Automated Data Extraction from Scholarly Line Graphs

Sagnik Ray Choudhury
Information Sciences and Technology
Pennsylvania State University
sagnik@psu.edu

Shuting Wang
Computer Science and Engineering
Pennsylvania State University
sxw327@psu.edu

Prasenjit Mitra, C. Lee Giles
Information Sciences and Technology
Pennsylvania State University
pmitra@ist.psu.edu, giles@ist.psu.edu

***Abstract*—Line graphs are ubiquitous in scholarly papers. They are usually generated from a data table and often used to compare performances of various methods. The data in these figures can not be accessed. Manual extraction of this data is hard and not scalable. On the other hand, automated systems for such data extraction task is not yet available. We report an analysis of line graphs to explain the challenges of building a fully automated data extraction system. Next, we describe a system for automated data extraction from color line graphs. Our system has multiple components: image classification for identifying line graphs; text extraction from the figures and curve extraction. For the classification, we show that unsupervised feature learning outperforms traditional low-level image descriptors by 10%. For the text extraction, our heuristics outperforms the accuracy of the previous method by 29%. We also propose a novel curve extraction method that has an average accuracy of 82%. A large partially annotated dataset for future research is described.**

I. Introduction

Scholarly papers usually contain many figures and tables. For example, among 10,000 randomly selected articles published in top 50 computer science conferences [1] between 2004 and 2014, more than 70% contained at least one figure, more than 43% contained at least one table, and more than 36% contained at least one figure and one table. While there have been many works about extraction and understanding of tables, figures have received less attention [2].

Previously, Ray Chaudhury et al. [3] explored figure metadata (caption/mention) extraction. A recent paper by Clark et al. [4] has proposed methods for automated extraction of figures from scholarly PDF documents. Using their method we extracted more than 40,000 figures from scholarly papers. This paper focuses on line graphs that are a subset of these figures. Preliminary analysis of the dataset is presented in section IV.

Line graphs are plots with single or multiple curves in the plotting region. These figures are used to compare multiple methods and are generated from data tables. For example, see figure 1, a line graph that was generated from a data table that can not be accessed from the paper. It is extremely beneficial to regenerate that table, but manual methods [2] are tedious and not scalable. A fully automated system doesn't exist till date, and this work aims to bridge that gap.

We analyzed more than hundred randomly selected line graphs from our dataset to understand the specific challenges of building a fully automated data extraction system. From our analysis (section III), we were able to identify easy and hard cases for data extraction. Our current system focuses on easy cases: line graphs where the curves or data points are drawn with separate colors.

The first module in our system is a classifier that classifies an input image as a positive (color line graph) or negative (bar graphs, pie chart, photograph) instance. While multiple features for this problem has been proposed, we are the first to show that unsupervised feature learning outperforms traditional feature descriptors such as SIFT or HoG.

For color line graphs, the generic algorithm for data extraction is:

1) Extract text "words" from the figure.
2) Classify the words as X-axis/Y-axis value/labels or legends. For the data extraction, every point in the plot region needs to be mapped into a (X-value,Y-value) pair. Assuming the plot scales are linear, two pairs of X-axis values and Y-axis values are necessary and sufficient.
3) Extract the curves and associate them with the legends.

Kataria et al. [9] proposed heuristics for text extraction from plot images, but their method was not evaluated using the metrics standardized by document analysis community [8] in recent times. Our system improves on their method.

Next, extracted text is classified in seven classes such as axes value/labels (see section VI-C). It is easy to see that once two X-axis values and two Y-axis values are identified properly, every pixel in the plot region can be mapped to a "data point". The next step is the curve segmentation i.e. assigning each non-text pixel in the plot region to one of the curves. As we consider color plots, it might appear that the process is trivial, and a color based segmentation would suffice. While a color plot usually contains less than ten "visually distinguishable" colors, it might have more than one thousand distinct RGB color values due to anti-aliasing. Our algorithm solves that problem.

