.

Currently, we attempt to classify images contained within documents into five classes: photograph, 2-D plot, 3-D plot, diagram, and others. In the future, more image classes may be defined depending on user requirements and the potential for successful algorithm development (Fig. 3).

Fig. 3 Some example non-photographic figures extracted from prior publications. **a** 2-D Plot, **b** 3-D Plot, **c** Diagram



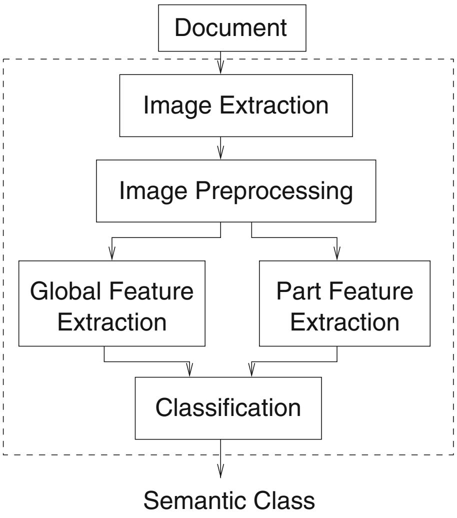


Fig. 4 The process for automatic classification of images within documents

3.2 Classification process overview

The process for automatic classification of images within documents is illustrated in Fig. 4. First, images within documents are extracted. Next, extracted images are preprocessed and converted to a common format. Then, both global image features and part image features [41] are extracted to capture the global patterns of images as well as properties of specific objects within images. Finally, a statistical model is trained to classify images into different classes.

3.3 Image extraction and preprocessing

We use off-the-shelf tools to extract images from the embedding documents. For instance, images in PDF documents can be extracted by Adobe Acrobat image extraction tools. Images contained within HTML document can be extracted by special HTML parsers. Images extracted from PDF are usually in “PNG” format. Web images are typically in GIF format.

Based on our observations, the majority of images extracted from PDF documents are stored in raster format and may also contain color information. Typically, humans do not need to see the images in full color in order to determine the class label of an image, though full color certainly helps in understanding the meanings of the images. Thus, we convert all images to gray scale format in order to standardize the input format of our system. Specifically, we convert all images to the Portable Gray Map (PGM) format, a gray scale image format which is easy to manipulate.

3.4 Extracting texture features

Global features refer to global properties of an image, e.g., the average gray level. Statistics about small image blocks are calculated as global texture features of an image. The global texture features are designed to discriminate photographs from non-photographic images.

First, we apply a region-based document image classification algorithm developed by Li and Gray [21]. We now provide a brief explanation of the algorithm. In the algorithm, every image is divided into non-overlapping small blocks (e.g., blocks of size 4×4 or 8×8 pixels) based on the needs of the application. The one-level Haar wavelet transform is performed on every image block and wavelet coefficients in high frequency bands are recorded. Vetterli and Kovacevic have reported that wavelet coefficients of photographs in the high frequency bands tend to follow Laplacian distributions [43]. The goodness of fit between the distribution of wavelet coefficients in every image block and the Laplacian distribution is calculated. In addition, the likelihood of the wavelet coefficients being composed of highly concentrated values is calculated. Combining these two values, the algorithm uses context-based multiscale classification techniques to classify every image block into one of three classes: photograph, text, and background. Then, the relative frequencies of photograph, text, and background image blocks are calculated as: $f_i = n_i/N, i = 1, 2, 3$, where N is the total number of image blocks in an image, n_i represents the number of photograph, text, and background image blocks, respectively.

3.5 Extracting line features

A part feature refers to a part of an image with some special properties, e.g., a circle or a line. Based on our definitions of several non-photographic image classes and our experimental data, we observed correlations of certain objects with corresponding image classes. For example, a two-dimensional coordinate system, consisting of two axes, are commonly seen in 2-D plots; rectangles, ovals and diamonds are common objects in diagrams. Thus, we attempt to design part image features for basic objects in non-photographic images and use them to discriminate different classes of non-photographic images.

We detect straight lines in an image and choose the longest lines, i.e., the most significant lines, as part features for the image. Specifically, the Canny edge detector [4] is applied to an image and an edge map (binary image) is generated for the original image. On the binary edge image, the Hough transform algorithm [8] is applied to detect straight lines. As shown in Fig. 5, the Hough transform performs a mapping from the original image space (also called x–y space) to the $\rho - \theta$ space (parameter space for straight line), where ρ, θ represent solutions of the line equation $x \cos\theta + y \sin\theta = \rho$. Thus, every cell in the $\rho - \theta$ space represents a potential line position in the image space, and the corresponding line in the image space can possibly be constructed under certain parameters such as the minimum number of points on the

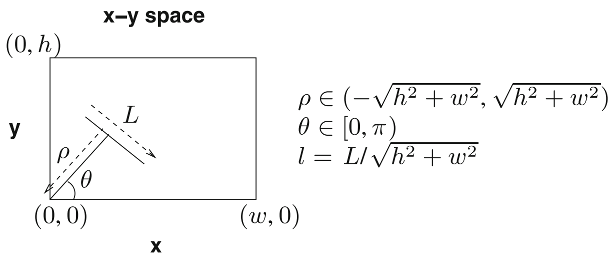


Fig. 5 A line in the image space is represented by a cell in the $\rho - \theta$ space

line, maximum gap between neighboring points, etc. In the Hough transform algorithm, every foreground point in the image space votes for the corresponding cells in the $\rho - \theta$ space. After the voting process, the cells in $\rho - \theta$ space with the top votes are retrieved and corresponding lines in the image space are constructed. In this work, we select the longest lines in an image and record their (ρ, θ, l) values as part image features.

3.6 Classification

We attempt to classify images into different classes based on extracted global and local features. In the feature space, optimal boundaries between different classes of images need to be found, which makes a machine-learning based approach naturally suitable.

We choose the Support Vector Machine (SVM) as the learning and classification tool due to its good generalization performance and ability to handle high-dimensional data [18]. For every image, the SVM uses a feature vector consisting of global and local features. In the current system, every feature vector has 45 dimensions including five global features and forty line features representing the ten longest lines in an image. The five global features include height and width of the image and three global texture features. For every selected line, there are four features representing ρ, θ, and (x, y) coordinate of the end point closer to the bottom left corner of the image.

