A SEMI-AUTOMATED ALGORITHM FOR DATA EXTRACTION FROM IMAGES OF BAR CHARTS AND SCATTER PLOTS USING TENSOR FIELDS

Komal Dadhich

Master of Science by Research Thesis

June 2021

International Institute of Information Technology, Bangalore

A SEMI-AUTOMATED ALGORITHM FOR DATA EXTRACTION FROM IMAGES OF BAR CHARTS AND SCATTER PLOTS USING TENSOR FIELDS

Submitted to International Institute of Information Technology,
Bangalore
in Partial Fulfillment of
the Requirements for the Award of
Master of Science by Research

by

Komal Dadhich
MS2018007

International Institute of Information Technology, Bangalore
June 2021

Dedicated to my family

Thesis Certificate

This is to certify that the thesis titled **A Semi-automated Algorithm for Data Extraction from Images of Bar Charts and Scatter Plots Using Tensor Fields** submitted to the International Institute of Information Technology, Bangalore, for the award of the degree of **Master of Science by Research** is a bona fide record of the research work done by **Komal Dadhich**, **MS2018007**, under my supervision. The contents of this thesis, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

Prof. Jaya Sreevalsan Nair

Bangalore,

The 4th of June, 2021.

A SEMI-AUTOMATED ALGORITHM FOR DATA EXTRACTION FROM IMAGES OF BAR CHARTS AND SCATTER PLOTS USING TENSOR FIELDS

Abstract

Statistical plots are some of the simplest yet effective visualizations as these plots have been part of different stages of school education. Currently, they are used extensively in academic curriculum, newspapers, digital platforms, and research articles in image format. The most used chart types in the curriculum are bar charts, scatter plots, line charts, and pie charts. Among them, bar charts are the most popular choice because of the availability of various representations such as grouped bar and stacked bar for plotting as per data requirement. The chart images in published articles may interest the user to explore more about the data as well as select a different visualization for ease of understanding. In such a scenario, the unavailability of source data used to visualize and create the chart image in focus comes out as a blocker in the analysis process. The publications or articles are mostly converted to speech for visually impaired readers. However, the existing systems are unable to provide or integrate similar information from the chart images embedded in the documents.

The previous systems and models for chart analysis, interpretation, and data extraction demonstrate the need for a generic system that can be used for various types of charts. As an initial step towards a computational model for data extraction from chart images, we introduce a semi-automated algorithm with components for chart type classifier, tensor field computation, and rule-based data extraction. Our algorithm has currently been completed for simple bar charts, histograms, and scatter plots. Our CNN-based chart classification model identifies the chart-type and sub-type that determine the

extraction steps for the given chart image. We propose implementing local geometric descriptors like structure tensor, tensor vote, and tensor voting field after anisotropic diffusion to exploit the geometry of graphical objects such as bars/columns or points. The clustering of critical points, filtered through the topological analysis of tensor fields, shows the corners in bars/column and center in point/dot in scatter plots. These corners of bar/column in the bar chart and center of scatter point in scatter plots encode the information in the source image and provide the pixel-based data value depicted by individual chart object.

As a next step, we extend our algorithm by introducing additional components such as a classifier to identify sub-type of a given bar chart image and chart annotation module for data extraction from multi-series/multi-class bar chart and scatter plots. Our data extraction component provides the data in pixel space for an input image and is integrated with text detection and localization models to extract the final data.

Here, our scope lies in providing a system that given an image of a bar chart or scatter plot, identifies the chart type and sub-type plotted in the image, and extracts the data represented in the given image. In this thesis, we discuss the two important steps of the algorithm for data extraction, namely, (i) chart classification and (ii) tensor field computation. We demonstrate how our algorithm is extensible to complex forms of a chart, e.g., from simple bar charts to grouped and stacked charts, from single-class scatter plots to multi-class scatter plots. In this thesis, we address the challenges associated with chart images, limitations, experiments with chart images covering different formatting specifications, and discuss the future work that can improve the process.

Acknowledgements

"My sincere gratitude to my advisor Prof. Jaya Sreevalsan Nair for her invaluable advice, continuous encouragement and support. It has been my pleasure to work under her guidance. I wish to express my sincere gratitude to the Machine Intelligence and Robotics Center at IIIT-Bangalore for generous funding support. I would also like to thank Ms. Reddy Rani Vangimalla for providing such a healthy working environment and sharing her invaluable experience. Discussing with her was always fun and stress relieving. I would also like to thank my lab mates Ms. Harshitha Ravindra and Ms. Siri Chandana Daggubati for helping me and working with me. Last but not least, I would like to thank my family for maintaining their belief and faith in me without which completing this journey would not be possible."

— Komal Dadhich

List of Publications

1.1 J. Sreevalsan-Nair, K. Dadhich, and S. C. Daggubati, "Tensor Fields for Data Extraction from Chart Images: Bar Charts and Scatter Plots," (to appear) in Topological Methods in Visualization: Theory, Software and Applications, Ingrid Hotz, Talha Bin Masood, Filip Sadlo, and Julien Tierny (Eds.). Springer-Verlag, 2020; preprint at arXiv (2020), October 2020.

1.2 K. Dadhich, S. C. Daggubati, and J. Sreevalsan-Nair, "BarChartAnalyzer: Digitizing Images of Bar Charts," in the Proceedings of the International Conference on Image Processing and Vision Engineering (IMPROVE 2021), pp 17–28, April 2021. https://doi.org/10.5220/0010408300170028

1.3 K. Dadhich, S. C. Daggubati, and J. Sreevalsan-Nair, "ScatterPlotAnalyzer: Digitizing Images of Charts Using Tensor-based Computational Model" (to appear) in the Proceedings of the International Conference on Computational Science (ICCS) 2021, June 2021.

Contents

List of Figures

List of Tables

List of Abbreviations

DBSCAN . . . . . Density-Based Spatial Clustering of Applications with Noise

CNN . . . . . Convolutional Neural Network

OCR . . . . . Optical Character Recognition

ST . . . . . Structure Tensor

TV . . . . . Tensor Vote

TVAD . . . . . Tensor Voting Field after Anisotropic Diffusion

2D . . . . . Two Dimensional

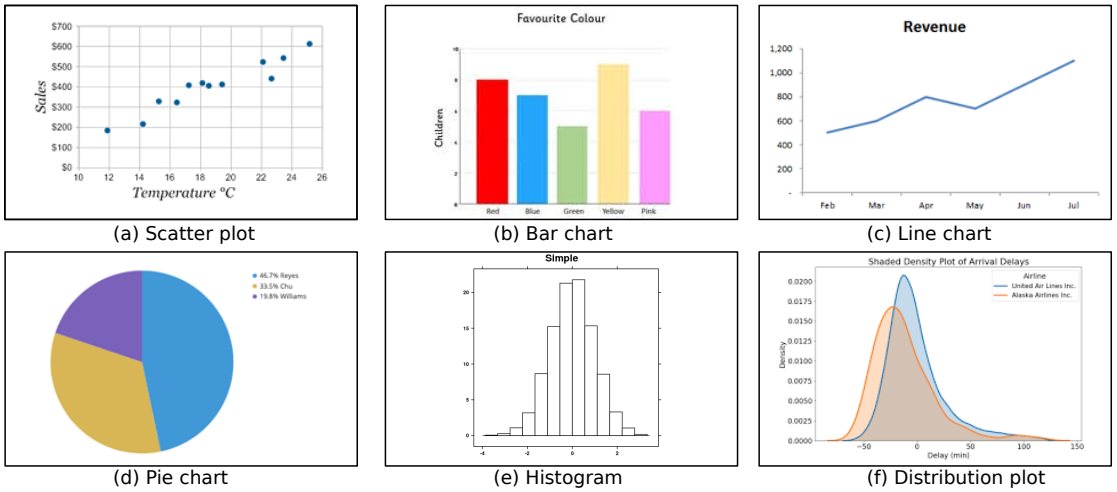
3D . . . . . Three Dimensional

CHAPTER 1

INTRODUCTION

"Use a picture. It's worth a thousand words." As Arthur Brisbane suggested, visualization simplifies complex data sets to highlight the important aspects—numerical patterns, trends, distributions, and other more abstract insights. Charts are one of the most comprehensible depictions of data typically found in documents, images, magazines, and articles, hence becoming the most widely used visualization method to illustrate data and associated trends. Charts, as part of graphical literacy, are important methods to develop and improve statistical understanding. The visualizations using charts of various types, also known as statistical plots, are among the most prevalent ways to explore, analyze, and describe the large pool of data, increasing day by day. Thus, graph/chart comprehension has become part of the academic curriculum at various levels.

Different levels of complexity are introduced across different grades in the school curriculum to enable students to interpret charts of different types, requirements, and difficulties. Some of the chart types are shown in Figure FC1.1. Bar charts and scatter plots become the most frequently used representation for data analysis and are available in textbooks, newspapers, and on the internet for data representation, as these chart types are taught at the school level. A Japanese educator, Kimura, had suggested a six-level scheme for the statistical ability of understanding charts [3] that describes the ability to comprehend graphs with the help of six levels starting from basic level-A with



(a) Scatter plot (b) Bar chart (c) Line chart

(d) Pie chart (e) Histogram (f) Distribution plot

Figure FC1.1: Various charts introduced at different levels in a school curriculum.

four sub-levels (A1-A4) to the advanced level-F. The chart understanding associated with each level starts from basic chart reading with sub-level A and reaches level-F, which points to a person's ability to use information depicted by charts to make new inferences from given information.

The choice of chart type for visualizing a given dataset is based on the simplicity of design for easy comprehension and non-cluttered visual encoding. However, chart comprehension in the initial learning stage, specifically in the case of multi-series charts, can be intimidating for students considering the wide variety of charts available across the curriculum. For example, a grouped bar chart can be challenging to decode due to the number of groups depicted in charts, color scheme selected for group representation, etc. However, these charts can be better understood when redesigned. This brings forth the gap in today's technology in automated chart reading, which can further be improvised for the redesign.

Building automated tools for chart reading is fraught with several limitations. One such is the unavailability of data for building machine learning systems for charts images. Let's take the example of such technology to aid graphicacy for visually impaired students. Although *screen readers* have provided an assistive approach using text-to-

speech conversion, these applications do not provide any narrative for images included in the documents. For such applications, the lack of availability of data is a limitation for using machine learning approaches. Dataset creation for training such models is also complex owing to the large design space for chart image creation, which includes chart formats, layouts, styles, text formats, etc.

Data extraction is one of the key processes in an automated tool for chart reading that infers knowledge from the data extracted from chart images. The data extraction process in itself is important for building assistive devices and learning aids for reading charts. The extracted data can be presented in the required format*, i.e.,* braille or excel sheets, needed for the creation of tactile diagrams, used widely by visually impaired people to learn by touch sense. The manual process of tactile diagram creation for each chart given in the textbook is a time-consuming and expensive process, as it requires user interaction to create the image in the required format before it can be converted into a tactile diagram. Hence, automating data extraction from chart images is an important problem to study.

The state-of-the-art in automating data extraction from chart images, or question-answer systems with chart images, involves image processing using handcrafted features or machine learning approaches. The machine learning approach is data-dependent and does not work for certain cases like stacked bar chart digitization and small image corpus for training, as mentioned in ChartSense [4]. We consider using handcrafted features to understand the nature of these images, with a scope of using our learnings in building appropriate machine learning approaches in the future. Our tensor field computation allows us to exploit geometrical attributes of chart objects and helps in locating them in image space. The tensor field computation is performed at pixel level and hence is impacted by image quality as well. Hence, we want to extend our tensor field analysis in integration with any deep learning based object detection model to locate chart objects effectively in the future. Our tensor field can be used as a next step to identify the

junctions in stacked bars located by object detection models. Graphical perception is about human understanding of a given chart with minimal interaction*, i.e.,* visualizing the data enables inferring knowledge from it. This perceptual understanding helps the user to have a rough idea about features of data to make future inferences like trends and summary. Chart perception is view-dependent as it follows the path of YOLO (You Only Look Once); hence, the exploration and summarization processes using visualization leave gaps in graphicacy for visually impaired students. Charts predominantly use geometry and color for visual encoding a given dataset so that a user can capture with a view of the chart only. Local features of color, geometry, and location extracted from the image of a chart play a key role in chart reading. Hence, these features play a key role in automated systems for chart reading.

In our work, we exploit local geometric features for data extraction from chart images. Let us look at the working of our proposed algorithm for data extraction for a relatively complex chart, say, a grouped bar chart. The grouped bar chart is useful to get inter-class and intra-class information in a single chart. However, for a user with limitations in using such charts, e.g., dyslexic learners, the graphical information has to be provided using simpler components. The redesigning of complex or multi-series charts requires source data that is used to generate the original chart as well as information about the classes/groups being represented in the image. A field study among different students from different middle and secondary school grades magnifies the approach of solving a complex visual problem by dividing it into simpler problems. Our data extraction approach can work as a primary step for reconstruction or re-plotting the data differently, as shown in Figure FC1.2, where a grouped bar chart can be redesigned as two simple bar charts or as scatter plots with the help of extracted data.

Our algorithm involves several tasks to accomplish data extraction, such as identifying the chart type and component annotation. A chart image is a combination of different components with specific functionality and location within the chart. The im-

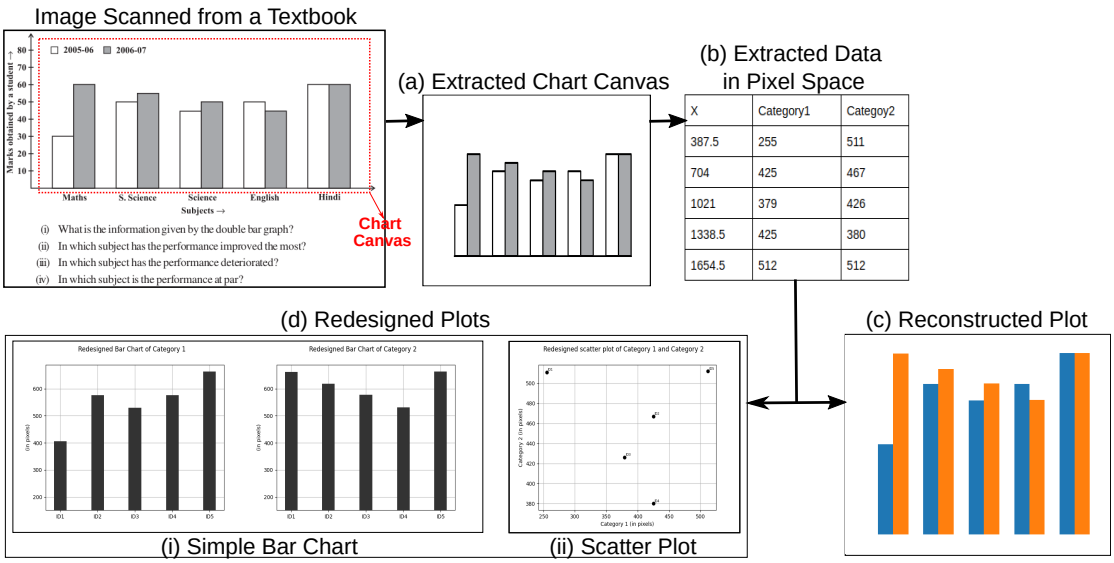


Figure FC1.2: Data extraction in pixel space from an image of a grouped bar chart using our method, for chart reconstruction and chart redesign. The input image is from a Grade-8 mathematics textbook. (Source: NCERT `http://ncert.nic.in/textbook/textbook.htm`).

portant components include axes, labels, legends, and chart objects. The interpretation of a given image depends on extracting these components and contextualizing them, e.g., providing a value with specific physical units to the height of the bar. The chart objects play a significant role in chart understanding, which is supported by other components. Text in the form of labels and titles in the chart and its legend is important for contextualization.

Here, we define different components as shown in Figure FC1.3, including the chart objects, to formalize the inputs to our proposed algorithm.

Definition 1.1. Chart Objects: The graphical objects or marks in the chart representation that encode the data are called chart objects. Chart objects are objects with non-geometric/geometric shapes, e.g., bars/columns in a bar chart and a scatterpoint in a scatter plot.

Definition 1.2. Chart Components: The chart components are regions in the charts based on their location and roles. They include x- and y-axes that set the reference lines in two-dimensional space, x- and y-labels that provide the values being represented by

Figure FC1.3: Various components of chart image highlighted in bounding boxes. (A) Chart components such as xy labels, title, and legend etc., (B) Chart objects such as bars/columns in this image, (C) Chart Canvas excluding legend, (D) The region inside bounding box is known as object interior, pixels outside this bounding box create object boundary.

the chart, title for both chart as well as axes, a legend used for multi-series/multi-class charts. The title and legend provide meta-information of context and groups in charts.

Definition 1.3. Chart Canvas: The chart canvas is a region of the image of a chart that contains only chart objects, such as bars or scatterpoints.

Definition 1.4. Object Boundary: Object boundary is created by the pixels for which the immediate 8-connected neighborhood contains neighboring pixels that do not belong to the chart object.

Definition 1.5. Object Interior: Object interior is a collection of pixels that have neighbors belonging to chart object.

1.1 Problem Statement

Data extraction from images of charts supports the automation of chart interpretation. This kind of chart interpretation process can be rightly called *chart digitization*. Chart digitization from images consists of four stages, namely, chart classification, text extraction, object detection, and data extraction [5]. Our primary goal is to create a system that takes a chart image as an input, labels the input image based on chart-type and sub-type, and extracts the data being represented in the chart image, as shown in Figure FC1.4 for the bar chart and its sub-types specifically.

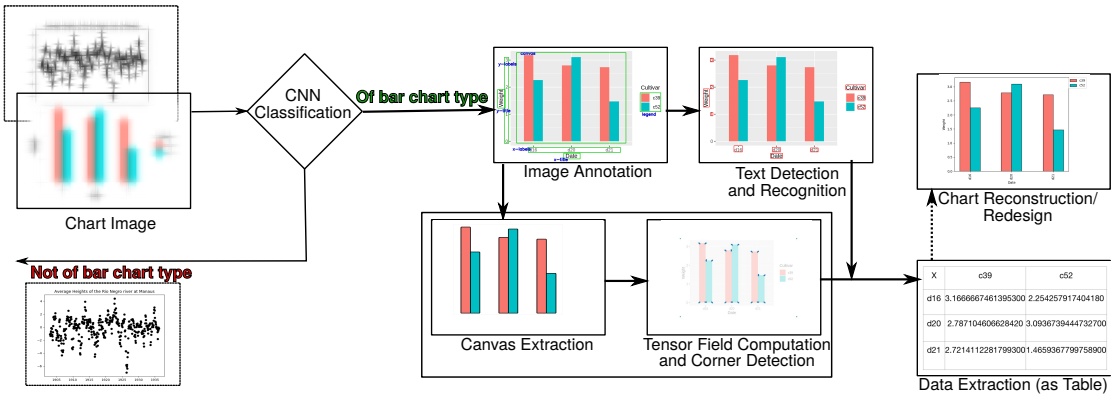


Figure FC1.4: Summary of the problem statement. We perform image classification and tensor field analysis on chart images and extract data associated with the given image.

We use image format over other forms of input representations of charts because of the wide availability of chart images in documents and on the internet. Our objective is to address the following problems:

- Classify the input image based on the chart types used for visualization.
  - The vast design space, including the variety of charts, makes them amenable to summarize the data for a broad spectrum of user requirements. Different chart visualizations entail different steps in visually depicting data. Hence, reverse engineering for data extraction from these chart types would require

an understanding of the primary steps that had been used for chart plotting in the first place. Even within a chart type, a different representation is needed for uni-, bi-, and multi-variate data; hence, the understanding of chart sub-type is also needed.

- Implement a pixel-based method like tensor field computation and analysis for data extraction from a raster image.
  - The recent attempts at chart interpretation and digitization have implemented machine learning methods to locate and identify chart objects that encode the values visualized in the image. As the datasets for such detection and localization models need to be large to train the model efficiently, we explore the traditional image processing techniques for our goal. Charts inherently have mathematical structures in their visualizations that can be exploited in reducing the requirement of large training sets.
- Improve the genericity of the chart digitization algorithm for different chart types.
  - Chart types and steps associated with decoding, once identified, build an algorithm for chart digitization. Such an algorithm has the goal of automating sub-levels of Level-A of statistical ability, as described by Kimura. The sub-levels A1 corresponds to the basic reading of chart image using the title, unit, and values, and the sub-level A2, to the reading of key features, e.g., minimum and maximum values, value differences, etc. It is straightforward to tackle one chart type at a time. However, there is value in exploiting common features in different chart types to improve the genericity of our algorithm.