II. RelatedWork

Classification of computer generated charts is a well-studied problem, and multiple features have been proposed. Shao et al. [15] and Huang et al. [7] used low-level graphemes extracted from vector graphics. Prasad et al. [12] used a set of very complicated features, extracted through various transformations on the image. Most recent work by Savva et al.

[1] http://academic.research.microsoft.com/
[2] http://arohatgi.info/WebPlotDigitizer/app/

client number	times	
	global locking	object range locking
4	0.005	5
8	0.01	9
12	0.12	11

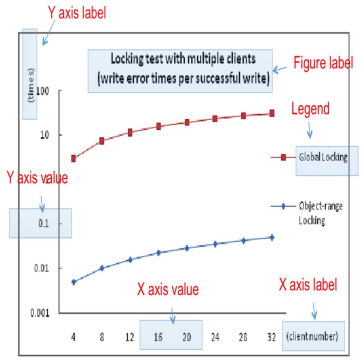


Fig. 1. An example 2D line graph and the data table from which it was generated. It also shows the seven classes of text inside the figure. The mapping between these classes and the data table is apparent.

[13] showed unsupervised feature learning outperforms feature engineering. Our work reinforces that finding (section V).

Early papers by Futrelle [5] introduced the concept of "Generalized Equivalence Relations" (GER) between graphical elements of a figure (symbols, lines, curves) and proposed a pyramidal spatial index for efficient computations of GERs. Subsequent work proposed a more complete grammar for parsing the graphemes [6]. While these works have greatly advanced the understanding of scholarly figures, they did not focus much on the automated data extraction problem itself. For example, they don't clarify much how these graphemes were extracted from the images, which would be the most important step for the data extraction. In more recent works by Shao et al. [14], [15] these graphemes were extracted from vector graphics embedded in PDFs and not raster graphics. Therefore, it would be hard to generalize their approaches to all figures. Kataria et al. [9] proposed an architecture for automated data extraction from 2D plots and their work is most relevant to ours. A more comprehensive version of their architecture [11] reported curve extraction techniques for line graphs, but only for continuous curves. We explore another aspect of the problem.

III. Analysis of Line Graphs for Data Extraction

A line graph has curves and text words. A word can be classified as one of the following: 1. X-axis value, 2. Y-axis value 3. X-axis label, 4. Y-axis label, 5. Figure label, 6. Legend and 7. Other text. The metadata structure for a line graph is a table as shown in figure 1. Axes values are used to generate the "data value" for the curve pixels. The axes labels are the headers for the columns in the table. For one X-value, there are multiple Y-values, corresponding to multiple curves. Legend texts are the column headers for these curves. The figure label is also a metadata for the graph, along with captions and mentions. An automated system for data extraction would need to extract all these information from the figure. While previous work has explored parts of the problem, they are hard to integrate into an architecture due to the variability of the data.

Our research questions are: 1. What are the variations in the plotting styles that make the text/graphics extraction difficult? And 2. Can we identify easy and hard instances of the problem? To answer them, we analyzed more than hundred line graphs sampled from a large collection (section IV).

Many problems arise from the variations in the plotting styles. Previous work [11] assumes that a line graph can be segmented into X-axis, Y-axis and plotting region. Another assumption is that the plotting region is always "ideal", i.e., contains only two components: 1. Curves and 2. A legend region. We find that is often not the case. Only 58% of the plots in our dataset had such an ideal plotting region. We observed that there were four main reasons for plots being "nonideal": 1. The plotting region had a grid structure (as in figure 2): 87%; 2. Legend region was not present (15%) or not in the plotting region (13%); 3. There were text/ graphic elements in the plotting region which were neither legend nor curve (15%) and 4. Plotting region background was nonwhite (10%). Note that these characteristics were not exclusive, i.e. there were nonideal plotting regions that had grid structures, as well as legend regions were not inside the plotting region.

These statistics indicate two problems with existing methods: 1. The "grid" structure and non-white background needs to be removed from the plotting region before applying any algorithm, and 2. The assumption of legend regions being inside the plotting region is not valid.

