Chart Mining: A Survey of Methods for Automated Chart Analysis

Kenny Davila, *Member, IEEE*, Srirangaraj Setlur, *Senior Member, IEEE*, David Doermann, *Fellow, IEEE*, Bhargava Urala Kota, *Member, IEEE*, and Venu Govindaraju, *Fellow, IEEE*

Abstract—Charts are useful communication tools for the presentation of data in a visually appealing format that facilitates comprehension. There have been many studies dedicated to chart mining, which refers to the process of automatic detection, extraction and analysis of charts to reproduce the tabular data that was originally used to create them. By allowing access to data which might not be available in other formats, chart mining facilitates the creation of many downstream applications. This paper presents a comprehensive survey of approaches across all components of the automated chart mining pipeline, such as (i) automated extraction of charts from documents; (ii) processing of multi-panel charts; (iii) automatic image classifiers to collect chart images at scale; (iv) automated extraction of data from each chart image, for popular chart types as well as selected specialized classes; (v) applications of chart mining; and (vi) datasets for training and evaluation, and the methods that were used to build them. Finally, we summarize the main trends found in the literature and provide pointers to areas for further research in chart mining.

Index Terms—Chart survey, chart extraction, multi-panel chart segmentation, chart image classification, chart understanding, chart data extraction, chart datasets

1 Introduction

Data visualizations can be used to communicate information in an efficient manner. In many cases, data which is hard to convey through text and/or tables becomes easily interpretable through data visualizations. A great deal of literature has been devoted to the analysis of data visualization in order to understand how to use them more effectively [1], [2], [3]. In this work, we concentrate on the technical aspects of automatic extraction, classification and understanding of a specific family of data visualizations. *viz*. charts.

Purchase [4] defines a diagram as a set of indivisible visual elements (or graphics) depicted on a two-dimensional plane in order to represent a concept. Charts are abstract diagrams with simple rules but strong representative power, and have been widely adopted in multiple domains [4]. In academic papers, charts can be used as powerful summarization tools [5], which allow researchers to navigate quickly through results and comprehend them [6]. Charts usually complement the facts described in the main text of a document [7], but their data is often not made available in other formats. Charts are ubiquitous in documents, and research in automatic chart processing is expected to provide access to the rich quantitative data hidden in these graphical objects.

• *The authors are with the Department of Computer Science and Engineering, University at Buffalo, Buffalo, NY 14260 USA. E-mail: {kennydav, setlur, doermann, buralako, govind}@buffalo.edu.*

Manuscript received 20 Aug. 2019; revised 10 Mar. 2020; accepted 28 Apr. 2020. Date of publication 4 May 2020; date of current version 1 Oct. 2021. (Corresponding author: Kenny Davila.) Recommended for acceptance by C. V. Jawahar PhD. Digital Object Identifier no. 10.1109/TPAMI.2020.2992028

In this survey, we concentrate on published methods for automated chart analysis and their applications. Besides seminal works, we cover primarily research papers from the past 15 years. Earlier methods for diagram analysis have been covered in the survey by Blostein *et al*. [8]. The survey by Purchase [4] presents a high level classification of the research topics related to diagrams that were presented at the *International Conference on the Theory and Application of Diagrams* from 2000 to 2012. The review by Liu *et al*. [9] covers earlier methods for chart extraction, classification and recognition [9], while our work also covers datasets and applications of chart analysis, as well as new methods from the past 8 years. While there is a significant amount of literature on chart mining, we also highlight and draw attention to multiple open problems which require the attention of researchers to advance this field.

This work is organized around the main steps required for automated chart mining (see Fig. 1). In Section 2, we describe methods for automated extraction of charts from documents. In Section 3, we present methods for processing multi-panel charts. In Section 4, we discuss methods used for chart image classification. In Section 5, we cover approaches used for automatic data extraction from charts. In Section 6, we present applications for automated analysis of charts. In Section 7, we present a summary of existing datasets and their creation methodology. Finally, we discuss our findings and open challenges in Section 8.

2 Extracting Charts from Documents

In highly structured documents (e.g., academic papers, technical reports, patents, etc.), charts are commonly included within figures. We define *figure* as a container for a given

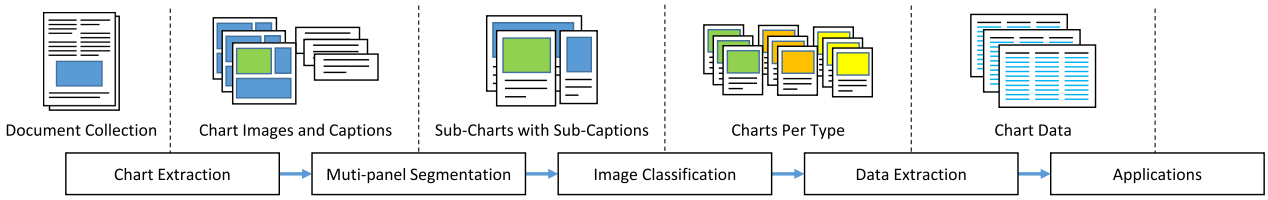


Fig. 1. Summary of the chart mining pipeline. The process starts with a collection of documents from which charts and their captions are extracted. Additional segmentation is required for multi-panel images. Charts are then identified by their type through image classification methods. An approximation of the data used to create each chart is produced by chart-specific data extraction models. Finally, the extracted data can be used to support multiple target applications.

visualization, its labels and its *caption* (textual description). In this section, we concentrate on methods used for extracting figures, including charts, from highly-structured documents.

Different approaches have been proposed for both digitally-born and scanned documents. The extraction process can be divided into two high level steps (see Fig. 2): document segmentation and linking figures to captions. The first locates and extracts figure and caption candidates, while the second aims to link candidate figures to their corresponding captions.

Multiple metrics have been used to *evaluate* figure extraction systems. Figures should be extracted with the correct page number and captions should match the ground truth text [5]. Region-based metrics such as intersection over target box, intersection over candidate box, and intersection over union (IOU) [5], [10] have been used to match the extracted figure candidates to ground truth elements. Then, thresholds over these metrics [5] and greedy 1-to-1 matching [10] have been used to determine the final recall and precision metrics. Recent figure extraction systems include PDF Figures 2 [5], PDFFigCapX [11], and DeepFigures [12]. Both PDFFigures 2.0 and DeepFigures have been used by Semantic Scholar [13] for large scale extraction of figures from academic papers.

2.1 Document Segmentation

Most methods for document segmentation can be classified as either top-down or bottom-up. The first group starts with complete pages and divides them into regions, while the second group creates these regions by grouping small graphical or textual units. Based on the input format, these methods can also be classified as: raster-based, vector-based, and hybrid. The trend for academic papers has been to focus on vector-based documents and graphics. However, vector graphics can be converted to images, and these can be processed using the raster-based methods also used for scanned documents.

2.1.1 Raster-Based Segmentation

Traditional methods use heuristics both to segment pages into uniform regions in either a top-down [14] or bottom-up [15], [16] manner, and to further classify these regions as either text or graphics [14], [16]. Deep neural networks are used by recent methods to locate, extract, and classify graphic regions from document page images directly [12], [17], [18]. Saliency-based attention models can improve general figure detection, but might degrade on charts containing large empty areas with little saliency [18]. Text regions that surround graphic regions can be treated as caption candidates, while text regions that overlap graphic regions can be considered textual elements within the graphic [15].

2.1.2 Vector-Based Segmentation

These methods extract figures from vector-based documents using analysis of instructions (e.g., PDF operators) alone. This is challenging, since vector graphics related to figures can be mixed with other operators without proper demarcation, and text operators can be used to represent complete words, lines, or paragraphs without any consistency [6]. PDF documents use a state-based system, which needs to be tracked by custom parsers, as well as operators to render primitives such as text, paths, and raster images. These state-dependent operations can be converted to self-contained objects to simplify further analysis [19]. Heuristics can be used to estimate the number of figures on a given page [10]. Different methods (e.g., k-means clustering) are then used to find groups of PDF operators representing figures [7], [10]. Other graphic elements such as logos or tables must be separated from figures using heuristics [7] or machine learning [10]. Figure caption candidates are finally located by analyzing the text operators directly [7], [20].

2.1.3 Hybrid Segmentation

These methods combine vector-based document analysis with rendering for further raster-based segmentation. Many

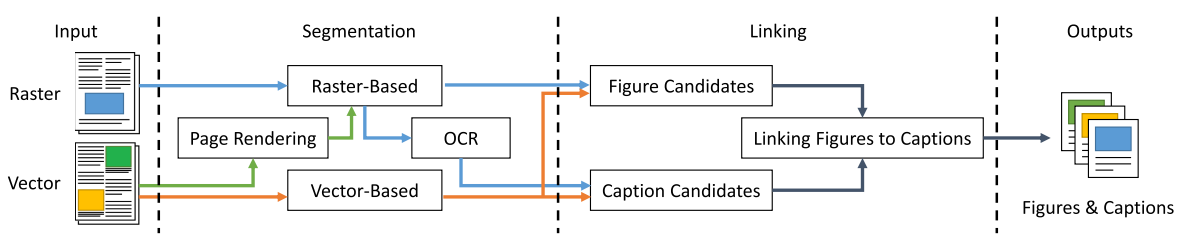


Fig. 2. Summary of approaches used for figure extraction. Depending on whether the input document is raster or vector, different segmentation mechanisms are applied to extract figure and caption candidates. In the next stage, each method needs to link each figure to its corresponding caption.

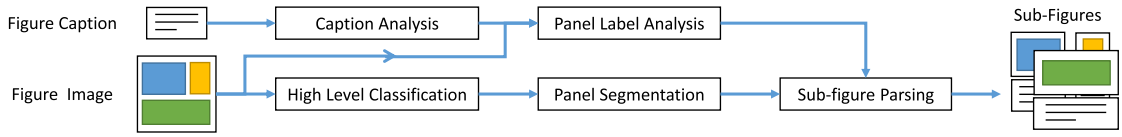


Fig. 3. Segmentation of multi-panel figures. Figure images and their captions are analyzed. Most methods use a classification step to determine if segmentation is needed. Image-based segmentation results are refined to match caption analysis results. The output is a set of individual sub-figures with sub-captions.

works rely on existing tools (e.g., PDFBOX [21]) for the extraction of text and figure candidates from documents [11], [22], but these generic PDF parsers usually ignore captions [6]. Hence, some works have developed their own custom PDF parsers which analyze text operators directly. Some approaches use harder assumptions about layouts for academic papers to locate caption candidates [6], [11], [23], [24]. The non-text PDF operators are then rendered, sometimes using only approximations of the regions affected by each PDF operator to accelerate the process [6], [25], [26]. Raster-based segmentation models can then be used to find figure candidate on the rendered graphics. The original PDF operators can be used both to refine the resulting figures and to extract them in vector formats such as SVG [6], [24]. Finally, text regions overlapping these figure candidates can be added to them as figure text. Overall, these methods work better on documents that adhere to standard layouts [6].

2.2 Linking Figures to Captions

This process takes as input the figure and caption candidates and produces the final figure-caption pairs. Captions are assumed to be close to the associated figures, usually within the same page, but exceptions have been found in practice [6]. Geometric properties of the captions [6], [7] and other heuristics based on document layouts [5], [12] have been used to compute the cost of linking captions to figure candidates. Then, one can iteratively select the best matches in an greedy fashion [7], or use the Hungarian method to find the optimal assignments that minimize such costs [5], [12]. Some methods allow linking one caption to multiple figure candidates which then become a single multi-panel figure [6]. Some of the rejected caption candidates can be further extracted as figure mentions [22].

3 Multi-panel Chart Segmentation

A large percentage of figures, including charts, in academic papers are multi-panel (up to 60 percent) [27], [28]. A generalized pipeline of figure segmentation methods is presented in Fig. 3. For multi-panel chart images, some elements (e.g., the legend) may be shared by many panels. General techniques might over-segment these images resulting in the isolation of relevant sub-components that are required for the correct interpretation of all sub-charts [27]. Captions need to be split and each portion should be linked to the specific chart it describes. Therefore, advanced segmentation models are required to ensure that data extraction will be possible later.

Multi-panel figure segmentation methods are *evaluated* using recall, precision, and f1-score of correctly segmented sub-figures. Different protocols have used different criteria, usually based on percentages of overlap between candidates and ground truth panels, to determine which sub-figure candidates are considered correctly segmented [28], [29], [30].

3.1 High Level Classification

A figure can be either single-panel or multi-panel, and different classification approaches have been used to identify and handle multi-panel figures correctly. Classification can also help to determine the appropriate segmentation algorithm that should be used for a given image, and might be more efficient than running segmentation algorithms arbitrarily and risk over-segmentation of single panel figures [31]. During ImageCLEF 2015, a Compound Figure Classification task was introduced [30], and multi-panel figure classification approaches can be evaluated using data from this competition and its more recent editions. We further describe methods used for chart image classification in Section 4.

3.2 Figure Caption Analysis

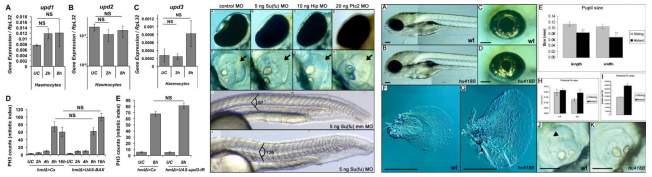
The goal of figure caption analysis is to identify sub-caption delimiters which can be used to estimate the number of sub-panels in the figure to inform and validate the image-based analysis [7], [29], [32]. Heuristic rules are typically used to identify specific strings (such as "(A)", "(B)", etc.) used as caption delimiters that link each sub-caption to one or more specific figure sub-panels [32]. The next step is to split the caption into sub-captions and associate them to their corresponding panels [29].

3.3 Panel Label Extraction

Panel labels embedded in the image itself are detected and recognized by some methods. Such labels can be helpful during segmentation and facilitate the correct linkage between panels and sub-captions [33]. Earlier models relied on high contrast assumptions and used connected component (CC) analysis to detect panel labels [7], [29], while recent methods have used path-based classifiers [34] and deep neural networks [35]. Detected labels are then recognized, but the expected classes are usually limited to consistent sets of strings commonly used as panel labels [29], [34]. Different methods such as heuristic rules [7], Markov random fields [29], convolutional neural networks [34], and beam search optimization [34], [35] have been used to ensure this consistency and to remove false positives. These methods might fail due to low image quality and irregular panel layouts [29].

3.4 Panel Segmentation

Multi-panel figures can be created by leaving gaps between panels, or by stitching together multiple images, or using a



(a) Gap-based (b) Stitched (c) Mixed

Fig. 4. Types of multi-panel figures. We show multi-panel figures using: (a) gaps, extracted from [36]; (b) image stitching, extracted from [37]; and (c) a mixture which combines gap-based and stitched panels, extracted from [37].

mixture of these (see Fig. 4). Here we cover three types of segmentation algorithms intended to work on each case.

3.4.1 Gap-Based Segmentation

These methods rely on the existence of narrow spaces (gaps) of contrasting colors between panels. First, the gaps are identified using pixel profiles [32] and other methods [26]. Then, many methods find the sub-figures using a top-down recursive splitting of the image gaps (X-Y cutting). However, under-segmentation can happen when the gaps lack enough saliency [29], [32]. These methods can over-segment charts because they typically contain gaps as part of the image [28], [29], [32]. However, these errors can be mitigated using different criteria (e.g., number of expected panels) for the early termination of the recursive splitting [23], [32]. Alternatively, the over-segmented panels can be merged using heuristic features and machine learning [27].