The purpose of this thesis is to explore and analyze diverse methods that have been implemented for chart digitization and introduce our approach as an alternative and equally efficient solution. We deal with chart images with visualizations such as bar, scatter plots for uni-, bi- and multi-variate data.

1.2 Contributions

We propose an algorithm that, for a given chart image, provides a label for image classification, and further for predetermined chart types, proceeds to image annotation, and tensor field generation and analysis for data extraction. Image annotation is required for canvas extraction and chart component labeling.

- The novelty of our work lies in the exploitation of geometry of chart objects by using local geometric descriptors such as structure tensor and tensor voting to extract features like the corners of bars.
- We visualize different local geometric descriptors as well as saliency values to fix parameters for clustering salient pixels. Thus, our algorithm is semi-automated, with minimal user interaction.
- We focus on two types of charts, namely, bar charts and scatter plots. We also consider histograms as a special case of bar charts. Our experiments cover uni- and bi-variate data representation. The choice of charts is towards improving the genericity of our algorithm.
- As these charts are available as colored images, we have experimented with different color models such as RGB and CIE-Lab to analyze the impact of the color model on our computation model.
- We discuss different deep learning models used for chart classification along with the complexities and limitations associated with them. We introduce our classification model for identifying chart type and sub-type for a given image.
- We annotate a collection of chart images to create a dataset that can be used as a training dataset for object detection models based on deep learning. This can

improve the automation for canvas extraction task or the entire chart digitization process.

- We identify different applications where our chart digitization algorithm fits as a module, e.g., assistive solutions to a selected audience.
- Our work also discusses the limitation associated with the chart digitization process and our findings of specific bottlenecks.

1.3 Thesis Structure

The thesis structure is as follows: in Chapter 2, we provide a literature survey on chart analysis and digitization, emphasizing the variety of implementations and their corresponding limitations. In Chapter 3, we discuss our algorithm for data extraction from chart images using tensor field analysis along with the experiments made. In Chapter 4, we elucidate our classification models and annotation steps. Chapter 5 focuses on complex charts for multi-series or multi-variate data, such as grouped bar, stacked bar, and multi-class scatter plots and our experiments. In Chapter 6, we discuss the factors influencing our method, e.g., shape and size of different chart objects, color model, etc. We also address the limitations of our method, demonstrated by cases with poor results. In Chapter 7, we conclude the thesis with the potential applications of our work, with a note on complete systems that can use our algorithm as a module and the scope of future work towards improving our algorithm.

CHAPTER 2

LITERATURE SURVEY

The six-level scheme by Kimura suggests the ability to understand and comprehend details from charts at various stages. We target to develop a workflow that can meet sub-levels of level-A of Kimura's statistical ability scheme, *i.e.,* basic reading and extraction of key features. Chart interpretation and analysis is an interesting problem that has been explored by various implementations. In this chapter, we analyze previous attempts in this domain and identify the limitations along with the tasks to be performed for chart digitization and analysis. Chart analysis consists of smaller tasks, such as chart type classification, data extraction, and reconstruction or re-design if required. In this chapter, we discuss the relevant paths followed to introduce a chart interpreting system.

2.1 Chart Interpretation

Chart interpretation has been studied extensively in cognitive science [6] and document engineering [7] domains. The findings in [7] continue to be extensively used in separating text and graphics in charts using Optical Character Recognition (OCR) techniques [5, 8]. Our literature survey in the cognitive science domain has provided information about the constructs used in charts that have helped in our current work. The guiding principles in cognitive science provided by Hegarty [6] refer to features we have developed using local structure descriptors. The importance of the proximity of

objects supports the use of spatial locality-based approaches, specifically for the scatter plot [9]. Bar charts and other charts have demonstrated properties of integral, configural, and separable/object displays depending on the mapping of the variable [10]. The information integration using bar charts has been found more useful [11]. While there have been several studies done in chart interpretation in parts, or whole for few chart types, there is still a gap in standardized and generalized methods for chart interpretation, which will work for more than one type of chart [12].

In our work, we detect corners in bars and histogram, based on spatial proximity, as these corner pixels are important in providing the height of the bar or bin, respectively, and consequently, for data extraction. Similarly, we extend the use of local geometric descriptors such as tensor fields to locate scatterpoints in image space.

2.2 Chart Classification

Visualization of data is an effective way of seeing the trends represented in data. Charts are a universally accepted form of visualization that has been discussed in the academic curriculum. Hence, charts are also the most frequently used visualization form due to their familiarity. The selection of chart type for visualizing the data is dependent on the data attributes and trends we want to showcase in the final chart. As different chart types will plot the data differently and may highlight different points of data, chart comprehension is highly dependent on the chart type selected for the visualization. Similar to chart analysis, data extraction steps for a given chart image are specific to the chart type. ReVision is one of the earlier works that introduced the idea of using feature vectors and geometric structures to extract visual elements and data encoded in the chart [13]. ReVision performed text type classification using feature vector generated using the geometric property of text along with mark type/chart type classification using a fine-tuned AlexNet. Image classification has been a highly explored problem in

computer vision; hence, many machine learning models have been introduced for chart classification as well. A web-based system Beagle classifies charts images in scalable vector graphics format [14]. In another study, the AlexNet classifier has been fine-tuned to take a bitmap image and classify it based on the estimated mark type [8]. Following the same approach of using a pre-trained model for chart classification, ChartSense uses GoogleNet to identify chart type from categories line, bar, pie, scatter plot charts, map, and table types, specifically [4] and addresses a limitation in digitizing stacked bar charts. The same fine-tuning approach is applied in FigureSeer [15], which is an end-to-end framework for the summarization of line charts exclusively.

In our work, we use a convolutional neural network based classifier to identify the chart type represented in the given image. We further classify the bar charts based on the sub-types such as simple, grouped and, stacked bar charts.

2.3 Data Extraction

In WebPlotDigitizer, the user is provided with an option to use the automated or manual procedure for data extraction from the given chart image [16]. The tool requires the user to select the chart type for the uploaded image from a given list, align the axis and mask the chart component by drawing multiple points on the graphical object. The data extraction process fails to get group/sub-group information from grouped and stacked bar charts. Similar to WebPlotDigitizer, Scatterscanner requires user interactivity for data extraction [17]. However, the scope of this system is limited to scatter plots. Scatteract is an automated system that extracts data from scatter plot images by mapping pixels to the coordinate system of the chart with the help of OCR [18]. ReVision uses chart layouts for bar and pie charts for pixel-based data extraction [13]. Choi et al. [5] have used different approaches for data extraction from bar charts, pie charts, and line charts. A Darknet neural network as an object detection model in combination with

OCR for text detection is used from bar charts. The proportion of pixels belonging to color within the periphery of the circle is utilized for pie charts. Automated data extraction for bar charts has been done by identifying graphical components and text regions independently [19]. Their system extracts the chart data using inference. ChartSense detects bars using the connected component method with the x-axis and extracts data effectively for simple bar charts [4]. However, the system provides incorrect results for multi-series bar charts (e.g., grouped bars) and does not extract data from stacked bar charts. PlotQA is used as an analysis tool for reasoning over scientific plots that uses a more accurate neural network for object detection for visual elements, such as bars [20]. However, the manual creation of bounding boxes does not extend to complex bar charts in a straightforward manner. The existing methods that use object detection methods to locate bars and to extract data have not considered one category of bar chart, *i.e.,* stacked bar. Data extraction from a stacked bar chart differs from the extraction from simple as well as grouped bar charts. Also, data extraction using the machine learning based object detection model depends on locating the bar object in the image; however, in a stacked bar chart, the bar junctions denoting the stack of each class/series are important to extract the data.

Hence, we refer to hand-crafted features based on image processing techniques such as tensor field analysis to locate bars as well as stack height in bar chart images.

2.4 Tensor Field Analysis

Extraction of perceptual information from point clouds has been performed using tensor voting [21]. Moreno et al. [22] have proposed the use of tensor voting for structure estimation from a variety of images. Their work also explains the similarities between tensor voting and structure tensor and extends tensor voting for grey-scale, vector- and tensor-valued images. The closed-form solution of the tensor voting com-

putation for n-dimensional data using a structure-aware tensor [23] helps in direct and efficient computation. The closed-form solution is different from the conventional one, where components like stick-, plate- and ball-voting for the tensor field are computed separately. The closed-form solution also allows any second-order symmetric tensor as a starting tensor for gathering votes, thus making the methodology more generic and functional in implementation.

We, thus, extend tensor field computation and analysis to locate and cluster degenerate points in 2D images of charts. The clusters of centroids allow us to locate scatter-points as well as provide information on bar heights in pixel space. We further map this pixel space data to original data dimension/units using OCR. Our work would be the closest to the use of the Hough transform in image space to recognize bar charts [24] in the context of deriving features in image space and finding generic patterns.

CHAPTER 3

TENSOR VOTING FOR CHART IMAGES

Chart analysis is a step-by-step process that also includes locating chart objects in the given image to decode the information. Object detection models locate the chart objects such as bars and scatter points but limits in identifying division in stacked bars. Here, we try to understand graphical perception, which highlights the importance of chart objects as the user tends to look at these objects as the first step towards chart analysis. In cognitive science, the perceptual understanding of charts requires global properties of the chart to be processed before local properties such as color or geometry of chart objects [25]. However, using local attributes to compute global features is more efficient [26]. The visual understanding of the chart uses the spatial proximity of chart objects to understand relative features as well as exact values. Hence, Spatial proximity, in particular, is important in charts of our interests, as the closeness in pixels of geometric objects, such as bars and scatterpoints, helps in extracting data and in making comparisons and trend inferences. Integrated tasks involve information integration, including object integration [11]. Since local geometry plays a significant role in the chart analysis as given by the PCP (Proximity Compatibility Principle), our proposed tensor field representation is for the chart canvas alone. Here, the chart canvas is the plot itself, after stripping off the axes and textual information, which are predominantly contextual and geometrical, to a lesser extent. We target to introduce a computational model for data extraction from images of bar charts, scatter plots, and histograms.

Differential-based features are widely implemented for tasks like edge detection, corner detection, shape analysis, and feature tracking. We use local geometric descriptors in the form of positive semi-definite second-order tensor fields like structure tensor and tensor voting that determines geometric information of objects and global perceptual information by collecting votes at each entity based on the normal tensors of its local neighbors. It is efficient for tasks such as Image Segmentation, completion, and reconstruction, where global context plays an important role. The votes at each entity are collected using different components, stick-, plate-, and ball-tensors, in 3D data, and stick- and ball-tensors in 2D. Tensor votes are widely used to capture perceptual information in natural images. Our aim is to implement tensor voting to get insight from a given chart image, such as the position of chart objects and trends followed by the encodings represented in the chart image.

The location and shape of chart objects in the image encode the information being represented in chart form. For bar chart representation, corners of bars can help us to detect the location and height of the individual bar. Similarly, a scatter plot encodes data, with each scatterpoint representing the information at its center. We want to exploit the geometric and spatial properties of charts to get these encodings and use local geometric descriptors such as structure tensor and tensor voting to detect corner points. Tensors are mathematical structures/objects defined by *dimension* and *order*.

3.1 Structure Tensor

Structure tensor T_s is an outer product of the gradient vector, and it stores information of the direction of the gradient of the local neighborhood. Although structure tensor is applicable to higher-dimensional domains, its application in the image processing/computer vision domains is highly appreciated.

T_s is computed from the gradient tensor, $T_g = G^T G$, using the gradient vector, $G =$

$\begin{bmatrix} \frac{\partial I}{\partial x} & \frac{\partial I}{\partial y} \end{bmatrix}$, at a pixel with the intensity I. We use the sobel operator to compute gradient vector in images and apply Gaussian function with zero mean and standard deviation ρ.

$$T_s = G_\rho * T_g \tag{Eqn 3.1}$$

where $*$ is a 2D convolution operator, in the case of 2D images.

We take each individual channel representing the given RGB image and normalize the values in the range [0,1] by dividing the intensity values by 255.0 for our 2D chart images. The normal intensity values are forwarded for derivative computation using a sobel operator that gives gradients across x and y directions. The xy-directional gradients generate structure tensor T_s and are convolved using Gaussian function with a standard deviation value of 0.1. Visualization of eigenvectors received post eigenvalue decomposition on this local geometric descriptor highlights the direction of the gradient.

3.2 Tensor Voting

Tensor voting is a process of collecting votes at each entity, *i.e.,* each pixel in case of 2D images, based on normal tensors of its neighbors. It has been implemented for extracting perceptual information from point cloud, particularly in 3D [21]. The method determines the likelihood of a point/pixel belonging to a surface, a curve, a junction, or an outlier. The voting tensor determines the global perceptual organization of an object [27]. The traditional implementation of tensor voting depends on the aggregation of stick-, plate- and ball- tensors in 3D data and stick- and ball-tensors in 2D data.

Wu et al. [23] have proposed a closed-form solution for tensor voting computation based on the structure-aware tensor. Here, a structure-aware tensor captures details such as surface, junctions, or curves in the case of 3D data. The visualization of structure-

aware tensor using ellipse or ellipsoid for 2D and 3D data respectively elaborates on tensor being a surface, plate, or ball type. The closed-Form solution calculates tensor vote at x_i induced by K_j located at x_j in d-dimensional space as:

$$S_{ij} = c_{ij} R_{ij} K_j R'_{ij}, \tag{Eqn 3.2}$$

$$\text{where } R_{ij} = (I_d - 2 r_{ij} r_{ij}^T);\ R'_{ij} = (I_d - \tfrac{1}{2} r_{ij} r_{ij}^T) R_{ij}$$

I_d is the d-dimensional identity matrix and direction vector $r_{ij} = \hat{d_{ij}}$, where the distance vector $d_{ij} = x_j - x_i$; $c_{ij} = \exp\big(-\big(\sigma_d^{-1}.\|d_{ij}\|_2^2\big)\big)$; and σ_d is the scale parameter.

In the closed-form solution, various K_j values can be implemented that give distinguished voting fields such as plate, ball, or stick based on value selection. The use of a generic second-order tensor in place of K_j eliminates the requirement of computing the voting field. As we compute structure tensor T_s at each pixel using equation (Eqn 3.1), the same is used for K_j value. Once S_{ij} is calculated at each pixel with its immediate neighbors, called von Neumann (or 4-) neighborhood $\mathbb{N}_4$ using equation (Eqn 3.2). The votes imposed at each pixel induced from all its neighbors are aggregated using summation with $\sigma_d = 4$, based on neighborhood size, we get the aggregated positive semidefinite second-order tensor as:

$$T_v = \sum_{k=0}^{(d-1)} \sum_{j \in \mathbb{N}_4} S_{ij}(d) \tag{Eqn 3.3}$$

3.3 Anisotropic Diffusion

The tensor votes induced by T_v are in normal space, and thus, their aggregated tensor is also in the normal space following Gestalt's principles of perceptual organization. However, we need the tensor field to encode the geometry of chart objects like bars, scatterpoints, etc., which implies that the tensor has to be transformed in tangential

space from normal space. We implement anisotropic diffusion on the tensor voting field used for 3D point cloud for our 2D chart images [28, 29].

The anisotropic diffusion for the 2D case requires eigenvalues of T_v, $\lambda_0 \geq \lambda_1$, and corresponding eigenvectors v_0 and v_1. The tensor after anisotropic diffusion using diffusion parameter δ, for which a widely used value is 0.16 [28, 29], is:

$$T_{\text{v-ad}} = \sum_{k=0}^{1} \lambda'_k . v_k v_k^T, \qquad \text{(Eqn 3.4)}$$

$$\text{where } \lambda'_k = \exp\Big(-\frac{\lambda_k}{\delta}\Big)$$

Consequently, tensor voting upon anisotropic diffusion gives a new tensor field, $T_{\text{v-ad}}$.

3.4 Saliency Map Computation

The eigenvalue decomposition of the local geometric descriptor of a 3D point provides information of a point belonging to either a line-, surface-, or point-type feature. For our 2D chart images, eigenvalues of local geometric descriptor $T_{\text{v-ad}}$ computed after anisotropic diffusion provides the probabilistic geometric classification of pixels to the line- and point-type features. As this is a probabilistic classification, the sum of probabilities of a point being point-type (C_p) and line-type (C_l) is 1. The C_l and C_p values are computed using the eigenvalues of $T_{\text{v-ad}}$ as:

$$C_l = \frac{\lambda_0 - \lambda_1}{\lambda_0 + \lambda_1} \; and \; C_p = \frac{2\lambda_1}{\lambda_0 + \lambda_1}, \qquad \text{(Eqn 3.5)}$$

where eigenvalues of $T_{\text{v-ad}}$ for the pixel with $\lambda_0 \geq \lambda_1$ are used. Tensor field analysis with probabilistic geometric classification provides a degenerate point at a pixel with $C_p \approx 1.0$, attributed to the anisotropic local neighborhood.

3.5 DBSCAN Clustering

We observe that the degenerate points computed from both bars and scatterpoints do not show any specific pattern, shape, or cluster size. Hence, the degenerate points need to be clustered. In our case, the number of clusters is data-dependent, owing to which clustering methods using the number of clusters as a hyperparameter can not be used. Similarly, methods that cater to specific shape or size of clusters are not applicable here, e.g., k-means clustering works with spherical clusters effectively.

Density-Based Spatial Clustering of Applications with Noise (DBSCAN) is a density-based well-known clustering algorithm that groups a set of points together based on distance (usually euclidian distance eps) and the minimum number of points (minPts) [30]. The distance parameter ”eps” specifies the value of proximity of points for them to be considered as co-existing in a cluster. It means that if the distance between two points is lower or equal to this value (eps), these points are considered neighbors. The parameter minPts indicates the minimum number of points required to form a dense region in a specific application, e.g., $minPts = 5$ implies that we need at least five points to form a dense region/cluster. DBSCAN initiates the process of finding clusters using a random point p and identifies all points density-reachable from it with respect to eps and minPts. This process is iterative and classifies point p as core point or border point. If point p is the core point, DBSCAN provides a cluster id to point p. No points will be density-reachable to point p if point p is border point. In this case, DBSCAN checks for the next point in the database. The global density parameters ”eps” and ”minPts” are used to determine if a point is a core point or can be classified as noise. If the neighborhood of a point in radius eps does not contain points more than minPts, the point is classified as noise. These noise-classified points are revisited if they are density-reachable to the point in interest and are classified with cluster id.

In general, small eps values cause large data not to be clustered together by wrongly

identifying those points as outliers. At the same time, large eps values introduce cluster merging, and hence, the majority of objects will be in the same cluster. Thus, finding an optimal eps value is a requirement, which requires domain knowledge. In our case, we use the visualization of critical/degenerate points identified using the saliency map to perform parameter estimation process for DBSCAN.

3.6 Data Extraction

The data extraction process is the same as chart analysis. Different chart types having different shapes of geometric objects encode the data in image impacts the information depicted by the chart; similarly, the data extraction process needs to consider the chart type to determine the steps to be followed in reverse order.

For scatter plots, as each point encodes xy-coordinates labeled on the xy-axis, the extracted point in pixel space should give us the location of scatterpoint in the image. To achieve this goal, we identify clusters using DBSCAN and then calculate the centers for each cluster. As the clusters are generated due to the scatterpoint, the center denotes the location for the point in concern.

The bar charts are plotted in a specific manner, thus, needing a different method for data extraction. Our degenerate points guide us to the corner of each bar. The final cluster centers calculated after DBSCAN clustering are listed as corner points belonging to each bar. An individual bar has four corners depicted by four cluster centers. These centers allow us to get the baseline, bar width, and final height of the bar in pixel space.

3.7 Our Proposed Algorithm

Our algorithm combines the components mentioned in the above sections to extract data from the given chart image. As our computational model is implemented at the pixel level to gather geometric information from graphical objects, the first module of the workflow needs to cleanly identify the objects and separate these objects from other image components such as axis lines, grid lines, etc. Our algorithm uses morphological operations to remove such fine components from the source image, adds a border to the object to remove effects of pixelation, and gives chart canvas having only chart objects as entities.

The preprocessed image is then fed through the tensor voting computation module that also provides saliency mapping for each pixel. The saliency map identifies the degenerate points from the given set of pixels. The degenerate points are clustered to get the data encoded with the chart object in pixel space [31].

3.8 Error Analysis for Data Extraction

We perform qualitative and quantitative error analysis by reconstructing charts of the same type from extracted data and Earth Mover's Distance, respectively. The accuracy of chart interpretation and reading is based on the accuracy with which the data/quantitative information can be extracted from the given visualization. Hence, Any chart interpretation and extraction model can be efficient only if the error in data extraction is minimal. The problem with quantitative analysis lies in the loss of context of chart text, axes, and legend. As the extracted data is in pixel space, we calculate appropriate distance measures to compare distributions.