For curve extraction, line graphs can broadly be classified in two classes: 1. **Binary/ grayscale plots** where curves are plotted with black/gray pixels and are distinguished by markers or other patterns. Liu et al. [11] proposed methods for this problem, but they assumed that the curves were always connected. That assumption is not valid in most real graphs (see figure 2). 2.**Color plots**where curves can be distinguished by their color. 42% of plots in our dataset are color plots. Obviously, a color plot doesn't automatically imply that each curve is plotted with a separate color. However, for most of these plots (89%) that is the case. Naturally, curve extraction is easier for these plots.

To summarize:

1) The hardness of the problem lies in the variation of the plotting styles, and that aspect has gone largely unnoticed in previous works.
2) There are quite a few "easy" instances of the problem that have not been explored properly. These plots are the focus of this paper.

There are three challenges in curve extraction from color plots: 1. Removal of the grid structure and non-white background, 2. Identifying "visually distinguishable colors" and 3. Overlapping curves. Section VI-C reports our methods for these problems. Also, for the text extraction and classification, we do not make any assumption about the legend region being inside the plot region as in the previous work.

IV. Dataset

Our entire dataset consists of 40,000 figures extracted from 10,000 articles published in top fifty computer science conferences between 2004 and 2014. The figures were extracted using a recently released system by Clark et al. known as "pdffigures" [4]. Their system produces a grayscale image file and a JSON metadata file for each figure in the document. The metadata contains following information:

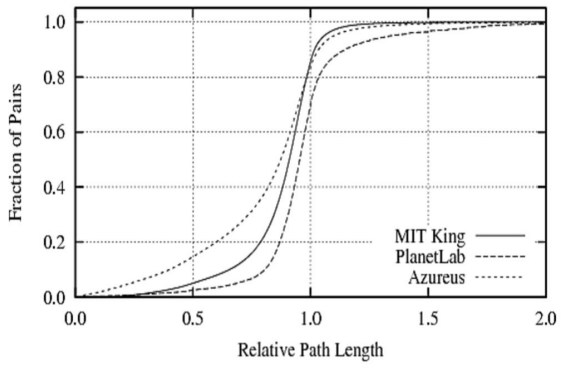


Fig. 2. A monochrome plot (extracted from [10]) where the curves can only be separated by their patterns.

- Page number for the page where the figure appears.
- Location of the figure on the page (bounding box).
- Caption of the figure.
- If the text inside the figure can be extracted from the PDF itself, then for each text word in the figure following metadata is available: 1. The text of the word, 2. The bounding box of the word, and 3. The orientation (horizontal/vertical). Typically, this happens when figures are embedded as vector graphics (eps/ps/PDF).

We made two modifications to the existing system: 1. From the metadata, we re-extracted the figure as a color image. However, around 60% of these images were originally embedded in the PDF as a grayscale image, therefore, were extracted as such. 2. We modified the metadata to include the paragraphs in the document where the figure was mentioned [3].

We manually examined the images and found around 50% of these figures contained sub-figures: often a combination of line graphs, bar charts, and pie charts. From a random selection of 10,000 figures, we manually selected 2250 plots that were either a line graph or contained at least one line graph as a sub-figure. Among them, 882 were color plots. These 882 images were subsequently sampled for various experiments in this paper except for the figure classification experiment (to maintain consistency with previous work). A hundred of these figures were manually tagged with word class labels as discussed in section VI-B). To avoid bias in the algorithm design, a completely separate sample of 120 figures were used for the problem analysis (section III). This dataset will be made publicly available.

V. Classification of Figures

The first step in our architecture is a classification problem: an input image is classified as a line graph or not. Previous works have explored a similar multiclass classification problem, where images are classified as a line graph, a bar graph or a pie chart, etc. Surprisingly, none of the previous methods has used common low-level image descriptors such as SIFT and HoG. We used these descriptors in a bag of words (BoG) model.