3.4.2 Edge-Based Segmentation

Some multi-panel images are created by stitching together images without gaps. Edges might be created around the panel boundaries, and methods in this category try to detect them for segmentation. Different image processing techniques are typically applied to the image in order to enhance and detect axis-aligned, long, non-contiguous lines representing panel boundaries [33], [38]. Panels can be segmented using recursive top-down cuts based on these boundaries [28], or by detecting corners of panel boundaries formed by these edges [33]. These methods can under-segment when no edges are found at the panel boundaries [28], and over-segment when sharp edges exist within the panels [33].

3.4.3 Object-Based Segmentation

These methods treat panel segmentation as an object detection problem. Earlier methods used CC analysis to find the panel candidates [7], [38], but these methods can over-segment charts and other figure types [38], [39]. Recent methods use convolutional neural networks (CNNs) to locate figure panels [40], [41], but without constraints, these might produce overlapping panel regions, under-segmentation for similar looking stitched panels, and over-segmentation for panels containing charts. By contrast, specialized CNN architectures which explicitly model possible figure layouts work much better [35], [41]. Unified models have also been proposed which can simultaneously identify panel and panel label candidates on the image with higher accuracy and faster performance than using separated detectors [35].

3.5 Sub-Figure Parsing

The parsing step generates the final list of panels with their associated sub-captions. Any differences between initial panel count estimates (e.g., from caption analysis) and the actual number of panels produced by the segmentation algorithm are resolved in this step. For bottom-up models with estimated over-segmentation, further clustering of panel candidates can be used to resolve the difference [38]. When multiple segmentation hypotheses are available, one option is to pick the best one based on a given heuristic criteria (e.g., panel count is the closest to initial estimates) [32]. Methods which detect panel labels on the image and/or caption can apply more complex refinements. Panels with matching labels can be accepted first [29], and layout heuristics can be used to estimate and fix panel segmentation errors [7], [29]. However, this is prone to failure due to OCR errors and false positives in panel label detection results [7], [29].

4 Chart Classification

Extracting data from charts at scale requires image classification to split charts from other visualizations. In this section, we overview figure taxonomies, and then we focus on methods used for classification of chart images.

4.1 Figure Taxonomies

Visualizations in general have been classified by their functionality [1], by their structure [2], and by visual information seeking tasks [42]. Complex hierarchical taxonomies have been defined for figures based on large scale analysis of published papers [43]. The visualization process and existing data visualizations types, including charts, have been reviewed by Khan and Khan [44]. For the scope of this survey, we are interested in the classification between chart and non-chart images, followed by classification of images per chart type.

4.2 Methods for Classification

Many figure classification approaches first discriminate between broad categories of visualizations (e.g., figure types), and then further classify sub-types of each broad category (e.g., chart types). The original figure context can also provide helpful information during classification, especially in the case of multi-panel figures [45]. We can roughly group existing classification methods into four families: model-based, heuristic-based, bags of visual words (BoVW), and deep learning. Table 1 presents an overview of the classes covered by different methods used for image classification, and Fig. 5 shows examples of some of these classes.

4.2.1 Model-Based

These approaches create models for each class using domain knowledge and heuristics [46], [47], [48] or trainable models such as Hidden Markov Models [49], multiple-instance learning [50], and decision trees [51]. Images are first analyzed to locate specific chart elements (e.g., bars) and other basic shapes. Most methods use raster-to-vector conversions for this purpose [46], [47], [50], [51]. Alternatively, trainable models may use feature sets which can capture these elements implicitly [49]. During classification, these basic objects and their layout are compared against each chart model to

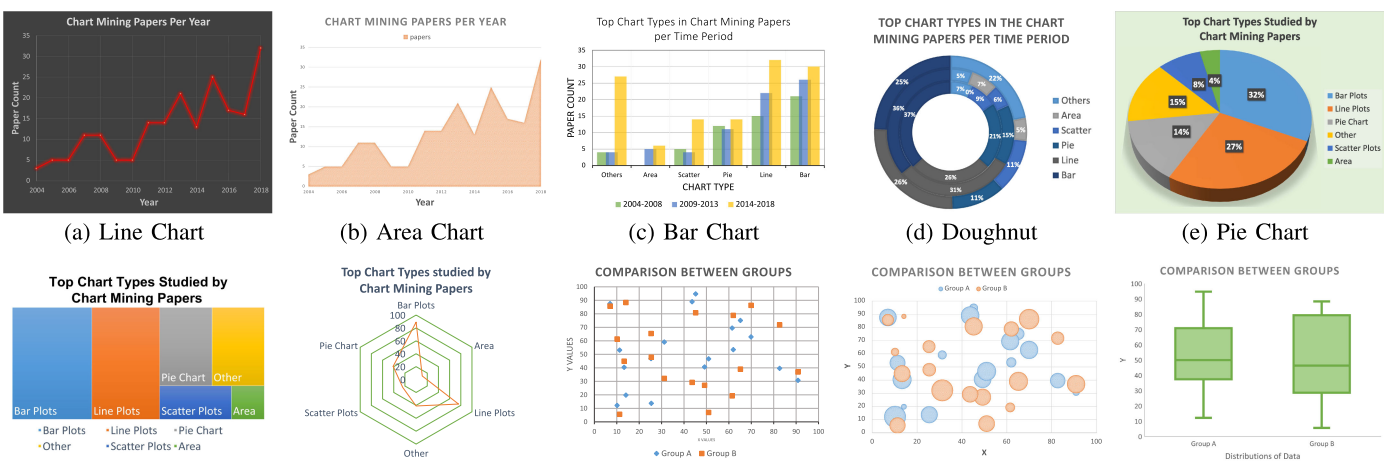
TABLE 1
Summary of Classes Handled by Multiple Chart/Figure Classification Methods

Classes	Classifier Family: Model-based	Heuristic Features	Bags of Visual Words	Deep Learning
High Level Classification: Coarse Figure Types				
Figures / Tables / Other		[54, 57, 59]	[31]	[23, 45, 61, 62, 63, 64, 65]
Panels (Single / Multi)		[28, 38, 57]	[24, 31, 60]	[45, 66]
Low-Level Classification: Charts Per Type				
Area	[47, 48]	[56, 67]	[68]	[65, 67, 69, 70, 71]
Bar/Column	[46, 47, 49, 50, 51]	[19, 25, 52, 53, 55, 56, 67]	[24, 68, 72]	[23, 65, 67, 69, 70, 71, 73, 74, 75, 76, 77]
Bubble	[51]	[56]		[65]
Doughnut	[50]	[55, 56]		
Flow				[23, 65, 73]
Geo Map		[56]	[68]	[65, 70, 71]
High-Low-Close	[49]			
Line	[46, 47, 49, 50, 51]	[19, 52, 55, 56, 58, 59, 67]	[24, 68, 72]	[65, 67, 69, 70, 71, 73, 74, 77]
Pareto			[68]	[65, 70, 71]
Pie	[46, 47, 50, 51]	[52, 55, 56, 67]	[68, 72]	[17, 65, 67, 69, 70, 71, 73, 74, 75, 76, 77]
Radar			[68]	[65, 70, 71, 77]
Scatter		[19, 52, 56]	[67, 68]	[23, 65, 67, 69, 70, 71, 73, 77]
TreeMap		[56, 67]		[67]
Venn			[68]	[65, 70, 71]
Other		[52, 56, 67]	[68]	[23, 65, 67, 70, 71]
Rejection	[47, 51]	[25, 53, 58, 59]	[24]	[23, 74]

determine the likelihood that the input belongs to that class. The model with the highest likelihood is selected as the image class, unless this likelihood is too low and/or the image does not contain all of the expected elements for that class [50]. These methods cannot distinguish between broader figure types. They also break easily due to failures during the recognition of basic chart elements and also on valid charts which do not follow the original assumptions for their class [50]. Note that the objects located during the classification process can be re-used during data extraction [50].

4.2.2 *Heuristic Features*

Methods in this category generate vector-based image representations using heuristic-based features for later classification through machine learning. Both visual and textual features have been considered. Many approaches use features to describe objects such as lines [25], curves [52], rectangles [53], closed contours [54], CCs [54], among others [19], [55]. Such object-based features are easier to obtain from vector-based graphics [56]. For images in gray-scale, the pixel intensities are used to extract multiple features such as basic statistics, histograms, moments, co-occurrence matrix and entropy [53], [55]. For binary images, other features are typically used based on density, basic statistics, and profiles [15], [16]. From edge images, other features such as direction histograms, distance histograms, and profiles have been used [52]. From color images, color-based features are also used [28], [38]. Finally, many models use different texture descriptors such as skewness, entropy, uniformity, smoothness, and edge difference [28], [57]. Other texture descriptors used include Local Binary Patterns (LBP) [58], Hough transform (HT) [54], wavelet transform [59] and histogram of oriented gradients (HOG) [52], [58]. For textual feature extraction, the caption and/or mentions are usually normalized by removing stop-words and stemming the remaining words [57], and then different features are extracted such as keywords, bag-of-words, n-grams, word embedding, TF-IDF, and sub-figure mentions or delimiters [24], [45], [54], [57], [60]. Some works use feature selection methods to make the final vector representation smaller [54]. The classification process is then carried out



(a) Line Chart (b) Area Chart (c) Bar Chart (d) Doughnut (e) Pie Chart

(f) Tree Map (g) Radar (h) Scatter Chart (i) Bubble Chart (j) Box Plot

Fig. 5. Popular chart types examples. Approximated counts of chart mining papers published from 2004 to 2018 are presented using both a (a) line chart and an (b) area chart. This count is further divided in three time periods, and we show the number of papers related to popular chart types using a (c) bar chart, and a (d) doughnut chart. The overall proportion of the coverage given to these most popular chart types is depicted using a (e) pie chart, a (f) tree-map and (g) radar chart. Finally, we use synthetic data to show examples of (i) scatter chart, (j) bubble chart, and (k) box plot.

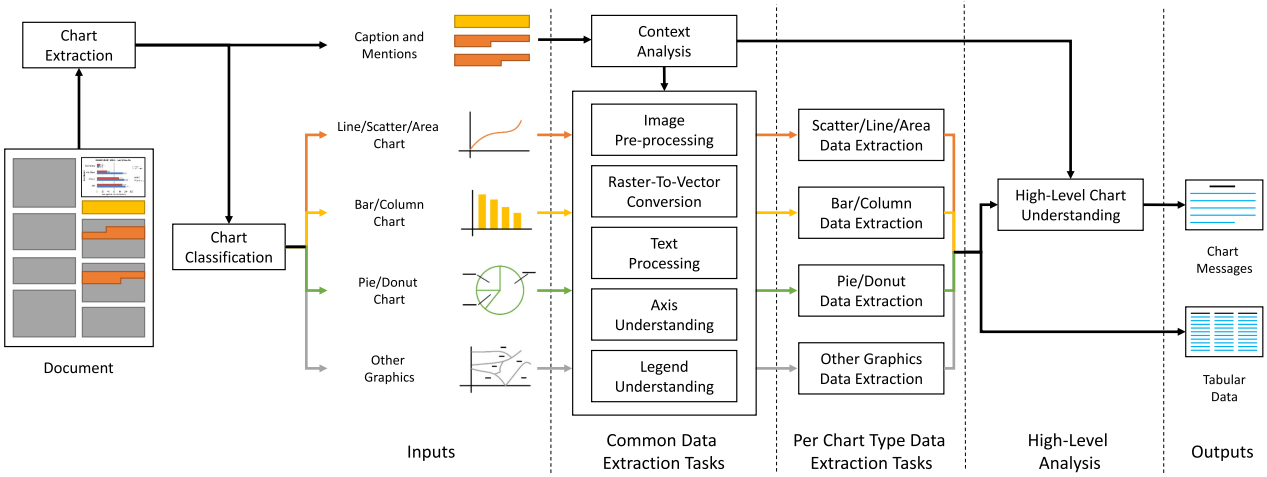


Fig. 6. Overview of the chart data extraction process. In general, chart images are processed through multiple distinguishable steps which are executed in different order by existing methodologies. Most methods conclude the process with a data extraction step which is chart-type dependent. Afterwards, many works analyze the tabular data along with the contextual information to extract the chart messages.

using machine learning techniques such as K-nearest neighbors [55], decision trees [15], logistic regression [28], shallow neural networks [55], SVM [59], boosting [52], or ensemble methods [55], [56].

4.2.3 Bags of Visual Words

These methods learn vector-based image representations through visual dictionaries made of recurring image patches or features (visual words). Patch-based methods first resize all images to a fixed resolution [68]. During training, a fixed number of visual words are sampled either densely or pseudo-randomly (e.g., by rejecting low variance patches) [24], [68]. A visual dictionary is then generated by clustering the sampled visual words (e.g., using K-means). Images are then represented by creating histograms of responses between the visual dictionary and visual words extracted from them. It is common to build multiple histograms using image regions (e.g., quadrants), and concatenating them into a single vector representation [68]. Additional heuristic features (e.g., textual features) can also be added to further improve the classification accuracy [24]. Finally, machine learning techniques such as SVM [24], [31], [68] or random forests [60] are trained for image classification.

4.2.4 Deep Learning

State-of-the-art image classification methods are based on deep neural networks. Some methods use heuristic features as the input for deep neural network classifiers [61], [69], but the majority of methods use deep convolutional neural networks (CNNs) in order to learn a feature representation from training images directly. The output of certain network layers can also be used as a trainable feature representation which can be fed to other classifiers such as SVMs [65], [69]. Common image classification network architectures such as AlexNet [78], VGG-19 [79], ResNet [80], DenseNet [81], and others, have been used with few to no changes for chart classification. Comparisons between some of these architectures and heuristic-based features have also been carried out [67]. To deal with the lack of large scale datasets on this domain, networks can be pre-trained on large image datasets such as ImageNet [82]. Image classification accuracy can be further improved by using fusion techniques that combine the deep convolutional features with heuristic-based features (e.g. textual features) [45], [63], or Fisher Vector encoders [65]. Figure extraction and classification can be combined using object detection networks [17].

5 Extracting Data from Charts

The goal of chart data extraction is to recover the tabular data used to create a chart. In the literature, we find methods for both semi and fully automatic data extraction from charts. A few examples of semi-automatic systems include: Dagra [83], Plot Digitizer [84], Engauge Digitizer [85], and DataThief [86]. Semi-automatic models can be very accurate, but they can be hardly used at scale because they require humans in the loop. Therefore, we focus on methods used for fully-automatic data extraction from chart images.

Chart data extraction is a very challenging process with strict requirements since both text and graphics need to be interpreted correctly [50]. Creating a style-independent chart recognition system is very difficult because charts can be very diverse in design [49]. Colored data marks are generally easier to extract, but many charts are gray-scale, and they might also have heavy clutter or deformations [23]. Despite these challenges, digitally-born chart images do not have other types of noise commonly found in natural images. In this section, we first survey methods used to deal with common chart data extraction challenges such as context analysis, image pre-processing, raster-to-vector conversion, text processing, and detection and understanding of axes and legends. Then, we discuss type-specific chart interpretation methods. Finally, we present methods used for higher level interpretations of charts. The entire process of data extraction is illustrated in Fig. 6.

5.1 Context Analysis

Important information can be extracted from the original context of a chart. As discussed previously, this information is useful during multi-panel figure segmentation (Section 3) and figure classification (Section 4). It is also useful during later

processes such as inferring the message of a chart (Section 5.8), chart retrieval (Section 6.4), and chart summarization (Section 6.2). In particular, many methods analyze the captions and mentions of charts, which are typically extracted alongside the charts (Section 2). This analysis requires techniques from natural language processing such as: stop-word removal, stemming, parsing, and named-entity recognition among others.