4 Extracting text and numerical data from 2-D plots

Two-dimensional (2-D) plots represent a quantitative relationship between a dependent variable and an independent variable. Extracting data from 2-D plots and converting them to a machine-processible form will enable users to analyze the data and compare them with other data. Extracting the metadata related to 2-D plots will enable retrieval of plots and corresponding documents and will help in the interpretation of the data. We developed a system for extracting metadata from single-part 2-D plot images, i.e., a single 2-D plot in the 2-D plot image.

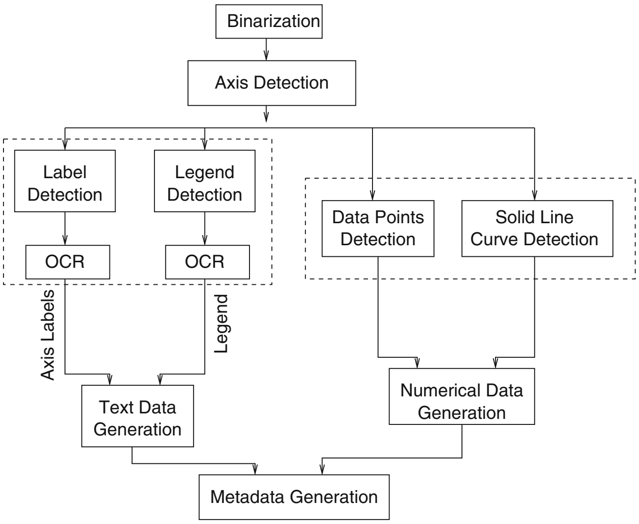


Fig. 6 The process for automatic data extraction from 2-D plots

4.1 Extraction process overview

The process for extracting text and numerical data from 2-D plots is illustrated in Fig. 6. First, images are preprocessed and binarized. Then, axis lines in 2-D plots are detected and the image is segmented into different regions. After that, text data, including axis labels and legends, are extracted using text block extraction techniques. Finally, numerical data are extracted by detecting special data points or solid line curves in 2-D plots.

4.2 Axis detection and plot segmentation

Detecting axis lines in 2-D plots consists of two steps: detecting candidate lines and finding axis lines. In the first step, a customized Hough transform algorithm is designed to detect candidate axis lines. Axis lines are then selected using a set of heuristic rules.

The two axis lines in 2-D plots are almost always perpendicular and lie in horizontal and vertical directions respectively. Therefore, the customized Hough transform is designed to detect lines in these two directions. An illustration of the customized Hough transform is shown in Fig. 7. In the figure, the total rectangle area represents the two-dimensional parameter space ($\rho-\theta$ space) of a Hough transform for line detection, while the shaded area represents the parameter space of the customized Hough transform for detecting candidate axis lines. Specifically, the shaded area where θ is close to 90 correspond to parameter cells for vertical lines and the shaded area where θ is close to 0 or 180 correspond to parameter cells for horizontal lines. The width of the shaded

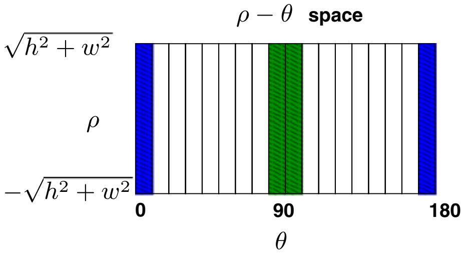


Fig. 7 In the customized Hough transform algorithm, the size of θ dimension in the $\rho - \theta$ space is greatly reduced to detect potential candidate axis lines

area, corresponding to the range of θ values in the customized Hough transform, is dependent upon the desired tolerance for slight skewness of 2-D plot images. Compared with the Hough transform, the customized Hough transform improves both memory and computation efficiency.

After candidate axis lines are detected, a set of rules are used to determine the pair of axis lines in a 2-D plot. The rules are designed to represent expected visual characteristics of axis lines: the relative position between axis lines, the location of axis lines in a 2-D plot, and the relative length of axis lines in a 2-D plot.

Relative position between a pair of axis lines. As shown in Fig. 8(1), two pairs of features, $\{d1, d2\}$ and $\{d3, d4\}$, are used to define the relative position between a pair of axis lines. Specifically, $d1/d2$ represents the position of the vertical axis line relative to the horizontal axis line, while $d3/d4$ represents the position of the horizontal axis line in relative to the vertical axis line. The rules for these features are: $-t1 < d1/d2 < t1$ and $-t2 < d3/d4 < t2$, where $t1$ and $t2$ are adjustable threshold values in the range of (0.1, 0.2).

Location of an axis line in a 2-D plot. As shown in Fig. 8(2), $d5/width$ and $d6/height$ represent locations of the vertical axis line and the horizontal axis line in a 2-D plot respectively. The rules for these features are: $d5/width < t3$ and $d6/height < t4$, where $t3$ and $t4$ are adjustable threshold values in the range of (0.2, 0.5).

Relative length of a axis lines in a 2-D plot. As shown in Fig. 8(3), $d7/width$ and $d8/height$ represent the lengths of the horizontal axis line and the vertical axis line relative to the image width and height, respectively. The rules for these features are: $d7/width > t5$ and $d8/height > t6$, where $t5$ and $t6$ are adjustable threshold values in the range of (0.4, 0.6).

4.3 Extracting text

Text data present in 2-D plots have special characteristics: (a) the distribution of text data is sparse, (b) there may be many very short text strings, and (c) heterogeneous mixture of text and graphical objects in local areas. Based on these characteristics, we adopt a connected components based approach because texture-based methods generally have difficulties in separating graphics and text in a small local area, e.g., in the legend of 2-D plots. In order to increase the accuracy of text recognition, which is important for our application, since there may be very few data in a 2-D plot image, we devised a connected component based method to detect the bounding box of every text character and to separate overlapped characters.