The SIFT feature descriptor tries to find key points in an image using scale-space extrema in a difference-of-Gaussians (DoG) pyramid. A Gaussian pyramid is created by repeated smoothing and subsampling of an image by a Gaussian kernel, and the DoG pyramid is created by computing the difference between the adjacent levels in the Gaussian pyramid. Histogram of Oriented Gradients (HOG) algorithm counts occurrences of gradient orientation in localized portions of an image and combines them to produce a final feature descriptor. Both HoG and SIFT have been extensively used in object detection and image classification. Our motivation for using them was to represent specific structures such as bars in the bar graphs and curves in the line graphs in the feature space.

The BoG method in our case works in two steps. First, interest points and descriptors for these points are extracted from the image and then clustered to create a "visual words" dictionary or codebook. Then, each image is represented as a frequency distribution over these visual words. This distribution is used as a feature vector.

For the same classification problem, Savva et al. [13] used random patches of pixels in a BoG model instead of image descriptors. Their method is motivated by recent advances in unsupervised feature learning that has shown excellent promise in natural image classification. We used their dataset for the sake of comparison. In summary, we had 1130 images divided in eight categories: bar charts (215, 19%), line graphs (147 13%), maps (200 17.7%), Pareto charts (117, 10.3%), pie charts (118 10.4%), radar plots (86 7.6%), scatter plots (159 14.2%) and Venn diagrams (88 7.8%). We used stratified k-fold cross validation and compared the accuracies on the test data. The results are shown in table I. In terms of average accuracy (mean of class specific accuracies), HoG descriptor (68%) performed better than the SIFT descriptor (40%) but neither of them outperformed the random patches method (80%). This indicates that the scholarly charts usually do not contain complex structures and randomly selected patches are sufficiently good representations.

	Random (Savva et al. [13])	HoG	SIFT
Curve Plot	73%	64%	48%
Maps	84%	68%	52%
Pareto Plot	85%	67%	51%
Pie Plot	79%	71%	42%
Radar Plot	88%	70%	44%
Scatter Plot	79%	63%	40%
Bar Plot	78%	66%	39%
Table	86%	72%	46%
Venn Plot	75%	68%	42%
Average	80%	68%	40%

TABLE I. Accuracy results for computer generated charts classification: random patch based unsupervised feature learning outperforms low-level image descriptors such as HoG and SIFT.

VI. Word Extraction and Classification

The goal of this module is to extract the text words from the image and classify them to aid the data extraction process.

A. Word Extraction

Text extraction in our case is easier than natural scene images. One choice is to use off-the-shelf OCR systems such as Tesseract, but they have less accuracy on these images because of the text sparsity [9], [17]. Zhu et al. [17] used a convolutional K-means approach to extract text regions from patent images. While their method achieved good accuracy (71%), it

was computationally very expensive because they used multi-scale sliding windows. Kataria et al. [9] used a much simpler approach based on connected components. In their method, first an edge map of the image is created and connected component (CC)s are extracted. These CCs are the candidates for the text characters. Therefore, the ones with an area greater than 20% of the image are discarded because rarely characters in scholarly figures are larger than that. The rest of the CCs are merged to form "words" if they are "close" to each other. Two CCs C_i and C_j are considered "close" if $P(C_i, C_j) > 0.05$ where $P(C_i, C_j) = e^{(C^y_{ij}-a)/2s_1^2} . e^{(C^x_{ij}-b)/2s_2^2}$. C^y_{ij} and C^x_{ij} are the vertical and horizontal distances between C_i and C_j, respectively. The parameters a,b,s_1 and s_2 are the mean and standard deviations of vertical and horizontal distances on the training data.

While the CC approach was computationally efficient, we found several problems with it. Most text characters had an area smaller than 1% of the image area. Also, all CCs having area <1% were not text characters. Therefore, a filtering step was needed to remove noise. The merging heuristic generated large text regions often containing multiple words that needed further segmentation. Also, the proposed merging algorithm was non-iterative and depended on the order of inputs. In recent times, the document analysis community has proposed standardized evaluation metrics for text extraction from images. The previous algorithm was not evaluated using them.