5.2 Raster Image Pre-Processing

Different image processing techniques are typically used to prepare raster images for data extraction. Many methods assume that segmentation of chart elements (e.g., bars or lines) is easier on certain color spaces (e.g., binary, gray-scale, HSV, and LAB), and they use color transformations. Based on the types of noise expected from the image source, different methods use noise removal techniques such as Gaussian smoothing, median filters, bilateral filters and morphological operations. Some systems attempt to split text from other graphical components using CC analysis and more recently deep neural networks for semantic segmentation. Afterwards, many works, especially the ones using grammars, convert the graphics image to vector as described in Section 5.3. Finally, some works remove the background grids that some charts include on the data region.

5.3 Raster-To-Vector Conversion

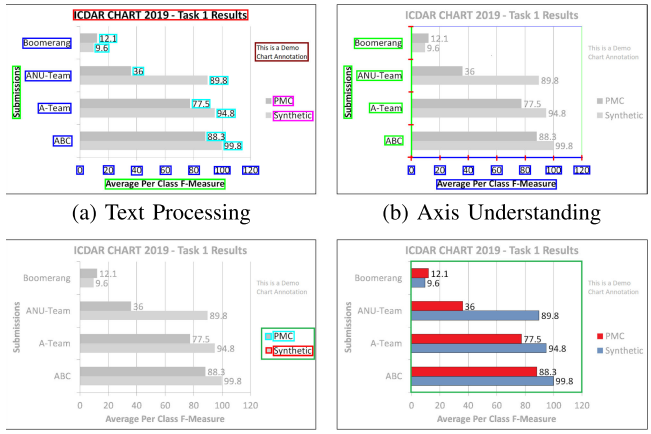
The raster to vector conversion process takes as input a raster image and produces a decomposed version of the image using vector primitives such as lines, and arcs (circular and elliptic). Many methods start with an image where text has been removed and multiple image processing techniques have been used to enhance basic shapes (see Section 5.2). One way to vectorize the image is to identify small line segments which can be grouped into larger straight line segments or circular/elliptic arcs (bottom-up) [51], [87]. Curve fitting algorithms can be used over arc candidates to validate and replace them with parametrized representations [87]. Other methods use CC-based analysis, where each CC which is not a line can be replaced by its contour for further segmentation into primitives [46]. Finally, iterative tracing algorithms have also been used which start with individual pixels that do not belong to any curve and trace them until an intersection is found. This process is then repeated until every pixel belongs to a curve [14].

5.4 Processing Text in Charts

Chart understanding heavily depends on accurate chart text processing due to the different roles that text plays in chart images. Most approaches start by locating text regions in the image, then apply optical character recognition (OCR) to recover their content, followed by classification of the role of each text region. One example of the output of this process is shown in Fig. 7a.

5.4.1 Text Detection

The complexity of the challenges for text detection in chart images lies somewhere between scanned document images and natural images [88]. Chart images have a sparse distribution of short text strings with semi-structured layouts,



(a) Text Processing (b) Axis Understanding

(c) Legend Understanding (d) Data Extraction

	Avg. Per Class F-measure	
Submission	PMC	Synt.
ABC	88.3	99.8
A-Team	77.5	94.8
ANU-Team	36.0	89.8
Boomerang	12.1	9.6

(e) Tabular Data

ICDAR CHART 2019 – Task 1 Results

This [bar chart] presents [ICDAR CHART 2019 - Task 1 Results]. Two [datasets] were used: [PMC] and [Synthetic].

There were four [submissions] to the task: [Boomerang], [ANU-Team], [A-Team], and [ABC].

Overall, the [Average Per Class F-Measure] is higher for most [submissions] on [Synthetic].

The [Highest value] is [99.8] on [Synthetic] for [ABC]. The [Lowest value] is [9.6] on [Synthetic] for [Boomerang].

(f) Chart Message

Fig. 7. Chart data extraction example. After the initial pre-processing, the data extraction process continues with: (a) text processing, (b) axis understanding, and (c) legend understanding. The order of these steps varies from method to method. Then, (d) data extraction takes place producing (e) tabular data. Methods for high-level chart understanding produce additional outputs such as the (f) chart messages.

multiple orientations, fonts, colors and sizes. Chart text detection can be more challenging due to confusions caused by the presence of other graphical objects such as plot markers. Some methods have used existing OCR systems for detection and recognition of text in charts [7], [51], but the majority of approaches use custom text detection algorithms based on: CC analysis, texture analysis, and CNNs.

Most systems using CC analysis rely on image preprocessing to binarize the image and extract the CCs. The next challenge is to identify text CCs and existing methods have used geometric features (normalized CC height, width and area), structural features (CC pixel density, binary patterns, mass to area ratio, orientation of edges), location features (centroids, bounding box corners), and texture features (Gabor filters) [14], [49]. Recent approaches have used neural networks for semantic segmentation to identify text CCs [70], [77]. After isolating the text CCs, the next challenge is to group them into words or lines for recognition. This has been done using rules about layout [59], [89], [90], segmentation trees, projection profiles [46], Newton's gravitational force formula [50], morphological operations [91], clustering algorithms [92], and Hough transform with heuristics-based line splitting [88], [93]. In their review work, Boschen *et al.* [92] compare several configurations for CC-based text detection on scholarly images using a generic pipeline. They found that adaptive binarization, dilation-based CC clustering and dilation-based orientation determination yielded best overall performance [92].

General techniques for text detection in natural scenes have also been adopted for charts [72], [74], [94], [95]. Recent approaches mainly use deep neural networks. Object detection CNNs can be used to find text alongside other chart

elements (e.g., data marks or axes) in images [75], [76], [96]. It is also possible to combine both text detection and text role classification using these networks [76].

Text detection algorithms are *evaluated* on the basis of recall and precision of detected text elements. This is described in more detail on existing benchmarks for general text detection such as the ICDAR Robust Reading Competition [97].

5.4.2 Text Recognition

After locating the text regions, the next task is to recognize their content. Most charts are digitally born and hence standard OCR engines for typeset text recognition can be used. This includes open source and commercial systems such as: Microsoft OCR [98], Tesseract [99], ABBYY FineReader [100], and Ocropy [101]. The performance of these OCR systems has been empirically compared [92]. Running the OCR system for multiple orientations can help to improve the overall recognition accuracy [92], [95]. Convolutional Recurrent Neural Networks [102] have also been used for chart text recognition [74], [75].

The next step is to refine the OCR results. Traditional strategies used for document OCR were found to provide no improvements on figure images [92]. This is most likely due to the isolated nature of text regions in charts. One option is to use text from context (captions and mentions) to apply lexical corrections based on edit distance, part-of-speech and named entity recognition [103]. However, this method is sensitive to tokenization of words especially in the presence of compound nouns [103]. Recognized text can be *evaluated* using metrics such as character error rate, word error rate, Levenshtein distance and Gestalt Pattern Matching.

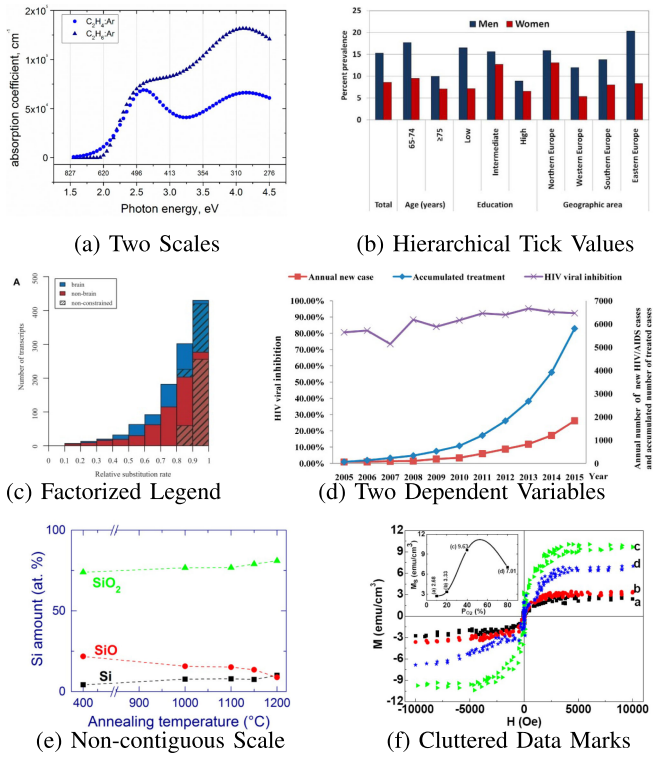
5.4.3 Text Role Classification

Accurate chart data extraction requires understanding the role of each text region on a given chart image. The most common roles include: chart title, axis title, tick labels (values associated with axis positions), legend title, and legend entry [20], [50], [70], [74], [77], [91]. It is common to further associate some roles (axis title, tick value) with their corresponding axis. Other less common roles include: data mark names, data mark values, unit labels, and other. Lack of balance between classes makes this problem harder.

Many heuristic features have been used for text role classification including geometric features, layout features, and text-based features [50], [70], [74], [77], [91]. Geometric and layout-based features of text bounding boxes include: corner locations, aspect ratio, centre coordinates, distance to image borders, angle to image centers, and angle of rotation. Text-based features include capitalization, string length and whether the string is numeric or not. Classifiers based on SVMs, Random Forests, Decision Trees and Naive Bayes have been used with these features to determine text roles. Object detection networks can simultaneously predict both the location and the role of text regions [76].

5.5 Axis Detection and Understanding

Axes recognition is required for the extraction of data from certain chart types such as line, scatter, bar and others. To reconstruct the values of data marks in the original chart space, the scale and range of each axis must be inferred by



(a) Two Scales (b) Hierarchical Tick Values
(c) Factorized Legend (d) Two Dependent Variables
(e) Non-contiguous Scale (f) Cluttered Data Marks

Fig. 8. Examples of charts complexities. (a) Two scales for a single axis, from [104]. (b) Hierarchical tick values, from [105]. A (c) factorized legend, where data series are combinations of legend entries, from [106]. (d) Two dependent variables, from [107]. (e) Non-contiguous axis scale, from [108]. A chart with (f) cluttered data marks, where individual points are hard to identify, and the data region contains addition of a sub-chart, from [109].

detecting the axes lines, tick marks and tick values (see Fig. 7b). Otherwise, only relative coordinates can be produced [110]. While most charts include two lines (x and y axes), there are many 3D visualizations which add a third line for the z axis. Other situations that make axes recognition hard are illustrated in Fig. 8. In this section, we concentrate on automated approaches for axes recognition.

5.5.1 Detecting Axes Lines

Most models assume that axes have a fixed location (e.g. y on the left and x on bottom), without any skew or line splitting. Axes lines are commonly expected to be the largest lines in a chart image. For raster images, axes lines have been detected using methods based on projection profiles, Hough Tranform (HT) and CC analysis. Methods based on projection profiles assume that axes lines will produce peaks in profiles generated from raw binary pixels [14], [89], [110], [111], [112] and tick mark locations [96], and candidate axes lines are chosen by locating these peaks. One advantage of these methods is that they can handle broken axes lines [110]. Methods based on the HT first use edge detection algorithms (e.g Canny) and then apply the HT to find the axes lines [50], [58], [59], [113], [114]. Comparisons between these two families of methods have found that HT is less sensitive to small rotations of the axes [93]. Methods using CC analysis identify the axes lines by assuming these should be connected thus representing a very large CC in the image [91]. For vector-based graphics, axes candidates

can be identified directly from straight line objects using heuristics [46], [50], [51], [115].

The final axes lines are then selected from the set of candidates. Grid lines and/or borders create bad candidates, and many methods use filtering rules based on location and length of lines, and the location of tick marks to select the final candidates. Methods that first recognize other chart elements such as text regions and data marks can use these to select the axes lines [50], [91]. For example, the baseline of the bars can be used to hypothesize the location of the x-axis in a bar chart [91]. Text role classification results can also be really helpful since axes are located around tick labels in most cases.

5.5.2 Detecting Tick Marks

These are marks located within the axes lines which help to define the chart scale. Normally these can be detected using heuristics based on their location relative to axes candidates. Under the assumption that tick marks are located at fixed intervals, some methods detect them using the Fast Fourier Transform over pixel profiles of the axes [89]. Recent models have used deep neural networks trained specifically for tick mark detection [96].

5.5.3 Inferring the Range and Scale of Axes

First, the recognized values of tick labels should be associated with their corresponding tick marks or axis locations. For axis using linear scales, the mapping from the pixel space to the axis coordinate space is an affine transformation which can be estimated from pixel-wise distances between tick values [23], and methods such as RANSAC can be used to deal with potential OCR errors in the tick values [96]. However, it is important to bear in mind that many charts use other non-linear scales (e.g., logarithmic), or have interruptions in their scales (see Fig. 8e). Nominal patterns representing units or scale multipliers (e.g., k for kilo) can be included within text regions, and they must be considered for correct inference of the range and scale of the data being extracted [70]. Some charts include hierarchical tick values (see Fig. 8b) which need further analysis to extract them correctly.

5.6 Legend Detection and Understanding

Charts with multiple data series usually include legends to identify them. Methods which handle legends can associate each data series with its original name (see Fig. 7c). In most cases, the legend entries become the headers in the extracted tabular data (see Fig. 7e). Exceptions include cases where the final set of data series is a combination of legend entries (see Fig. 8c), and charts with legends used to group categorical values within the same data series.

Some methods assume that legends are delimited rectangular regions containing text elements as well as data marks patches [24], [46], [58], and they detect legends by finding a large rectangle which holds these assumptions [46]. CC analysis can be used for this purpose [91]. Other methods rely on text role classification to detect the legend region [23], [24], [58]. They associate data marks to legend entries using position-based heuristics [23], [24], optimization models (e.g., Hungarian method) [24], and more recently relational networks [75]. Long data series names which are split into multiple text regions can make these methods fail [24].

5.7 Per Chart Type Data Extraction

5.7.1 Line, Area and Scatter Charts

These charts are semantically similar since they present one or more sequences of 2D points on a given Cartesian plane, and related methods have been used to recognize them in the literature. The *Scatter chart* is the most basic kind in this group, and when lines or curves are used to connect its data points then it becomes a *line chart* (also called *curve chart*). When the goal is to highlight the area under each curve, typically by using coloring or texture patterns, then it becomes an *area chart*. These types of charts are commonly used in academic papers, mostly to represent experimental data [26]. They can be challenging to recognize for several reasons including: the presence of arbitrary text and/or grids on the data region, cluttering, non-white backgrounds, lack of legends, overlapping non-colored curves, and the use of colors which are hard to distinguish visually [26]. Some of these complexities are illustrated in Fig. 8, including the case of a chart which contains a sub-chart in the data region (Fig. 8f).

Methods for recognition of line, scatter and area charts work under multiple assumptions. The majority assume that charts are created using a regular process [14], without unexpected elements in the data region [110]. Many consider that the chart might have grid lines, but these should not be more visually distinctive than the data marks [116]. Other approaches implicitly assume that data is located in the first quadrant of the Cartesian plane, while a few have no such constraints. Some methods can only handle a single data series in the chart [110], [116], and for those supporting multiple data series, it is common to assume that these will use different colors [74], [115], [116]. Very few works support line charts where a single x value might be associated with multiple y values on the same curve (e.g., step function) [110].

For *line charts*, the next step is to recognize the lines. Legend analysis results can provide an estimation of the number of lines in the chart. For digitally-born vector graphics (e.g., SVG, D3), chart lines can be extracted accurately as long as they can be correctly identified, and overlapping curves and dashed lines do not pose a challenge as they do for raster images [24], [117]. Methods that work with raster images require different approaches to extract the lines such as: sampling, tracing, segmentation, and grammars.