Earth Mover's Distance, d_{EMD}, is a measure of cross-bin distances to identify dis-

Algorithm 1: Data extraction using tensor fields from chart images.

Input: Chart image C_i, chart-type C_t
Output: Data table D

Initialize $D \leftarrow \varnothing$
Initialize $S_{\text{deg-pt}} \leftarrow \varnothing$ `// Set of degenerate points`

Initialize $D_{cq} \leftarrow \varnothing$ `// Cluster centroids of degenerate points`

$C_c \leftarrow$ chart-canvas-extraction(C_i) `// from Algorithm 1`

for *pixel i in* C_i **do**
    $N \leftarrow$ find-N_8-local-neighborhood(i)
    $T_{\text{geom}} \leftarrow$ compute-tensor(descriptor-type, N) `// Local geometric descriptor`

    $C_l, C_p \leftarrow$ compute-saliency-map(T_{geom})
    `/* Check if the pixel is a strong degenerate point */`

    if $C_p > \tau_{cp}$ *and* $trace(T_{geom}) > \tau_{wd}$ **then**
        $S_{\text{deg-pt}} \leftarrow$ set-union($S_{\text{deg-pt}}$, i)
$C_{\text{deg-pt}} \leftarrow$ DBScan($S_{\text{deg-pt}}$) `// Cluster degenerate points`

for *cluster q in* $C_{deg\text{-}pt}$ **do**
    $C_q \leftarrow$ compute-centroid(q)
    $D_{cq} \leftarrow$ set-union(D_{cq}, C_q)
if C_t *is bar-chart* **then**
    $D_{cq} \leftarrow$ set-union(D_{cq}, missing-points) `// Rule-based occurrence patterns`

    $D_{cq} \leftarrow$ sort-first-by-x-and-sort-second-by-y(D_{cq})
    for *unique x-value in (x,y) in* D_{cq} **do**
        $\delta_y \leftarrow$ find-y-intervals(x, D_{cq})
        $D \leftarrow$ set-union(D, (x, δ_y)) `// Add to data table`

else if C_t *is a scatter-plot* **then**
    for *(x,y) in* D_{cq} **do**
        $D \leftarrow$ set-union(D, (x,y)) `// Add to data table`

return D

similarity between two multi-dimensional distribution, which uses ground distance measures [32]. We compute the Earth Mover's Distance between:

- the univariate distributions for the extracted data and original data in the case of bar charts, $d_{\text{EMD-BC}}$.
- the extracted data and the frequency table of the original data in the case of histograms, $d_{\text{EMD-HG}}$.
- the 2D point clouds of the extracted data and original data in the case of scatter plots, $d_{\text{EMD-SP}}$.

We normalize the extracted data as well as source data to compute EMD.

3.9 Visual Analysis of Tensor Fields

Here, we propose a computational model for data extraction from chart images based on local geometric descriptors. The model can be made optimal using a comparative study of various geometric descriptors such as T_s, T_v, and $T_{\text{v-ad}}$. We propose using saliency values, shape, and orientation of the tensor as metrics to compare the outcomes of all descriptors by visualizing these measures. The tensor field visualization helps in data exploration, feature/pattern identification, and decision making. We use the following methods for visualizing different entities:

- Dot plots with color-mapping to study patterns across points/pixels based on saliency value, including degenerate points.
- Ellipsoid glyph visualization for tensor field analysis based on orientation and shape of feature descriptor.

3.9.1 Dot Plot with Color-map for Saliency Value

As saliency values denote the likelihood of a point/pixel belonging to different feature classes: line-, surface-, and point- type, we use C_l, C_p values generated for our 2D images and map the saliency values at each pixel with color-blind safe divergent color palette, namely the coolwarm palette. As $C_l + C_p = 1.0$, we take C_l values from 0 to 1 and visualize the same using dot plots with color mapping. The degenerate points are identified as blue points in the visualization, pointing to a high C_p value, as shown in Figure FC3.1, row D.

3.9.2 Tensor Glyph Visualization

Glyphs are marks used in visualizations, such as arrows, to study the properties at a single point or instance, and the glyph size, shape, color, and position are used as channels. Here, we use ellipsoid glyphs to visualize the shape and orientation of tensors at each point. We take eigenvectors of our second-order tensor calculated for each pixel and use the same with quiver plot to visualize glyph at each pixel. The glyphs are assigned a color based on the color mapping associated with major and minor eigenvector*, i.e.,* the major eigenvector is colored red, and the minor eigenvector is assigned blue color. The visualizations for structure tensor T_s and tensor voting after anisotropic diffusion $T_{\text{v-ad}}$ are shown in Figure FC3.1, row B, and C.

3.9.3 Dot Plot for Degenerate Point Visualization

The critical points are identified on filtering the saliency values at all pixels. As these pixels create a sparse cluster, the visualization helps to locate these clusters on the source image, as shown in Figure FC3.2. We use the dot plot to visualize these points and use the source image as a watermark to locate the critical points in the image.

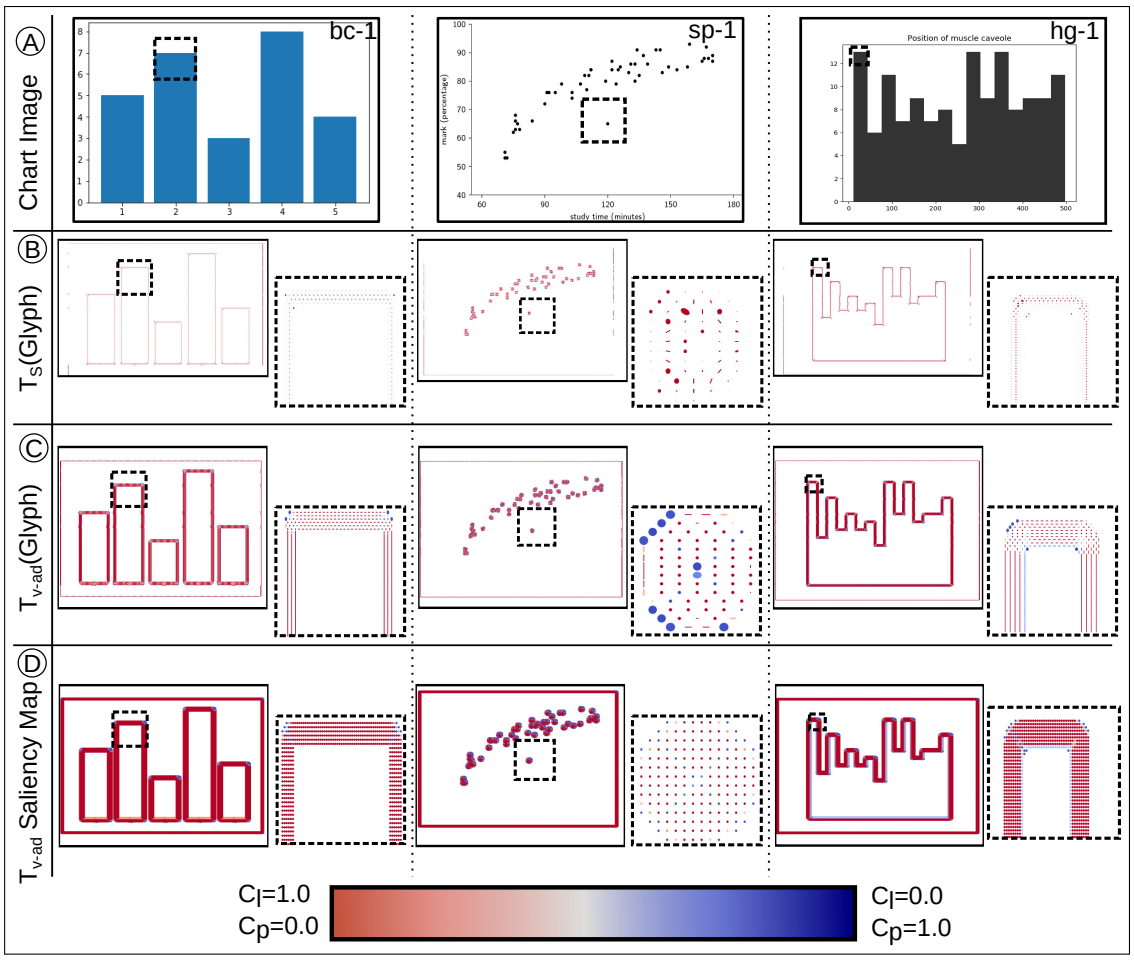


Figure FC3.1: The tensor fields computed from images of charts using our approach. (A) Input images of bar chart, scatter plot and histograms. Tensor fields computed on the images, visualized using glyphs and colored by saliency values, include (B) structure tensor T_s, and (C) tensor voting field of T_s after anisotropic diffusion $T_{\text{v-ad}}$. (D) The saliency values of $T_{\text{v-ad}}$ visualized using dot plots. The coolwarm color mapping associated with corresponding C_l and C_p saliency values, used in the visualizations, is shown in the colorbar.

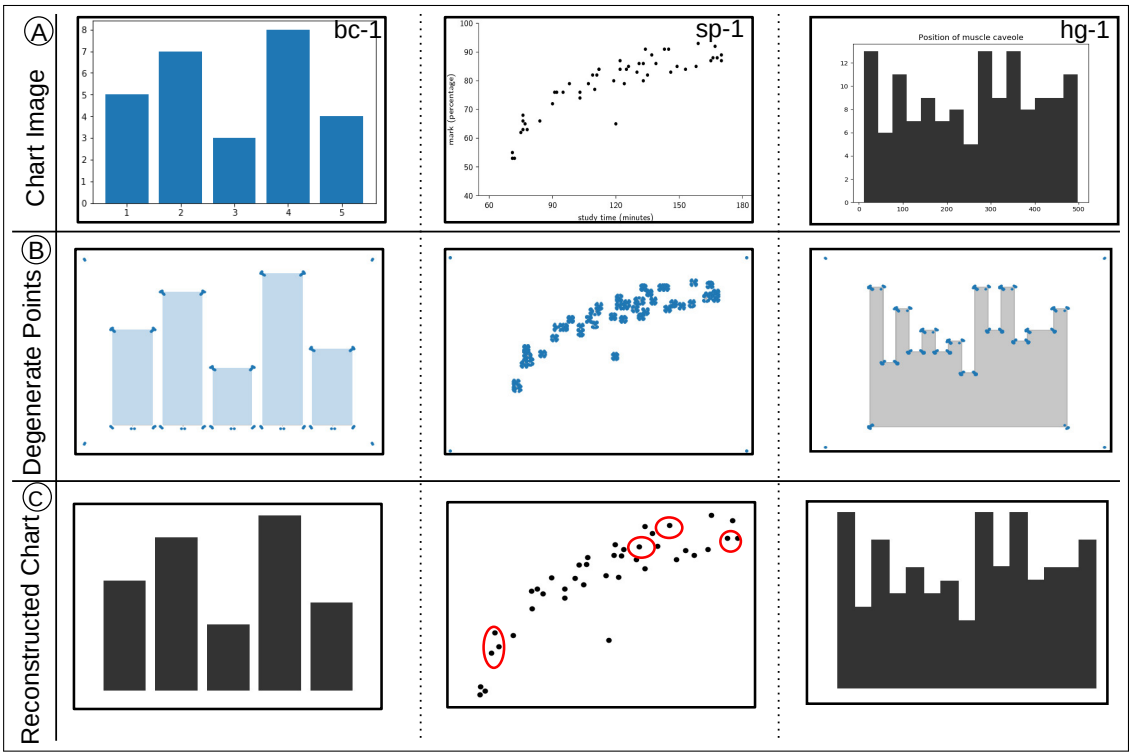


Figure FC3.2: Use of degenerate points in $T_{\text{v-ad}}$ field for data extraction from chart images, and validated using chart reconstruction. (A) Input images of bar chart, scatter plot and histogram. (B) Visualization of the pixels corresponding to degenerate points based on C_l and C_p values of $T_{\text{v-ad}}$ at the corners of bar/bin for and near the center of scatterpoint, (C) The reconstructed charts from the extracted data for the input images.

3.10 Experiments

Tensor fields are computed at pixel level, and hence, the image resolution affects the identification of degenerate points. For experiments, we have prepared our own dataset by collecting data both from web images and synthetically generated images. The synthetic images have been generated using the Python library, `matplotlib.pyplot` [33], and are stored in .png image format. The experiment set contains images of bar charts, scatter plots, and histograms. For all chart images of test datasets, we constructed the tensor fields and reconstructed data. We specifically used this library since it generates high-resolution plots. The tensor field computation works best in high-resolution images, thus making our pixel-based algorithm sensitive to the input image resolution.

For bar charts, we included various cases where a large number of bars are plotted in a single visualization, non uniformly placed bars, smaller set of bars, and bar charts with large variation in the bar heights. Scatter plot images contain positive and negatively correlated data representation and overlapping scatterpoints. For histograms, we plotted data that showcase variations in the number of bins, with close-to-zero frequencies in some of the histogram bins, with several large variations in frequencies represented as several peaks and valleys in the histogram, and with the close-to-normal distribution. As we have the available data tables used for synthetic chart images, we also perform quantity-based error analysis by computing Earth Mover's Distance.

3.10.1 Results

The visualizations for tensor fields, quiver plots of eigenvectors, color map for saliency values, and degenerate point are generated using `matplotlib.pyplot`. Our results are shown in Figures FC3.3–FC3.8, and Table TC3.1, and are further analyzed here.

3.10.1.1 Tensor Field Analysis

Our local geometric descriptors T_s, T_v, and $T_{\text{v-ad}}$ encode geometric attributes by exploiting spatial locality. The tensor $T_{\text{v-ad}}$ provides stronger degenerate points with higher C_p values as compared to T_s in our experiments listed in Figures FC3.3–FC3.4–FC3.5–FC3.6–FC3.7–FC3.8. Hence we use $T_{\text{v-ad}}$ for our data extraction process.

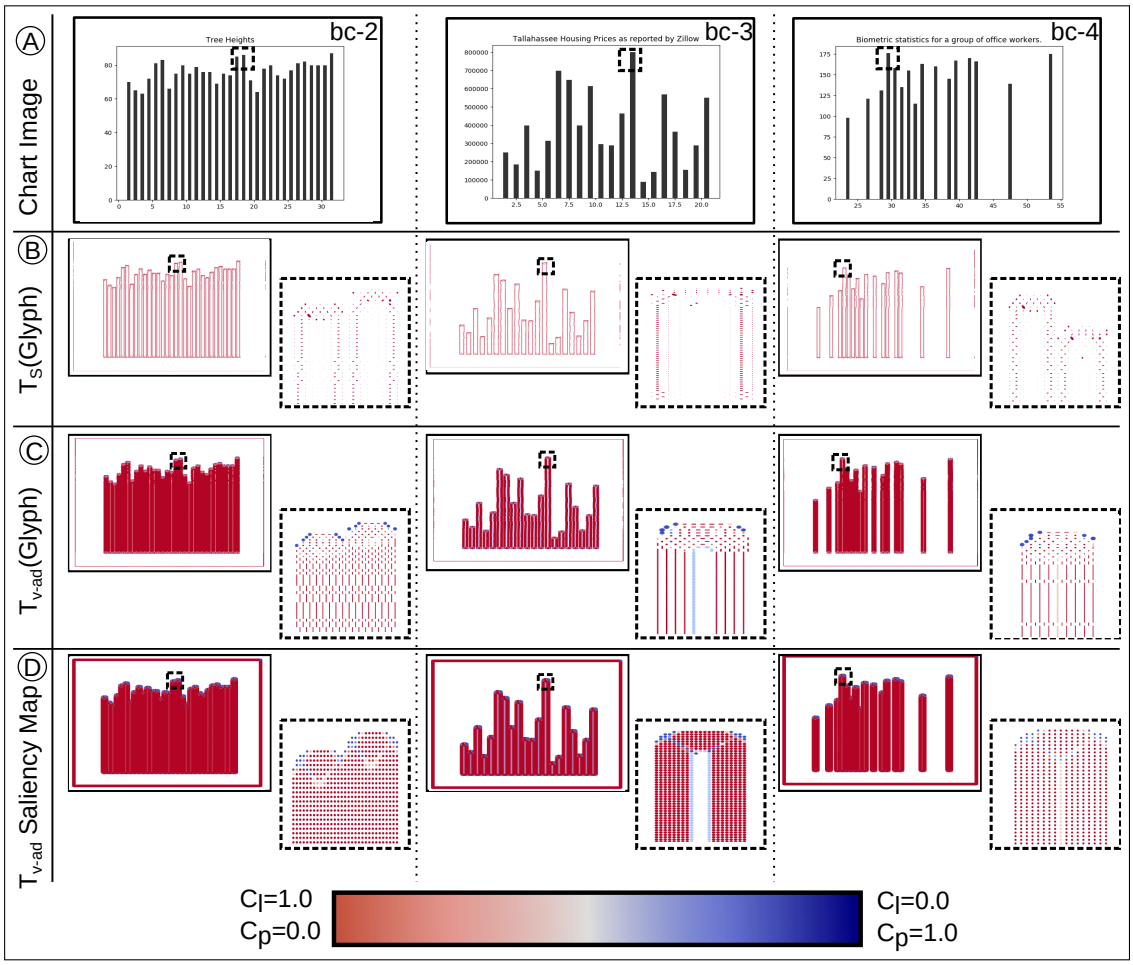


Figure FC3.3: The tensor fields computed from images of charts using our approach. (A) Input images of different variants of simple bar chart. Tensor fields computed on the images, visualized using glyphs and colored by saliency values, include (B) structure tensor T_s, and (C) tensor voting field of T_s after anisotropic diffusion $T_{\text{v-ad}}$. (D) The saliency values of $T_{\text{v-ad}}$ visualized using dot plots. The coolwarm color mapping associated with corresponding C_l and C_p saliency values, used in the visualizations, is shown in the colorbar.

The tensor field visualization using glyphs and dot plots color mapped based on

saliency value shown in Figures FC3.3–FC3.4 gives degenerate points having high saliency value. The sparse cluster of such degenerate points can be seen at corners/junction of bars and histograms and centers of scatterpoints, as displayed in Figures FC3.6–FC3.8. For clean visualization, we plotted the tensor field and saliency values at every third pixel.

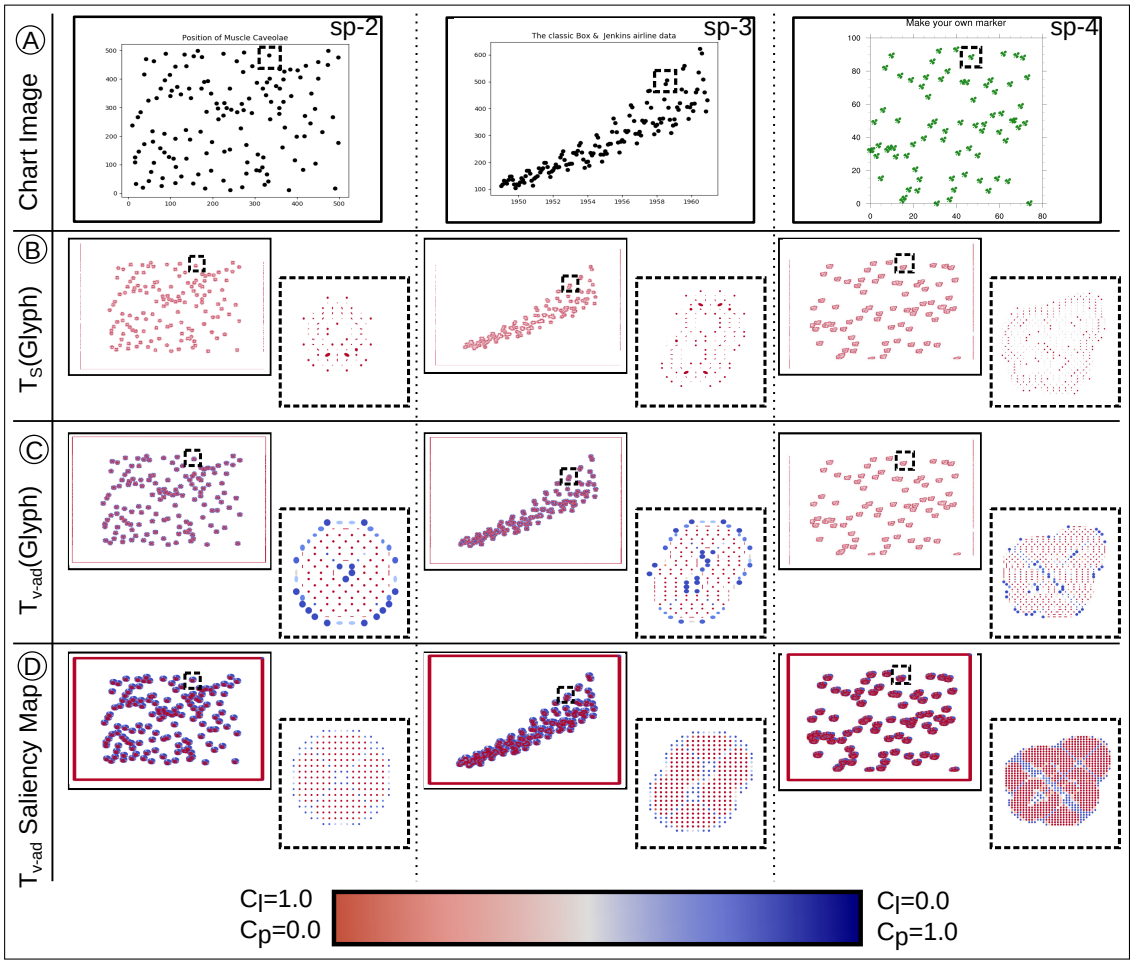


Figure FC3.4: The tensor fields computed from images of charts using our approach. (A) Input images of different variants of simple scatter plots. Tensor fields computed on the images, visualized using glyphs and colored by saliency values, include (B) structure tensor T_s, and (C) tensor voting field of T_s after anisotropic diffusion $T_{\text{v-ad}}$. (D) The saliency values of $T_{\text{v-ad}}$ visualized using dot plots. The coolwarm color mapping associated with corresponding C_l and C_p saliency values, used in the visualizations, is shown in the colorbar.