We made following changes to the existing algorithm:

- CCs having area<1% of the image area were considered as candidates for the text characters.
- Candidate CCs were filtered to remove noise. We found that text in the color line graphs is almost always written using black pixels. Therefore, color based filter sufficed. However, it is hard to identify black, gray or white pixels from an image because of anti-aliasing. In section VI-C we propose an algorithm for that. In future, trained classifiers can be used. The output from this step was a set of CCs that are the text characters. In the next step, they were merged to form words.
- We observed that most texts in these figures were horizontal, except for Y-axis labels. Therefore, it was beneficial to use these two distances separately rather than combining them as in the previous work. CCs that are within 10% of the image width from left are the candidates for Y-axis labels and merged vertically if the vertical distances between them are less than a threshold. Other CCs are merged horizontally using a separate distance threshold. We experimented with several distance thresholds such as minimum, second minimum, median and mean of the horizontal/vertical distances. The second minimum heuristic worked best among them. The merging process is run iteratively until there is no merge. The process is usually terminated in two or three iterations.

1) Experiments and Results: From our dataset (see section IV) of extracted figures, we selected 200 color line graphs that were originally embedded as vector graphics. These figures being vector graphics, bounding boxes of the words were available, and we used them as the gold standard. Our evaluation protocol is adopted from Wolf et al. [16] that has been used extensively in other works and competitions [8] for text extraction from images. For each image, ground truth G contains coordinates of i bounding boxes for the words. A predicted result P contains coordinates of j bounding boxes. For each box b_G in G, the box in P (b_P) having maximum overlap is defined as the equivalence. This is defined as one-to-one matching. Note that Wolf et al. allows for other types of matching such as many-to-one (multiple boxes predicted for a single box) and one-to-many matching (one box predicted for multiple boxes in the gold standard). These other matchings are not helpful for us. For a pair (P,G), the precision (P_{OB}) and recall (R_{OB}) scores are calculated as follows:

- for all pairs (b_P,b_G)
  - if Area-Precision(b_P,b_G) >= σ_P and Area-Recall(b_P,b_G)>=σ_R
  - no. of correctly identified rectangles+=1
- $P_{OB} = \frac{\text{no. of correctly identified rectangles}}{|P|}$
- $R_{OB} = \frac{\text{no. of correctly identified rectangles}}{|G|}$

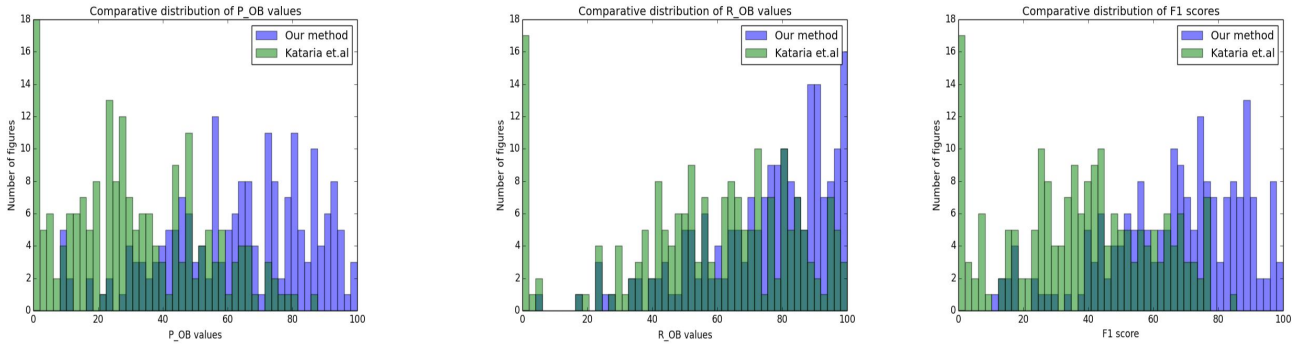
Area-Precision is defined as the ratio of the overlap between b_P, b_G and the area of b_P. Area-Recall is defined as the ratio of the overlap between b_P, b_G and the area of b_G. The thresholds σ_R and σ_P are both set to 0.4. The standard practice for these values are 0.4 and 0.8 [8] respectively, but many-to-one and one-to-many matchings are also valid in those cases.