Sampling-based methods find the points where chart lines intersect a set of vertical lines [118] Then, these points are clustered into lines using local descriptors. These methods can handle broken and dashed lines.

Tracing-based methods scan the data region to find pixels that belong to chart lines and trace them using pixels [110], connected line segments [46], or patches [58]. Only the patch-based tracing methods can handle broken and dashed lines, but they might break when chart lines have sharp gradients [58].

Methods based on segmentation aim to split entire chart lines from the background pixels. This has been done using color-based heuristics (e.g., Hue-based color quantization) [74], [115]. Other methods have used deep neural networks to

learn embeddings for data mark patches, which in turn are used to segment out lines by minimizing the cost of matching patches from lines to data mark patches from the legend [23]. Some of these methods can handle broken lines [115].

Methods based on grammars start with a vectorized image, and using bottom-up parsing they recognize the chart lines from low-level elements such as straight lines and elliptic arcs [14], [47]. The main drawback of these models is the cost of defining grammars that can properly handle large collections of charts [47]. Some of these methods can only parse multiple lines if they have different colors [14].

For *scatter charts*, the next step is to locate the data marks. If the legend is recognized, then shape [111] and color [115] can be used to identify data marks by data series. Further processing might be required to split overlapping data marks [111]. For methods relying on color-based heuristics, colored text in the data region might cause errors [115]. Recent methods locate data marks in charts by using object detection networks [96], but they can over-segment data marks and might not work well on dense charts.

Some charts contain both lines and data marks, and are generally considered line charts, but sometimes the lines represent trend lines on scatter charts. Some models treat these as scatter charts by removing the lines using k-median filters and estimations of line thickness [59], [89]. For charts in vector format, repeated shapes can be grouped together if they are all intersected by the same line candidate [115].

The pixel coordinates of data marks and/or extracted lines are projected onto the axes to determine their relative coordinates. If axes were fully recognized, these relative coordinates can be projected onto the original data space. The output is a table with the data points represented in the chart in their original coordinate system. Data value labels cause problems for many methods, but these should be used to refine local estimates [118]. The *evaluation* of these data extraction methods is challenging because ground truth is limited. Some methods use qualitative evaluation based on chart reconstruction [58]. Recent methods are using fully annotated synthetic datasets which allow them to evaluate their models using recall and precision of the extracted data marks [96].

5.7.2 Bar and Column Plots

These charts are commonly used to visualize data series which have a categorical independent variable. The chart sub-type (either bars or columns) is usually determined either during classification or later by analyzing the widths and heights of the candidate bars. The same techniques are usually applied on both chart sub-types, and these work under a variety of assumptions. Many assume that bars should be in the first quadrant of the chart [91], and they should have a single solid color filling, without 3D effects or text written on the bars [68]. Very few methods handle stacked bars and they expect these to have different colors [76]. Grouped bar charts are also expected to have large spacing between each group of bars [77]. Many methods assume no grid or interpolation lines should be on the chart.

The next main step is the detection of bars/columns. Methods working with vector graphics can easily identify the bar rectangles with high accuracy [117], but different families of approaches have been used for raster graphics. Multiple works parse charts using grammars [14], [49] and other forms of explicit bar chart models [46], [47], [50]. These operate in a bottom-up fashion by first identifying small primitives such as lines using methods for edge detection [49], vectorization [46], [47], and color-based segmentation [14]. The primitives are then grouped into higher level components such as bars, and this can be done through optimization processes that test multiple grouping hypotheses against layout constraints involving other chart elements (e.g., text, axes), and keep the most promising bar candidates for further data extraction [47], [49]. Grammars, however, rely heavily on heuristics and hard assumptions which do not cover many real charts, and early mistakes can affect the whole recognition process. An alternative is to learn the models from data [49], where heuristic features can be used to map specific components of bar charts (e.g., beginning and ending of bars).

Other methods aim to directly detect bars on chart images, and they have used a variety of methods based on: CC analysis, contour analysis, projection profiles, and deep neural networks. Methods using CC analysis assume that bars should be solid single-colored regions. Bars are differenciated from other rectangular regions (e.g., the legend, grid cells) using classifiers and heuristic features based on shape, pixel densities, color uniformity, and relative distances to axes [68], [76], [77], [91]. However, these methods might fail to detect small bars [68]. Methods using contours assume that bars have a solid border which can be detected using contour tracing algorithms [114], and further heuristic rules can be used to identify bar candidates from these contours. Methods using pixel projection profiles also assume that bars have solid borders and/or backgrounds which can be detected as plateaus in the pixel projection profiles over the x-axis [119]. Methods based on deep neural networks use object detection architectures to locate the bars [74], [75], [120].

Further data extraction requires full text processing as well as axis and legend recognition. These processes identify the categorical values which are then linked to bars using mostly layout-based heuristics. Legend recognition is used to identify the names of data series on stacked and grouped bar charts. A recent approach has used relational networks to identify links between bars, legend entries and categorical values [75].

The final step is to determine the bar values by projecting their widths or heights into the dependent axis. Using axes recognition results, these relative values are then projected into the original chart data space to produce the final tabular data (see Figs. 7d and 7e). Normally, many bars share a common baseline usually aligned with the independent axis, and the minimums and maximums of each bar can be used to identify it [68]. This baseline must be considered to interpret bar charts with stacked bars and/or negative values correctly. Overall, all methods are sensitive to OCR errors, and rotated labels are a common source of errors. Most of the methods described here cannot deal with bars using non-solid color patterns.

5.7.3 Pie and Donut Charts

These charts can be used to display a single data series, where each pie slice represents a data tuple of the form "(category, value)" [68]. The categorical values can be specified using

either the legend or text labels around the corresponding slices. Methods for recognition of these charts work under assumptions such as having solid single-colored pie slices without 3D effects [17], [74], [75], [76].

Extracting data from these charts requires locating the pie and its slices. This can be done accurately on vector graphics [117]. For raster images, different families of methods have been proposed including: grammars, curve fitting, CC analysis and deep learning. Methods using grammars typically recognize the pies using bottom-up parsing from vectorization results [14], [46], [47], [87]. Pie slices can be formed by combining straight lines and elliptic arcs, and multiple slices are combined to form a single pie or doughnut. Methods using curve fitting locate the pie by finding ellipsis candidates on gradient or edge images (e.g., by using RANSAC) [68], [76]. The slices can be further located by finding sharp transitions between colors (the slice boundaries) on one or many smaller concentric ellipses [68]. Methods using CC analysis locate the pie slices under the assumption that these have single solid-color patterns [17]. Recent methods based on deep learning have used object detectors to locate the pie [74], [75], and some a rotation fitting component to predict the angle of each slice [75]. Slice detection and legend analysis can be combined through relational networks [75].

After finding the slices, the percentage associated with each slice can be found by dividing: the angle of each slice by 360 degrees [50], the area of each slice by the total pie area [17], [74], and the number of pixels from each slice by the total number of pixels in a circular sample of the pie [76]. For charts without a perspective distortion, the values obtained by these methods are close to the true values. However, charts with perspective distortions (e.g., 3D charts) require the projection of the elliptical pies into circles.

These methods can fail due to several reasons including small slices that go undetected, and categorical values that are associated with the wrong slices because they are the closest [68]. Systems working with gray-scale images might accidentally merge neighboring pie slices [17], [74]. Many methods degrade when used over 3D pies.

5.7.4 Miscellaneous Chart Types

Due to their highly specialized nature, many chart types have very little coverage in the existing chart recognition literature. Some of these chart types include: high-low charts [49], topographic maps [121], map charts [122], meteorological facsimile charts [123], limnigraf charts [124], and phase diagrams [125].

5.8 High Level Chart Understanding

Many charts are designed to simply display data, but some are created to highlight specific trends and other relevant features of the data itself [126]. This is the high level message of the chart which can be enough to summarize the entire graphic [126], as illustrated in Fig. 7f. Many methods aim to infer these messages from chart images, and here we discuss the considerable amount of work done towards this goal.

The high level messages that can be transmitted by different types of charts have been explored in the literature including bar charts with one [126] or multiple [127] data series, line charts [128] and pie charts [129]. Based on what the chart messages describe, they can be coarsely grouped as: trends, ranks, gaps, relationships, saliencies, entity comparisons, computations, and others. Some charts might illustrate multiple messages, but they attempt to identify the most relevant and therefore other relevant ideas might be missed [130]. They evaluate their method in terms of its accuracy in ranking first (with confidence $> 50\%$) the top chart message as annotated by human coders [126].

Carberry *et al.*, use a plan inference technique based on Bayesian Networks [131] for recognition of chart messages. Nodes at the top-level of the network represent all the possible messages for the current chart type, and the probabilities at these nodes represent their likelihood based on the available evidence. The next level represents the evidence nodes which are provided by communicative signals detected on the chart such as the relative effort required by the viewer to perform a given perceptual or cognitive task, saliency of graphical elements, and text-based signals from the caption. Here, a perceptual task represents actions such as comparing bar heights, while a cognitive task represents deductive work such as making computations from the chart. The relative effort of performing some visual tasks was studied using eye-tracking in order to estimate these efforts on new charts [132]. Saliency of graphical elements is determined using color, references to such elements on the caption, or any highlighting on the chart [130]. Finally, the caption might provide strong cues about the chart message by using certain verbs or adjectives, and also by mentioning the dependent and independent variables of the chart [133]. Note that is possible to make inferences using this model even when some of the evidence is missing.

Other methods have been proposed to predict high-level chart messages. A multi-modal approach which combines textual and convolutional features using deep learning has been proposed for prediction of messages in line charts [134]. Other works have used natural lenguage processing to extract explicit ontologies, triplets of the form "(subject, predicate, object)", to describe 2D charts at a high level [119].

6 Chart Analysis Applications

6.1 Redesigning and Improving Chart Visualization

A chart might be redesigned for many reasons such as making them easier to understand, reducing their design bias or simply to improve their aesthetics. In this section, we cover methods for chart redesign which include automatic components for data extraction from the original charts. More general tools used to design and redesign visualizations are presented in the survey by Mei *et al.* [135].

To redesign charts in raster format, the first step is usually classification followed by chart data extraction, which in many cases is carried out semi-automatically [14], [68], [71], [136], [137]. Any limitations on these processes will also place limits on the redesigning capabilities of a system. Other systems work with programmatic vector graphics such as D3 which can include the original data in the file [117]. By automatically inferring the mappings between data fields and attributes of data marks in the graphic, it is

possible to handle a variety of visualizations without explicitly classifying them [117].

A chart can be redesigned in multiple ways after its data has been extracted. Galleries of redesigned versions of the chart can be created [68], which are helpful to let the user chose a new design. This process can consider aspects of the user (preference, experience, visual/verbal working memory, etc) in order to redesign charts specifically for them [138]. Then, interactive interfaces can be provided to let the users modify the new chart design [14], [137], or a chart specification can be exported to edit it with external tools for chart design [70].

The original chart design can be abstracted into templates [117], [139], or it can be enhanced by adding graphic overlays [136], [140] such as reference structures, highlights, summary statistics, and descriptive text, even from the source document [141]. Qualitative comparisons and evaluations of redesign tools can be carried out through user studies [71].

6.2 Textual Summarization of Charts

Textual summarization of charts is one of the most targeted applications in the literature. Chart summaries can further facilitate other chart applications such as retrieval or accessibility. According to Demir *et al.* [142], the challenges of creating textual descriptions from numeric data include: (1) selection of the most relevant information, (2) which must be organized into coherent/fluent text, (3) using complex but understandable sentence structures, (4) produced in a given language (e.g English) with appropriate expressions in the text. In this section, we focus on abstractive summarization methods, where both content and context of a chart are used to create a meaningful textual description.

Summaries are based on chart facts that are collected using automated methods for data extraction (see Section 5), or by relying on meta-data made available during the creation of the chart (e.g., by using plug-ins [143]). Facts can be extracted from the chart image, its data and related text. Textual facts are based on analysis of captions and mentions [24], [144], other text surrounding the chart [48] and the remaining document text [144]. Further text analysis can produce additional derived information such as a referent for the dependent axis [145], and short descriptors for long lists of categorical values making the summaries more concise [145]. Based on the specific chart type, additional facts can be derived based on existing trends [24], [91], the shape of data marks [48], and salient chart elements [91], and any inferred chart message.

Not all facts are required in a text summary, especially in interactive environments where users can request them [146]. A study [146] found that users prefer initial summaries containing descriptions of salient chart elements as well as propositions which helped to avoid drawing false conclusions from the summary. To produce an initial summary, the facts and/or document sentences can be initially ranked based on training data [128], similarity between them and chart mentions and captions [144], and other ranking algorithms such as PageRank [147]. Adding one fact to the summary will reduce the relevance of the remaining ones due to information coverage, summary length, and redundancy. As such, many methods will iteratively add the top ranking fact and then re-weight the remaining ones [144], [147].

After selecting facts, the textual summaries can be generated. The simplest method is to use templates [48], [114]. Other works first represent the extracted facts using trees [146] and other graph-based structures [91]. Sentences in natural language can be then generated by navigating these data structures (e.g., by using protoforms [91]). Tree structures can be modified to re-organize the same facts (tree nodes) in a way that the overall complexity is reduced thus resulting in a better summary [142]. The vocabulary level of the original document can be considered to produce summaries that are consistent with it [147]. Finally, the generated summary can be used to replace the chart in the original document, but it should be placed around the most relevant paragraph [148].

Different systems have been developed for automatic generation of textual summaries of charts [143], [146] including bar charts [91], [114], [142], line charts [24], [128], [147], area charts [48]. Other systems summarize figures in general [144]. *Evaluation* of textual summaries of charts has been carried out using precision oriented metrics such as fact accuracy (correct or approximately correct facts) and summary relevance [91], [145]. However, these metrics do not account for relevant facts missed by the summary. Overall, chart summarization methods are very sensitive to OCR errors and other mistakes made by the automatic data extraction process [91].

6.3 Charts and Accessibility

Accessibility systems such as screen readers handle images by relying on source-based tags which are generally not enough to explain charts [46], [143]. To overcome this, chart accessibility systems aim to help users with visual impairments understand charts. This has been done by providing tools for accessing chart data directly, and also by providing meaningful summaries of the chart data (see Section 6.2).

A naive system can simply read out loud all data points from a chart. However, other models use sounds and tactile representations. For line charts, one can play a continuous sound with varying pitch based on the shape of each line as a function of the x axis [149]. This makes it easier to determine the slopes and can also be very effective in delivering mathematical concepts such as symmetry or monotonicity [149]. However, this might not work well on complex visualizations, and recent models combine sound with interactive exploration [150]. Methods using tactile representations translated the extracted chart data to Braille. One example is the VIEW system which works for bar, pie and line charts [72].

Methods based on textual summaries use speech synthesis. Short summaries can be provided first, and interactive options allow the users to request additional facts [151], [152] (see Section 6.2). The raw data can also be accessed and described using sounds [113] and other methods described earlier. Congenitally blind users who have never seen charts might learn more from descriptions of the chart message than from descriptions of what the chart looks like [53]

Many tools for chart accessibility have been implemented as web browser extensions that complement existing screen readers. Some of them automatically detect charts on web pages and extract their data. After this, some tools replace

these chart images with sound files [113], HTML tables [74], and other interactive documents to explore the chart data [72] and its textual summary [151]. Other web-based tools avoid chart analysis but depend on counterparts that need to be used during the chart generation process [152]. The evoGraphs jQuery plugin [153] can be used to create visualizations that are directly compatible with screen readers. Recently, interactive tools have been proposed to let visually impaired people create visualizations for sighted users [154].