We take a look at the impact of bar thickness on tensor field computation in images of bar charts. Bars with standard or more width demonstrate homogeneity in the

bar interior, as we observe zero-tensors near the centroid, as shown in Figure FC3.1, `bc-1`. Hence, non-zero tensors for the non-zero gradient clearly demarcate the boundaries of thick bars. However, the tensor field in the case of thinner bars, as shown in Figure FC3.3, demonstrates that the boundary of the bar is not clearly demarcated. We observe that there is the degenerate points at the different corners of the bar are closer. The data extraction process relies on correctly identifying two distinct corners for each bar, which may fail in the case of thinner bars if the clustering algorithm is not adjusted. Hence, the visualization of degenerate points is required for changing hyperparameters of the DBSCAN clustering algorithm.

3.10.1.2 Data Extraction

Data extraction using our tensor field model for different chart types and images is as shown in Figures FC3.6, FC3.7, and FC3.8. As our module works at the pixel level on chart canvas, without any text component, the extracted data is based on the pixel location occupied by the chart objects. Threshold-based filtering of degenerate points removes noise and hence helps in the data extraction process.

3.10.1.3 Error Analysis

The reconstructed charts shown in Figures FC3.6, FC3.7, FC3.8 are generated using extracted data in pixel space and hence, should be compared with the source image visually. As the source data is available only for the synthetic images, We calculated Earth Mover's Distance between the extracted data and the source data for the synthetic plots only. The EMD values are listed in Table TC3.1 for synthetic scatter plots, bar charts, and histograms.

The visual comparison of reconstructed charts and input chart image also points out the missing values/scatterpoints in Figure FC3.7 marked in red contour. The type-2

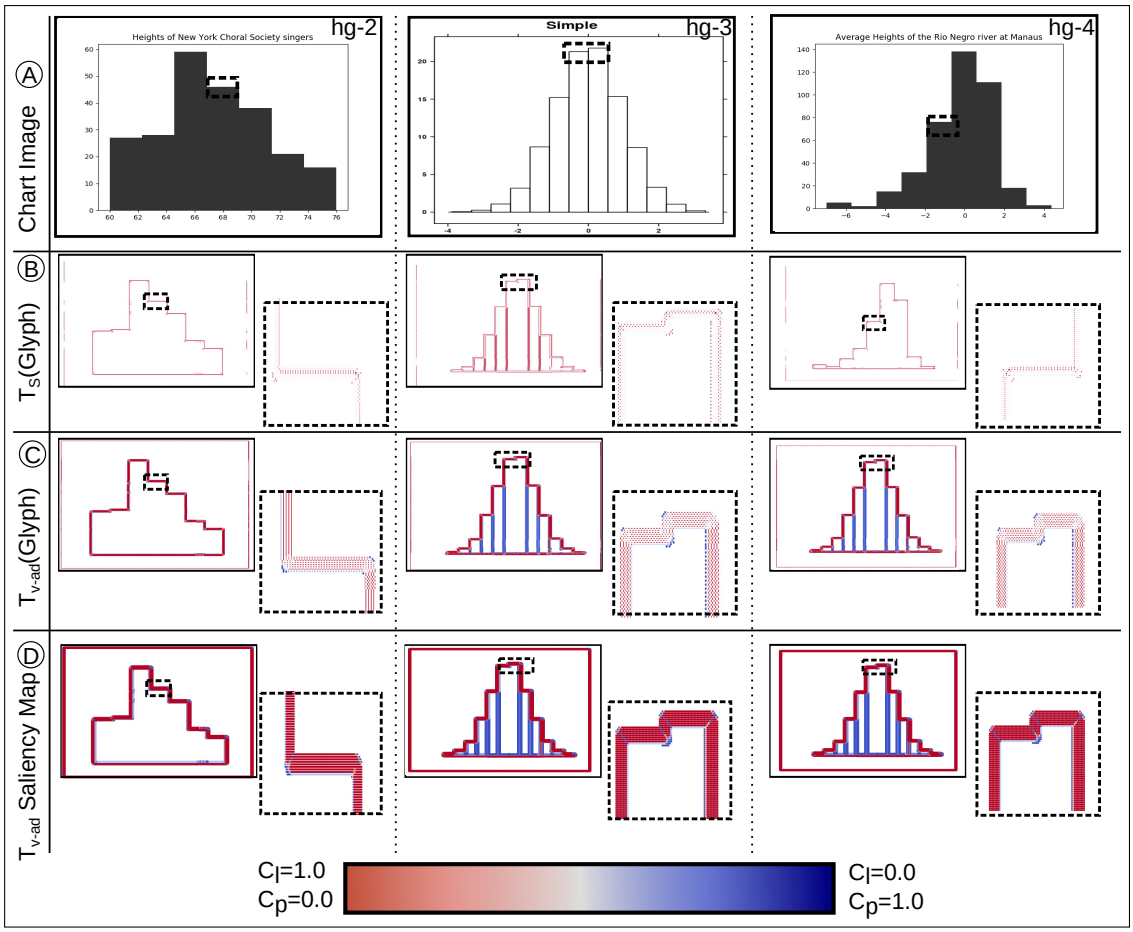


Figure FC3.5: The tensor fields computed from images of charts using our approach. (A) Input images of different variants of simple histograms. Tensor fields computed on the images, visualized using glyphs and colored by saliency values, include (B) structure tensor T_s, and (C) tensor voting field of T_s after anisotropic diffusion $T_{\text{v-ad}}$. (D) The saliency values of $T_{\text{v-ad}}$ visualized using dot plots. The coolwarm color mapping associated with corresponding C_l and C_p saliency values, used in the visualizations, is shown in the colorbar.

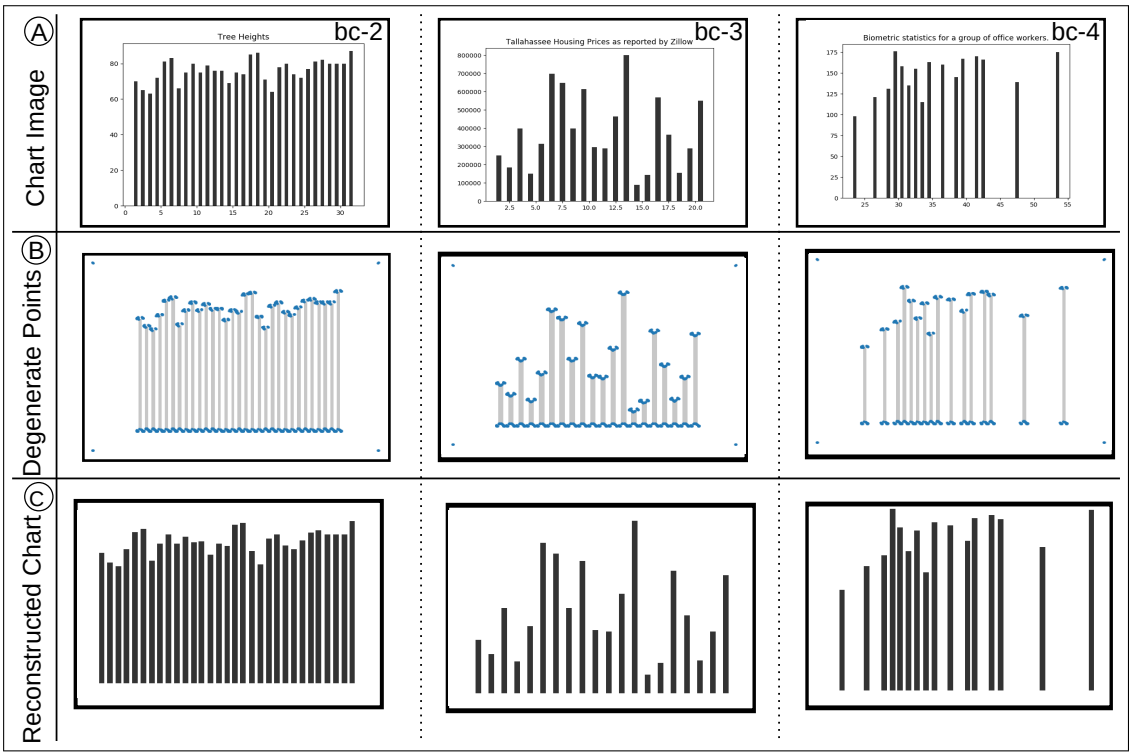


Figure FC3.6: Use of degenerate points in $T_{\text{v-ad}}$ field for data extraction from images of bar charts, and validated using chart reconstruction. (A) Input images of different variants of simple bar chart. (B) Visualization of the pixels corresponding to degenerate points based on C_l and C_p values of $T_{\text{v-ad}}$ at the corners of bar/bin for and near the center of scatterpoint, (C) The reconstructed charts from the extracted data for the input images.

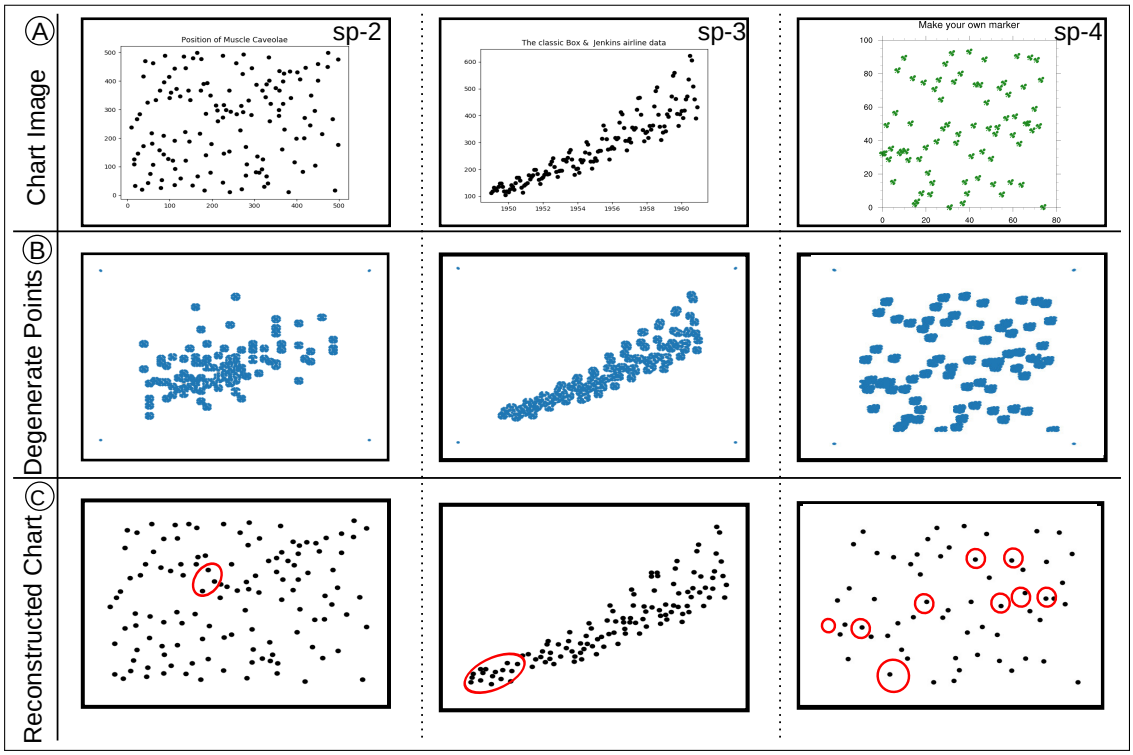


Figure FC3.7: Use of degenerate points in $T_{\text{v-ad}}$ field for data extraction from images of scatter plots, and validated using chart reconstruction. (A) Input images of different variants of simple scatter plot. (B) Visualization of the pixels corresponding to degenerate points based on C_l and C_p values of $T_{\text{v-ad}}$ at the corners of bar/bin for and near the center of scatterpoint, (C) The reconstructed charts from the extracted data for the input images, with red ellipses indicating omission errors.

(omission) error occurs as a result of scatterpoints being clustered together and overlapped on each other. The overlapped scatter plots do not follow any specific cluster patterns such as standard distance or the number of points in a cluster. Hence, For such cases, DBSCAN hyperparameters have to be reconsidered and modified for each computation belonging to different scatter plot images.

We compare our result of data extraction for scatter plot originally listed in Scatteract [18] as shown in Figure FC3.7, `sp-4`. The chart contains unique "clover" marks for scatterpoint notation that attributes to several false positives (type-1 errors) using Scatteract. In this case, the type-1 error occurs due to three different scatterpoints identified at each "clover" mark. We avoid such false positives by detecting the "clover" marks

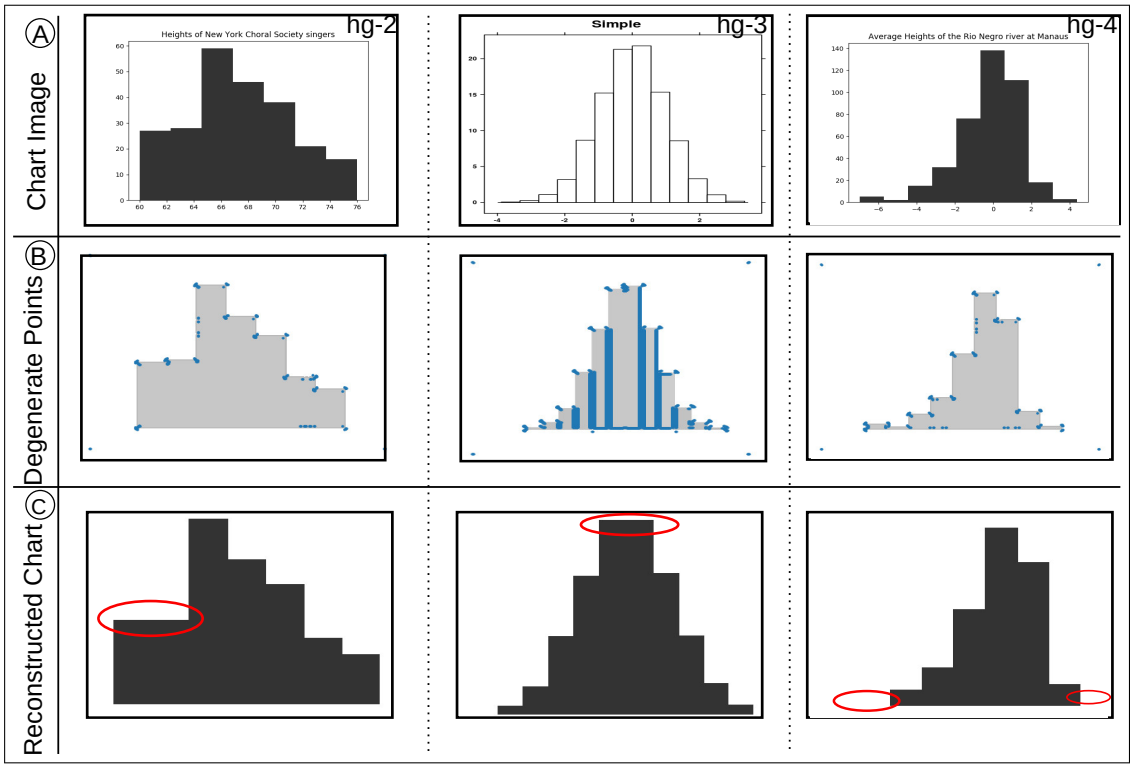


Figure FC3.8: Use of degenerate points in $T_{\text{v-ad}}$ field for data extraction from images of histograms, and validated using chart reconstruction. (A) Input images of different variants of histograms. (B) Visualization of the pixels corresponding to degenerate points based on C_l and C_p values of $T_{\text{v-ad}}$ at the corners of bar/bin for and near the center of scatterpoint, (C) The reconstructed charts from the extracted data for the input images, with red ellipses indicating omission errors.

as centroids of clusters. We change cluster distance parameters "eps" of DBSCAN clustering to include all degenerate points of "clover" mark.

The data extraction process using tensor field analysis performs well on histogram images listed in Figure FC3.8. The errors are highlighted for both histograms in Figure FC3.8, where the insignificant bin height difference is not captured as well as the bin close to the x-axis is missing.

The qualitative error analysis supports the values received for EMD calculation, *i.e.*, higher d_{EMD} value denoting higher error for scatter plots as compared to bar charts as a result of omission errors. The histograms of tailed distributions containing $\sim$0-

Table TC3.1: Error computation using Earth Mover's Distance of distributions of normalized values of original data and reconstructed data belonging to original and reconstructed images, respectively. $d_{\text{EMD}} > 0.10$, in boldface, can be considered relatively high.

Scatter plot		Bar Chart		Histogram	
Figure FC3.2–FC3.7	d_{EMD}	Figure FC3.6	d_{EMD}	Figure FC3.2–FC3.8	d_{EMD}
sp-1	5.4e-2	bc-2	4.4e-3	hg-1	1.9e-2
sp-2	1.1e-2	bc-3	2.6e-3	hg-2	6.3e-3
sp-3	5.5e-2	bc-4	3.0e-3	hg-4	3.9e-2

frequency bins at the tails cause larger errors Figure FC3.8, `hg-4` in comparison to other histograms.

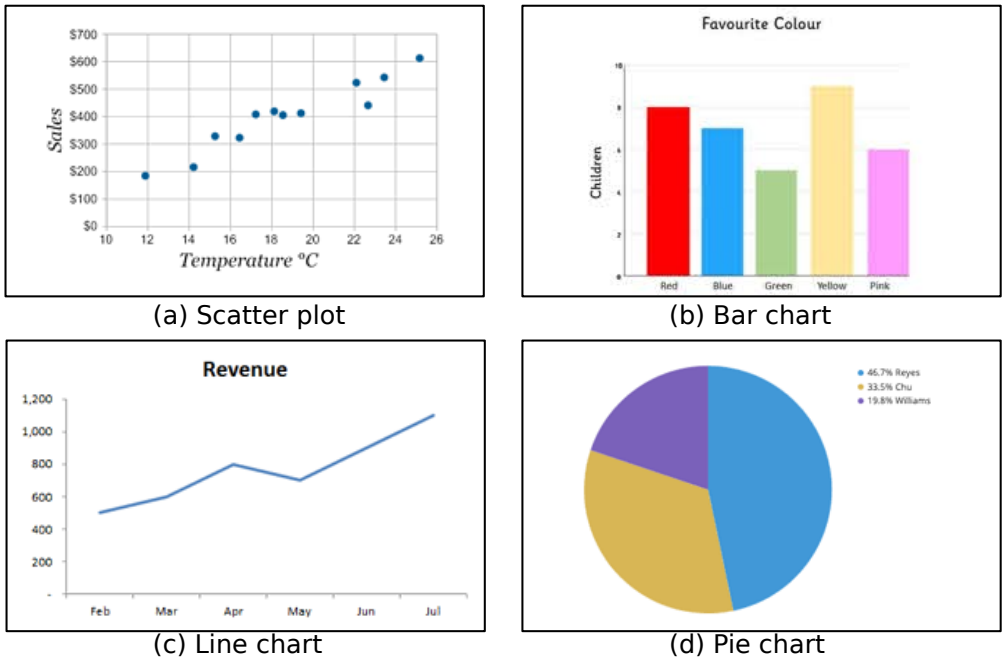
CHAPTER 4

CHART IMAGE CLASSIFICATION AND ANNOTATION

Data can be represented using different visualizations, such as the same data can be plotted using a bar chart as well as a pie chart or a scatter plot. Data representation in charts showcases different aspects and properties of data based on what type of chart has been selected for visualization. In the intelligent mathematics problem-solving system related to high school, the classification of the statistical chart is a key step. Consequently, the classification of statistical charts has become an urgent problem to be solved. Similarly, to deal with the graphical perception of the chart, it is important to know the type of chart in order to follow a correct approach to analyze the same. Figure FC4.1 lists basic chart types introduced at a primary level of education.

