SCATTERPLOTANALYZER: DIGITIZING IMAGES OF CHARTS USING TENSOR-BASED COMPUTATIONAL MODEL

A PREPRINT

Komal Dadhich **Siri Chandana Daggubati** **Jaya Sreevalsan-Nair***
Graphics-Visualization-Computing Lab,
International Institute of Information Technology Bangalore, Karnataka 560100, India
http://www.iiitb.ac.in/gvcl

This is a peer-reviewed article accepted for publication, prior to its camera-ready version, in the Proceedings of the International Conference on Computational Science, ICCS 2021

March 31, 2021

ABSTRACT

Charts or scientific plots are widely used visualizations for efficient knowledge dissemination from datasets. Nowadays, these charts are predominantly available in image format in print media, the internet, and research publications. There are various scenarios where these images are to be interpreted in the absence of datasets that were originally used to generate the charts. This leads to a pertinent need for automating data extraction from an available chart image. We narrow down our scope to scatter plots and propose a semi-automated algorithm, ScatterPlotAnalyzer, for data extraction from chart images. Our algorithm is designed around the use of second-order tensor fields to model the chart image. ScatterPlotAnalyzer integrates the following tasks in sequence: chart type classification, image annotation, object detection, text detection and recognition, data transformation, text summarization, and optionally, chart redesign. The novelty of our algorithm is in analyzing both simple and multi-class scatter plots. Our results show that our algorithm can effectively extract data from images of different resolutions. We also discuss specific test cases where ScatterPlotAnalyzer fails.

1 Introduction

Plots or charts are one of the simplest visualizations for data analysis, of which bar charts and scatter plots are commonly used. Today, most charts are widely available in image format in print media, the internet, research publications, etc. Digitizing chart images is a pertinent requirement for automating chart interpretation, which has applications in assisted technologies for STEM (Science, Technology, Education, and Mathematics) education of the visually impaired. Recently, machine and deep learning solutions have been widely used for reasoning over [1, 2], and to a lesser extent, data extraction [3]. However, a generalized computational model for such images can also alternatively lead to data extraction and its applications, such as chart reconstruction for improving image resolution or quality or chart redesign for improving interpretation.

The simple design of bar charts and scatter plots provide high levels of interpretability for low-dimensional data, even with complex trends. However, uni- or bi-variate data analysis is less efficient in the light of multivariate data collections being the norm. Hence, more complex forms of these chart types are used to address the insufficiency of its simpler forms. While the simpler forms are used widely in the early stages of chart graphicacy in school education, widely available chart images tend to be complex variants. The studies on both bar charts and scatter plots publish results for accuracy in text detection and/or data extraction [3]. The complex forms of bar charts, such as grouped bar charts, have been used for chart image analysis [4, 5]. At the same time, limited studies demonstrate similar analysis of complex forms of scatter plots, where one such study looks at scatter points or marks with different types

*jnair@iiitb.ac.in

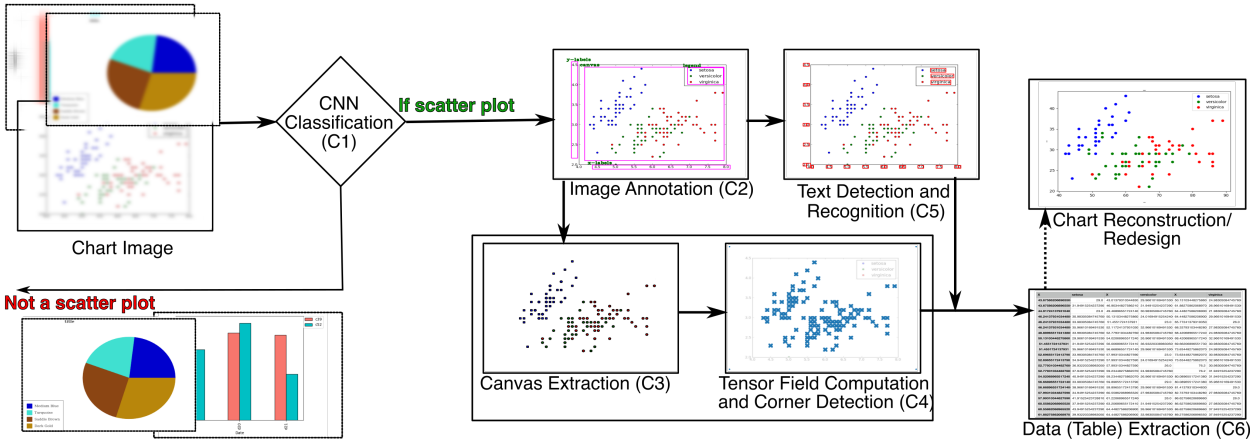

Figure 1: Our proposed workflow for data extraction from a given image of a scatter plot with six components (C1-C6), where C1-C3 are preprocessing steps, C4 is the computational modeling step, C5 and C6 are postprocessing steps. The output is a data table, which can be optionally used for chart reconstruction or redesign.