To create accessibility systems, it is important to include prospective users at every stage of the process [74], [154]. During evaluation, it is common to include both sighted and visually impaired people to test the effectiveness of the systems including their interaction capabilities [151], [152]. These studies have shown that blind users generally adopt very different follow-up strategies when compared to non-blind users [152]. Based on their findings, authors recommend keeping in mind the limitations of screen readers with graphic material, to place chart descriptions very close to them, and to be consistent when listing chart elements using text [152]. Additional studies and technologies for chart accessibility as well as existing guidelines for creating more accessible charts have been reviewed in the work by Martinez *et al*. [155].

6.4 Chart Retrieval

Charts can be retrieval targets in many situations, especially when they contain data not available elsewhere. The goal of chart retrieval is to produce search engines which consider information from the charts themselves (data, text, etc) instead of relying on text from meta-data or context like most traditional search engines do [156]. Here we cover methods for indexing and retrieval of charts.

Chart search engines first need to create an index based on image and text analysis. Multi-panel figures can be segmented and indexed individually [26]. Further image classification will enable retrieval of specific image types [26], [31]. Images are typically indexed based on text extracted from captions [31], [156], [157], the image itself [20], [23], [31], [157], the chart message [157], and textual summaries of the charts [157]. Chart text can be indexed based on its role [20], [23]. To increase the chances of matching text, one can use techniques such as stemming [31], acronym expansion [156] and other high-level natural language processing methods [26]. In addition, some applications such as style-based retrieval of info-graphics [158] require indexing images by features such as HoG, LBP, GIST, and color histograms. Charts in text formats (e.g., Vega-Lite) can be indexed by style attributes directly [159].

After indexing, the next step is retrieval which can be divided into query matching and ranking. Many systems deal with key-word based queries [20], [31], where these can then be matched to specific text roles [20], or even chart style properties [159]. To improve precision, some methods prefer longer queries which can be compared against full textual descriptions of charts [157]. The goal is to automatically identify specific entities (e.g., the dependent and independent axes) from the query, and match these to what is stored in the index [157]. To improve recall, some methods consider query expansion by adding synonyms of query key-words [20], [157]. Unfortunately, systems relying on chart text extraction for indexing can be very sensitive to OCR errors [23]. Besides text-based queries, other systems allow query by example, where for a given query image, the system finds the closest images based on a given similarity function [158], [159].

Finally, ranking sorts the matching images by decreasing relevance to the query. For keyword-based queries, state-of-the-art methods from general text-based search engines can be used for this purpose. For text indexed by its role, one can use TF-IDF scores for each role, and then all query terms can be combined using role-wise weights to produce the final ranking [20]. Each document can have its own overall score (e.g., based on impact estimates [31]) which can also be used to rank the images it contains. One can also use machine learning to predict the relevance of a given image to a given query using both text-based [20], [157] and/or image-based features [160]. Many methods use triplets of the form (query, candidate, relevance) to learn to predict candidate relevance [20], but humans are generally inconsistent in rating candidates. For this reason, other methods learn similarity metrics using triples of the form (image, similar, dissimilar) [160], which humans can produce more consistently. These methods can also collect data by focusing on adversarial examples [161], and the data collection process can be up-scaled using Crowdsourcing [158]. Using this training data, different methods have been trained to rank figure and chart candidates including mixture models [157], weight-based distance metrics [20], [158], and neural networks [160]. In some applications, diversity of results is considered an important factor and search results might be shown in randomized order [159].

Some figure retrieval systems in the literature have considered charts and diagrams explicitly [20], [23], [45], [157], [158]. Other related systems for general figure retrieval are covered in the review by Sanyal *et al*. [162].

6.5 Visual Question Answering

The goal of visual question answering (VQA) systems is to provide useful responses to natural language questions with respect to specific images. General approaches for VQA can be found in the survey by Wu *et al*. [163]. In this section, we focus on strategies for chart question answering (CQA).

Chart questions have been categorized as structure understanding, data retrieval, and reasoning [164]. Different procedures can be used for each type of question [50], [120], [164]. Data retrieval questions can be answered by simply executing the required operations on the extracted chart data [50], and some systems can accurately determine these operations by relying on template-based questions which explicitly state the data source and variables involved [23]. Many other questions, specially the structural ones, can be answered using classification methods that choose one answer from a fixed vocabulary [120], [164]. However, there are many open-ended questions requiring out-of-vocabulary (OOV) answers which must be produced through more sophisticated methods. We note that existing models rarely consider rejecting invalid questions, and will forcefully produce invalid answers for them.

Initial results show that out-of-the-box state-of-the-art methods for VQA do not perform well on charts, except for

questions related to structure understanding, involving chart properties and/or basic counting operations [164]. Apart from traditional image and question embeddings, CQA methods usually consider chart-specific operations such as detection and recognition of chart elements, especially text [120], [164], [165]. Many of these models deal with OOV answers by creating image-specific dictionaries to dynamically encode the detected chart text regions and any reference to these on the input questions [164], [165], [166]. After this, the system can combine the image-specific text encoding with a fixed vocabulary to select the final answer through classification, usually involving some form of attention [164], [165], [167]. An alternative to these encodings is to use a regression branch to directly identify text regions containing the answer [167].

Other families of methods first convert the input chart into a table, and then use table-based question answering systems [120]. This can be useful with charts formats such as Vega-Lite which already includes the original tabular data used to create the chart [168]. Data-related questions requiring computations can be handled effectively by these systems [120], [168], but the input questions should be modified by converting all references to visual properties of chart elements into their corresponding data references [168]. The query constructed by the QA system can also be used as the basis to provide an explanation of the answer to the final users based on templates and the reverse mappings between data fields and visual attributes of chart elements [168]. Overall, methods for VQA on raster images are really sensitive to OCR errors [166]. Finally, the usage of Iterative CQA has been proposed to approximately reconstruct the original tabular data [166].

6.6 Bibliometrics

The field of bibliometrics studies ways to help researchers discover important papers among the increasing pool of academic literature [23]. Some works have studied which types of visualizations are commonly used per discipline, and how these visualizations and their corresponding captions affect the citations that a publication receives [5], [31]. In a study with 5 million figures [31], authors found that papers with more citations have more diagrams (e.g., schematics, conceptual diagrams, flow charts, architecture diagrams, illustrations) than charts. Two potential explanations were given - papers with good visualizations are more effective, hence producing higher impact; novel ideas require more visual explanations (more diagrams, less charts with experimental results). Another study [5] found a small correlation between paper citations and their figure and table count, especially when these have longer captions.

6.7 Other Applications

There are many other helpful applications for chart analysis that have received little attention from the research community. In this section we briefly cover some of these.

6.7.1 Chart Quality Assessment

Principles from educational psychology and cognitive theories have been used to create quantitative metrics of chart quality i.e. effectiveness of information transmission from the graphic to the reader [51]. These metrics include: spatial location quality, label completeness, graphic contrast, and multi-modality consistency. Computing these metrics requires image and natural language processing, and they are very sensitive to OCR and vectorization errors.

6.7.2 Automated Chart Grading

Questions related to charts and diagrams in student assignments can be automatically graded by comparing similarity of submitted charts to the ideal answer. This has been done for Venn and Euler diagrams in SVG format [169], where similarity is measured using labels of sets, curves used for sets, regions in the diagram, number of elements, and shaded zones.

6.7.3 Chart Plagiarism Detection

Similarity metrics can also be used to detect potentially plagiarized charts. Extracted text and chart data have been used to test for plagiarism of bar charts [170]. However, such methods cannot distinguish between authorized reproductions and plagiarized charts.

6.7.4 Chart Data Preservation

Historical records contain large amounts of data only available on non-digital charts which can be preserved and digitized by automatic data extraction methods. Axis analysis and image processing techniques have been used to preserve historical autographic weather charts (thermographs, microbarographs, hydrographs) [112].

6.7.5 Predicting Hashtags

Hashtags can be used to support applications such as search and browsing of graphical content including infographics. Text extraction along with object detection and classification having been used to generate textual and visual hashtags from Infographics [171].

7 Datasets

Chart mining solutions require chart datasets for training and evaluation. Most works in the literature use small, often private, datasets. This is due to the fact that creating manually annotated chart datasets is often a complex and time consuming task. However, recent works have started using semi and fully automatic approaches to create larger datasets which are also public. Here we cover existing chart datasets and the methods used to create them.

7.1 Manual Dataset Generation

A collection of chart images can be turned into a dataset with the help of human annotators. Before the labeling process takes place, it is common to apply some quality controls to the data [158]. The original image meta-data can provide some relevant labels [56], but others need to be captured through special user interfaces [172]. Crowdsourcing can up-scale the annotation process but it requires strict quality controls to reduce noisy labels [134], [141], [158].

The annotation process depends on the labels required to train and evaluate the task being targeted. Classification tasks require chart images and/or chart elements to be

TABLE 2
Summary of Datasets Used for Training and Evaluation Methods for Extraction of Figures From Documents

Domain	Size	Unit	Used By
Multi	979	Pages	[15]
Bio Med	2,256	Articles	[7]
Multi	207	Articles	[6]
CS	346	Articles	CS-Large [5]
CS	20,000	Articles	[23]
Multi	$>$ 5,500,000	Articles	DeepFigures [12]

associated with classes from predefined sets [173]. Detection tasks usually require collecting the position, size and orientation of certain chart elements such as text regions, legends, axes, and data marks [173]. For text regions, their transcription and roles are also collected. High-level recognition tasks require collecting certain relationships among chart elements (e.g., linking data marks to data series), or even between the document and specific chart elements [141]. Retrieval tasks require relevance assessments which are commonly captured as scores for query-candidate pairs, or alternatively by capturing the user preference over image candidates for a given target [158], [160]. For chart summarization, some authors have collected textual summaries for isolated charts [172].

7.2 Semi-Automatic Dataset Generation

The semi-automatic approach combines manual labeling with automated methods to speed-up the annotation process. This approach works better on tasks which are too hard to fully annotate by hand. For example, the generation of ground truth at the pixel-level and/or vector-level [174]. Depending of the complexity of the labeling process, the users can provide inputs and/or correct the outputs for automatic algorithms in one or multiple interactive rounds. For example, an automated text detection algorithm can be used on a chart image to obtain candidate text regions which can be corrected by the user [174], and afterwards the transcriptions for each text region can be generated using an OCR engine followed by manual corrections [68], [173]. For classification tasks, a small but diverse set of images can be manually labeled. These labels are then propagated to similar images followed by manual verification. This process can be repeated until a large number of images has been annotated [65]. For CVQA tasks, human annotators can be source of question templates which can be used to automatically generate thousands of questions based on arbitrary chart data [120], [168], which in turn can be paraphrased manually by humans [120] or automatically by translator systems [165].

7.3 Automatic Data Generation

Automatic chart dataset generation requires data sources and models for synthetic chart generation. The data sources can also be fully synthetic, real or derived from real data [165], [173]. Considering a noise model during the generation process might also lead to better data to train more robust systems [67]. For example, a method proposed printing, scanning and cropping synthetic charts to add realistic noise [116], while other approaches simulate such noise programmatically [174].

Chart rendering tools capable of producing ground truth at different levels for each chart image are required. Despite the fact that relying on a single chart rendering tool has considerable limitations in terms of variety [174], most works tend to use a single tool for their entire chart rendering process. A few works have used commercial software such as Microsoft Excel [75], [114], SAP reports [14], or MATLAB [116]. However, the majority of works prefer non-commercial packages such the XML/SWF Charts tool [47], [175], and the Vega language [67], [70], [176]. Some early works created their own custom chart generation tools in order to gain full control of the data generation [174]. This allowed them to create ground truth at any level including pixels and vector primitives. Many recent works use the Matplotlib library [177] to create large scale synthetic datasets because it allows them to generate ground truth at the level of chart elements [75], [96], [164], [165], [173].

Synthetic datasets have also been created for document segmentation tasks related to chart mining (see Section 2). Using existing collections of papers which include both a PDF version of each paper and its corresponding sources (e.g., LaTeX or XML plus figures), it is possible to apply simple heuristics to automatically generate labels for figure locations, and these can be used to train and test methods for extraction of figures from documents [12]. For multi-panel figure segmentation, it is possible to use collections of single panel figures to generate synthetic multi-panel figures with random layouts [40]. Ground-truth can be generated for these images at the level of both pixels and bounding boxes.

TABLE 3
Summary of Datasets Used for Image Classification

Size	Datasets
Classes: Single vs Multi Panel, Figure Type	
20,000	ImageCLEF 2015 [28], [33], [39], [64], [66]
21,000	ImageCLEF 2016 [40], [41], [45], [62], [63], [64], [66]
Classes: Figure Type	
$\leq$ 1500	[24], [61]
33,070	DocFigure [65]
Classes: Chart Type	
$\leq$ 2000	VIEW [72] , [52], [69], [49]
2,500	ReVision [68], [70], [74]
5,000	DeepChart [73]
6,997	ChartSense [71]
4,242	CHART 2019 - PMC [173]
11,174	Chart Decoder [77]
17,154	[67]
202,550	CHART 2019 - Synthetic [173]

Different sets of classes have been considered: single-panel vs multi-panel, figures per type, and charts per type

7.4 Existing Chart Datasets

In this subsection, we present a summary of datasets that have been used for different chart mining tasks. In Table 2, we present a summary of datasets for extraction of figures from documents (Section 2). These generally include document pages annotated with the locations of their figures. Many of these also annotate the locations of tables, and the captions for both figures and tables. In Table 3, we present datasets used for image classification (Section 4). This table

TABLE 4
Summary of Datasets Used for Chart Data Extraction and Other Related Applications

Graphics	Domain	Size	Used By
Line	Media	215	SIGHT [128]
Bar	Media	330	SIGHT [127]
Bar, Line, Pie	Multi	200	CHIME-R [50], [92]
Line	CS	882	[90]
2D Charts	CS	332	[70]
Bar	Multi	213	[91]
Bar	Synth	300,000	DVQA [164]
2D Charts	Synth	224,377	PlotQA [120]
2D Charts	Synth	250,000	LEAF-QA [165]
2D Charts	Bio Med	400	CHART 2019 [173]
2D Charts	Synth	202,550	CHART 2019 [173]
Infographics	Multi	29,000	Visually29K [171]

TABLE 5
Summary of Datasets Used for Evaluation of Text Processing in Figures

Graphics	Domain	TDR	TRC	Size	Used By
Figures	Multi	Yes	No	441	[92]
Figures	News	Yes	Yes	475	[70]
Figures	Bio Med	Yes	Yes	10,642	[34]
2D Charts	Bio Med	Yes	Yes	400	[173]
2D Charts	Multi	Yes	Yes	202,550	[173]

The main sub-tasks are: text detection and recognition (TDR) and text role classification (TRC)

includes different hierarchies of image classes: single-panel vs multi-panel, figure per type, chart per type. Table 4 presents datasets which have been used for recognition of charts and other graphics (Section 5). Note that very few of these datasets have been made publicly available, and even fewer of these are reaching the scale required by state-of-the-art techniques such as deep learning. In addition, most of the large datasets are synthetic resulting in multiple limitations despite the scale. Finally, Table 5 presents datasets used for chart text processing (Section 5.4). Two main sub-tasks are considered here: text detection and recognition (TDR), and text role classification (TRC).

8 Discussion and Conclusions

8.1 Discussion

Automatic extraction of the data encoded in charts is difficult due to the diversity of their types, styles, structures and noise [9]. There are many technical challenges in the process which have not been completely addressed in the existing literature, but rather deferred through hard assumptions in most cases. In a previous analysis of charts extracted from PubMedCentral [173], we found that most of these hard assumptions are broken by real charts from academic papers. For example, the empty space of the chart data regions is commonly re-used to added smaller sub-charts (see Fig. 8f). Many charts display more than one dependent variable (y-axis), each one with its own scale (see Fig. 8d). Some charts have interruptions on the axes lines which can be interpreted as a non-contiguous scale (see Fig. 8e). Despite being common, none of the works covered here dealt explicitly with these and other chart complexities. In addition, many recent methods prefer working with charts in vector formats, but there are vast amounts of charts only available in raster formats.