The analysis and interpretation of these chart types require different steps and understanding. The reverse engineering on these chart images requires similar steps that are initially used to create the chart from source data. All the chart types require different data handling, plotting, and extraction mechanism. For example, the height of bars denotes the value represented by that particular bar, and the width has no contribution to the evaluation of data. Similarly, a slice of the pie chart shows the value/percentage contribution of different elements.

A Single chart type can visualize the data in various forms, e.g., a bar chart type has multiple representations such as grouped bar, stacked bar, and depending on the



(a) Scatter plot (b) Bar chart

(c) Line chart (d) Pie chart

Figure FC4.1: Basic chart types introduced for chart graphicacy at the primary education level.

orientation assigned to the chart, it can be a vertical bar or horizontal chart. These details and sub-types play a vital role in the extraction process.

Here, our goal is to provide a component that can provide the chart type and chart sub-type for a given chart image. These details are used for further computation on the image. We have trained our models for performing classification tasks on chart images.

4.1 Chart Type Classification

The divergent approach associated with chart creation and analysis for different datasets and requirements needs to be considered while doing reverse engineering operations such as data extraction from the chart. Hence, we have taken into account some of the basic charts that are being widely used through classroom programs in schools. The categories included for our classification model are:

- Bar chart
- Scatter plot
- Line chart
- Pie chart

4.1.1 Dataset for Type Classification

Even though Machine Learning has been in light since the 1950s, major development, problems, and solutions came into the picture in the late 1990s. Extravagant progress in the ML area is due to the availability of larger datasets to perform different tasks. This observation promotes the importance of good quality/quantity data for ML models. Dataset is a collection of similar instances sharing the same properties. For machine learning models to understand how to perform various actions, training datasets must first be fed into the machine learning algorithm, followed by validation datasets (or testing datasets) to ensure that the model is interpreting this data accurately.

CNN-based model for classification requires a large dataset for training. A corpus of chart images with correct tagging of chart types plays an important role in accurate chart type classification. We collected data from different sources such as FigureQA, ReVision, and google search results. To train and test our chart type classifier, we have used the FigureQA [34] dataset, ReVision dataset [13], and Vega dataset [8]. The reason to use three different datasets to train our model comes from including different images possessing different sizes, quality, and representation of statistical plots. The dataset provides high-quality chart images belonging to categories of bars, scatter plots, lines, and pie charts. The variety in the dataset helps ML models to learn features efficiently that can be generalized for a large number of images.

For natural images, augmentation helps in creating large datasets while applying

different transformations to the collected set of images. The transformation can be resizing, rotation, smoothening, etc. For chart type identification, such transformations on images may not contribute much to the dataset due to sparse and structured representation in charts that might change the sense of representation if rotated and might not change any details in case of operations like resizing and smoothening. We have collected approximately 2000 images for training chart type classifier.

4.1.1.1 Preprocessing

The charts from different sources mentioned in section 4.1.1 contain images of different sizes and quality. The input to a fully connected layer is ”flattened”, and then the number of weights is determined by the number of input elements (channel and spatial combined) and the number of outputs. If the size of the input image changes, the number of weights to be trained in a fully connected layer also changes, which requires the whole model to retrain. As we have two fully connected layers at the end part of our model to declare a class of image, we can not perform training using a variable-sized image dataset. To avoid errors while training, we have preprocessed the dataset by resizing images to a fixed size (200, 200) using resize() function from the Python imaging (PIL) library. The function requests hyperparameters like new dimensions and resampling method. We have selected PIL.Image.ANTIALIAS (a high-quality downsampling filter)to get a high precision image. We have another resize option from the OpenCV library as cv2.resize(), which does not provide ANTIALIAS as one of the options for resampling methods, hence compromises image quality.

4.1.1.2 Image labelling

Dataset needs to be in a particular format in order to solve an image classification problem. The dataset should be divided into two folders, one for the train set and the

other for the test set. The training folder needs a CSV file that contains the names of all the training images and their corresponding true labels/ ground truth. The CSV file in the test set is different from the one present in the training set. This test set CSV file contains the names of all the test images, but they do not have any corresponding labels. The model will be trained on the images and labels present in the training set, and the label predictions will happen on the testing set images.

Based on the discussed dataset requirements, we separate all training images in different directories based on their chart type. As we do not have pre-labeled data, this task is performed with a python script. The script requires a path of image files belonging to the bar, scatter plot, line, and pie chart stored in different folders (named as corresponding chart type). Our labelling script reads through each image from different folders and adds the image path in the label.csv file, where the label of the image is stored with the image path. The image label is decided on the basis of the folder name in which the image was placed. Hence, the way in which images are stored in directories named with chart types is important to our classifier. The directory structure to store images is as shown in Figure FC4.2

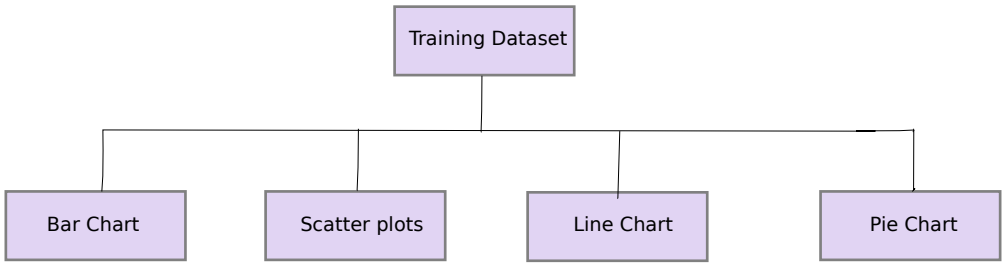


Figure FC4.2: Training dataset and directory structure

4.1.2 Inception Model

The “Inception” micro-architecture was first introduced in their 2014 paper [1]. We use a pre-trained Deep Learning Convolutional Neural Network model called Inception. The Inception network was an important milestone in the development of CNN

classifiers. Prior to its development as a module to GoogleNet, most popular CNNs just stacked convolution layers deeper and deeper, hoping to get better performance. This model has been pre-trained for the ImageNet Large Visual Recognition Challenge using the data from 2012, and it can differentiate between 1,000 different classes. We retrain the model for our chart images by providing labeled images of different types of charts collected from FigureQA and ReVision datasets. The prediction is provided as a probability of the source image belonging to a particular class.

4.1.2.1 Architecture

The image in Figure FC4.3 is the "naive" inception module. It performs convolution on input, with three different sizes of filters (1x1, 3x3, 5x5). Additionally, max pooling is also performed. The outputs are concatenated and sent to the next inception module. The combination of multiple inception modules creates GoogLeNet. Inception v2 and Inception v3 upgrade on the initial version, which increased the accuracy and reduced the computational complexity.

The basic architecture of Inception was introduced by Szegedy et al. [1] that is represented in Figure FC4.4. The Inception network is complex, heavily engineered, and the first one with batch normalization. Inception-v3 is a successor to Inception-v1.

4.1.3 Limitations of Pre-trained Models

As we know, CNN is a machine learning algorithm for machines to understand the features of the image with foresight and remember the features to guess whether the name of the new image fed to the machine. ImageNet challenge brought out the best classification models such as GoogleNet, ResNet, AlexNet, etc. The datasets comprised approximately 1 million images and 1,000 object classes. As these models are part of different machine learning libraries, it is recommended to use the pre-trained models

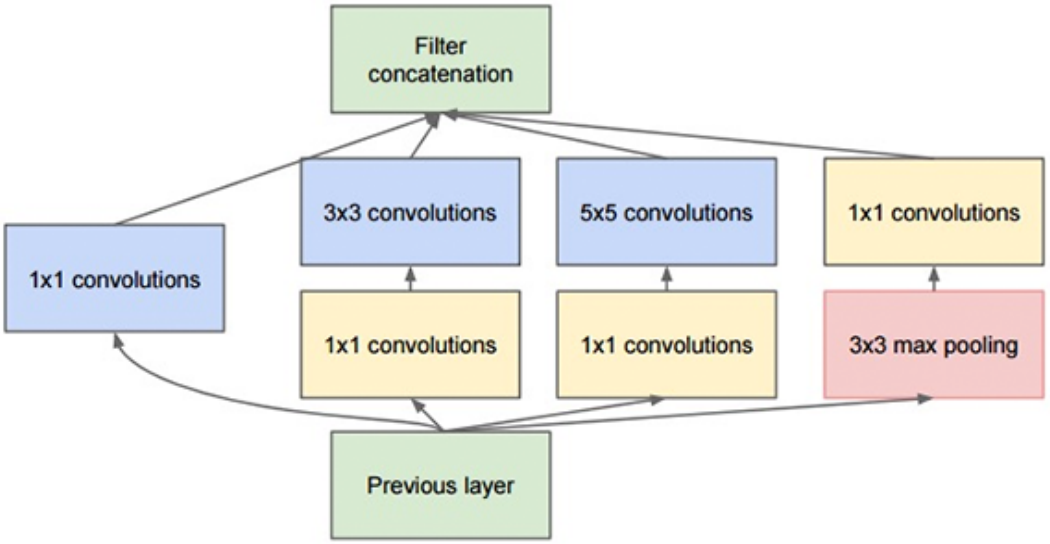


Figure FC4.3: Naive Inception module described by Szegedy et al. [1].

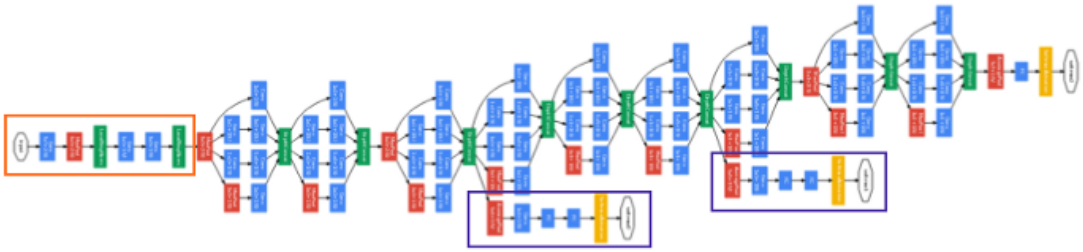


Figure FC4.4: GoogleNet Architecture described by Szegedy et al. [1]. The stem, marked by the orange box, performs preliminary convolutions. The auxiliary classifiers are marked by purple boxes, and the remaining structures are inception modules.

in order to save time and effort. Transfer learning and fine-tuning these models have made research in the machine learning area much easier and less time-consuming, as training the model might take an hour to months. Also, transfer learning removes the requirement of large data for training the model.

The reason behind not using transfer learning for our classifier directs us to a fundamental definition of transfer learning. "Transfer learning makes use of the knowledge gained while solving one problem and applying it to a different but related problem."

The definition stands for usage of transfer learning when there is a relation between our dataset and the pre-trained model. For example, we can use a different pre-trained model over ImageNet dataset with Cifar10,100 datasets, but we can not use the pre-trained model of ImageNet with the biomedical images because ImageNet does not contain images belonging to the biomedical field, so in such case, we should train the model from scratch over biomedical images and then we can uses this model for transfer learning. Similarly, we haven't seen any description of ImageNet dataset containing statistical plots or any chart image data. Hence we need to train these models from scratch, which limits us back to training with a sufficiently large dataset to avoid overfitting. Training these models from scratch will also lead us to high time consumption.

4.2 VGGNet Classifier

As our images are sparse and can be classified mostly based on the shape of objects in the image, instead of starting over from scratch with a model we don't know much about, we have tried to create our own model to classify the type of chart that can also be implemented for another task named as sub-type classification illustrated in 4.3. As the initial hidden layers of any CNN model try to capture features such as identifying different components and then recognizing the shape created by these components, we have added four convolutional layers, each being followed by pooling layers.

VGGNet was introduced in the same year 2014 as GoogleNet, and was selected as the second-best model for ImageNet challenge [2]. The network architecture is known for its simplicity and has famously been used for tasks like object detection and segmentation. Hence, we use VGGNet architecture to implement our classifier that can be further extended for detecting and locating chart objects in image to extract clean canvas. VGGNet uses only 3×3 convolutional layers stacked on top of each other in increasing depth. VGGNet models are named VGG11, VGG13, VGG16 with suffix 11, 13, 16, etc., where the numeric suffix stands for the number of weight layers in the network.

4.2.1 Architecture

Following traditional CNN-based architecture, VGGNet architecture uses a stack of convolutional layers for network creation. CNN image classifier takes an input image, processes it, and classifies it under certain categories. We use VGG architecture with the kernel size of (5,5) as shown in FC4.5, containing convolutional layers and pooling layers followed by fully connected layers, also known as dense layers [35] [36].

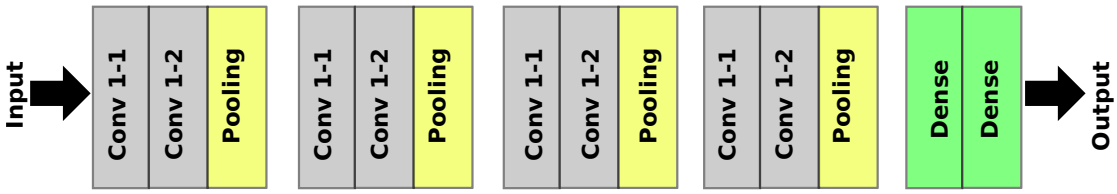


Figure FC4.5: Our VGGNet inspired classification model with convolutional layers stacked along with max-pooling layers with tailing fully connected layers shown in LeNet style. Our model has total of 10 layers with 2 convolutional layers before each pooling layers similar to VGG13 [2] architecture with filters (3x3) and (5x5).

Convolution is the first layer to extract features from an input image. Convolution preserves the relationship between pixels by learning image features using small squares of input data, also known as kernels/filters. For an input image X and filter f, the convolved image (Z) can be represented as:

Z=X * f

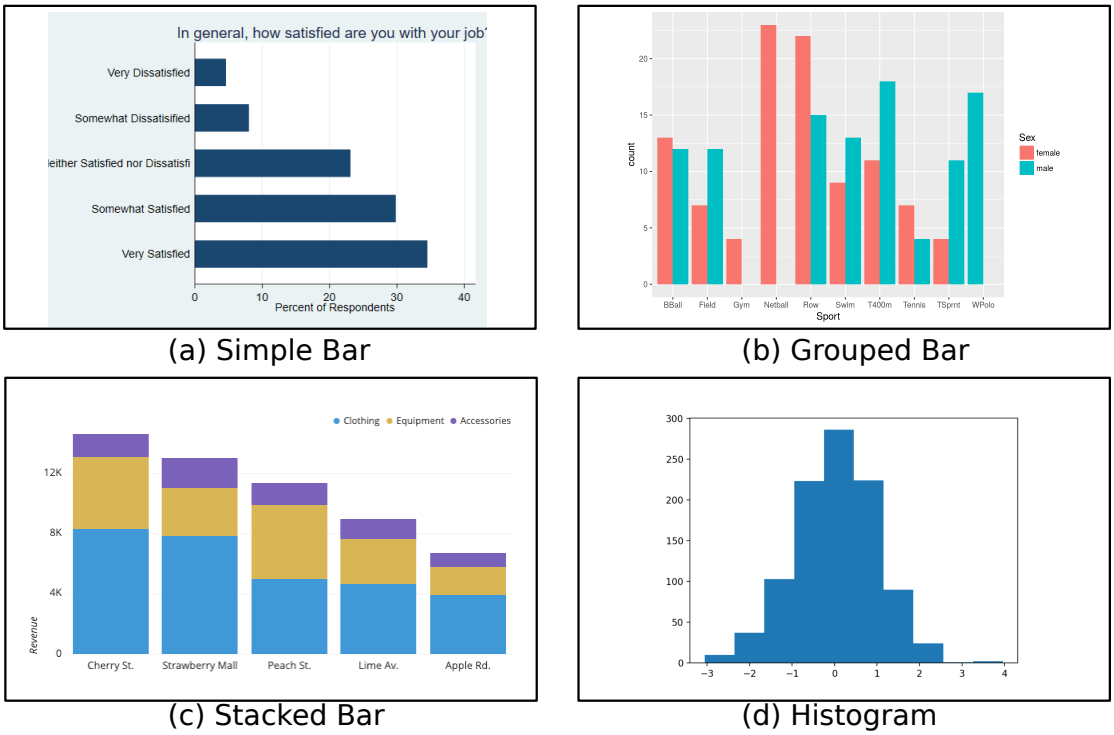
It is a mathematical operation represented by * that takes two inputs, an image matrix, and a filter or kernel. We use rectified linear unit (ReLu) (Eqn 4.1) as an activation function in layers other than the last fully connected/dense layer. The function returns 0 if it receives any negative input x but return input x as it is for any positive input x.

$$f(x) = \max(0, x) \tag{Eqn 4.1}$$

The pooling layers section would reduce the number of parameters when the images are too large. Spatial pooling, also called subsampling or downsampling, reduces the dimensionality of each map but retains important information. Spatial pooling can be of different types such as max pooling, average pooling and sum pooling. We use max pooling, the most commonly used type for our CNN architecture. The max-pooling takes the largest element from the rectified feature map. The fully connected layer provides the class label for a given input image, and we use the softmax activation function (Eqn 4.2). The output from the last pooling layer needs to be flattened and passed through the fully connected layer to get a class label.

$$f_j(z) = \frac{e^{z_j}}{\sum_k e^{z_k}} \tag{Eqn 4.2}$$

In softmax activation function (Eqn 4.2), all the z_i values are the elements of the input vector and can take any real value. The the normalization term at the bottom which ensures that all the output values of the function will sum to 1, thus constituting a valid probability distribution.



(a) Simple Bar (b) Grouped Bar

(c) Stacked Bar (d) Histogram

Figure FC4.6: Examples of different sub-types of the bar chart with horizontal and vertical orientation.

4.3 Chart Sub-type Classification

The problem of chart type classification introduces another step to be taken for sub-type classification as the data represented using one class of chart type can still be different in terms of design space and visual elements. For example, in many datasets, visualization using three simple bar charts can also be done with the help of a single grouped bar chart or stacked bar chart. Another example associates us with the chart orientation, *i.e.,* the horizontal and vertical bar objects are both highly used in the data visualization domain. Some sample bar charts belonging to different sub-categories and orientations are shown in FC4.6.

Our algorithm requires pre-existing characteristics of different charts for fine-tuning its data extraction. Hence, the need to identify the sub-type is a requirement for our

reverse engineering task.

4.3.1 Dataset

The FigureQA dataset is not currently inclusive of all types of bar charts, even the ones which are commonly used. Hence, we have created our own dataset for sub-type classification pertaining to only bar chart categories. The image corpus contains 1000 images and is created with google downloaded chart images that were separated into different folders for the labelling process. We consider seven total categories, including horizontal and vertical orientation of simple, grouped, stacked bar. We also include histograms as bar sub-type as design space among various tools such as Google Sheets and Microsoft excel ®allow users to plot histograms with the help of column charts due to geometric similarities between bin and column/bar. The dataset also contains images of scatter plot, line, and pie chart in the ”Other” category that helps our classifier to predict if the given image does not belong to bar chart types and discards the image. We have downloaded approximately 1000 images of various types of bar charts to train chart sub-type classifier.

4.4 Chart Annotation

Image Annotation is the process of labelling data in the various mediums of images, text, or video. The labels are usually predetermined by a machine learning engineer or computer vision scientist based on the requirements and are selected to provide the computer vision model information on objects depicted in the image. Hence, annotation becomes a key step to generate a training dataset for computer vision tasks such as object detection and segmentation. Image annotation task is performed in various ways like bounding box, polygon, line, and point annotation. A bounding box is the most commonly used and simplest annotation method. This requires labelers to draw a box to

enclose the region of interest in an image. This annotation method is often used in training data generation for object classification, localization, and detection models. A wide range of annotation tools is available on the internet to perform requirements-based or domain-based annotation including, LabelImg, Labelbox, VGG Image Annotator, etc.