Figures 3a, 3b and 3c show the comparative histograms of P_{OB}, R_{OB} and $F1$ values for the text extraction from 200 images, with our method and Kataria et al. [9] as baseline. The baseline method segments the image into X-axis region (area below X-axis), Y-axis region and plot region. The text extraction depends on this step, but the segmentation is heuristic, i.e. the longest pair of intersecting vertical and horizontal lines are considered as axes lines. We found that assumption to be wrong for most of our images. We believe that attributed to the poor result. Our average precision, recall, and F1 values are 63%, 75%, and 67% respectively, which is better than the baseline by 32%, 18%, and 29%.

B. Word Classification

Extracted words need to be classified in one of the following seven classes: 1. X-axis value, 2. Y-axis value 3. X-axis label, 4. Y-axis label, 5. Figure label, 6. Legend and 7. Other text. While most of the words can be classified as one of the first six classes, sometimes line graphs contain words that are used to show specific points of interest. These words are classified as "other text". For the data extraction, we need to map each point in the plotting region to a data value.Two X-axis values and two Y-axis values suffice for that step. Also, other words can be mapped to the metadata structure for the line graph as shown in figure 1. While this classification step is essential in an automated data extraction process, it has gone largely unnoticed in the previous works.

We only have two features for each word: 1. The text of the word and 2. The location of the word on the image. Therefore, it is hard to train a classifier. However, the probability of a text having a particular class label is dependent on the labels of its neighbors. For example, if a word is classified as an "X-axis label", nearby words have a higher probability of being an X-axis label. This is similar to pixel labeling problems in image



(a) (b) (c)

Fig. 3. Comprative results for text extraction: our method and Kataria et al. [9]

processing [1], where an energy equation such as equation 1) is minimized:

$$E(L) = \sum_{p \in P} D_p(L_p) + \sum_{p,q \in N} V_{p,q}(L_p, L_q). \quad (1)$$

Here $L = \{L_p | p \in P\}$ is a labeling of the image P, $D_p(L_p)$ is a cost function for the pixel p having the label L_p (also known as a "data penalty function"), and $V_{p,q}$ models the interaction between two neighboring pixels p and q having labels L_p and L_q (also known as "n interaction potential"). N is the set of all pairs of neighboring pixels.

We are trying to predict the class label for the words instead of pixels. The minimization problem is NP-hard for more than two labels, and we have seven labels. Efficient approximations based on max-flow/min-cut techniques exist [1], but we developed a much simpler algorithm to model the interaction between neighboring words. The key observation is that the labels can change iteratively depending on the neighboring labels.

We designed heuristics for the class prediction problem that are summarized in table II. Each heuristics in the table II can be considered as a confidence point. While it is easy to design heuristics for classes such as axes values and axes labels, classes such as legend or "other text" don't have intuitive heuristics.

In the first step, each text is classified as one of the classes in the table II or "other text". We observed that the heuristics were very accurate for the following three classes: X-axis value, Y-axis value and Y-axis label (see table III). In the seconds step, each word that was not classified as one of these classes was relabeled as "other text", if a word of class "other text" was close to it. A word is considered to be "close" to another, if the horizontal or vertical distance between them is less than 5% of the image width or image height, respectively.

Usually, the legend region is the most textually dense region of the figure. That motivated our third step, where we employed a region growing algorithm to label some of the "other text" words as legends. A region is defined as the "legend box" which is initially empty. Iteratively the words labeled as "other text" are selected. If they are close to each other, they are merged, and the bounding box is updated with the merged region. This process is continued until there is no merge possible. Finally, the words inside the "legend box" are classified as legends and rest of the words are classified as "other text".

For the experiment, we manually tagged the 2000 words from 100 figures in the aforementioned seven classes. One versus all precision, recall and f1 scores are reported in table III. It is important to note that we predicted the class labels for two classes "legends" and "other text" with 91% and 57% accuracy, without desigining any heuristic. Most words are classified correctly, except for the class "other text". We believe this is because they are randomly spread over the images and do not follow a logical layout structure.