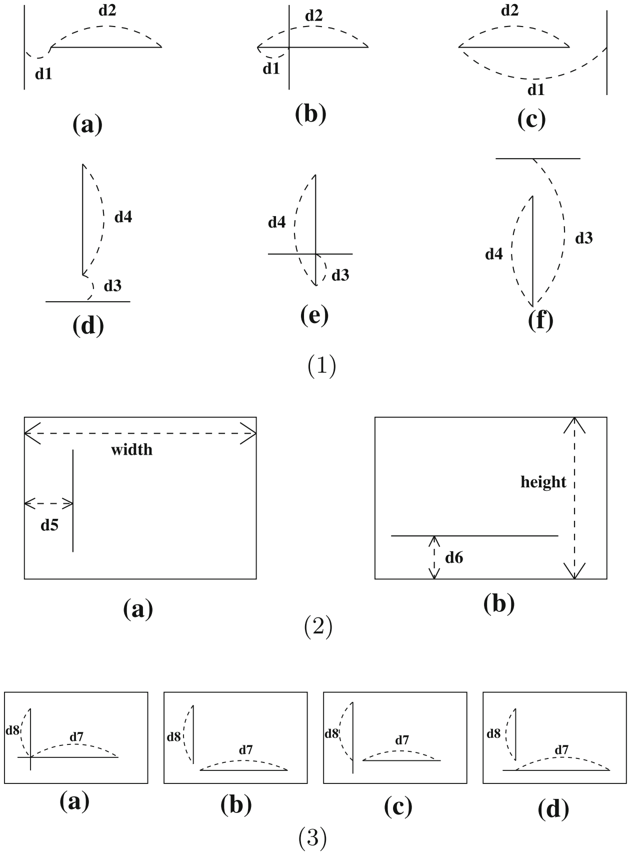


Fig. 8 Illustration of the features used for finding the pair of axis lines. **1** Relative position between candidate axis lines, **2** Relative position of candidate axis lines, **3** Relative length of candidate axis lines

To perform the component analysis, we apply the connected component labeling scheme [9] that tags all the different connected components in the image. Text possesses certain spatial properties such as horizontal alignment and a certain spacing between characters, which distinguish the characters of a text string from other components in the image. These spatial characteristics of the text components may be utilized to perform the component analysis that provides the probabilistic location of the text in the 2-D plot. Specifically, we employ fuzzy rules based upon the spatial alignment of the characters to cluster the components into potential strings of text, which can be recognized using existing OCR software.

Let $I(j, k)$ denote the raster matrix of a 2-D figure. We have text blocks in figures occupy only a small area in them.

Therefore, after connected component labeling, components that have an area above a certain threshold are not considered as candidates for being extracted as text blocks. The result of performing the connected component labeling is a numbered set of pixel matrices corresponding to each component, i.e., $CCL(I(j,k)) = \{C_i\}_{i=1}^{N}$, where the area of the component is below the threshold.

After obtaining the connected components representing single letters, we need to determine which letters constitute a single text block. This could be accomplished through proposing thresholds by which we would determine if two letters are adjacent. However, defining a fixed threshold would not lead to good accuracy since different figures use different fonts and spacing. Therefore, we use the distribution of the spacing and the vertical distance of the connected components, to identify which letters are adjacent and which are parts of separate blocks. Let $X_i^h, X_i^l, Y_i^h, Y_i^l$ represent the corresponding maximum and minimum x and y coordinates of the i^{th} component, which indicate the height and width of a component. Let C_{ij}^y denote the vertical distance between any two components C_i and C_j, i.e. $C_{ij}^y = Y_i^l - Y_j^l$; and C_{ij}^x denote the horizontal distance or *spacing* between C_i and C_j i.e. $C_{ij}^x = X_i^l - X_j^h$. The spatial alignment of the text characters can be encoded using the following fuzzy rule:

$$\text{IF } C_{ij}^y \approx a \quad \text{AND} \quad C_{ij}^x \approx b, \text{THEN } C_i \leftarrow C_i \cup C_j. \quad (1)$$

In the above equation, a and b approximate the spatial alignment parameters. The parameter a is the mean of the distribution of C_{ij}^y for all pairs of i and j. The parameter b is defined similarly. By applying the fuzzy rule, adjacent letters can be merged to create a text block. In some cases, the text appears vertically aligned, e.g., in the case of labels on the y-axis. For these characters, a similar fuzzy rule can be defined by replacing x with y in the equations above. The fuzzy rule in Eq. 1 can be encoded as a bivariate Gaussian unit as follows:

$$P(R_{ij}) = e^{-\frac{(C_{ij}^y - a)^2}{2s_1^2}} e^{-\frac{(C_{ij}^x - b)^2}{2s_2^2}} \quad (2)$$

In the above equation, s_1 is the standard deviation corresponding to a and denotes the allowed difference, i.e, vertical spread, and s_2 is the standard deviation corresponding to b and denotes the allowed difference in the x-dimension. R_{ij} is a random variable that gives the probability of C_i and C_j belonging to the same string or text block. However, the location of the strings of text in X and Y regions (area below the horizontal axis and to the left of vertical axis, respectively) is easy to predict using the region profile, for example, see the horizontal profile of only the X-axis region in Fig. 9b. This extra information is useful in accounting for the differences between various font shapes and can be used to approximate the values of parameters of Eq. 2 on a per figure basis.

```
input  : Binarized Image matrix Segmented with Regions
output: Strings set for OCR input
Initialize:
$strings - set = \phi$;
$cc - set = \phi$;
$\delta$ = 20% of the image area;
For X/Y Region:
$cc - set = ConnectedComponentLabeling$;
Detect the horizontal position of the text using horizontal
profile for X region and Vertical profile for Y region;
separate the text strings using Vertical Profile of the text
identified in previous step;
Approximate the parameters of Eq.(2);
For Curve Region:
cc-set = Connected Component Labeling;
foreach each component $C_i$ do
    if $Area(C_i) > \delta$ then
        cc-set=cc-set - $C_i$
end
foreach pair of components $C_i$&$C_j$ do
    Calculate $P(R_{ij})$ using Eq(2);
    if $P(R_{ij}) > 0.05$ then
        $string - set = C_i, C_j$;
        $strings - set = string - set$;
end
return strings-set;
```

Algorithm 1: Text data detection

4.3.1 Separation of overlapping characters

Once text regions have been detected in the image and extracted, they are suitable for parsing by Optical Character Recognition (OCR) software. We have used the GOCR software[1] amongst others, an open source optical character recognition engine which was deemed flexible enough for the current application. While this and other software tools prove highly reliable for well-posed text, to tackle the unique challenges provided by the text in figures, the text blocks need to be pre-processed. For instance, the frequent overlap of characters within text blocks is a potential cause for error. This feature may be attributed to a variety of factors including software drivers, user editing, small fonts and pixel noise. We propose a recursive segmentation algorithm that deals with this particular preprocessing requirement. Our connected component analysis treats the fused characters as a single component. Overlap of a few characters in a bag of strings can be detected by using profile-based heuristics as shown below. The algorithm determines the average size of the individual connected components, most of which are single letters, and the standard deviation σ of the width of these letters. For very large connected components, we compute the vertical profile. Usually, at the point of contact between two letters there will be a minima in the profile. The fused

[1] http://jocr.sourceforge.net/.