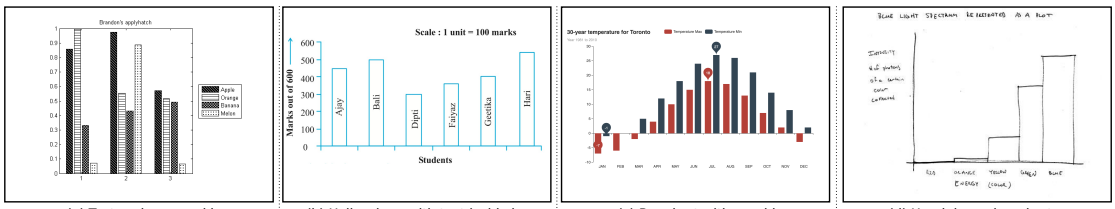
Image annotation is a manual task and requires user interaction to select ROI and label it accordingly. For chart images, manual marking and annotation of bounding boxes for ROIs have been widely used [5, 20]. Hence, the selection of an annotation tool depends on the steps to be followed to annotate the image components and user interface of the tool. As our chart images are sparser as well as clean and structured, the requirement associated with the annotation tool is only to draw and adjust the bounding box around the object, assign a label to selected ROI and save the annotation in simpler formats to process it further if needed.

Figure FC4.7: Image annotation on an image of bar chart using LabelImg tool to locate chart components in the given image.

LabelImg [37], a Python tool with a user-friendly graphical interface based on Qt, helps us to achieve the final annotation for our chart images by marking and annotating chart components. We label different chart components as canvas, x-axis, y-axis, x-

labels, y-labels, legend, title, x-title, and y-title based on their position and role in the visualization. The tool allows saving annotation in XML files in PASCAL VOC format as well as a text file in YOLO format. The XML annotation stores information about the bounding box of each label by saving minimum and maximum x and y coordinates. We use XML annotation to extract the canvas region as well as for text localization. The labels assigned to different components can be seen in the right tab of the tool. The selection of a label from the right tab highlights the component in the uploaded image, and hence, the bounding box can be readjusted for the selected component if needed. A curated dataset containing 1500 images has been annotated using LabelImg for charts. The GUI and annotations using LabelImg are shown in Figure FC4.7.

Annotated canvas region is considered as an input for our preprocessing module that performs morphological operations like dilation and erosion on the region to remove grid line and overlaid legends, etc. This process is named as canvas extraction process that gives image having only graphical objects such as bars, scatter points, etc. The output is used for the tensor voting computation module. Annotation of text components is used with text detection and recognition models to increase model accuracy.



(a) Textured grouped bar (b) Hollow bars with text inside bar (c) Bar chart with graphics (d) Hand drawn bar chart

Figure FC4.8: Bar charts drawn with different design formats, that fail with either our chart classifier or our data extraction algorithm.

4.5 Experiments

The annotations saved in XML files help us in improving the canvas extraction process and hence enable our workflow to perform chart digitization for high-resolution

images with $\sim$100% accuracy.

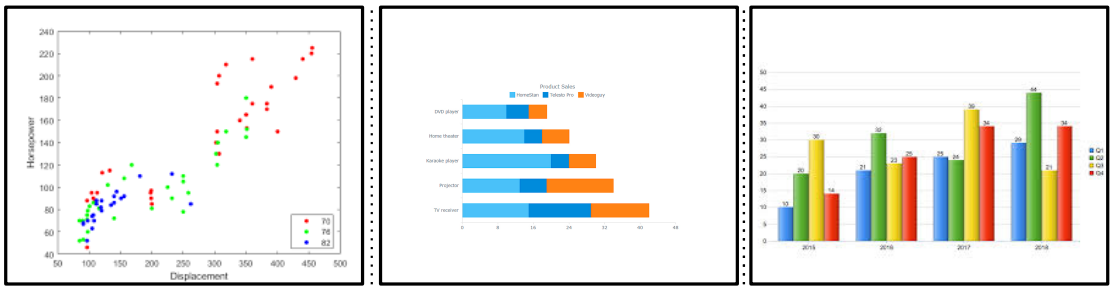
For testing our chart-type classifier, we prepared a test dataset that contains images of bar charts, scatter plots, line charts, and pie charts. As our main focus is on images of bar charts and scatter plots, we have collected 100 images of bar charts and observed that the classifier labels $\sim$94% of images correctly as a bar chart. Similarly, we have collected 100 images of scatterplots and observe $\sim$81% accurate labels identified by our classifier.

Our chart type classifier works with $\sim$93% accuracy and sub-type classification models work with $\sim$90% accuracy for bar chart images, where our training set covers several variants. The classification fails for certain cases where the standard plotting approach is not used. For instance, the textured grouped and hollow bar charts shown in Figure FC4.8 either do not get classified correctly in our image classifier or fail in our chart object extraction process. We are working on including images of bar charts from various design spaces as well as hand-drawn charts to improve our classifier for sub-type classification.

CHAPTER 5

MULTI-CLASS AND MULTI-SERIES CHARTS

Uni-variate data analysis is less efficient in the light of multivariate data collections being the norm. Hence, while simple bar charts and scatter plots serve the purpose of simpler analysis, more complex forms of charts are needed in the current scenario. Most charts available from the sources of chart images also tend to be more complex than simple charts, where the latter is used only in the early stages of chart graphicacy.



(a) Scatter plot (b) Stacked Bar (c) Grouped Bar

Figure FC5.1: Examples of multi-series/multi-class bar chart and scatter plot.

One of the widely used complex charts is where the data is plotted juxtaposed or stacked in a single chart to get more clarity and context. The plots that contain information of two different data points or features are called multi-series charts and are used most frequently for comparative studies, for example, a company's revenue vs. its costs over time or New users per day vs returning users per day, etc. These charts are often used to demonstrate the trends across different data series in a single visualization that can elaborate on the relationship among these data series. The examples of multi-series

charts belonging to the bar chart and scatter plot are listed in Figure FC5.1.

In this chapter, we aim to extend the tensor voting analysis explained in chapter 3 for multi-class/multi-series bar charts and scatter plots by improving the rule-based extraction method to capture data from multi-series charts. Many previous attempts, such as [4], have limited the data extraction process for simple and grouped bar charts, leaving the stacked bar chart unattended. Our goal is to suggest a more generic approach for the chart digitization process for a larger variety of charts.

5.1 Grouped Bar Chart

As the feature set grows, a simple chart is not sufficient in representing multiple features/ grouped/ sub-groups. Then, the simple charts bring in complexity to represent multiple features or classes/groups. For instance, bar charts are plotted with the help of separate bars representing each of the sub-groups, and the sub-groups are colored, shaded, or textured differently to distinguish between them. Like a simple bar, the height of each bar depicts value, and the color/shade of each bar in the visualization is associated with the group information from the source data. The questions about the bars belonging to certain sub-groups are then completely answered using the legend provided in visualization. This bar chart category is also known as a clustered bar chart, multi-set bar chart, or grouped column chart. The side-by-side arrangement of bars of grouped categories makes the interpretation of the differences inside a group and even between the same category across groups easier.

However, the grouped bar charts can become complex if the creator of the chart encodes too many classes in a single visualization. Thus, very few variables, but (> 1), are usually used in such charts. This limitation also indirectly supports the redesign of complex charts to simpler charts by breaking down the composite visualization.

5.2 Stacked Bar Chart

The stacked bar chart is another way to visualize the sub-groups in which the sub-groups are stacked on the same bar. Similar to a grouped bar chart, different colors/shades are used for different sub-groups. This representation is useful, where the goal is to encode the total size of groups and the proportion between groups. The total height or length of the bar shows the total size, and different colors or shadings are used to indicate the relative contribution of the different sub-groups. These stacked bar chart attributes are the reason behind the profound use in financial reports and other corporate settings. Similar to a grouped bar chart, a stacked bar chart can also become intimidating if the encoding is not correct or if the data has a large number of sub-groups.

5.3 Multi-class Scatter Plot

Similar to simple scatter plots discussed in chapter 3, multi-class scatter plots also demonstrate mapping between the x- and y-axes values. However, this representation extends the simple scatter plots by additionally encoding class/group information with each scatterpoint using the color-mapping selected for classes. Multi-class scatter plots are mainly used to organize the display of the clustering in datasets, correlation, etc. For example, multi-class scatter plots are highly used in machine learning to visualize multi-class classification. Multi-class scatter plots can be treated as several overlays of simple scatter plots using the same coordinate system and units and scales on the axes.

5.4 Data Extraction

The data extraction algorithm for multi-series charts is similar to simple charts discussed in chapter 3 till DBSCAN clustering and baseline computation using cluster

centers. However, as CIE-Lab color space is highly recommended for the perceptual understanding of colors, we convert our RGB image to CIE-Lab color space. The final image is processed for gradient tensor for structure tensor computation. The values used for normalizing in CIE-Lab follow range as below:

- L has values in [0, 100]
- a has values in [-127, 127]
- b has values in [-127, 127]

We proceed with structure tensor T_s to compute voting tensor T_v and voting tensor after anisotropic diffusion $T_{\text{v-ad}}$ at each pixel. Degenerate points are filtered from the set of non-zero tensors based on the saliency value. The final set of degenerate points are processed through DBSCAN for clustering. Once the clusters and corresponding centers are identified, the computation in the case of the stacked bar chart specifically changes as the total bar/column height are no longer provides value for groups. We include rule-based steps to be taken if the bar chart is classified in grouped or stacked categories which includes the following steps [36]:

- Pick the legend colors.
- Identify the number of classes depicted in the chart
- Calculate bar/stack heights.

In the case of scatter plots, the cluster centers work as the xy-coordinates of scatter-point in pixel space. The pixel to data mapping based on legend and classes is mention in section 5.4.1.

5.4.1 Legend Mapping

The analysis of multi-series or multi-class visualization is highly dependent on the legend as the color encoding point out to the class a graphical object belongs to. As we annotate our data in the initial stage of the algorithm, we mark the legend on the image and save the information in XML annotation.

We render through the XML and get the rectangular/square bounding box coordinates in terms of maximum and minimum of both x and y coordinates in pixel space to get the legend component. We process the legend bounding box through morphological operations similar to the canvas extraction module discussed in chapter 3 to get the RGB value of color from legend [36] [35]. The number of colors in legend also denotes the number of classes depicted in the original image. This can be cross verified with the text detected using OCR, *i.e.,* if the two colors are found in legend, then the text detection on the legend module should return two class labels.

As of now, the extracted data is in pixel space; we need to map the pixel location with the image data itself. We perform this task using our annotations done in the primary phase of the algorithm and text detected through OCR. We normalize the center coordinates and the coordinates of bounding boxes of labels that encapsulate the designated text region. This gives us the exact data being represented by the original image.

5.5 Experiments

For experiments, we create a test dataset for the bar chart and scatter plots by collecting images synthetically generated as well as downloaded from the internet. We have generated the synthetic test dataset using the `matplotlib.pyplot` library in python [33]. The results are shown in Figures FC5.2–FC5.7, which also show the intermediate ones for each module of our algorithm using a subset of test case images.

We have uploaded our results for more images of bar charts on our demo page BarChartAnalyzer.

For error analysis, we perform a qualitative comparison of the source image with the reconstructed chart, as these images are downloaded from the web and do not have the source data for comparison. We can also look through the final data extracted post-OCR to get a better idea.

For bar charts, we test our algorithm using different types of bars, *i.e.,* simple, grouped, and stacked bar charts. The orientation of charts impacts the rule-based extraction method; so, we also include both horizontal and vertical bar in our test cases.

For scatter plots, we have used plots having scatterpoint of different sizes. The variation in size allows us to see the various patterns depicted by the degenrate points for a small geometric object. The images of both bar and scatter plots are of high quality.

5.5.1 Results

The degenerate points in the case of grouped bar chart show the same patterns as a simple bar near the corners of the bar/column as shown in Figure FC5.4, row B, (ii). Whereas, for a stacked bar chart, tensor field analysis provides degenerate points at the corners of the bar as well as at the junction of each stack as shown in Figure FC5.4, row B, (iii). This establishes the association of tensor voting with perceptual grouping.

We observe that the extracted values after scaling pixel space data to original dimension using OCR have numerical precision errors predominantly. Hence, to compare the difference between the extracted values and the source values, we compute the normalized Mean Absolute Error (nMAE), and the Mean Absolute Percentage Error (MAPE) for the synthetic images, which are bounded in [0,1]. MAPE is commonly reported

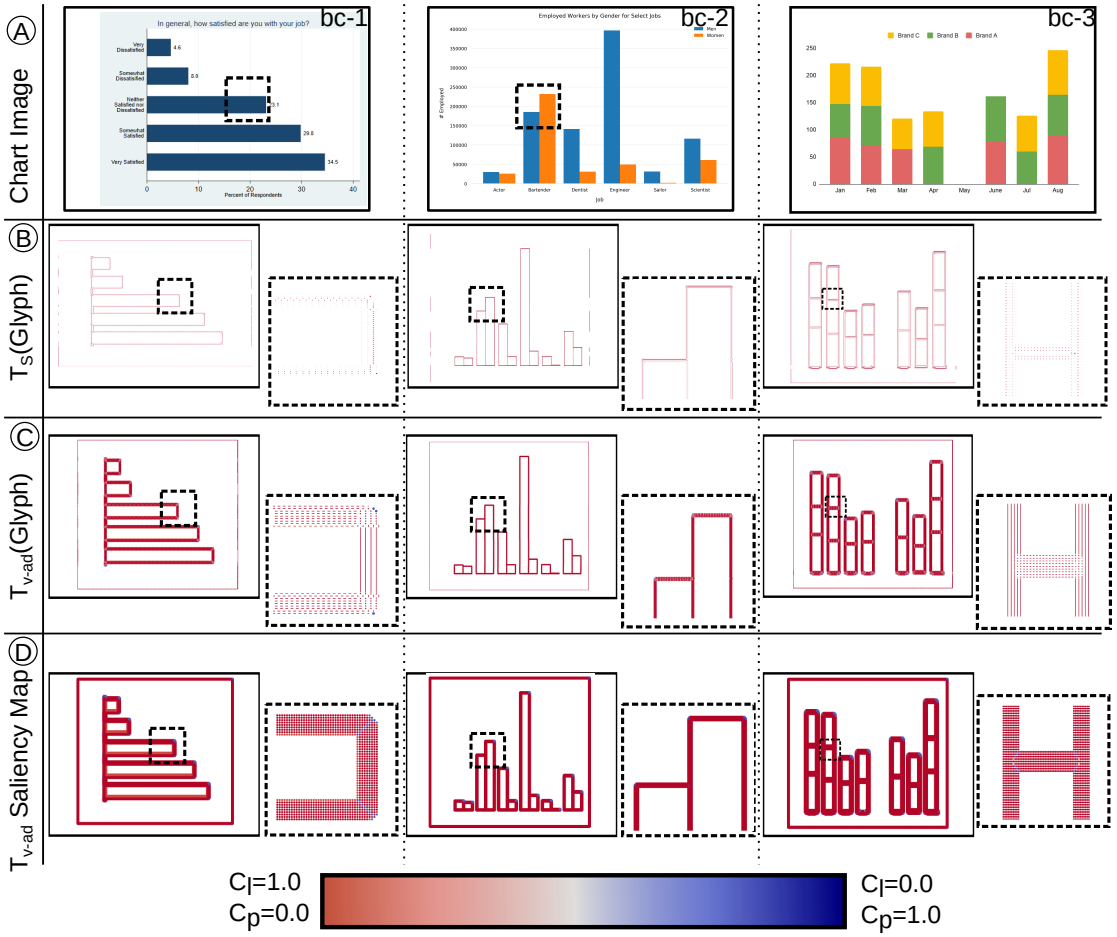


Figure FC5.2: The tensor fields computed from images of charts using our approach. (A) Input images of multi-series bar charts. Tensor fields computed on the images, visualized using glyphs and colored by saliency values, include (B) structure tensor T_s, and (C) tensor voting field of T_s after anisotropic diffusion $T_{\text{v-ad}}$. (D) The saliency values of $T_{\text{v-ad}}$ visualized using dot plots. The coolwarm color mapping associated with corresponding C_l and C_p saliency values, used in the visualizations, is shown in the colorbar.

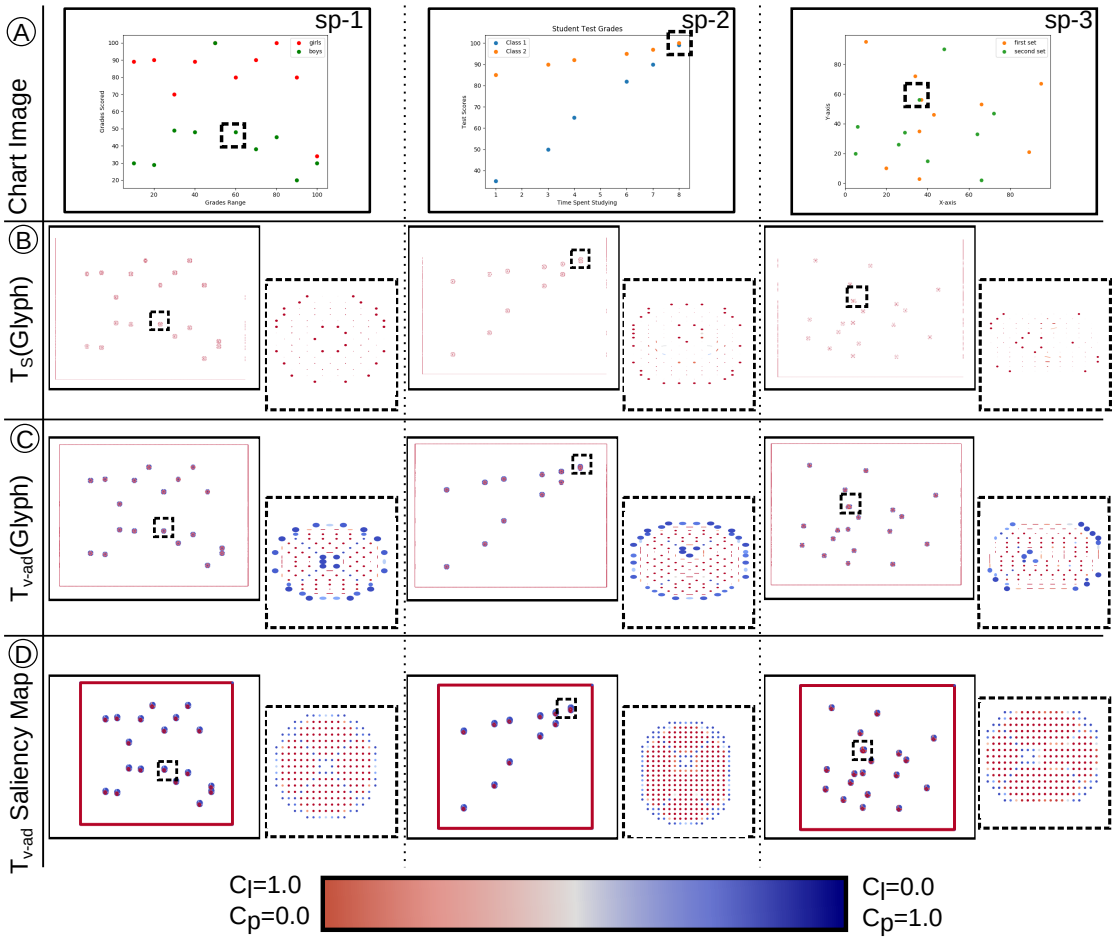


Figure FC5.3: The tensor fields computed from images of charts using our approach. (A) Input images of multi-class scatter plots. Tensor fields computed on the images, visualized using glyphs and colored by saliency values, include (B) structure tensor T_s, and (C) tensor voting field of T_s after anisotropic diffusion $T_{\text{v-ad}}$. (D) The saliency values of $T_{\text{v-ad}}$ visualized using dot plots. The coolwarm color mapping associated with corresponding C_l and C_p saliency values, used in the visualizations, is shown in the colorbar.

Figure FC5.4: Use of degenerate points in $T_{\text{v-ad}}$ field for data extraction from images of multi-series bar charts, and validated using chart reconstruction. (A) Input images of charts. (B) Visualization of the pixels corresponding to degenerate points based on C_l and C_p values of $T_{\text{v-ad}}$ at the corners of bar/bin for and near the center of scatterpoint, (C) The reconstructed charts from the extracted data for the input images.