of formatting [6]. Overall, there is a gap in the analysis of complex variants of scatter plots. Here, we consider such scatter plots, particularly those with color, as a visual encoding.

The simple scatter plot is where scatter points represent (x,y) tuples, and its location encodes the data. Whereas the complex variants of scatter plots additionally use shape, size, and color of scatter points for encoding extra dimensions of the data. One such variant is where color is used to encode different classes or groups in the bi-variate data used, which we refer to as *multi-class scatter plots*. Multi-class scatter plots are predominantly used to organize the display of the clustering in datasets, correlation, etc. For example, such charts are used in machine learning to visualize multi-class classification. In visualization parlance, multi-class scatter plots can be considered to be two or more overlays of simple scatter plots using the same coordinate system and units and scales on the axes. Hence, we hypothesize that the data extraction algorithm for a simple scatter plot can be effectively extended to multi-class scatter plots using a data-parallel construct. As a result, we propose a generalized algorithm that performs data extraction from both simple and multi-class scatter plots.

Our algorithm uses perceptual information from the charts to extract data, where preprocessing is implemented using widely used image processing and artificial intelligence techniques. This perceptual information is represented using a second-order tensor field, which is a computational model, as has been done in our previous work [7]. These tensor fields exploit local geometry to extract objects such as scatter points from the chart in pixel space. The next step in data extraction is to devise an algorithm that uses these tensor fields to extract data in pixel space, which is further used with text extracted from the images to output a data table [4]. We propose such a data extraction algorithm from images for simple and multi-class scatter plots, ScatterPlotAnalyzer (Figure 1).

2 Related Work

Chart analysis for data extraction depends on processes, such as chart type classification, chart image annotation, feature extraction, and text recognition. Several tools and techniques have already been devised to automatically interpret charts from their images, where the outcomes correspond to different aspects of chart analysis. 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 [5]. In WebPlotDigitizer, the user is provided with an option to use the automated or manual procedure for data extraction from the given chart image [8]. 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.

Just as WebPlotDigitizer, Scatterscanner requires user interactivity for data extraction from scatter plots, specifically [9]. 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 [6].

Similar to many computer vision problems, machine and deep learning models have been introduced for chart classification and object detection. A web-based system Beagle takes charts in scalable vector graphics format and classifies charts found as visualizations [10]. ChartSense uses GoogleNet to perform chart classification for line, bar, pie, scatter

charts, map, and table types [11]. FigureSeer uses a similar fine-tuning approach [12] to localize and classify result-figures in research papers. Text interpretation is equally important for data extraction. A Darknet neural network as an object detection model in combination with OCR for text detection is used from bar charts, whereas the pixels of a specific color within the periphery of the circle are utilized for pie charts [13].

3 Tensor Field as a Computational Model

Tensor fields have been widely used to exploit geometric properties of objects for edge detection in natural images using structure tensor and tensor voting [14]. In our previous work, we have used tensor voting for extracting bars and scatter points in images of simple charts, thus generating a computational model using second-order tensor fields [7]. This model has shown promising results for data extraction using perceptual information in pixel space. Hence, we use this tensor-based computational model for extracting data from both simple and multi-class scatter plots. The computational model is one of the components of ScatterPlotAnalyzer, which completes the step of converting the extracted values in pixel space to data space.