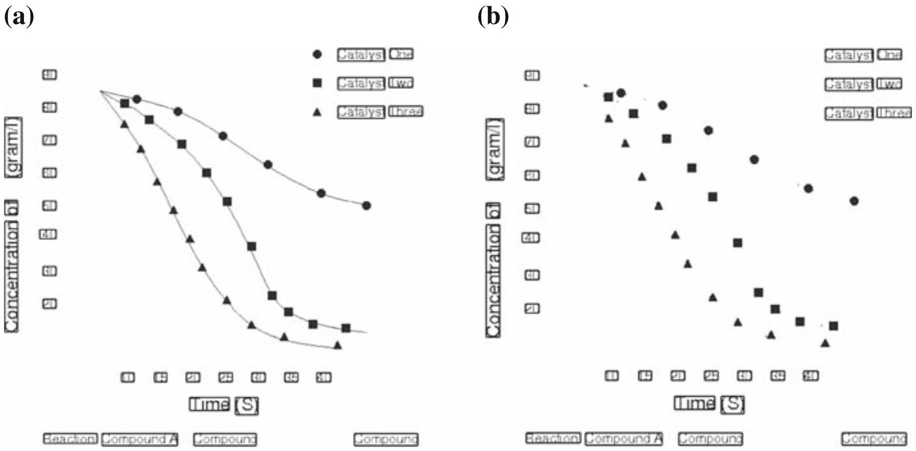


Fig. 9 Text and data point extraction on a sample 2-D figure. **a** Text detection, **b** Data points detection

letters can be separated at the point where these minima occur, by examining the pixel columns immediately adjacent. Should the output width still exceed $X + \sigma$, then the heuristic algorithm is repeated recursively, until the output character(s) have widths that fall within one standard deviation.

```
input : Connected components, output of text detection algorithm
output: Connected components, overlapped characters segmented
Calculate average character width $x$ and standard deviation $\sigma$ along horizontal dimension for input;
foreach character $a_{ij}$ do
    if width exceed $X + \sigma$ then
        Create vector $b_j$ by summing along $y$ and locate minima with indicies $\epsilon_j$;
end
foreach minima index $\epsilon_j$ do
    examine pixels $c$ with indicies $\alpha_i$ along dimension $y$ in original character $a_{ij}$;
    if $c_{\alpha_i,\epsilon_j}$ is bounded either side by zeros && $Sum(c_{\alpha_i,\epsilon_j - 1}) > Sum(c_{\alpha_i,\epsilon_j}) < Sum(c_{\alpha_i,\epsilon_j + 1})$ then
        this corresponds to a point of contact, and the compound character may be separated at the column with index $\epsilon_j$;
end
```

Algorithm 2: Character overlap detection and separation

4.4 Detecting data points

Once the text is identified in the figure, next, our system locates the data points in the data region. Based on our assumption that curves have similar pixel width as axes, we adapt the k-median filtering algorithm [32] to filter out curve lines. The K-median algorithm is a well-known technique to remove noise from images. For the benefit of reader, a summary of the algorithm follows. A raster scan of the image is performed with a window of $(k \times k)$ size with $k = 2 \times w + 1$. With each new position in the window, the intensity of the most central pixel in the window is assigned with the median intensity value of all the pixels that lie in that window. The K-median filtering algorithm is especially chosen to remove point pixel noise and the noise present on the contours. Appropriate size of k is chosen to preserve the curvature of the filtered features.

The K-median filtering algorithm uses a square window. However, our system chooses two windows of size $(1 \times k)$ and $(k \times 1)$ where $k = 2 \times w + 1$ and then performs the raster scan of the image. This choice ensures the consideration of pixels not within the central region of a single square window, which would otherwise fail to be examined owing to the predominance of background pixels. Further, using two one-dimensional windows preserves even those pixels at the edges of the two-dimensional data point objects reasonably well, but removes narrow lines from the figure (as desired). The width and orientation of the data points is different to pixels constituting the plotted curves. Additionally, the average pixel width of the curve is very similar to the axis width. Therefore, w is set to be at most the value of axis width (recall that the value of w determines the value of k). We calculate the width of the axis during the plot segmentation stage using the profile of the 2-D plot. Figure 9b shows the result of the operation of two modified K-median filters.

4.5 Extracting data from solid line curves

In many 2-D plots, there are no special data points in the data region. Thus, the numerical data is presented only by line curves. We designed an automated algorithm for extracting data from line curves in 2-D plots.

A line segment is a connected line without branches within it. In our work, we extract data points from each single *solid line curve* in a 2-D plot. A *solid line curve* is defined as a special type of line composite, i.e., possibly consisting of multiple line segments, with the following requirements:

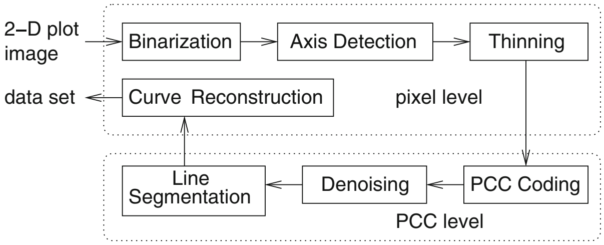


Fig. 10 The process for automatic data extraction from 2-D plots

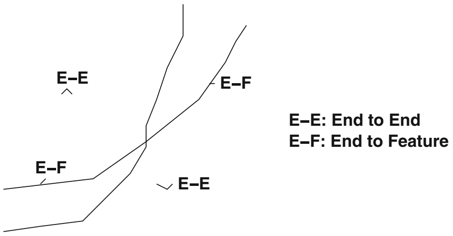


Fig. 11 Various types of noise on PCC representation level

Connectivity: All components in a single solid line curve are connected.

Connectivity between line segments: For any pair of neighboring line segments in a curve, the left end point of a line segment should be connected with the right end point of the other line segment.

Uniqueness: Any line segment should belong to at most one curve except the intersection line segments which may be shared among several curves.

Smoothness: A curve is as smooth as possible. At any intersection point or area, a curve continues in the direction that minimizes the change of derivative at the intersection.

Maximum length: A curve has the maximum possible length as long as it fulfills the above requirements.

The process for extracting numerical data from solid line curves within 2-D plots is illustrated in Fig. 10. First, images are preprocessed and binarized. Then, axis lines in 2-D plots are detected and data regions are located. After that, thinning, PCC coding, and noise reduction techniques are applied to the data areas to obtain line segments. Finally, solid line curves, corresponding to a special kind of composite line, are constructed using the curve construction algorithm.

4.5.1 Thinning, chain coding, and noise reduction

After the axis lines in a 2-D plot have been detected, the data region, where the curves are present, is processed. The term "image" in the rest of this section refers to the part of the original 2-D plot corresponding to the data region. Image thinning, chain coding, and chain code based noise reduction are applied to extract valid thin line segments.