Class	Heuristics for classification
X-axis value	1. Text is a number, 2. Within 10% of image height from the bottom of the image.
X-axis label	2. Text is not a number, 2. There's nothing below the text.
Y-axis value	1. Text is a number, 2. Within 10% from image width from the left of the image.
Y-axis label	1. Text is vertical, 2. Not a number, 3. There's nothing to the left of the text.
Figure label	1. Within 10% of the image height from the top, 2. There's nothing above the text.

TABLE II. PRECISION, RECALL AND F1-SCORES FOR THE CLASSIFICATION OF WORDS IN THE FIGURES.

	precision	recall	f1-score
X-axis value	99%	90%	95%
X-axis label	82%	95%	88%
Y-axis value	97%	95%	95%
Y-axis label	95%	97%	96%
Figure label	83%	73%	78%
Legend	90%	93%	91%
Other text	48%	71%	57%

TABLE III. PRECISION, RECALL AND F1-SCORES FOR THE CLASSIFICATION OF WORDS IN THE FIGURES.

C. Curve Segmentation and Assignment

The final module in our system is a heuristic algorithm that separates out the curves in the plots and matches the plots with the legend text. While a color plot usually has very few "visually distinguishable" colors, it has more than thousand distinct RGB values due to "anti-aliasing", as these plots are extracted from PDFs. RGB space is not a "natural" representation of colors as a small Euclidean distance between two RGB values doesn't imply that the colors are visually

close. Therefore, we converted the images into HSV color space. In HSV, the hue component determines the color and the saturation and value components determine the shade of the color. Therefore, we first quantized the pixels in 36 color bins based on their hue value (bin size=10, min hue value 0, max 360).

Three important observations are: 1. Text in the plots are almost always written in black colors, 2. White and gray pixels are almost always background values and 3. In color plots very rarely curves are drawn with black pixels. Therefore, it was evident that white, gray and black pixels can be removed. These pixels have low saturation or value. However, determining such threshold is hard. Our experiments suggested that if the "value" parameter is less than six and the saturation parameter is less than eleven, that pixel could be classified as white,black or gray, and hence removed. Note that in section III we mentioned that one problem with color graphs is the presence of the grid structure in the plotting region. This approach solved that problem. Indeed, some curves are drawn with black pixels, and hence would be removed by our algorithm. We sacrifice the recall in favor of precision. In future, we plan to investigate this problem further.

Given a set of quantized "color" pixels, the problem reduces to determining the actual number of colors. We sorted the color bins based on their frequency (number of pixels in each bin). The bins with high-frequency values correspond to the curves. Bins are selected iteratively from higher to lower frequency: from 1 to i such that frequency of bin (i-1) - frequency of bin (i) $>$ 0.5* frequency of bin (i-1). This heuristic is motivated by the idea of the rate of change in a variable.

Overlapping curves pose a challenge when the curves are drawn by solid bold lines. But for curves with dotted lines, even overlapping ones can be extracted easily. In future, a sequential error correction model common in OCR can be used.

For evaluation, we manually examined the original image and the reconstructed curves and predicted the number of colors. On a random sample of 165 images, our algorithm correctly predicted the number of colors in 82% of cases. Once the curves are segmented and the text from the figure is classified, the curve-legend assignment is easy for color curves. We observed that for almost all plots, the color symbol appears left or right to the legend text.

VII. Conclusion

Line graphs are abundant in scholarly papers. These figures are generated from tabular data and that data, while very important, can not be accessed or searched on. We report an architecture for automated data extraction for such line graphs. We present an analysis of line graphs to identify easy and hard cases. Next, we present algorithms for text extraction, classification and curve extraction from line graphs. We also briefly discuss our dataset of figures extracted from scholarly papers. In future, we plan to improve the text extraction accuracy and consider curve extraction from BW/grayscale line graphs.

VIII. Acknowledgments

We gratefully acknowledge partial support from the National Science Foundation and NPRP grant # 4-029-1-007 from the Qatar National Research Fund (a member of Qatar Foundation).