Tensor Voting Using Gradient Tensor: We use a local geometric descriptor in the form of a second-order tensor, which is required for tensor voting computation [15]. Tensor voting itself returns a second-order tensor that is geometry-aware in a more global context [14]. Hence, the tensor voting field is used for scatter point extraction from a scatter plot image. Structure tensor T_s at a pixel provides the orientation of the gradient computed from the local neighborhood.

$$T_s = \mathcal{G}_\rho * (G^T G), \text{ where } G = \begin{bmatrix} \frac{\partial I}{\partial x} & \frac{\partial I}{\partial y} \end{bmatrix}$$

is the gradient tensor at the pixel with the intensity I; convolved (using $*$ operator) with Gaussian function $\mathcal{G}$ with zero mean and standard deviation ρ. The tensor vote cast at x_i by x_j using a second-order tensor K_j in d-dimensional space is, as per the closed-form equation [16]: $S_{ij} = c_{ij} R_{ij} K_j R'_{ij}$,

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

I_d is the d-dimensional identity matrix; unit vector of direction vector $r_{ij} = \hat{d_{ij}}$, with $d_{ij} = x_j - x_i$; σ_d is the scale parameter; and $c_{ij} = \exp\big(-\big(\sigma_d^{-1}.\|d_{ij}\|_2^2\big)\big)$. The gradient T_g can be used as K_j [17].

Anisotropic Diffusion: As the tensor votes T_v in normal space has to encode object geometry in tangential space, we perform anisotropic diffusion to transform T_v to tangential space [7, 15]. The eigenvalue decomposition of the two-dimensional T_v yields ordered eigenvalues, $\lambda_0 \geq \lambda_1$, and corresponding eigenvectors v_0 and v_1, respectively. Anisotropic diffusion of T_v using parameter δ is,

$$T_{\text{v-ad}} = \sum_{k=0}^{1} \lambda'_k . v_k v_k^T, \text{ where } \lambda'_k = \exp\Big(-\frac{\lambda_k}{\delta}\Big).$$

Diffusion parameter value ($\delta = 0.16$) is widely used [18, 7].

Saliency Computation: The saliency of a pixel belonging to geometry features of line- or junction/point-type is determined by the eigenvalues of $T_{\text{v-ad}}$ [7]. We get the saliency maps at each pixel of an image of its likelihood for being a line- or junction-type feature, C_l and C_p, respectively, $C_l = \frac{\lambda_0 - \lambda_1}{\lambda_0 + \lambda_1}$ and $C_p = \frac{2\lambda_1}{\lambda_0 + \lambda_1}$,
using eigenvalues of $T_{\text{v-ad}}$ of the pixel, such that, $\lambda_0 \geq \lambda_1$. The pixel with $C_p \approx 1.0$ is referred to as a critical point or degenerate point, in the parlance of tensor fields. Our goal is to find all the critical points in the chart image in the C4 step (Figure 1), as a few of them are significant for data extraction in pixel space.

4 Components of ScatterPlotAnalyzer

We propose an algorithm, ScatterPlotAnalyzer, that extracts data from a given chart image after determining it is a scatter plot. ScatterPlotAnalyzer has six main components to meet this requirement (Figure 1), namely, chart type classification, chart image annotation, canvas extraction, tensor field computation, text recognition, and data table extraction.

Chart Type Classification (C1): Different types of charts, *e.g.,* bar chart, scatter plot, pie chart, etc., encode data differently. Hence, data extraction from their images needs to exploit these characteristics of visual encoding for reverse engineering. Visual encodings include the location of scatter point for data tuple (x,y), the height of the bar for y-value, and sector size in a pie chart for percentage value. Separating these visual encodings is important for

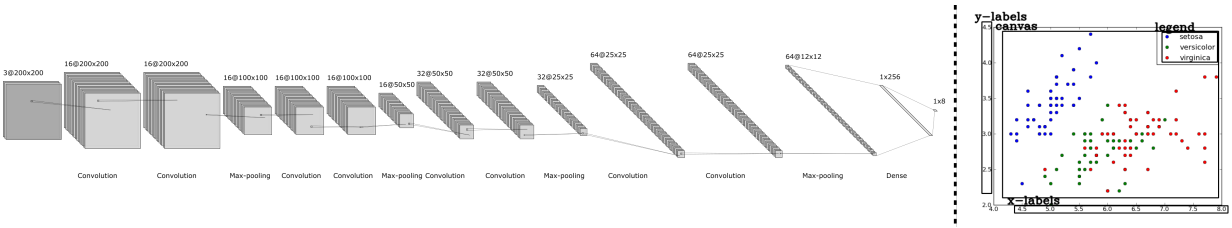

Figure 2: The preprocessing steps of chart classification and annotation of ScatterPlotAnalyzer. (Left) The architecture diagram of our CNN-based classifier for identifying bar chart subtypes. (Right) Human-guided annotation of the chart image, using an example of scatter plot, to extract the chart canvas, legend, and text needed for object detection, multi-class inferences, and contextualization.

abstractions leading to a generalized data extraction from different chart types, which unifies a data extraction system for all charts.