We adopt the image thinning algorithm proposed by O'Gorman [13], which peels the boundaries of foreground regions, one layer of foreground pixels at a time, until the regions have been reduced to thin lines.

After an image has been thinned, a chain coding based representation of the image is obtained to facilitate subsequent thin line analysis. In a chain coding based representation, instead of storing the "ON" and "OFF" values for every pixel as in the raster format, lists of "ON"-valued pixels along lines and the direction from every pixel to the next pixel in its neighborhood are stored. This chain coding based representation is convenient for tracing thin lines, which is necessary for identifying the curve in the later steps. We choose the primitive chain code (PCC) [32], an extension of the popular Freeman [10] chain code, which is designed to preserve connection, branching, and junction topology. In PCC codes, there are pixel chain codes and feature codes, representing pixel connections, junction and end-point features.

Besides providing a high-level line representation, the primitive chain code also provides some noise reduction. Here, the noise refers to any unwanted "ON" pixels (foreground pixels) that may have been brought into the image during the original imaging process or the following image processing steps. At the PCC level, it is possible to find correlation among some types of noise and combinations of chain codes. For example, as shown in Fig. 11, two types of noise in the thinned image, isolated pixels and spurious pixels branching from lines, correspond to short "End to End" and "End to Feature" line segments respectively. Thus, an effective noise reduction method is to set threshold length values for different types of line segments and filter out segments shorter than the corresponding threshold length.

4.5.2 Curve construction

Based on the definition of a solid line curve above, there are five requirements that need to be considered in constructing a curve. Among those, the "connectivity between line segments" and "uniqueness" requirements present critical issues and need to be taken care of before a curve can be identified.

Connectivity between line segments

A line segment has two end points, called left end point and right end point according to their locations in the image. In the case that the two end points have the same X-coordinates values in the $x - y$ space of the image, either one of them can be arbitrarily called the left end point and the other called the right end point.

Two line segments are connected if there exist two end points, one from each line segment, which are connected

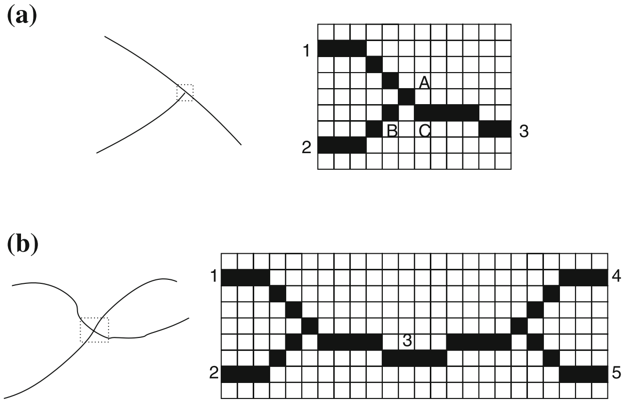


Fig. 12 Illustration of the connection point between line segments and the intersection area between two curves. **a** Connection point between line segments. **b** Intersection area between two curves

according to eight-connectivity. However, there are cases that two end points must be treated as connected even though they are not in each other's eight-connectivity neighborhood. Figure 12a shows the cross-over area of a 2-D plot, where there are three line segments, numbered 1, 2, and 3. There are three special points, labeled as A, B, and C. A is the right end point of line segment 1, B is the right end point of line segment 2, and C is the left end point of line segment 3. Based on eight-connectivity, A is connected to both B and C. Even though B and C are not in each other's neighborhood, they should be treated as connected since they both are connected to a common point. Figure 12a illustrates the case where two end points are connected via another "bridge" point. Two end points may also be connected via multiple hops of connected "bridge" points.

Intersection between curves

An intersection between multiple curves may appear as a single point or a collection of points (constituting a single line segment or multiple line segments). In Fig. 12b, we see that two curves intersect. The pixel-level illustration of the intersection area shows that the intersection appears as a line segment, namely line segment 3.

According to the "uniqueness" requirement in the definition of a solid line curve, an intersection line segment may be shared by multiple curves while a non-intersection line segment should belong to only one curve. Thus, it is necessary to identify intersection line segments so that it can be treated differently. Furthermore, according to the "smoothness" requirement, there exist intersection areas where the continuation of multiple curves need to be decided based on the derivative of line segments.

A manual examination of about five hundred 2-D plots from the CiteSeer [12] digital library and from the Internet showed that even though intersections are common in plots, it is not common to have more than two curves intersect at the exact same point. In our work, we assume that any intersection segment is shared by two curves. This assumption simplifies the problem. The handling of more general cases is beyond the scope of this work. Based on this assumption, we have defined three types of intersection segments, as illustrated in Fig. 13, namely L-type, R-type and M-type intersection segments. A L-type intersection segment has zero left connections and two right connections; a R-type intersection segment has two left connections and zero right connections; a M-type intersection segment has two left connections and two right connections. Based on this definition, the line segment 3 in Fig. 12b is a M-type intersection.

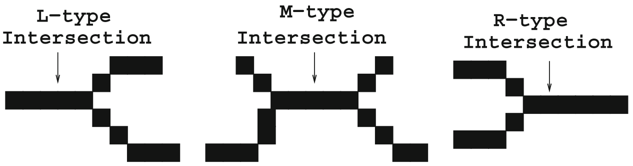


Fig. 13 Three categories of intersection areas between two crossing curves

Curve construction algorithm

An algorithm designed to construct curves contained within a 2-D plot based on the set of line segments present in the image is shown in Algorithm 3. In the Algorithm 3, the *Sort_Line_Segments* function sorts line segments from left to right based on the left end point of every line segment. If there is a tie among multiple line segments, they can be ordered in any way. The *Check_Intersection_Segment* function determines if a line segment is an intersection segment based on its left and right connected line segments. For an intersection line segment, two tickets are assigned to it allowing it to be visited twice. The *Select_Right_Connection* procedure selects the next line segments following j, which minimizes the change of derivative compared with line segment j or the line segment before j on the curve.

The curve construction algorithm generates a set of solid line curves. Depending on application requirements, post processing may filter out invalid curves.