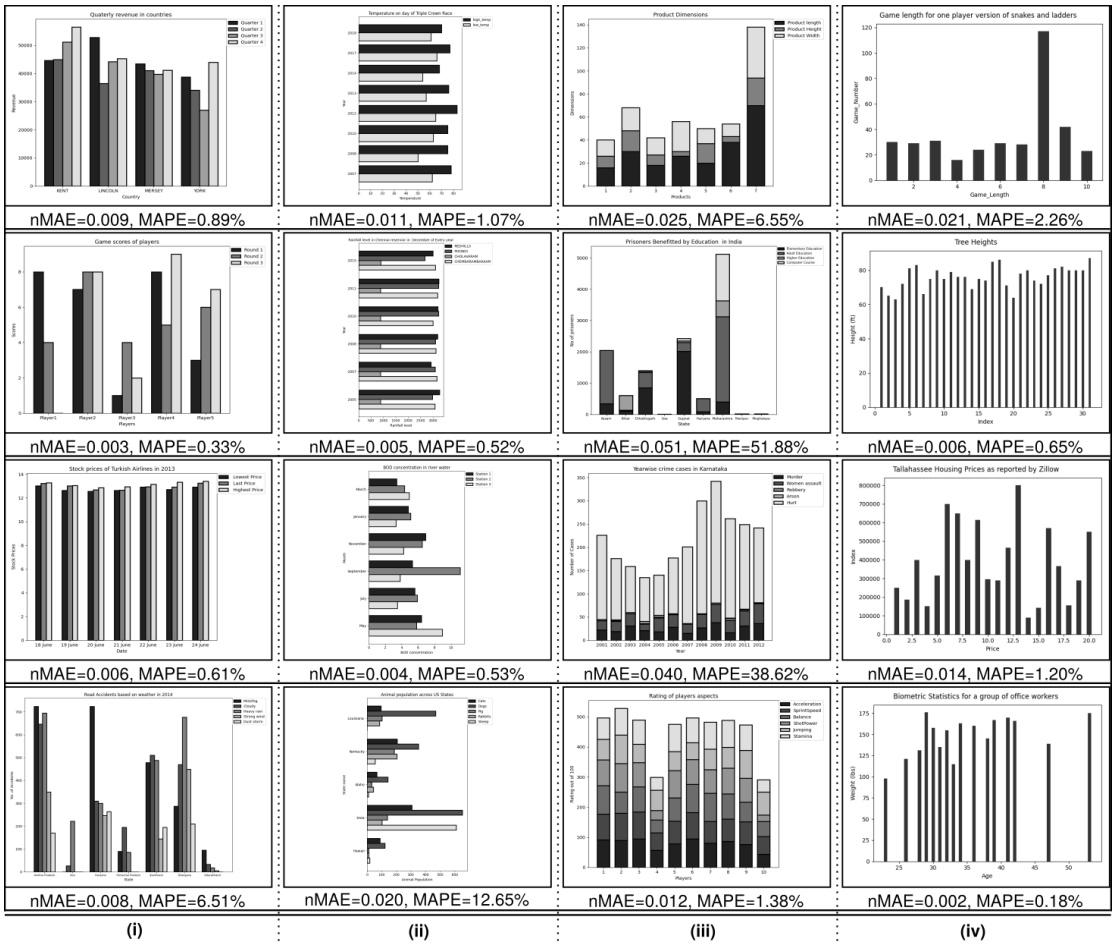


Figure FC5.5: Reconstruction of synthetically generated bar chart images with their error evaluation in normalized mean absolute error (nMAE) and mean absolute percentage error (MAPE).

in a percentage format. MAPE is augmented in the case of missing extracted data in grouped bar charts Figure FC5.5(ii) and stacked bar charts Figure FC5.5(iii) owing to relatively short bars or bar segments. For N data items with source data value x_i and its corresponding extracted value $x_i^{(e)}$,

$$\text{nMAE}=\frac{\sum_{i=1}^{N}|x_i-x_i^e|}{\sum_{i=1}^{N}x_i}; \text{ and } MAPE = \frac{1}{N}.\sum_{i=1}^{N}\left|\frac{x_i-x_i^{(e)}}{x}\right|.$$

In our representative examples in Figure FC5.5, we observe relatively low nMAE values. Histograms are not included in this analysis as the source and extracted data in its case is a frequency table, different from a data table in the case of bar charts.

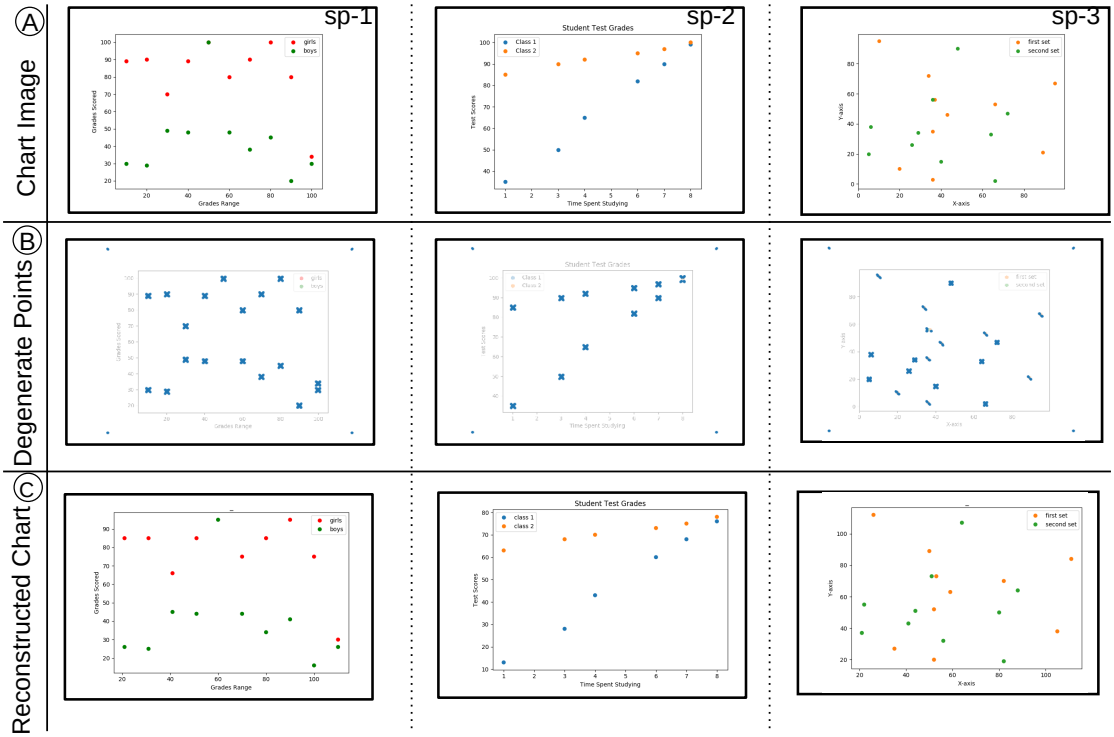


Figure FC5.6: Use of degenerate points in $T_{\text{v-ad}}$ field for data extraction from images of multi-class scatter plots, and validated using chart reconstruction. (A) Input images of charts. (B) Visualization of the pixels corresponding to degenerate points based on C_l and C_p values of $T_{\text{v-ad}}$ at the corners of bar/bin for and near the center of scatterpoint, (C) The reconstructed charts from the extracted data for the input images.

As Figures FC5.3–FC5.6 show our workflow for scatter plot images generated synthetically, we have also performed experiments on images of scatter plots downloaded from the internet. Figure FC5.7 shows the different steps for both simple and multi-class

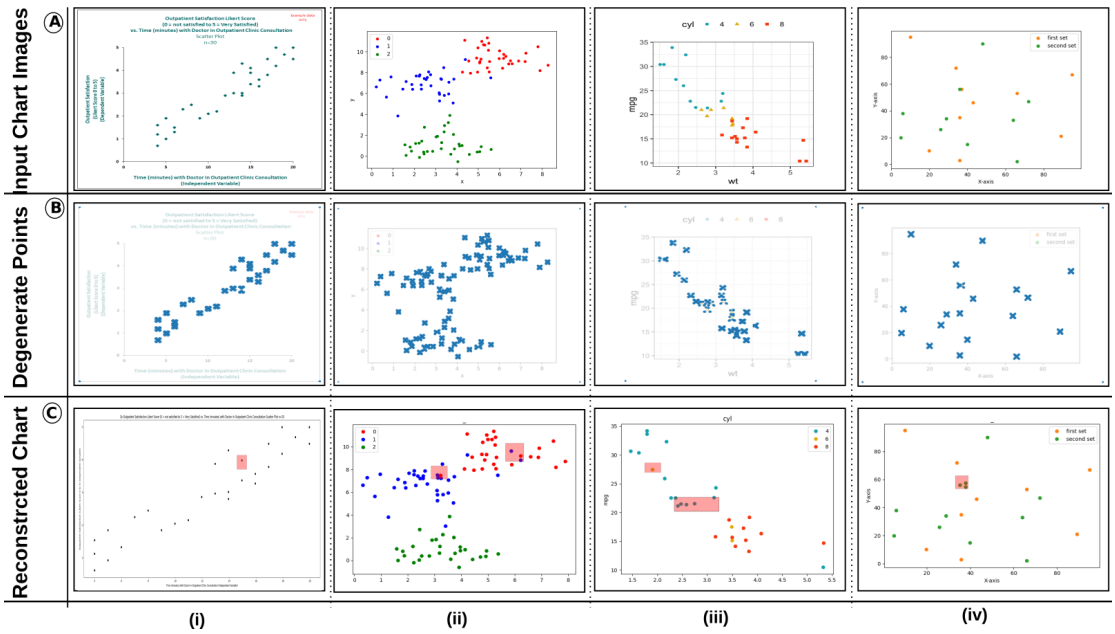


Figure FC5.7: The tensor field computation steps for scatter plot images downloaded from internet (A) Input images of scatter plots. (B) Visualization of the pixels corresponding to degenerate points based on C_l and C_p values of $T_{\text{v-ad}}$ near the center of scatterpoint, (C) The reconstructed charts from the extracted data for the input images.

scatter plots. We observe that the overlapping points in scatter plots do not get extracted accurately, as only perceptually visible scatter points are extracted. At the same time, we observe that the human eye can detect partial overlaps; however, our tensor field is not able to extract the overlapping points as multiple points. Hence, we observe *omission errors*. The degenerate points in case of overlapping points in simple scatter do not provide any information on the points belonging to different data entries. However, overlapping points that belong to different classes provide two sets of degenerate points that are distinguishable by distance during our visual analysis step.

The degenerate points in the case of scatter plots show similar patterns for the same-sized scatterpoints. The patterns or arrangement of these degenerate points for a bigger scatterpoint are slightly different. This can be compared in Figure FC5.6, row B, (i) and (iii) where image (i) is plotted with default size provided by matplotlib library whereas image (iii) is generated with a scatterpoint size 20.

In terms of quantifying the error in our data extraction, we use synthetic datasets for both simple and multi-class scatter plots. We plot the data using `matplotlib`, a Python plotting library, extract the data table and reconstruct the image. We compute the Pearson's correlation of synthetic datasets and their extracted counterparts. We have reported the differences in correlation coefficient *r* for simple scatter plots in Figure FC5.8, and the same for multi-class scatter plots in Figure FC5.9. We observe that the errors in correlation coefficients are proportional to the density of scatter points in the plot. In the case of multi-class scatter plots, the errors in correlation coefficients are additionally proportional to the density of points in regions where both classes overlap in the image. We observe that comparing correlation coefficients in original and reconstructed charts helps in comparing the overall appearance of the charts, which is more significant for text summarization than exact data extraction.

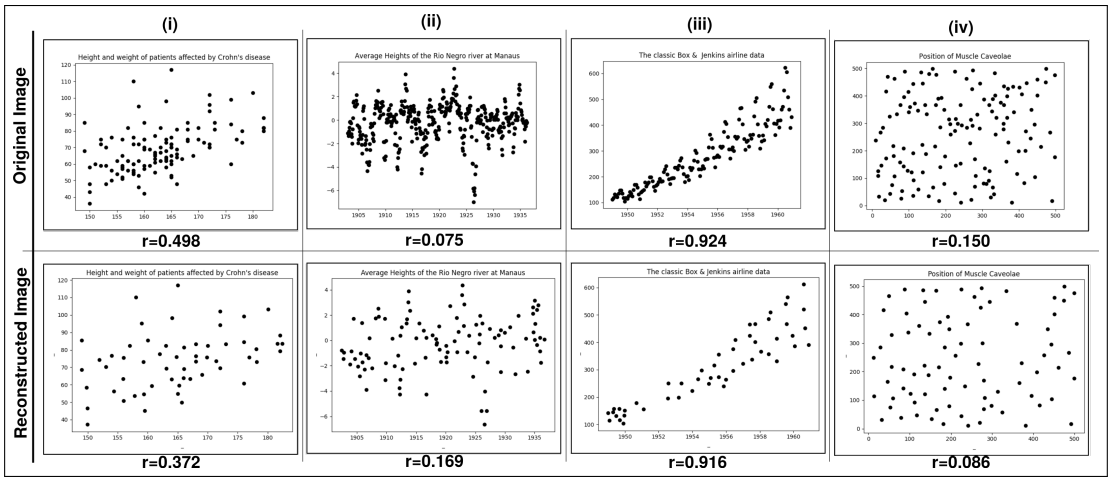


Figure FC5.8: Correlation coefficient (r) values in both original and reconstructed images of simple scatter plots.

However, similar to simple scatter plots, our data extraction algorithm for multi-class scatter plots suffer from omission errors. If the scatterpoints that belong to the same class overlap, the data extraction process will not be able to distinguish all the points. Thus, the consequent exclusion of the obscured points leads to omission errors. Omission errors in scatter plots lead to errors in the correlation coefficient computed

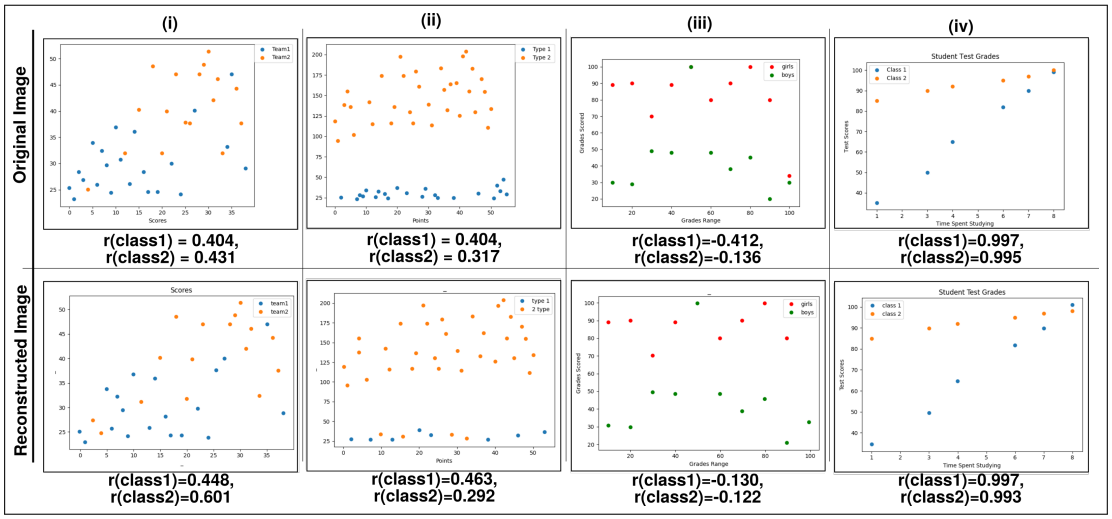


Figure FC5.9: Correlation coefficient (r) values in both original and reconstructed images of multi-class scatter plots.

from the extracted data table. For such cases of overlapping scatterpoints of the same class, a component/model that can capture repetitive patterns like scatterpoint shape needs to be implemented to capture overlapping scatterpoints that do not plot standard patterns on images.

We compare our work with Scatteract where the data extraction success rate is maximum of 89.2% with F_1 score > 0.8. In our experiments, we get a continuous uniform distribution for $F_1 \geq 0.4$, unlike Scatteract, i.e. very low and very high values. Hence, we relax the constraint appropriately to $F_1 > 0.5$, then Scatteract with the use of RANSAC regression for mapping pixel-to-chart coordinates has 89.5% success rate for simple scatterplots for procedurally generated ones, 78% for simple scatterplots from the web, and with other regression methods has 70.3% at its best. In comparison, our method has 73.3% success rate for simple scatterplots, which improves to 93.3% for $F_1 > 0.4$. We can likewise improve the success rate of our method by refining the pixel-to-chart coordinate mapping using RANSAC regression.

CHAPTER 6

DISCUSSION

Through this chapter, we want to address the limitations as well as the different factors that influence the results of our tensor field computation and analysis. As tensor voting is used for perceptual grouping, we experiment with CIE-Lab color space for tensor field computation. We also try to cover various aspects of the design space in our experiments. As all our tensor fields at each pixel are dependent on neighborhood changes, it becomes important to explore the various patterns that occur as a result of a change in either geometry of the object or the plotting mechanism used to generate the object.

6.1 Object Geometry

Our results, in Figure FC3.6, demonstrate the impact of the tensor field computation. The thinner bar generates clusters at a very small distance; hence, the clustering parameters need to be fine-tuned for such cases.

The variation in object geometry in the case of scatter plots is introduced based on either the scatterpoint size or shape. Hence, We have studied the influence of the choice of glyph/point size and shape in scatter plots listed in Figure FC6.1. The shapes cover commonly used representations for scatterpoints, including triangle, square and stan-

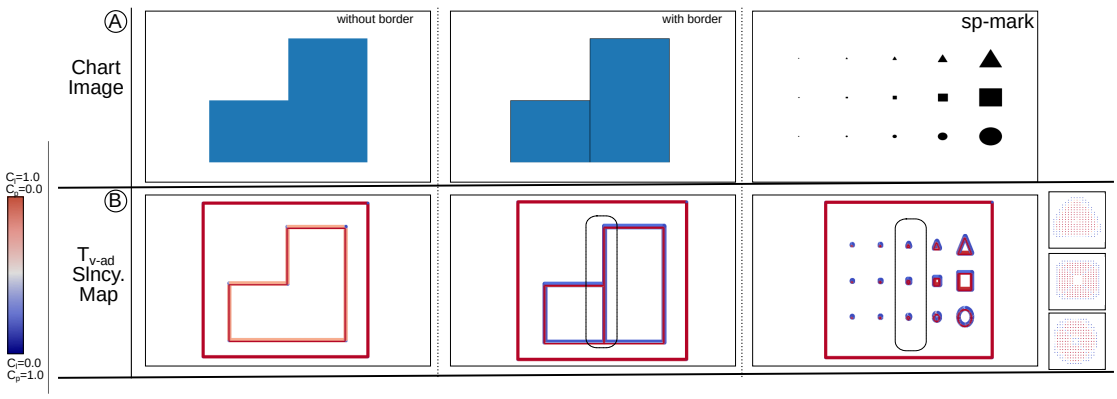


Figure FC6.1: The impact of border of bins in histograms, and scatter point (glyph) size and shape in scatter plots in our tensor field computation. We consider the following cases of histograms: (left) with the border and (middle) without the border on bins in a histogram; and (right) different glyph shapes and sizes used in scatter plots. The source images are in (A), and the saliency map visualization of the tensor field $T_{\text{v-ad}}$ is in (B). The coolwarm color mapping associated with corresponding C_l and C_p saliency values, used in the visualizations, is shown in the colorbar.

dard circle/point representation. We also check the impact of object size in the case of scatterpoints. In the visualization domain, the use of both very small and very large scatterpoints is discouraged. Hence, it is advised to use a moderate-sized scatterpoint/-mark for data encoding in any visualizations. All scatterpoint shapes in Figure FC6.1, `sp-mark` have degenerate points; however, the distribution of these clusters is not uniform and differs for different shapes, unlike bar charts. Other than the scatterpoint shape, the change in scatterpoint size also demonstrates the expected behavior of the appearance of a homogeneous region in the glyph centroid. This homogeneous region has zero tensors but is surrounded by degenerate points.