Methodologies used for chart recognition have evolved considerably as well. Many early methods concentrated on bottom-up approaches which usually started with raster-to-vector conversions when needed and then used grammar-based parsing to recognize chart elements from low-level to high-level. These methods were more generic in the initial steps of the process until the recognized elements had to be interpreted as data. Recent methods recognize type-specific chart elements from the outset. They also rely more on data and machine learning for high-level chart interpretation.

In general, state-of-the-art computer vision techniques have not been fully adopted by chart mining approaches, but we believe that these will be useful in overcoming many of the technical challenges discussed earlier. For tasks such as chart image classification and text processing, these computer vision methods have been relatively successful with little to no changes from their original domains. However, other tasks require domain adaptations as we have seen in some of the very recent works using end-to-end deep learning.

There has been very few comparisons between methods for chart mining. This is mostly due to the lack of public benchmarks. There have been competitions on related challenges such as figure extraction, multi-panel figure segmentation, figure classification, and figure text processing. For charts, the first effort was made at ICDAR 2019 through the Competition on HArvesting Raw Tables (CHART-info) [173], and their data and tools have been made publicly available.

Another major challenge in chart mining is the lack of large annotated datasets. So far, chart extraction and classification are the only tasks being tested with reasonably large datasets, probably due to the simplicity of the labels required for evaluation. Advanced data extraction tasks require complex labels which are hard to annotate manually. As such, they are still evaluated on considerably small datasets, which are often also private. We also noticed that many works use images from the web, but redistribution might be affected by copyrights.

The common alternative to large scale data annotation is synthetic data generation. However, these datasets are commonly generated using a single tool, and as such they often fail to capture the diversity and complexity found in real-world charts [173]. Many chart generation methods rely on unconstrained randomizations of style parameters of the chart rendering tool, but this is likely to overfit the design space for that tool in particular. The goal should be to create realistic charts using a diverse set of tools. Studies considering both large-scale synthetic and small-scale real chart datasets show similar trends where their methods perform well on the synthetic testing data, but perform significantly worse on real charts [67], [75], [76], [173].

In this review, we have only covered applications for chart mining that have been addressed in the existing literature. Many users would benefit from advances in each of these applications, but some of them have been rather under-explored by existing works. Also, like the main chart recognition problem, many of these applications suffer from the same

issues of over-simplification, lack of comparisons between existing methods and lack of large scale public datasets.

8.2 Open Challenges

There is a strong need to address the challenging problems in chart processing by moving away from hard assumptions and weak heuristics, and start moving towards more robust methods which can handle the varied characteristics of real chart data. For example, we can anticipate that vector graphics might become more commonplace in the future, but we still need more robust methods to handle raster graphics. Overcoming these challenges will have an immediate impact on a variety of applications in the document recognition community.

Chart mining methodologies continue to evolve, and we hope to see more methods taking advantage of progress in computer vision. There is a need to adopt fully trainable methodologies (e.g., Deep Neural Networks) that avoid hard heuristic assumptions such as we have seen in some of the most recent methods [75]. The advantage of these methods is the concurrent optimization of multiple portions of the chart recognition pipeline, especially with end-to-end networks. Also, processing of certain specialized diagrams in scientific disciplines requires the exploration of a combination of state-of-the-art methods for low-level chart element recognition with a mechanism to provide high-level domain knowledge and other input to the system for interpreting these diagrams. This should facilitate the rapid development of pipelines for recognition of highly specialized charts based on models pre-trained on more general charts.

The adoption of state-of-the-art computer vision techniques can be facilitated by the creation of large scale public datasets. Overall, these datasets should be more diverse and include charts from multiple sources and domains (news media, academic literature, technical reports, policy documents, white papers, patents, etc). The chart recognition community should take advantage of existing chart annotation tools [173] to create and release benchmarks in the future. These benchmarks might require the formal definition of new metrics which are capable of capturing the complexity of the task being evaluated and the finer improvements obtained with new methodologies for these tasks. Also, methods using synthetic data should combine charts from multiple chart generation tools, considering as well that it is not enough to simply use real data to generate synthetic charts, but the chart generation process itself should adopt user models that emulate the way that real people would create visualizations based on such data.

Finally, while some applications of chart mining are getting a reasonable amount of attention, there are more ambitious applications that have been under-explored by the community. As more accurate methods for chart data extraction emerge, we can anticipate that these applications will become more achievable.

8.3 Conclusion

In this work, we have not only reviewed the chart mining literature, but we have also analyzed its limitations and provided a few pointers for future work. We anticipate that chart literature will grow in the years to come, leading to more accurate chart mining methods that can power more creative applications. We expect this review to provide useful insights for the future developers of these methodologies.

Acknowledgments

This work was supported by the National Science Foundation under Grant No.1640867 (OAC/DMR). The authors would like to thank Fei Xu and Richard Zanibbi for their valuable comments.

References

[1] E. R. Tufte, *The Visual Display of Quantitative Information*. Cheshire, CT, USA: Graphics Press, 1983.
[2] J. Bertin, *Semiology of Graphics: Diagrams, Networks, Maps*. Madison, WI, USA: Univ. Wisconsin Press, 1983.
[3] W. S. Cleveland and W. S. Cleveland, *The Elements of Graphing Data*. Monterey, CA, USA: Wadsworth Advanced Books and Software, 1985.
[4] H. C. Purchase, "Twelve years of diagrams research," *J. Vis. Languages Comput.*, vol. 25, no. 2, pp. 57–75, 2014.
[5] C. Clark and S. Divvala, "Pdffigures 2.0: Mining figures from research papers," in *Proc. IEEE/ACM Joint Conf. Digital Libraries*, 2016, pp. 143–152.
[6] P. A. Praczyk, J. Nogueras-Iso, and S. Mele, "Automatic extraction of figures from scientific publications in high-energy physics," *Inf. Technol. Libraries*, vol. 32, no. 4, 2013, Art. no. 25.
[7] L. D. Lopez *et al.*, "A framework for biomedical figure segmentation towards image-based document retrieval," *BMC Syst. Biol.*, vol. 7, no. 4, 2013, Art. no. S8.
[8] D. Blostein, E. Lank, and R. Zanibbi, "Treatment of diagrams in document image analysis," in *Proc. Int. Conf. Theory Appl. Diagrams*, 2000, pp. 7–42.
[9] Y. Liu, X. Lu, Y. Qin, Z. Tang, and J. Xu, "Review of chart recognition in document images," in *Proc. Visu. Data Anal.*, International Society for Optics and Photonics, vol. 8654, pp. 384–391, 2013, doi: 10.1117/12.2008467.
[10] S. Ray Choudhury, P. Mitra, and C. L. Giles, "Automatic extraction of figures from scholarly documents," in *Proc. ACM Symp. Document Eng.*, 2015, pp. 47–50.
[11] P. Li, X. Jiang, and H. Shatkay, "Figure and caption extraction from biomedical documents," *Bioinformatics*, vol. 35, pp. 4381–4388, 2019.
[12] N. Siegel, N. Lourie, R. Power, and W. Ammar, "Extracting scientific figures with distantly supervised neural networks," in *Proc. 18th ACM/IEEE Joint Conf. Digital Libraries*, 2018, pp. 223–232.
[13] Semantic Scholar - An academic search engine for scientific articles, Accesssed: Mar. 10, 2020. [Online]. Available: https://www.semanticscholar.org
[14] J. P. Svendsen, "Chart detection and recognition in graphics intensive business documents," Ph.D. dissertation, Dept. Elect. Comput. Eng., University of Victoria, Victoria, BC, 2015.
[15] W. Huang and C.-L. Tan, "Locating charts from scanned document pages," in *Proc. 9th Int. Conf. Document Anal. Recognit.*, 2007, pp. 307–311.
[16] V. Karthikeyani and S. Nagarajan, "Scientific chart image property identification by connected component labeling in PDF files," in *Proc. 3rd Int. Conf. Electron. Comput. Technol.*, 2011, pp. 209–212.
[17] P. De, "Automatic data extraction from 2D and 3D pie chart images," in *Proc. IEEE 8th Int. Advance Comput. Conf.*, 2018, pp. 20–25.
[18] I. Kavasidis *et al.*,"A saliency-based convolutional neural network for table and chart detection in digitized documents," in *Proc. Int. Conf. Image Anal. Process.*, 2019, pp. 292–302.
[19] M. Shao and R. P. Futrelle, "Recognition and classification of figures in PDF documents," *Graphics Recognit. Ten Years Review Future Perspectives*, pp. 231–242, 2006.
[20] Z. Chen, M. Cafarella, and E. Adar, "Diagramflyer: A search engine for data-driven diagrams," in *Proc. 24th Int. Conf. World Wide Web*, 2015, pp. 183–186.
[21] Apache PDFBox | A Java PDF Library, Accessed: Mar. 10, 2020. [Online]. Available: https://pdfbox.apache.org/
[22] S. R. Choudhury *et al.*, "Figure metadata extraction from digital documents," in *Proc. 12th Int. Conf. Document Anal. Recognit.*, 2013, pp. 135–139.

[23] N. Siegel, Z. Horvitz, R. Levin, S. Divvala, and A. Farhadi, "Figureseer: Parsing result-figures in research papers," in *Proc. Eur. Conf. Comput. Vis.*, 2016, pp. 664–680.

[24] S. R. Choudhury, S. Wang, and C. L. Giles, "Scalable algorithms for scholarly figure mining and semantics," in *Proc. Int. Workshop Semantic Big Data*, 2016, Art. no. 1.

[25] R. P. Futrelle, M. Shao, C. Cieslik, and A. E. Grimes, "Extraction, layout analysis and classification of diagrams in PDF documents," in *Proc. 7th Int. Conf. Document Anal. Recognit.*, 2003, pp. 1007–1013.

[26] S. Ray Choudhury and C. L. Giles, "An architecture for information extraction from figures in digital libraries," in *Proc. 24th Int. Conf. World Wide Web*, 2015, pp. 667–672.

[27] P.-S. Lee and B. Howe, "Dismantling composite visualizations in the scientific literature," in *Proc. Int. Conf. Pattern Recognit. Appl. Methods*, 2015, pp. 79–91.

[28] M. Taschwer and O. Marques, "Automatic separation of compound figures in scientific articles," *Multimedia Tools Appl.*, vol. 77, no. 1, pp. 519–548, 2018.

[29] E. Apostolova, D. You, Z. Xue, S. Antani, D. Demner-Fushman, and G. R. Thoma, "Image retrieval from scientific publications: Text and image content processing to separate multipanel figures," *J. Assoc. Inf. Sci. Technol.*, vol. 64, no. 5, pp. 893–908, 2013.

[30] A. G. S. de Herrera, H. Müller, and S. Bromuri, "Overview of the imageclef 2015 medical classification task," in *Proc. Cross Lang. Eval. Forum*, 2015, vol. 1391. [Online]. Available: http://ceur-ws.org/Vol-1391/

[31] P.-S. Lee, J. D. West, and B. Howe, "Viziometrics: Analyzing visual information in the scientific literature," *IEEE Trans. Big Data*, vol. 4, no. 1, pp. 117–129, Mar. 2018.

[32] S. K. Antani, D. Demner-Fushman, J. Li, B. V. Srinivasan, and G. R. Thoma, "Exploring use of images in clinical articles for decision support in evidence-based medicine," in *Proc. Document Recognit. Retrieval XV*, 2008, Art. no. 68150Q.

[33] K. Santosh, Z. Xue, S. K. Antani, and G. R. Thoma, "NLM at imageCLEF2015: Biomedical multipanel figure separation," in *Proc. Cross Lang. Eval. Forum*, 2015, vol. 1391. [Online]. Available: http://ceur-ws.org/Vol-1391/

[34] J. Zou, S. Antani, and G. Thoma, "Localizing and recognizing labels for multi-panel figures in biomedical journals," in *Proc. Int. Conf. Document Anal. Recognit.*, 2017, pp. 753–758.

[35] J. Zou, G. Thoma, and S. Antani, "Unified deep neural network for segmentation and labeling of multipanel biomedical figures," *J. Assoc. Inf. Sci. Technol.*, 2020, doi: 10.1002/asi.24334.

[36] S. Chakrabarti, J. P. Dudzic, X. Li, E. J. Collas, J.-P. Boquete, and B. Lemaitre, "Remote control of intestinal stem cell activity by haemocytes in drosophila," *PLoS Genetics*, vol. 12, no. 5, 2016, Art. no. e1006089.

[37] M. J. Koudijs *et al.*, "The zebrafish mutants dre, uki, and lep encode negative regulators of the hedgehog signaling pathway," *PLoS Genetics*, vol. 1, no. 2, 2005, art. no. e19.

[38] B. Cheng, S. Antani, R. J. Stanley, and G. R. Thoma, "Automatic segmentation of subfigure image panels for multimodal biomedical document retrieval," *Document Recognit. Retrieval XVIII*, vol. 7874, pp. 294–304, 2011.

[39] X. Wang, X. Jiang, A. Kolagunda, H. Shatkay, and C. Kambhamettu, "CIS UDEL working notes on imageCLEF 2015: Compound figure detection task," in *Proc. Cross Lang. Eval. Forum*, 2015, vol. 1391. [Online]. Available: http://ceur-ws.org/Vol-1391/

[40] S. Tsutsui and D. J. Crandall, "A data driven approach for compound figure separation using convolutional neural networks," in *Proc. 14th IAPR Int. Conf. Document Anal. Recognit.*, 2017, pp. 533–540.

[41] X. Shi, Y. Wu, H. Cao, G. Burns, and P. Natarajan, "Layout-aware subfigure decomposition for complex figures in the biomedical literature," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, 2019, pp. 1343–1347.

[42] B. Shneiderman, "The eyes have it: A task by data type taxonomy for information visualizations," in *Proc. IEEE Symp. Vis. Languages*, 1996, pp. 336–343.

[43] H. Müller, J. Kalpathy-Cramer, D. Demner-Fushman, and S. Antani, "Creating a classification of image types in the medical literature for visual categorization," in *Proc SPIE Medical Imaging*, 2012, vol. 8319, pp. 83 190P–83 190P.

[44] M. Khan and S. S. Khan, "Data and information visualization methods, and interactive mechanisms: A survey," *Int. J. Comput. Appl.*, vol. 34, no. 1, pp. 1–14, 2011.

[45] A. Lagopoulos, N. Kapraras, V. Amanatiadis, A. Fachantidis, and G. Tsoumakas, "Classifying biomedical figures by modality via multi-label learning," *IEEE J. Biomed. Health Inform.*, vol. 23, no. 6, pp. 2230–2237, Nov. 2019.

[46] D. Chester and S. Elzer, "Getting computers to see information graphics so users do not have to," *Foundations Intell. Syst.*, in: M. S. Hacid, N. V. Murray, Z. W. Raś, and S. Tsumoto, Eds., Lecture Notes in Computer Science, pp. 660–668, 2005. [Online]. Available: https://doi.org/10.1007/11425274_68

[47] A. Mishchenko and N. Vassilieva, "Chart image understanding and numerical data extraction," in *Proc. 6th Int. Conf. Digital Inf. Manage.*, 2011, pp. 115–120.

[48] A. Mahmood, I. S. Bajwa, and K. Qazi, "An automated approach for interpretation of statistical graphics," in *Proc. 6th Int. Conf. Intell. Human-Mach. Syst. Cybern.*, 2014, pp. 376–379.