5 Experiments

We have implemented the algorithms mentioned above for automated analyses of images within documents and tested on real-world datasets. In this section, we present the experimental systems, tests, and results of automatic classification of images within documents, and data extraction from 2-D plot images.

```
Input: {lineSegments}
Output: {solidLineCurves}
Sort_Line_Segments;
foreach lineSegments(i) do
    Check_Intersection_Segment(i);
    if isIntersectionSegment(i) then
        v_ticket(i) = 2;
    else
        v_ticket(i) = 1;
    end
end
repeat
    Start_New_Curve_At(i);
    j = i;
    repeat
        Visit(j);
        v_ticket(j) = v_ticket(j) − 1;
        if hasUnvisitedRightConnection(j) then
            j = Select_Right_Connection(j);
    until noReightConnection;
until noUnvisitedSegments;
```

Algorithm 3: The curve construction algorithm.

Table 1 Composition of the figure dataset

Class	No. of figures	Percentage
Photograph	460	23
2-D plot	341	17
3-D plot	20	1
Diagram	93	5
Others	1,066	54
Total	1,980	100

5.1 Classification of images within documents

5.1.1 Experimental data

We collected a dataset from the CiteSeer scientific literature digital library. About two thousand PDF files were randomly selected in the experiments. The Adobe Acrobat image extraction tool is used to extract images from documents. After extracting figures from the dataset, we manually classify figures—required for the training and verification of our automated algorithms—to the pre-defined image classes: photograph, 2-D plot, 3-D plot, diagram, and others. The number of images in every class as well as their corresponding ratios are shown in Table 1.

Recall, that before we extract line features from contents of images, Canny edge detection is applied. We have tested various parameter settings of the Canny edge detector for our figure images. The parameters we used are: (1) the standard deviation of Gaussian smoothing filter is set to 0.6; (2) the high value to be used in hysteresis thresholding is set to 0.8, which specifies the percentage point in the histogram of magnitude of gradient; and (3) the low value to be used in hysteresis thresholding is set to 0.3, which specifies the fraction of the computed high threshold edge-strength value.

Table 2 Precision and recall of photograph versus non-photograph classification based on global image features

Class	No. of images	Precision (%)	Recall (%)
Photograph	460	68.3	77.8
Non-photograph	1,520	93.0	89.0

Three classification problems have been tested on our dataset: (1) ‘photograph’ versus ‘non-photograph’; (2) five classes of images: ‘photograph’, ‘2-D plot’, ‘3-D plot’, ‘diagram’, and ‘others’; and (3) ‘2-D plot’ versus other ‘non-photograph’.

For the first test, only global features are used. In the second test, both global features and part features (line features) are used because we need to test the effectiveness of combined features on discriminating all image classes. In the third experiment, we used line features and heuristics.

5.1.2 Performance measures

We use precision and recall to measure the classification performance. For a single figure class, we denote A as the number of figures correctly classified by the system, B as the number of figures classified by the system to that class, and C as the number of figures classified by a human in that class. The precision and recall of the classification are $precision = A/B$ and $recall = A/C$, respectively. Precision measures the percentage of correctly classified figures in relation to the number of figures classified by a computer to the class. Recall measures the percentage of the figures correctly classified to each class, compared with human classification.

5.1.3 Results

We used SVM Light [18] to classify photograph and non-photographic figures based on the global image features. Under default parameters, experimental results using five-fold cross-validation are reported in Table 2. As can be seen from the result, global image features are fairly effective in discriminating photographs from non-photographic figures.

We used the SVM Multi-class package [18] to classify figures into five classes using both global and part image features. We have adjusted one learning parameter, trade-off between training error and margin, by setting it to 1.0 based on our experimental results. Precision and recall for every class are presented in Table 3. The results show that precision and recall for photographs and 2-D plot figures are much higher than those for 3-D plot and diagram figures.

Table 3 Precision and recall of multi-class classification

Class	No. of images	Precision (%)	Recall (%)
Photograph	460	59	82
2-D plot	341	31	55
3-D plot	20	2	15
Diagram	93	8	20
Others	1,066	79	24

Table 4 Precision and recall of retrieving 2-D plots from non-photograph images

No. of 2-D plots	Precision (%)	Recall (%)
362	82	83

Table 5 Experimental results of text block detection

	Total	Recall (%)	Precision (%)
X Labels	504	84.9	96
Y Labels	504	87.5	96
Legend text	504	79.0	95

We have checked the classification results and analyzed the nature of several false cases. For instance, some non-photographic images that contain computer-generated graphics have been classified as photographs due to relatively smooth coloring within images. In addition, some non-photographic images have been classified as 2-D plots due to the existence of significant straight lines that are actually not axis lines. Even though line features show their potential for certain classes of figures, e.g., 2-D plot figures, they do not work well by themselves. It is essential to design more effective image features to improve the classification of non-photographic images.

We also tested the retrieval of 2-D plot images from non-photographic images using the set of rules for axis detection illustrated in Sect. 4.2. Under the heuristics, the existence of a pair of axis lines in a non-photographic image indicates that the image contains a 2-D plot. The performance is shown in Table 4.

5.2 Extracting text from 2-D plots

We randomly selected 504 2-D images for the text block detection. After using our tool to extract text data from these 2-D plots, we manually check the outputs for calculating experimental results. We consider a *X* label, *Y* label, or a legend text block correctly identified if at least 70% of the letters in a block are correctly extracted. Results for the sample 2-D plots are reported in Table 5.

Table 6 Experimental results of overlapping characters separation

No. of characters	Accuracy (%)
1320	83.5

Table 7 Experimental results of data point extraction

Total	# Correct (%)	Recall (%)
504	92	76

5.3 Separation of overlapping characters

We tested the separation of overlapping characters on a dataset of 1,320 random 16 point font Arial characters, with random case and word spacing. The performance of correct separation is reported in the Table 6.

5.4 Extracting data points from 2-D plots

After text blocks have been detected, we performed data extraction on 2-D plot images. We randomly selected 212 2-D images that had scattered or curve-fitted plots in it. After the data point detection process, we check the result manually. We accept a plot if more than 90% of the data points gets correctly extracted with their shape preserved, otherwise we treat it as a failure. Table 7 shows the recall for the data extraction algorithm.