6.2 Border Thickness

To test another scenario covering different design spaces, we generate bars with and without the border, as shown in Figure FC6.2. The border on each bar generates generalized behavior of tensor fields and their degenerate points. The same pattern

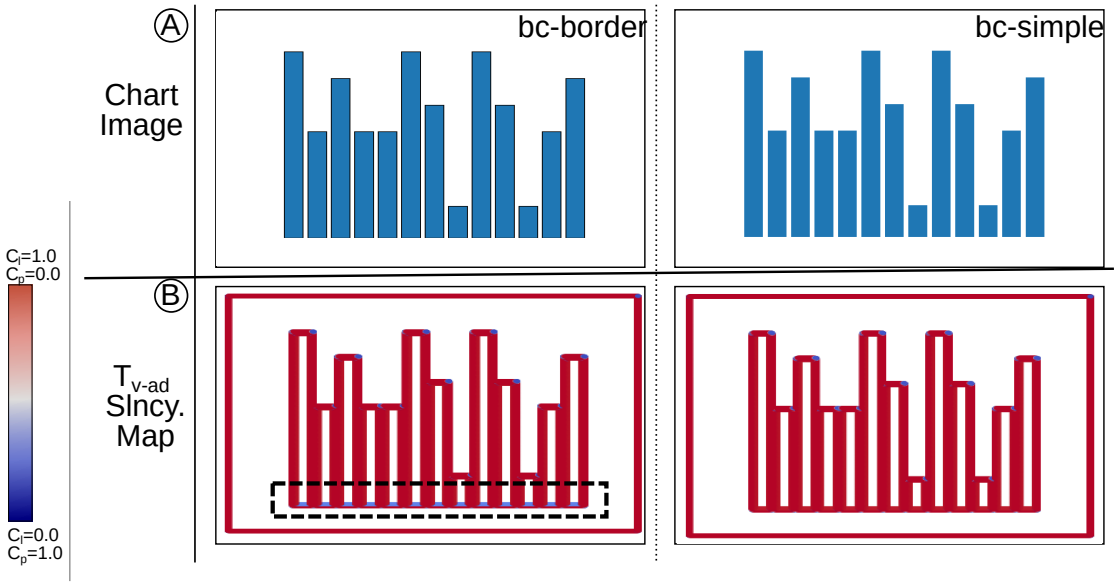


Figure FC6.2: The impact of border of bars in bar charts in our tensor field computation. We consider the following cases of bar charts: (left) with the border and (right) without the border on the bars. The source images are in (A), and the saliency map visualization of the tensor field $T_{\text{v-ad}}$ is in (B). The coolwarm color mapping associated with corresponding C_l and C_p saliency values, used in the visualizations, is shown in the colorbar.

is expected in scatterpoints as well. However, the bars with formatted borders show degenerate points at the baseline of the bar. We also test with the histogram creation with bin having a thick border. Similarly, if bins in histograms are plotted with borders, as shown in Figure FC6.1, the tensor field will generate weak degenerate points creating a periphery to the thick border as well. This border in the case of bars is only visible at the baseline because bars are not attached and have some spacing in between the columns.

6.3 Color Space

The CIELab color space is a profoundly used color model for identifying perceptual uniform colors where L* stands for perceptual lightness or luminance, and a* and b* represent four unique colors of human vision: red, green, blue, and yellow. The CIELab

color model has been recommended for computing tensor voting for color image denoising [38]. We modify our tensor voting computation that originally is working on RGB color components of the image to compute structure tensor using L*a*b* values.

We initially read the input image in RGB format and convert the image array to L*a*b* space using the OpenCV library. The difference between tensor field computation for RGB color image and L*a*b* color image comes from structure tensor computation. We take individual red, green and blue channels having values between [0,255], normalize the color values, and compute gradient tensor for all three channels individually for RGB image. For the CIE-Lab color image, we take account of all three components of the color model, normalize the values individually and compute the gradient.

The comparison of tensor field computation for both RGB and L*a*b* color models for chart images can be referred from figure FC6.3. Comparing saliency visualization for RGB and Lab color models proves that bars with different colors are identified better. We can see that the CIE-Lab color model provides a more precise set of degenerate points and eliminates weak critical points. Hence, this color model is more suitable for our multi-series or multi-class charts as colors play an important role in distinguishing chart objects belonging to one group/class. The change in the set of degenerate points introduces the requirement to tune the clustering hyper-parameters for DBSCAN.

6.4 Limitations

The tensor field computation is pixel-based and is dependent on neighboring pixels; hence, it might fail to capture the insignificant change in height, as shown in Figure FC3.8, `hg-2`, `hg-3` and, `hg-4`. Also, overlapping scatterpoints in scatter plots cause type-2 errors shown in Figure FC3.7, row B that lead to data loss.

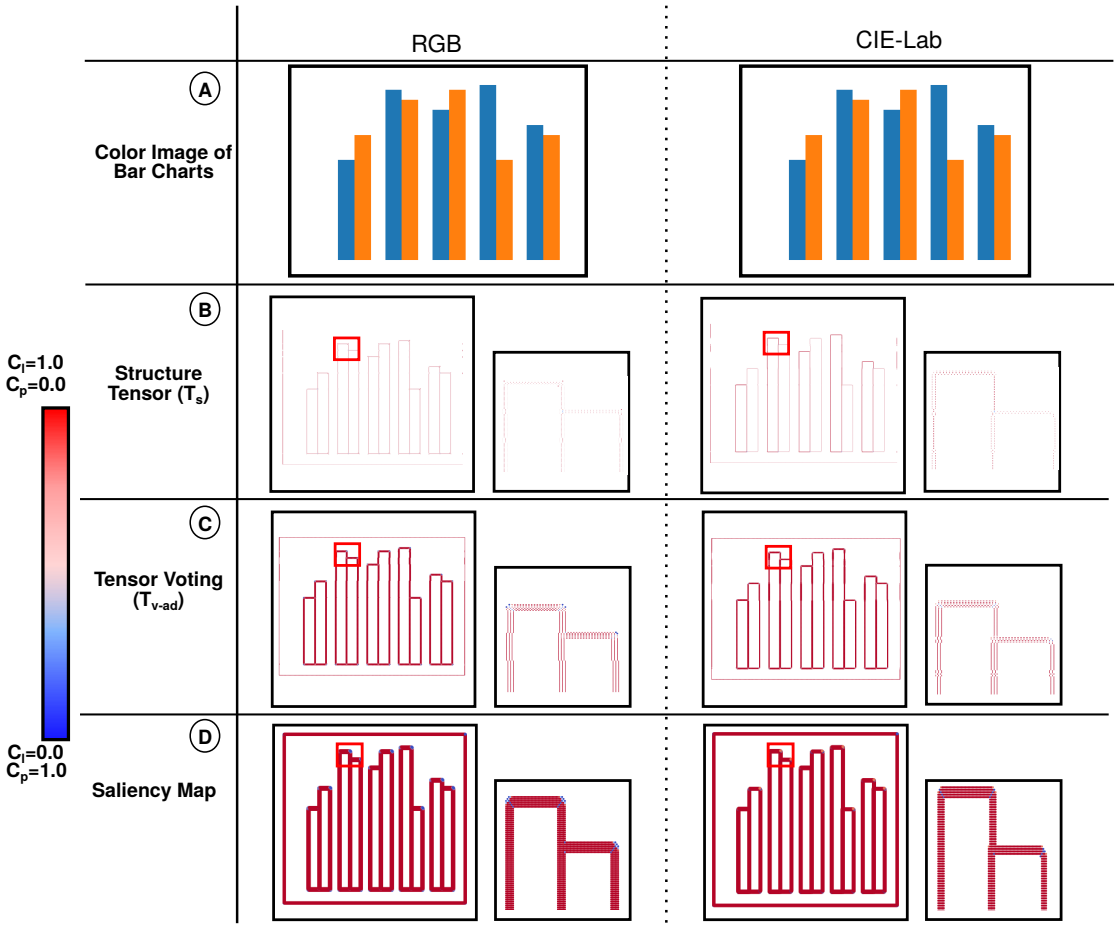


Figure FC6.3: The impact of color model in our tensor field computation. We consider the following cases of color image of grouped bar chart with color models: (left) RGB, and (right) CIELAB. (A) Input images of different variants of simple scatter plots. Tensor fields computed on the images, visualized using glyphs and colored by saliency values, include (B) structure tensor T_s, and (C) tensor voting field of T_s after anisotropic diffusion $T_{\text{v-ad}}$. (D) The saliency values of $T_{\text{v-ad}}$ visualized using dot plots. The coolwarm color mapping associated with corresponding C_l and C_p saliency values, used in the visualizations, is shown in the colorbar.

Our proposed method is limited to high-quality chart images, whereas the images available on the web can belong to the category of either low-quality images or highly processed images. In the case of such images, our tensor field computation would generate false degenerate points around the edges of the object and would need a better preprocessing for such pixelated images to get better results through our tensor fields.

Also, DBSCAN is one of the critical modules of the algorithm that needs its hyper-parameters to be set for identifying clusters for a given dataset/image. The tensor field visualization helps in identifying the values for eps and minPts. However, this requires user interaction and attention to analyze the proximity of critical points to frame clusters. Another module that needs user interaction is chart annotation. The annotation process needs a precise selection of the bounding box to encapsulate the component and provides erroneous results if not done properly during morphological operation and legend color extraction.

Our chart classifiers provide a generalized model to identify the chart type and sub-type of the given chart image. However, the chart type classification model is trained with images of bar charts, scatter plots, line charts, and pie charts. Hence, the other classes of charts, such as area charts, will not work with our classifier. Additionally, the sub-type classifier fails to classify bar charts and others with unconventional characteristics, e.g., hollow bars and handwritten text in bars. The texture/shaded bars are still out of scope for our workflow. These limitations can be overcome by training the model with other chart types as well as with chart images generated across various design spaces.

CHAPTER 7

CONCLUSIONS

Automated chart interpretation model has been a topic of interest in recent times due to the high demand for data-driven approaches. Several methods exist in the literature for chart digitization. These methods have certain constraints and limitations on certain chart types. We introduce a semi-automated algorithm that performs data extraction on a given chart image by exploiting local structure estimation. We take the human perception of charts that identify chart canvas as the main component of any chart visualization. We use the topology of positive semidefinite second-order tensor fields generated using tensor voting after anisotropic diffusion on chart canvas for data extraction. In this thesis, we discuss how degenerate points show specific patterns owing to the geometry of chart objects. Our work mainly focuses on bar charts, scatter plots, and histograms. While developing the framework, we also discuss the challenges, limitations, and solutions suggested so far for similar goals. Our classifier is novel in handling seven different bar chart sub-types. We use tensor field visualization as one of the main components to analyze prominent features like corners in chart objects and use the analysis for setting our clustering parameters.

Our current model performs data extraction with considerable accuracy and achieves Levels-A1 and A2 in Kimura's six-level scheme of statistical ability [3]. Our classification model also covers the sub-type classification that helps in extending the data

extraction process for multi-series charts like grouped and stacked bars. In summary, we propose a semi-automated algorithm for chart digitization using tensor field analysis and deep-learning models.

7.1 Future Work

The limitations discussed in chapter 6 lists the future steps to be taken in order to make a robust system for chart digitization. As a first step, our chart sub-type classifier needs to be prepared to label multi-class scatter plots. The user-dependent task such as hyperparameter-tuning of DBSCAN and chart annotation using LabelImg are required to be improved for developing a completely automated system. The mapping between pixel space data and the data space using the text detection model can be used for text summary generation for the given chart. The final generated data can be used to create different visualizations using color-blind friendly color schemes. Tactile diagram generation or text-to-speech converter such as `gtts` in Python can be integrated with our algorithm to provide assistance to visually impaired students. We also observe that bars are salient objects in images of bar charts and hence, can be located using saliency detection models. The extracted bar location can further be mapped with text detection model results to generate extracted data table.

The tensor field computation model process each pixel and requires to be more optimized for real-time computation. The dataset of annotated images can be used as a training set for the object detection model to extract chart canvas efficiently. As VGGNet has been famously used for object detection tasks, our goal is to improve our classifier to automate the canvas extraction step to reduce the dependency on the user.

Bibliography

[1] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. E. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," *CoRR*, vol. abs/1409.4842, 2014, doi: https://doi.org/10.1109/CVPR.2015.7298594.

[2] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2015.

[3] K. Aoyama and M. Stephens, "Graph Interpretation Aspects of Statistical Literacy: A Japanese Perspective," *Mathematics Education Research Journal*, vol. 15, no. 3, pp. 207–225, 2003, doi: https://doi.org/10.1007/bf03217380.

[4] D. Jung, W. Kim, H. Song, J.-i. Hwang, B. Lee, B. Kim, and J. Seo, "Chartsense: Interactive data extraction from chart images," in *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, ser. CHI '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 6706–6717, doi: https://doi.org/10.1145/3025453.3025957.

[5] 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. Wiley Online Library, 2019, pp. 249–260, doi: https://doi.org/10.1111/cgf.13686.

[6] M. Hegarty, "The cognitive science of visual-spatial displays: Implications for design," *Topics in cognitive science*, vol. 3, no. 3, pp. 446–474, 2011, doi:https://doi.org/10.1111/j.1756-8765.2011.01150.x.

[7] W. Huang and C. L. Tan, "A system for understanding imaged infographics and its applications," in *Proceedings of the 2007 ACM Symposium on Document Engineering*. ACM, 2007, pp. 9–18, doi: https://doi.org/10.1145/1284420.1284427.

[8] J. Poco and J. Heer, "Reverse-Engineering Visualizations: Recovering Visual Encodings from Chart Images," in *Computer Graphics Forum*, vol. 36, no. 3. Wiley Online Library, 2017, pp. 353–363, doi: https://doi.org/10.1111/cgf.13193.

[9] P. M. Jones, C. D. Wickens, and S. J. Deutsch, "The display of multivariate information: An experimental study of an information integration task," *Human Performance*, vol. 3, no. 1, pp. 1–17, 1990, doi: https://doi.org/10.1207/s15327043hup0301_1.

[10] K. B. Bennett and J. M. Flach, "Graphical Displays: Implications for Divided Attention, Focused Attention, and Problem Solving," *Human Factors*, vol. 34, no. 5, pp. 513–533, 1992, doi:https://doi.org/10.1177/001872089203400502.

[11] C. D. Wickens and C. M. Carswell, "The Proximity Compatibility Principle: Its Psychological Foundation and Relevance to Display Design," *Human Factors*, vol. 37, no. 3, pp. 473–494, 1995, doi:https://doi.org/10.1518/001872095779049408.

[12] Y. Liu, X. Lu, Y. Qin, Z. Tang, and J. Xu, "Review of Chart Recognition in Document Images," in *Visualization and Data Analysis 2013*, vol. 8654. International Society for Optics and Photonics, 2013, p. 865410, doi: https://doi.org/10.1117/12.2008467.

[13] M. Savva, N. Kong, A. Chhajta, L. Fei-Fei, M. Agrawala, and J. Heer, "Revision: Automated Classification, Analysis and Redesign of Chart Images," in *Proceed-*

ings of the 24th annual ACM symposium on User interface software and technology. ACM, 2011, pp. 393–402, doi: https://doi.org/10.1145/2047196.2047247.

[14] L. Battle, P. Duan, Z. Miranda, D. Mukusheva, R. Chang, and M. Stonebraker, "Beagle: Automated Extraction and Interpretation of Visualizations from the Web," in *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. ACM, 2018, p. 594, doi: https://doi.org/10.1145/3173574.3174168.

[15] N. Siegel, Z. Horvitz, R. Levin, S. Divvala, and A. Farhadi, "FigureSeer: Parsing result-figures in research papers," in *European Conference on Computer Vision*. Springer, 2016, pp. 664–680.

[16] A. Rohatgi, "Webplotdigitizer," 2011.

[17] A. Baucom and C. Echanique, "ScatterScanner: Data Extraction and Chart Restyling of Scatterplots," 2013.

[18] M. Cliche, D. Rosenberg, D. Madeka, and C. Yee, "Scatteract: Automated extraction of data from scatter plots," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 2017, pp. 135–150, doi: https://doi.org/10.1007/978-3-319-71249-9_9.

[19] R. A. Al-Zaidy and C. L. Giles, "Automatic Extraction of Data from Bar Charts," in *Proceedings of the 8th international conference on knowledge capture*, 2015, pp. 1–4, doi: https://doi.org/10.1145/2815833.2816956.

[20] N. Methani, P. Ganguly, M. M. Khapra, and P. Kumar, "PlotQA: Reasoning over Scientific Plots," in *The IEEE Winter Conference on Applications of Computer Vision*, 03 2020, pp. 1516–1525, doi: https://doi.org/10.1109/wacv45572.2020.9093523.

[21] G. Medioni, M.-S. Lee, and C.-K. Tang, *Computational Framework for Segmentation and Grouping*. USA: Elsevier Science Inc., 2000, doi:https://doi.org/10.1016/b978-0-444-50353-4.x5000-8.

[22] R. Moreno, L. Pizarro, B. Burgeth, J. Weickert, M. A. Garcia, and D. Puig, "Adaptation of tensor voting to image structure estimation," in *New Developments in the Visualization and Processing of Tensor Fields*. Springer, 2012, pp. 29–50.

[23] T.-P. Wu, S.-K. Yeung, J. Jia, C.-K. Tang, and G. Medioni, "A Closed-Form Solution to Tensor Voting: Theory and Applications," *arXiv preprint arXiv:1601.04888*, 2016, doi: https://doi.org/10.1109/tpami.2011.250.

[24] Y. Zhou and C. L. Tan, "Hough-based Model for Recognizing Bar Charts in Document Images," in *Document Recognition and Retrieval VIII*, vol. 4307. International Society for Optics and Photonics, 2000, pp. 333–340, doi: https://doi.org/10.1117/12.410854.

[25] J. Wagemans, J. Feldman, S. Gepshtein, R. Kimchi, J. R. Pomerantz, P. A. Van der Helm, and C. Van Leeuwen, "A Century of Gestalt Psychology in Visual Perception: II. Conceptual and Theoretical Foundations," *Psychological Bulletin*, vol. 138, no. 6, p. 1218, 2012, doi: https://doi.org/10.1037/a0029334.

[26] G. Guy and G. Medioni, "Inferring Global Perceptual Contours from Local Features," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 1993, pp. 786–787, doi: https://doi.org/10.1007/BF00144119.

[27] G. Medioni, C.-K. Tang, and M.-S. Lee, "Tensor Voting: Theory and Applications," *Proceedings of RFIA, Paris, France*, vol. 3, 2000.

[28] J. Sreevalsan-Nair and B. Kumari, *Local Geometric Descriptors for Multi-Scale Probabilistic Point Classification of Airborne LiDAR Point Clouds*.

Springer Cham, Mathematics and Visualization, 2017, pp. 175–200, doi: https://doi.org/10.1007/978-3-319-61358-1_8.

[29] S. Wang, T. Hou, S. Li, Z. Su, and H. Qin, "Anisotropic Elliptic PDEs for Feature Classification," *Visualization and Computer Graphics, IEEE Transactions on*, vol. 19, no. 10, pp. 1606–1618, 2013, doi: 0.1109/TVCG.2013.60.

[30] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, ser. KDD'96. AAAI Press, 1996, p. 226–231.

[31] J. Sreevalsan-Nair, K. Dadhich, and S. C. Daggubati, "Tensor Fields for Data Extraction from Chart Images: Bar Charts and Scatter Plots," in *Topological Methods in Visualization: Theory, Software and Applications (to appear)*, I. Hotz, T. B. Masood, F. Sadlo, and J. Tierny, Eds. Springer-Verlag, 2020.

[32] Y. Rubner, L. J. Guibas, and C. Tomasi, "The earth mover's distance, multi-dimensional scaling, and color-based image retrieval," in *Proceedings of the ARPA image understanding workshop*, vol. 661, 1997, p. 668.

[33] J. D. Hunter, "Matplotlib: A 2D Graphics Environment," *Computing in Science & Engineering*, vol. 9, no. 3, pp. 90–95, 2007, doi: https://doi.org/10.1109/MCSE.2007.55.

[34] S. E. Kahou, A. Atkinson, V. Michalski, Á. Kádár, A. Trischler, and Y. Bengio, "Figureqa: An annotated figure dataset for visual reasoning," *CoRR*, vol. abs/1710.07300, 2017.

[35] K. Dadhich, S. C. Daggubati, and J. Sreevalsan-Nair, "Scatterplotanalyzer: Digitizing images of charts using tensor-based computational model," in *Computational Science – ICCS 2021*, M. Paszynski, D. Kranzlmüller, V. V.

Krzhizhanovskaya, J. J. Dongarra, and P. M. Sloot, Eds. Cham: Springer International Publishing, 2021, pp. 70–83, doi: https://doi.org/10.1007/978-3-030-77977-1_6.

[36] K. Dadhich, S. C. Daggubati, and J. Sreevalsan-Nair, "Barchartanalyzer: Digitizing images of bar charts," in *Proceedings of the International Conference on Image Processing and Vision Engineering, IMPROVE 2021, Online Streaming, April 28-30, 2021*, F. H. Imai, C. Distante, and S. Battiato, Eds. SCITEPRESS, 2021, pp. 17–28, doi: https://doi.org/10.5220/0010408300170028.

[37] Tzutalin, "Labelimg," https://github.com/tzutalin/labelImg, 2015.

[38] R. Moreno, M. A. Garcia, D. Puig, and C. Julià, "Edge-preserving color image denoising through tensor voting," *Computer Vision and Image Understanding*, vol. 115, no. 11, pp. 1536–1551, 2011, doi: https://doi.org/10.1016/j.cviu.2011.07.005.