[49] Y. Zhou and C. L. Tan, "Learning-based scientific chart recognition," in *Pro. 4th IAPR Int. Workshop Graph. Recognit.*, 2001, pp. 482–492.

[50] W. Huang and C. L. Tan, "A system for understanding imaged infographics and its applications," in *Proc. ACM Symp. Document Eng.*, 2007, pp. 9–18.

[51] S. Shukla and A. Samal, "Recognition and quality assessment of data charts in mixed-mode documents," *Int. J. Document Anal. Recognit.*, vol. 11, no. 3, pp. 111–126, 2008.

[52] V. S. N. Prasad, B. Siddiquie, J. Golbeck, and L. S. Davis, "Classifying computer generated charts," in *Proc. Int. Workshop Content-Based Multimedia Indexing*, 2007, pp. 85–92.

[53] S. Elzer, E. Schwartz, S. Carberry, D. Chester, S. Demir, and P. Wu, "Accessible bar charts for visually impaired users," in *Proc. 4th Annu. IASTED Int. Conf. Telehealth Assistive Technol.*, 2008, pp. 55–60.

[54] B. Cheng, R. J. Stanley, S. Antani, and G. R. Thoma, "Graphical figure classification using data fusion for integrating text and image features," in *Proc. 12th Int. Conf. Document Anal. Recognit.*, 2013, pp. 693–697.

[55] V. Karthikeyani and S. Nagarajan, "Machine learning classification algorithms to recognize chart types in portable document format (PDF) files," *Int. J. Comput. Appl.*, vol. 39, no. 2, pp. 1–5, 2012.

[56] L. Battle, P. Duan, Z. Miranda, D. Mukusheva, R. Chang, and M. Stonebraker, "Beagle: Automated extraction and interpretation of visualizations from the web," in *Proc. CHI Conf. Human Factors Comput. Syst.*, 2018, Art. no. 594.

[57] D. Kim, B. P. Ramesh, and H. Yu, "Automatic figure classification in bioscience literature," *J. Biomed. Inform.*, vol. 44, no. 5, pp. 848–858, 2011.

[58] R. R. Nair, N. Sankaran, I. Nwogu, and V. Govindaraju, "Automated analysis of line plots in documents," in *Proc. 13th Int. Conf. Document Anal. Recognit.*, 2015, pp. 796–800.

[59] 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," *Int. J. Document Anal. Recognit.*, vol. 12, no. 2, pp. 65–81, 2009.

[60] O. Pelka and C. M. Friedrich, "FHDO biomedical computer science group at medical classification task of imageclef 2015," in *Proc. Cross Lang. Eval. Forum*, 2015, vol. 1391. [Online]. Available: http://ceur-ws.org/Vol-1391/

[61] T. Giannakopoulos, I. Foufoulas, E. Stamatogiannakis, H. Dimitropoulos, N. Manola, and Y. Ioannidis, "Visual-based classification of figures from scientific literature," in *Proc. 24th Int. Conf. World Wide Web*, 2015, pp. 1059–1060.

[62] I. Almakky, V. Palade, Y.-L. Hedley, and J. Yang, "A stacked deep autoencoder model for biomedical figure classification," in *Proc. IEEE 15th Int. Symp. Biomed. Imaging*, 2018, pp. 1134–1138.

[63] V. Andrearczyk and H. Müller, "Deep multimodal classification of image types in biomedical journal figures," in *Proc. Int. Conf. Cross-Lang. Eval. Forum Eur. Languages*, 2018, pp. 3–14.

[64] J. Zhang, Y. Xie, Q. Wu, and Y. Xia, "Medical image classification using synergic deep learning," *Med. Image Anal.*, vol. 54, pp. 10–19, 2019.

[65] K. Jobin, A. Mondal, and C. Jawahar, "Docfigure: A dataset for scientific document figure classification," in *Proc. Int. Conf. Document Anal. Recognit. Workshops*, 2019, pp. 74–79.

[66] S. L. Lee and M. R. Zare, "Biomedical compound figure detection using deep learning and fusion techniques," *IET Image Process.*, vol. 12, no. 6, pp. 1031–1037, 2018.

[67] P. Chagas *et al.*, "Evaluation of convolutional neural network architectures for chart image classification," in *Proc. Int. Joint Conf. Neural Netw.*, 2018, pp. 1–8.
[68] M. Savva, N. Kong, A. Chhajta, L. Fei-Fei, M. Agrawala, and J. Heer, "Revision: Automated classification, analysis and redesign of chart images," in *Proc. 24th Annu. ACM Symp. User Interface Softw. Technol.*, 2011, pp. 393–402.
[69] S. Kanjanawattana and M. Kimura, "Annsvm: A novel method for graph-type classification by utilization of fourier transformation, wavelet transformation, and hough transformation," *BRAIN. Broad Res. Artif. Intell. Neurosci.*, vol. 8, no. 2, pp. 5–25, 2017.
[70] J. Poco and J. Heer, "Reverse-engineering visualizations: Recovering visual encodings from chart images," in *Computer Graphics Forum*, vol. 36, Hoboken, NJ, USA: Wiley, 2017, pp. 353–363.
[71] D. Jung *et al.*, "Chartsense: Interactive data extraction from chart images," in *Proc. CHI Conf. Human Factors Comput. Syst.*, 2017, pp. 6706–6717.
[72] J. Gao, Y. Zhou, and K. E. Barner, "View: Visual information extraction widget for improving chart images accessibility," in *Proc. 19th IEEE Int. Conf. Image Process.*, 2012, pp. 2865–2868.
[73] B. Tang *et al.*, "Deepchart: Combining deep convolutional networks and deep belief networks in chart classification," *Signal Process.*, vol. 124, pp. 156–161, 2016.
[74] J. Choi, S. Jung, D. G. Park, J. Choo, and N. Elmqvist, "Visualizing for the non-visual: Enabling the visually impaired to use visualization," in *Computer Graphics Forum*, vol. 38, no. 3. Hoboken, NJ, USA: Wiley, 2019, pp. 249–260.
[75] X. Liu, D. Klabjan, and P. NBless, "Data extraction from charts via single deep neural network," 2019, *arXiv: 1906.11906*.
[76] A. Balaji, T. Ramanathan, and V. Sonathi, "Chart-text: A fully automated chart image descriptor," 2018, *arXiv: 1812.10636*.
[77] W. Dai, M. Wang, Z. Niu, and J. Zhang, "Chart decoder: Generating textual and numeric information from chart images automatically," *J. Vis. Languages Comput.*, vol. 48, pp. 101–109, 2018.
[78] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proc. Advances Neural Inf. Processi. Syst.*, 2012, pp. 1097–1105.
[79] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014, *arXiv:1409.1556*.
[80] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 770–778.
[81] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 4700–4708.
[82] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2009, pp. 248–255.
[83] Blue Leaf Software - Dagra, Accessed: Mar. 10, 2020. [Online]. Available: https://blueleafsoftware.com/
[84] Plot Digitizer, Accessed: March 10, 2020. [Online]. Available: http://plotdigitizer.sourceforge.net
[85] Engauge Digitizer, Accessed: March 10, 2020. [Online]. Available: http://markummitchell.github.io/engauge-digitizer/
[86] DataThief III, Accessed: March 10, 2020. [Online]. Available: https://datathief.org/
[87] W. Huang, C. L. Tan, and W. K. Leow, "Elliptic arc vectorization for 3D pie chart recognition," in *Proc. Int. Conf. Image Process.*, 2004, pp. 2889–2892.
[88] N. Vassilieva and Y. Fomina, "Text detection in chart images," *Pattern Recognit. Image Anal.*, vol. 23, no. 1, pp. 139–144, 2013.
[89] S. Kataria, W. Browuer, P. Mitra, and C. L. Giles, "Automatic extraction of data points and text blocks from 2-dimensional plots in digital documents," in *Proc. 23rd Nat. Conf. Artif. intell.*, 2008, pp. 1169–1174.
[90] S. R. Choudhury, S. Wang, and C. L. Giles, "Automated data extraction from scholarly line graphs," in *Proc. Int. Workshop Graph. Recognit.*, 2015.
[91] R. A. Al-Zaidy and C. L. Giles, "A machine learning approach for semantic structuring of scientific charts in scholarly documents," in *Proc. 31st AAAI Conf. Artif. Intell.*, 2017, pp. 4644–4649.
[92] F. Böschen, T. Beck, and A. Scherp, "Survey and empirical comparison of different approaches for text extraction from scholarly figures," *Multimedia Tools Appl.*, vol. 77, no. 22, pp. 29 475–29 505, 2018.
[93] Y. P. Zhou and C. L. Tan, "Hough-based model for recognizing bar charts in document images," in *Proc. Document Recognit. Retrieval*, 2001, pp. 333–340.
[94] D. Morris, P. Tang, and R. Ewerth, "A neural approach for text extraction from scholarly figures," in *Proc. Int. Conf. Document Anal. Recognit*, 2019, pp. 1438–1443.
[95] M. Jessen, F. Böschen, and A. Scherp, "Text localization in scientific figures using fully convolutional neural networks on limited training data," in *Proc. ACM Symp. Document Eng.*, 2019, pp. 1–10.
[96] M. Cliche, D. Rosenberg, D. Madeka, and C. Yee, "Scatteract: Automated extraction of data from scatter plots," in *Proc. Joint Eur. Conf. Mach. Learn. Knowl. Discovery Databases*, 2017, pp. 135–150.
[97] ICDAR 2019 RobustReading Competition, Accessed: Mar. 10, 2020. [Online]. Available: http://rrc.cvc.uab.es/
[98] Image Processing with the Computer Vision API | Microsoft Azure, Accessed: Mar. 10, 2020. [Online]. Available: http://azure.microsoft.com/en-us/services/cognitive-services/computer-vision/
[99] Tesseract OCR – opensource.google.com, Accessed: Mar. 10, 2020. [Online]. Available: https://opensource.google.com/projects/tesseract
[100] PDF Software with Text Recognition - ABBYY FineReader 14, Accessed: Aug. 20, 2019. [Online]. Available: https://www.abbyy.com/en-us/finereader/
[101] GitHub - tmbdev/ocropy: Python-based tools for document analysis and OCR, Accessed: Mar. 10, 2020. [Online]. Available: https://github.com/tmbdev/ocropy
[102] B. Shi, X. Bai, and C. Yao, "An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 11, pp. 2298–2304, Nov. 2016.
[103] S. Kanjanawattana and K. Masaomi, "Ontologies-based optical character recognition-error correction method for bar graphs," in *Proc. 10th Int. Conf. Advances Semantic Process.*, 2016, pp. 1–8.
[104] T. Chandrashekaraiah *et al.*, "Spectroscopic study of plasma polymerized ac: H films deposited by a dielectric barrier discharge," *Materials*, vol. 9, no. 7, 2016, Art. no. 594.
[105] A. Lugo, C. La Vecchia, S. Boccia, B. Murisic, and S. Gallus, "Patterns of smoking prevalence among the elderly in europe," *Int. J. Environ. Res. Public Health*, vol. 10, no. 9, pp. 4418–4431, 2013.
[106] J. Ponjavic, P. L. Oliver, G. Lunter, and C. P. Ponting, "Genomic and transcriptional co-localization of protein-coding and long non-coding rna pairs in the developing brain," *PLoS Genetics*, vol. 5, no. 8, 2009, Art. no. e1000617.
[107] Y. Yan *et al.*, "Shift in HIV/AIDS epidemic in southeastern china: A longitudinal study from 1987 to 2015," *Int. J. Environ. Res. Public Health*, vol. 13, no. 8, 2016, Art. no. 794.
[108] T. Nikitin and L. Khriachtchev, "Optical and structural properties of Si nanocrystals in SiO2 films," *Nanomaterials*, vol. 5, no. 2, pp. 614–655, 2015.
[109] S. Mahadeva, J. Fan, A. Biswas, K. Sreelatha, L. Belova, and K. Rao, "Magnetism of amorphous and nano-crystallized dc-sputter-deposited MgO thin films," *Nanomaterials*, vol. 3, no. 3, pp. 486–497, 2013.
[110] M. K. I. Molla, K. H. Talukder, and M. A. Hossain, "Line chart recognition and data extraction technique," in *Proc. Int. Conf. Intell. Data Eng. Automated Learn.*, 2003, pp. 865–870.
[111] W. Browuer, S. Kataria, S. Das, P. Mitra, and C. L. Giles, "Segregating and extracting overlapping data points in two-dimensional plots," in *Proc. 8th ACM/IEEE-CS Joint Conf. Digital Libraries*, 2008, pp. 276–279.
[112] A. S. Diwakar, D. Nayak, and P. Talwai, "Rescue and digitization of climate data by extraction from autographic weather charts," in *Proc. Int. Assoc. Comput. Sci. Inf. Technol.-Spring Conf.*, 2009, pp. 186–189.
[113] A. Kaur, D. Dani, and N. Mishra, "Improving web accessibility of graphs for visually impaired," *Int. J. Comput. Sci. Inf. Technol.*, vol. 2, no. 5, pp. 1979–1981, 2011.
[114] J. S. Kallimani, K. Srinivasa, and R. B. Eswara, "Extraction and interpretation of charts in technical documents," in *Proc. Int. Conf. Advances Comput. Commun. Inform.*, 2013, pp. 382–387.
[115] S. R. Choudhury, S. Wang, and C. L. Giles, "Curve separation for line graphs in scholarly documents," in *Proc. IEEE/ACM Joint Conf. Digital Libraries*, 2016, pp. 277–278.
[116] V. K. Reddy and C. Kaushik, "Image processing based data extraction from graphical representation," in *Proc. IEEE Int. Conf. Comput. Graph. Vis. Inf. Secur.*, 2015, pp. 190–194.

[117] J. Harper and M. Agrawala, "Converting basic D3 charts into reusable style templates," *IEEE Trans. Vis. Comput. Graphics*, vol. 24, no. 3, pp. 1274–1286, Mar. 2018.

[118] W. Huang, C. L. Tan, and W. K. Leow, "Associating text and graphics for scientific chart understanding," in *Proc. 8th Int. Conf. Document Anal. Recognit.*, 2005, pp. 580–584.

[119] S. Kanjanawattana and M. Kimura, "Extraction of graph information based on image contents and the use of ontology," in *Proc. Int. Conf. Internet Technol. Soc.*, Technol. Educ., 2016, pp. 19–26.

[120] N. Methani, P. Ganguly, M. M. Khapra, and P. Kumar, "Plotqa: Reasoning over scientific plots," in *Proc. Winter Conf. Appl. Comput. Vis.*, 2020, pp. 1527–1536.

[121] R.-Q. Wu, X.-R. Cheng, and C.-J. Yang, "Extracting contour lines from topographic maps based on cartography and graphics knowledge," *J. Comput. Sci. Technol.*, vol. 9, pp. 58–64, 2009.

[122] A. Mayhua, E. Gomez-Nieto, J. Heer, and J. Poco, "Extracting visual encodings from map chart images with color-encoded scalar values," in *Proc. 31st SIBGRAPI Conf. Graph. Patterns Images*, 2018, pp. 142–149.

[123] K. Mao, X. Chen, K. Zhu, D. Hu, and Y. Li, "A method to extract essential information from meteorological facsimile charts," *Int. J. Pattern Recognit. Artif. Intell.*, vol. 33, no. 01, 2019, Art. no. 1954001.

[124] A. Marchewka and R. Pasela, "Extraction of data from limnigraf chart images," in *Image Processing and Communications Challenges 5*. Berlin, Germany: Springer, 2014, pp. 263–269.