5.5 Extracting data from solid line curves in 2-D plots

5.5.1 Experiments

We tested our data extraction system on three datasets: from the CiteSeer [12] scientific literature digital library, the Internet, and a third set generated using MATLAB plotting tools.
Plots from CiteSeer. We randomly selected about 2,000 documents from the CiteSeer database, extracted images from documents, and collected 2-D plot images based on manual labels.
Plots from the Internet. Since Google image search engine is based on keyword search, we have tried combinations of keywords. Keywords have been tried including "curve", "plot", "2-D plot", "curve plot", etc, among which the "curve plot" search gives back the maximum number of 2-D plot images.
Synthetic plots. We use MATLAB plotting tools to generate a set of 2-D plots. Comparisons can be made between the original data and the extracted data from 2-D plots. The data set contains random linear and quadratic plots and combinations of them. We use "L", "Q", "LL", "LQ", "QQ", "LLQ", and "LQQ" to represent seven categories of plots containing a single linear curve, a single quadratic plot, two linear

Table 8 Correspondence between the original 2-D plots and the redrawn plots

Data set	# Curves	# Matched	Match ratio (%)
Synthetic	177	153	86
CiteSeer	77	48	62
Web	40	29	73

curves, one linear curve and one quadratic curve, two quadratic curves, two linear curves and one quadratic curve, one linear curve and two quadratic curves respectively.

5.5.2 Results

The performance of data extraction is measured by the correspondence between the original 2-D plots and the plots redrawn using extracted data. Since the original data values are not available for 2-D plots from published scientific documents and from the Web, we manually check the original plots and redrawn plots to obtain the ratio of matched curves between them. The recorded correspondence for the three data sets are shown in Table 8.

In addition to the correspondence measure, the correctness of extracted data are evaluated for the synthetic dataset generated using MATLAB by comparing the original data with the extracted data. Specifically, for every pair of matched original curve and redrawn curve, the X and Y dimension values are normalized to 0–10 and 0–100, respectively. A set of data points are selected from each curve so that their X dimension values are uniformly distributed over the range of valid X dimension values. Finally, the Mean Squared Error (MSE) between Y dimension values of the original data sets and the extracted data sets are calculated. The average MSE between the original data sets and the extracted data sets are illustrated in Table 9.

Many factors affect the correctness of data extraction: image preprocessing, thinning, curve construction algorithm, etc. Among them, curve tracing at cross-points is the most influential factor. We see a higher MSE for data extraction in "LQQ" plots since there are more cross-points in those plots.

Table 9 Mean squared error between original MATLAB plots and redrawn plots

Category	# Matched	MSE
L	15	0.0662
Q	12	0.1408
LL	30	1.1086
LQ	20	0.1609
QQ	22	0.4122
LLQ	27	0.6215
LQQ	27	4.7180
Average	153	1.2575

The experimental results using CiteSeer and Internet plots indicate that a higher error rate occurs in CiteSeer plots due to various reasons: scanned documents, skewness of documents, poor image quality, etc. Resolving these problems is beyond the scope of this work. In summary, since the tracing of a curve relies on connectivity, the method has difficulty in tolerating broken curves. In addition, random noise in poor quality images presents another challenge, since thinned noisy lines may accidentally be traced as curves. Post-processing of the extracted curves may alleviate the problem of noisy lines in many cases.

6 Conclusion and future work

We have argued that in digital libraries, it is essential to extract information from image content to facilitate efficient document retrieval without human intervention. Data represented in these images is often invaluable to the end-user. We defined several classes of images that are common in electronic documents and we outlined algorithms that enable us to classify the images into five classes. We have proposed methods for extracting both point data and lines from 2-D plots. We also show how one can extract the axes in the plots, their labels, identify legends in these plots and extract the data shapes and the text associated with them, and extract other textual metadata from 2-D plots. We demonstrated the effectiveness of our proposed algorithms on synthetic and real-world data obtained from the CiteSeer digital library as well as via an online search.

There are several steps toward extending this work. First, the performance of multi-class classification needs to be improved. Incorporating further part object features will improve classification of non-photographic figures. The performance of algorithms dedicated to extraction tasks may also be improved. For example, the application of the k-median algorithm to data point detection may be improved by the use of different window geometries.

Second, we need to design systems for extracting information from important data-rich diagrams and more difficult objects such as bitmapped or scanned documents. In this work, we focused on electronically generated PDF documents because they are more prevalent in recently generated documents. For bitmapped or scanned documents, pre-processing using connected components labeling in conjunction with Hough-based algorithms and heuristics should allow for the identification of bitmaps pertaining to useful image content and other extraneous objects.

Acknowledgments This work was supported in part by the US National Science Foundation under grants 0535656, 0347148, 0454052, and 0202007, Microsoft Research, and Internet Archive. The authors would like to thank Jia Li for providing the source code for the region-based document image classification [21]. We used the Canny edge detection program [15] and the SVM Light package available in the public domain. Anonymous reviewers have provided constructive suggestions on the manuscript.