References

[1] Y. Boykov and V. Kolmogorov, "An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 26, no. 9, pp. 1124–1137, 2004.

[2] S. Carberry, S. Elzer, and S. Demir, "Information graphics: an untapped resource for digital libraries," in *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, 2006, pp. 581–588.

[3] S. R. Choudhury, P. Mitra, A. Kirk, S. Szep, D. Pellegrino, S. Jones, and C. L. Giles, "Figure metadata extraction from digital documents," in *Document Analysis and Recognition (ICDAR), 2013 12th International Conference on*. IEEE, 2013, pp. 135–139.

[4] C. Clark and S. Divvala, "Looking beyond text: Extracting figures, tables, and captions from computer science paper," 2015.

[5] R. P. Futrelle, "Strategies for diagram understanding: Object/spatial data structures, animate vision, and generalized equivalence," in*Proceedings of the 10th ICPR*, 1990, pp. 403–408.

[6] R. P. Futrelle and N. Nikolakis, "Efficient analysis of complex diagrams using constraint-based parsing," in*Document Analysis and Recognition, 1995., Proceedings of the Third International Conference on*, vol. 2. IEEE, 1995, pp. 782–790.

[7] W. Huang, C. L. Tan, and W. K. Leow, "Model-based chart image recognition," in *Graphics Recognition. Recent Advances and Perspectives*. Springer, 2004, pp. 87–99.

[8] D. Karatzas, F. Shafait, S. Uchida, M. Iwamura, L. Gomez i Bigorda, S. Robles Mestre, J. Mas, D. Fernandez Mota, J. Almazan Almazan, and L.-P. de las Heras, "Icdar 2013 robust reading competition," in *Document Analysis and Recognition (ICDAR), 2013 12th International Conference on*. IEEE, 2013, pp. 1484–1493.

[9] S. Kataria, W. Browuer, P. Mitra, and C. Giles, "Automatic extraction of data points and text blocks from 2-dimensional plots in digital documents," in*Proceedings of the 23rd national conference on Artificial intelligence*, vol. 2, 2008, pp. 1169–1174.

[10] J. Ledlie, P. Gardner, and M. Seltzer, "Network coordinates in the wild," in *Proceedings of the 4th USENIX conference on Networked systems design & implementation*. USENIX Association, 2007, pp. 22–22.

[11] X. Lu, S. Kataria, W. J. Brouwer, J. Z. Wang, P. Mitra, and C. L. Giles, "Automated analysis of images in documents for intelligent document search,"*IJDAR*, vol. 12, no. 2, pp. 65–81, 2009.

[12] V. Prasad, B. Siddiquie, J. Golbeck, and L. Davis, "Classifying computer generated charts," in *Content-Based Multimedia Indexing, 2007. CBMI'07. International Workshop on*. IEEE, 2007, pp. 85–92.

[13] M. Savva, N. Kong, A. Chhajta, L. Fei-Fei, M. Agrawala, and J. Heer, "Revision: Automated classification, analysis and redesign of chart images," in *Proceedings of the 24th annual ACM symposium on User interface software and technology*. ACM, 2011, pp. 393–402.

[14] M. Shao and R. Futrelle, "Graphics recognition in pdf documents," in *Proc. of GREC*, 2005.

[15] M. Shao and R. P. Futrelle, "Recognition and classification of figures in pdf documents," in*Graphics Recognition. Ten Years Review and Future Perspectives*. Springer, 2006, pp. 231–242.

[16] C. Wolf and J.-M. Jolion, "Object count/area graphs for the evaluation of object detection and segmentation algorithms,"*International Journal of Document Analysis and Recognition (IJDAR)*, vol. 8, no. 4, pp. 280–296, 2006.

[17] S. Zhu and R. Zanibbi, "Label detection and recognition for uspto images using convolutional k-means feature quantization and adaboost," in *Document Analysis and Recognition (ICDAR), 2013 12th International Conference on*. IEEE, 2013, pp. 633–637.