[125] B. U. Kota *et al.*, "Automated extraction of data from binary phase diagrams for discovery of metallic glasses," in *Proc. Int. Workshop Graph. Recognit.*, 2017, pp. 3–16.

[126] S. Carberry, S. Elzer, and S. Demir, "Information graphics: an untapped resource for digital libraries," in *Proc. 29th Annu. Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2006, pp. 581–588.

[127] R. Burns, S. Carberry, and S. Elzer Schwartz, "An automated approach for the recognition of intended messages in grouped bar charts," *Comput. Intell.*, vol. 35, no. 4, pp. 955–1002, 2019.

[128] C. F. Greenbacker, P. Wu, S. Carberry, K. F. McCoy, and S. Elzer, "Abstractive summarization of line graphs from popular media," in *Proc. Workshop Autom. Summarization Different Genres Media Languages*, 2011, pp. 41–48.

[129] R. Burns, E. Balawejder, W. Domanowska, S. E. Schwartz, and S. Carberry, "Exploring the types of messages that pie charts convey in popular media," in *Proc. Int. Conf. Theory Appl. Diagrams*, 2016, pp. 265–271.

[130] S. Elzer, S. Carberry, and S. Demir, "Communicative signals as the key to automated understanding of simple bar charts," in *Diagrams*. Berlin, Germany: Springer, 2006, pp. 25–39.

[131] E. Charniak and R. P. Goldman, "A Bayesian model of plan recognition," *Artif. Intell.*, vol. 64, no. 1, pp. 53–79, 1993.

[132] R. Burns, S. Carberry, and S. Elzer, "Modeling relative task effort for grouped bar charts," in *Proc. Annu. Meeting Cogni. Sci. Soc.*, 2009.

[133] S. Demir, S. E. Schwartz, R. Burns, and S. Carberry, "What is being measured in an information graphic?" in *Proc. Int. Conf. Intell. Text Process. Comput. Linguistics*, 2013, pp. 501–512.

[134] E. Kim and K. F. McCoy, "Multimodal deep learning using images and text for information graphic classification," in *Proc. 20th Int. ACM SIGACCESS Conf. Comput. Accessibility*, 2018, pp. 143–148.

[135] H. Mei, Y. Ma, Y. Wei, and W. Chen, "The design space of construction tools for information visualization: A survey," *J. Vis. Languages Comput.*, vol. 44, pp. 120–132, 2017.

[136] N. Kong and M. Agrawala, "Graphical overlays: Using layered elements to aid chart reading," *IEEE Trans. Vis. Comput. Graphics*, vol. 18, no. 12, pp. 2631–2638, Dec. 2012.

[137] G. G. Méndez, M. A. Nacenta, and S. Vandenheste, "iVoLVER: Interactive visual language for visualization extraction and reconstruction," in *Proc. CHI Conf. Human Factors Comput. Syst.*, 2016, pp. 4073–4085.

[138] R. Burns, S. E. Schwartz, and S. Carberry, "Towards adapting information graphics to individual users to support recognizing intended messages," in *Proc. UMAP Workshops*, 2013, vol. 997. [Online]. Available: http://ceur-ws.org/Vol-997/

[139] Z. Chen, Y. Wang, Q. Wang, Y. Wang, and H. Qu, "Towards automated infographic design: Deep learning-based auto-extraction of extensible timeline," *IEEE Trans. Vis. Comput. Graphics*, vol. 26, no. 1, pp. 917–926, Jan. 2020.

[140] A. Srinivasan, S. M. Drucker, A. Endert, and J. Stasko, "Augmenting visualizations with interactive data facts to facilitate interpretation and communication," *IEEE Trans. Vis. Comput. Graphics*, vol. 25, no. 1, pp. 672–681, Jan. 2018.

[141] N. Kong, M. A. Hearst, and M. Agrawala, "Extracting references between text and charts via crowdsourcing," in *Proc. SIGCHI Conf. Human Factors Comput. Syst.*, 2014, pp. 31–40.

[142] S. Demir, S. Carberry, and K. F. McCoy, "Summarizing information graphics textually," *Comput. Linguistics*, vol. 38, no. 3, pp. 527–574, 2012.

[143] L. Ferres, P. Verkhogliad, G. Lindgaard, L. Boucher, A. Chretien, and M. Lachance, "Improving accessibility to statistical graphs: The igraph-lite system," in *Proc. 9th Int. ACM SIGACCESS Conf. Comput. Accessibility*, 2007, pp. 67–74.

[144] S. Bhatia and P. Mitra, "Summarizing figures, tables, and algorithms in scientific publications to augment search results," *ACM Trans. Inf. Syst.*, vol. 30, no. 1, 2012, Art. no. 3.

[145] S. Demir, S. Carberry, and S. Elzer, "Effectively realizing the inferred message of an information graphic," in *Proc. Int. Conf. Recent Advances Natural Lang. Process.*, 2007, pp. 150–156.

[146] S. Demir, S. Carberry, and K. F. McCoy, "Generating textual summaries of bar charts," in *Proc. 5th Int. Natural Lang. Gener. Conf.*, 2008, pp. 7–15.

[147] P. Moraes, G. Sina, K. McCoy, and S. Carberry, "Evaluating the accessibility of line graphs through textual summaries for visually impaired users," in *Proc. 16th Int. ACM SIGACCESS Conf. Comput. Accessibility*, 2014, pp. 83–90.

[148] C. F. Greenbacker *et al.*, "Improving the accessibility of line graphs in multimodal documents," in *Proc. 2nd Workshop Speech Lang. Process. Assistive Technol.*, 2011, pp. 52–62.

[149] D. L. Mansur, M. M. Blattner, and K. I. Joy, "Sound graphs: A numerical data analysis method for the blind," *J. Med. Syst.*, vol. 9, no. 3, pp. 163–174, 1985.

[150] D. Ahmetovic, C. Bernareggi, J. Guerreiro, S. Mascetti, and A. Capietto, "Audiofunctions. web: Multimodal exploration of mathematical function graphs," in *Proc. 16th Web For All Personalization - Personalizing Web*, to be published, doi: 10.1145/3315002.3317560.

[151] S. Demir, D. Oliver, E. Schwartz, S. Elzer, S. Carberry, and K. F. McCoy, "Interactive sight into information graphics," in *Proc. Int. Cross Disciplinary Conf. Web Accessibility*, 2010, Art. no. 16.

[152] L. Ferres, G. Lindgaard, L. Sumegi, and B. Tsuji, "Evaluating a tool for improving accessibility to charts and graphs," *ACM Trans. Comput.-Human Interaction*, vol. 20, no. 5, 2013, Art. no. 28.

[153] A. Sharif and B. Forouraghi, "evographs—a jquery plugin to create web accessible graphs," in *Proc. 15th IEEE Annu. Consum. Commun. Netw. Conf.*, 2018, pp. 1–4.

[154] A. J. R. Godfrey, P. Murrell, and V. Sorge, "An accessible interaction model for data visualisation in statistics," in *Proc. Int. Conf. Comput. Helping People Special Needs*, 2018, pp. 590–597.

[155] R. A. Martínez, M. R. Turró, and T. G. i Saltiveri, "La accesibilidad de los gráficos estadísticos para personas con baja visión y visión cromática deficiente," *Revista de la Asociación Interacción Persona Ordenador*, vol. 1, no. 1, pp. 59–75, 2020.

[156] J. Charbonnier, L. Sohmen, J. Rothman, B. Rohden, and C. Wartena, "Noa: A search engine for reusable scientific images beyond the life sciences," in *Proc. Eur. Conf. Inf. Retrieval*, 2018, pp. 797–800.

[157] Z. Li, S. Carberry, H. Fang, K. F. McCoy, K. Peterson, and M. Stagitis, "A novel methodology for retrieving infographics utilizing structure and message content," *Data Knowl. Eng.*, vol. 100, pp. 191–210, 2015.

[158] B. Saleh, M. Dontcheva, A. Hertzmann, and Z. Liu, "Learning style similarity for searching infographics," in *Proc. 41st Graph. Interface Conf.*, 2015, pp. 59–64.

[159] E. Hoque and M. Agrawala, "Searching the visual style and structure of D3 visualizations," *IEEE Trans. Vis. Comput. Graphics*, vol. 26, no. 1, pp. 1236–1245, Jan. 2020.

[160] Y. Ma, A. K. Tung, W. Wang, X. Gao, Z. Pan, and W. Chen, "Scatternet: A deep subjective similarity model for visual analysis of scatterplots," *IEEE Trans. Vis. Comput. Graphics*, vol. 26, no. 3, pp. 1562–1576, Mar. 2020.

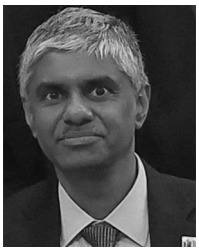
[161] Z. Li, M. Stagitis, S. Carberry, and K. F. McCoy, "Towards retrieving relevant information graphics," in *Proc. 36th Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2013, pp. 789–792.

[162] D. K. Sanyal, S. Chattopadhyay, and R. Chatterjee, "Figure retrieval from biomedical literature: An overview of techniques, tools, and challenges," in *Machine Learning in Bio-Signal Analysis and Diagnostic Imaging*. Amsterdam, The Netherlands: Elsevier, 2019, pp. 247–272.

[163] Q. Wu, D. Teney, P. Wang, C. Shen, A. Dick, and A. van den Hengel, "Visual question answering: A survey of methods and datasets," *Comput. Vis. Image Understanding*, vol. 163, pp. 21–40, 2017.

[164] K. Kafle, B. Price, S. Cohen, and C. Kanan, "Dvqa: Understanding data visualizations via question answering," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 5648–5656.
[165] R. Chaudhry, S. Shekhar, U. Gupta, P. Maneriker, P. Bansal, and A. Joshi, "LEAF-QA: Locate, encode & attend for figure question answering," in *Proc. IEEE Winter Conf. Appl. Comput. Vis.*, 2020, pp. 3512–3521.
[166] K. Kafle, R. Shrestha, S. Cohen, B. Price, and C. Kanan, "Answering questions about data visualizations using efficient bimodal fusion," in *Proc. IEEE Winter Conf. Appl. Comput. Vis.*, 2020, pp. 1498–1507.
[167] M. Sharma, S. Gupta, A. Chowdhury, and L. Vig, "Chartnet: Visual reasoning over statistical charts using mac-networks," in *Proc. Int. Joint Conf. Neural Netw.*, 2019, pp. 1–7.
[168] D. H. Kim, E. Hoque, and M. Agrawala, "Answering questions about charts and generating visual explanations," *ACM Human Factors Comput. Syst.*, 2020, pp. 1–13.
[169] D. B. Wijesinghe, J. Kadupitiya, S. Ranathunga, and G. Dias, "Automatic assessment of student answers consisting of venn and euler diagrams," in *Proc. IEEE 17th Int. Conf. Advanced Learn. Technol.*, 2017, pp. 243–247.
[170] M. M. Al-Dabbagh *et al.*, "Intelligent bar chart plagiarism detection in documents," *Sci. World J.*, vol. 2014, 2014, Art. no. 612787.
[171] S. Madan *et al.*, "Synthetically trained icon proposals for parsing and summarizing infographics," 2018, *arXiv: 1807.10441*.
[172] C. F. Greenbacker, S. Carberry, and K. F. McCoy, "A corpus of human-written summaries of line graphs," in *Proc. UCNLG+ Eval: Lang. Gener. Eva. Workshop*, 2011, pp. 23–27.
[173] K. Davila *et al.*, "ICDAR 2019 competition on harvesting raw tables from infographics (chart-infographics)," in *Proc. Int. Conf. Document Anal. Recognit.*, 2019, pp. 1594–1599.
[174] W. Huang, C. L. Tan, and J. Zhao, "Generating ground truthed dataset of chart images: Automatic or semi-automatic?" in *International Workshop on Graphics Recognition*. Berlin, Germany: Springer, 2007, pp. 266–277.
[175] XML/SWF Charts, Mar. 10, 2020. [Online]. Available: https://www.maani.us/xml_charts/
[176] A Visualization Grammar | Vega, Mar. 10, 2020. [Online]. Available: https://vega.github.io/vega/
[177] Matplotlib: Python plotting, Mar. 10, 2020. [Online]. Available: https://matplotlib.org/

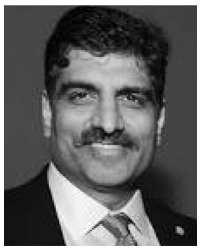
Kenny Davila (Member, IEEE) received the BE degree in computing systems engineering from Universidad Tecnológica Centroamericana (UNITEC), Tegugigalpa, Honduras, in 2009, and the MSc degree in computer science and PhD degree in computing and information sciences degrees from the Rochester Institute of Technology (RIT) Rochester, New York, in 2013 and 2017, respectively. In 2011, he was awarded a Foreign Fulbright Scholarship. In 2017, he joined the Center for Unified Biometrics and Sensors (CUBS), University at Buffalo as a postdoctoral associate. His current research interests include chart mining, lecture video analysis, handwriting recognition, and mathematical information retrieval. He has more than 15 academic publications, out of which three have received awards.



Srirangaraj Setlur (Senior Member, IEEE) is principal research scientist with the Center for Unified Biometrics and Sensors, and co-director of the NSF Center for Identification Technology Research, University at Buffalo, The State University of New York. His research interest includes machine learning applications in the fields of document analysis and recognition and biometrics. He has more than 60 publications on topics such as handwriting recognition, historical document processing, chart infographics processing, face and other biometric recognition technologies, and affective computing, and he has co-authored the first edited book on OCR of Indic Scripts. He has held leadership positions in several academic conferences in both Document Analysis and Biometrics.

David Doermann (Fellow, IEEE) is currently a professor of empire innovation and the director of the Artificial Intelligence Institute, University at Buffalo (UB). Prior to coming to UB, he was a program manager with the Information Innovation Office at the Defense Advanced Research Projects Agency (DARPA), where he developed, selected, and oversaw research and transition funding in the areas of computer vision, human language technologies and voice analytics. From 1993 to 2018, he was a member of the research faculty at the University of Maryland, College Park. In his role in the Institute for Advanced Computer Studies, he served as director of the Laboratory for Language and Media Processing, and as an adjunct member of the graduate faculty for the Department of Computer Science, Department of Electrical and Computer Engineering. He and his group of researchers focus on many innovative topics related to analysis and processing of document images and video including triage, visual indexing and retrieval, enhancement and recognition of both textual and structural components of visual media. David has over 250 publications in conferences and journals, is a fellow of the IAPR, has numerous awards including an honorary doctorate from the University of Oulu, Finland and is a founding editor-in-chief of the International Journal on Document Analysis and Recognition.

Bhargava Urala Kota (Member, IEEE) received the BTech degree in electronics and communication engineering from the National Institute of Technology, Karnataka, India, in 2011, and the MS degree in computer science from University at Buffalo, Buffalo, NY. He is currently working toward the PhD degree in the Department of Computer Science, University at Buffalo, Buffalo, NY. His primary research interests include detecting, recognizing and retrieving text in images and videos. He has previously worked on machine recognition of handwritten Indic scripts.



Venu Govindaraju (Fellow, IEEE) is currently a SUNY distinguished professor of Computer Science and Engineering and the founding director of the Center for Unified Biometrics and Sensors at the University at Buffalo, The State University of New York. He holds four patents, has received several major professional society awards and coauthored more then 425 scientific papers. He is a fellow of the Association of Computing Machinery, the American Association for the Advancement of Science, the International Association of Pattern Recognition, and the International Society of Optics and Photonics. He has served as general chair and in other leadership positions of a number of conferences in the areas of document analysis and biometrics. He is a recipient of the IEEE Technical Achievement Award, and has served as the President of the IEEE Biometrics Council.