References

1. Antani, S., Gargi, U., Crandall, D., Gandhi, T., Kasturi, R.: Extraction of text in video. Technical Report, Department of Computer Science and Engineering, Pennsylvania State University, CSE-99-016, August 30 (1999)
2. Blostein, D., Lank, E., Zanibbi, R.: Treatment of diagrams in document image analysis. In: Proceedings of the International Conference on Theory and Application of Diagrams, pp. 330–344 (2000)
3. Bouaziz, B., Mahdi, W., Hamadou, A.B.: A new video images text localization approach based on a fast Hough transform. In: Proceedings of the International Conference on Image Analysis and Recognition, pp. 414–425 (2006)
4. Canny, J.: A computational approach to edge detection. IEEE Trans. Pattern Anal. Mach. Intell. **8**(6), 679–698 (1986)
5. Carberry, S., Elzer, S., Demir, S.: Information graphics: an untapped resource for digital libraries. In: proceedings of the international conference on research and development in Information Retrieval. pp.581–588 (2006)
6. Chapelle, O., Haffner, P., Vapnik, V.N.: Support vector machines for histogram-based image classification. IEEE Trans. Neural Netw. **10**(5), 1055–1064 (1999)
7. Datta, R., Li, J., Wang, J.Z.: Content-based image retrieval—approaches and trends of the new age. In: Proceedings of the 7th International Workshop on Multimedia Information Retrieval, Singapore, November 2005, pp.253–262 (2005)
8. Duda, R.O., Hart, P.E.: Use of the Hough transform to detect lines and curves in pictures. Commun. ACM **15**, 11–15 (1972)
9. Fletcher, L.A., Kasturi, R.: A robust algorithm for text string separation from mixed text/graphics images. IEEE Trans. Pattern Anal. Mach. Intell. **10**(6), 910–918 (1988)
10. Freeman, H.: Computer processing of line-drawing images. ACM Comput. Surv. **6**(1), 57–97 (1974)
11. Futrelle, R.P.: Summarization of diagrams in documents. Advances in automated text summarization. MIT Press, Cambridge (1999)
12. Giles, C.L., Bollacker, K., Lawrence, S.: CiteSeer: An automated citation indexing system. In: Proceedings of the ACM Conference on Digital Libraries, pp. 89–98 (1998)
13. O'Gorman, L.: $k \times k$ thinning. Comput. Vis. Graph. Image Process. **51**(2), 195–215 (1990)
14. Haralick, R.M., Dinstein, I., Shanmugam, K.: Textural features for image classification. IEEE Trans. Syst. Man Cybernet. **3**, 610–621 (1973)
15. Heath, M., Sarkar, S., Sanocki, T., Bowyer, K.: A robust visual method assessing the relative performance of edge-detection algorithms. IEEE Trans. Pattern Anal. Mach. Intell. **19**(12), 1338–1359 (1997)
16. Jain, A.K., Karu, K.: Learning texture discrimination masks. IEEE Trans. Pattern Anal. Mach. Intell. **18**(2), 195–205 (1996)
17. Jeong, K.Y., Jung, K., Kim, E.Y., Kim, H.J.: Neural network-based text location for news video indexing. In: International Conference on Image Processing, pp.319–323 (1999)
18. Joachims, T.: Making Large-Scale Support Vector Machine Learning Practical. MIT Press, Cambridge (1998)
19. Jung, K., Kim, K.I., Jain, A.K.: Text information extraction in images and videos: a survey. Pattern Recognit. **37**(5), 977–997 (2004)
20. Lee, C.M., Kankanhalli, A.: Automatic extraction of characters in complex scene images. Int. J. Pattern Recognit. Artif. Intell. **9**(1), 67–82 (1995)
21. Li, J., Gray, R.M.: Context-based multiscale classification of document images using wavelet coefficient distributions. IEEE Trans. Image Process. **9**(9), 1604–1616 (2000)
22. Li, J., Najmi, A., Gray, R.M.: Image classification by a two-dimensional hidden Markov model. IEEE Trans. Signal Process. **48**(2), 517–533 (2000)
23. Li, J., Wang, J.Z.: Real-time computerized annotation of pictures. In: Proceedings of the ACM international conference on Multimedia, pp. 911–920, Santa Barbara (2006)
24. Li, H., Doerman, D., Kia, O.: Automatic text detection and tracking in digital video. IEEE Trans. Image Process. **9**(1), 147–156 (2000)
25. Liu, Y., Lu, H., Xue, X., Tan, Y.P.: Effective video text detection using line features. In: Proceedings of the International Conference on Control, Automation, Robotics, and Vision, pp. 1528–1532 (2004)
26. Liu, Q., Jung, C., Kim, S., Moon, Y., Kim, J.: Stroke filter for text localization in video images. In: Proceedings of the International Conference on Image Processing, pp. 1473–1476 (2006)
27. Mao, W., Chung, F., Lanm, K., Siu, W.: Hybrid Chinese/English text detection in images and video frames. In: Proceedings of International Conference on Pattern Recognition, pp. 31015–31018 (2002)
28. Maree, R., Geurts, P., Piater, J., Wehenkel, L.: Random subwindows for robust image classification. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition, pp. 34–40, June (2005)
29. Ohya, J., Shio, A., Akamatsu, S.: Recognizing characters in scene images. IEEE Trans. Pattern Anal. Mach. Intell. **16**(2), 214–224 (1994)
30. Okazaki, A., Kondo, T., Mori, K., Tsunekawa, S., Kawamoto, E.: An automatic circuit diagram reader with loop-structure-based symbol recognition. IEEE Trans. Pattern Anal. Mach. Intell. **10**(3), 331–341 (1988)
31. Pan, W., Bui, T., Suen, S.: Text segmentation from complex background using sparse representations. In: Proceedings of the International Conference on Document Analysis and Recognition, pp. 412–416 (2007)
32. Seul, M., O'Gorman, L., Sammon, M.J.: Practical Algorithms for Image Analysis. Cambridge University Press, Cambridge (2000)
33. Shao, M., Futrelle, R.P.: Recognition and classification of figures in PDF documents. In: Proceedings of the International Workshop on Graphics Recognition, pp. 231–242 (2005)
34. Shi, Z., Setlur, S., Govindaraju, V.: Text extraction from gray scale historical document images using adaptive local connectivity map. In: Proceedings of the International Conference on Document Analysis and Recognition, pp. 794–798 (2005)
35. Sin, B., Kim, S., Cho, B.: Locating characters in scene images using frequency features. In: Proceedings of International Conference on Pattern Recognition, pp. 489–492 (2002)
36. Smeulders, A.W.M., Worring, M., Santini, S., Gupta, A., Jain, R.: Content-based image retrieval at the end of the early years. IEEE Trans. Pattern Anal. Mach. Intell. **22**(12), 1349–1380 (2000)
37. Song, Y.J., Kim, K.C., Choi, Y.W., Byun, H.R., Kim, S.H., Chi, S.Y., Jang, D.K. Chung, Y.K. : Text region extraction and text segmentation on camera-captured document style images. In: Proceedings of the International Conference on Document Analysis and Recognition, pp. 172–176 (2005)
38. Srihari, R.K., Zhang, Z., Rao, A.: Intelligent indexing and semantic retrieval of multimodal documents. Inform. Retr. **2**(2), 245–275 (2000)

39. Subramanian, K., Natarajan, P., Decerbo, M., Castanon, D.: Character-stroke detection for text-localization and extraction. In: Proceedings of the International Conference on Document Analysis and Recognition, pp. 33–37 (2007)
40. Szummer, M., Picard, R.W.: Indoor–outdoor image classification. In: International Workshop on Content-Based Access of Image and Video Databases, Bombay, India, pp. 42–51 (1998)
41. Trucco, E., Verri, A.: Introductory Techniques for 3-D Computer Vision. Prentice-Hall, Englewood Cliffs (1998)
42. Vailaya, A., Jain, A., Zhang, H.: On image classification: "city images vs. landscapes". Pattern Recognit. **31**(12), 1921–1935 (1998)
43. Vetterli, M., Kovacevic, J.: Wavelets and Subband Coding. Prentice-Hall, Englewood Cliffs (1995)
44. Yang, C., Dong, M., Fotouhi, F.: Region based image annotation through multiple-instance learning. In: Proceedings of the ACM international conference on Multimedia, pp. 435–438, Singapore (2005)
45. Yu, Y., Samal, A., Seth, S.C.: A system for recognizing a large class of engineering drawings. IEEE Trans. Pattern Anal. Mach. Intell. **19**(8), 868–890 (1997)
46. Zhong, Y., Karu, K., Jain, A.K.: Locating text in complex color images. Int. Conf. Doc. Anal. Recognit. **1**, 146–149 (1995)