The chart objects such as bars, scatter points, and lines have specific geometric structure that can be exploited, unlike the objects found in natural images. The chart subtypes for bar charts are grouped, stacked, column, bar charts, and histograms as a special case. We now consider simple and multi-class scatter plots as chart subtypes in scatter plots. We observe that chart subtypes preserve the geometry in chart objects, as is in the parent chart type. While this similarity helps in generalizing data extraction workflow, the differences in geometry help in classification. Overall, the similarity in geometry across subtypes limit the applicability of contour-based techniques for chart type classification, which provides the granularity of subtypes. Hence, we use a convolutional neural network (CNN) for generic chart type classification, which is widely used for similar applications. Some of the pre-trained models have been used for chart type classification of images by imposing certain constraints, *e.g.,* training with a small image corpus. However, these classifiers have been found to perform with low accuracy. The classifier in ChartSense has used GoogleNet [11] and has been trained on different chart types, but performs well for six out of ten types (bar chart, line chart, map, pie chart, scatter plot, and table) with the small corpus.

We build our classifier using an architecture inspired by VGGNet (Figure 2). VGGNet architecture is popularly used for object detection tasks, and it requires the convolutional layers to be stacked. We add convolutional layers, followed by pooling layers, and finally, fully connected layers, kernel size of (5,5). We train the CNN model on our chart image dataset containing more than 1000 images of four chart types, namely, bar chart, scatter plot, pie chart and line chart. Our CNN based classifier is constrained by the requirement of input images of fixed size for training. Hence, we first resize the images in the dataset to 200×200 size. The image resizing and classifier implementation has been done using Python imaging (PIL) and Keras libraries, respectively.

Our classifier is sufficient for ScatterPlotAnalyzer to identify scatter plots, without the granularity of simple or multi-class typification. This is because this typification can be implemented more efficiently by interpreting the legend rather than using CNN.

Image Annotation (C2) and Canvas Extraction (C3): Image annotation is required to prepare a training dataset for object detection, segmentation, and similar computer vision applications. Image annotation provides labels to different regions of interest (ROI). The ROIs are detected and extracted using the predefined labels for such regions. Image annotation strictly requires contextual labels and appropriate associations between labels and ROIs. Automation of annotation is not a completely solved problem, owing to which human-guided annotation of images is widely used in practice.

Manual marking and annotation of bounding boxes for ROIs have been widely used for chart image analysis [13, 2]. We specifically use the following labels for specific ROIs needed for tensor field computation. These labels include canvas, x-axis, y-axis, x-labels, y-labels, legend, title, x-title, and y-title. We use LabelImg [19], a Python tool, for marking and annotating bounding boxes for ROIs. LabelImg has a graphical user interface (GUI) to select an image, trace a bounding box for an ROI, and label the ROI appropriately. We label the ROI that contains the chart objects such as bars, lines, or scatter points as *Canvas*. The extracted ROI is referred to as *chart canvas*, which is one of the chart image components [7]. LabelImg generates the annotation in an XML file that can be used further to automate the extraction of the canvas region and text localization. The former is used for chart extraction (C3), and the latter for text detection (C5). Figure 2 (right) shows an annotated scatter plot.

The canvas extraction step (C3) is implemented using image preprocessing methods to remove the remaining elements other than chart objects such as gridlines, overlaid legends, etc. This is required as tensor field computation in C4 is

sensitive to the presence of these extraneous elements. Marker-based watershed segmentation and contour detection algorithm have been used in combination to remove these extraneous components in the chart canvas effectively [7]. These steps also fill hollow bars and scatter points, as required, since the tensor field is computed effectively for filled chart objects. Highly pixellated edges in aliased images lead to uneven edges in each chart object. This issue is addressed by using the contour detection method to add a fixed-width border to chart objects [7]. Overall, the processes in chart canvas extraction are chosen specifically to extract chart objects using tensor fields.

Tensor field Computation (C4): We use tensor voting on gradient tensor to compute a second-order tensor field T_v, which is further improved using anisotropic diffusion (Section 3). The resulting field, $T_{\text{v-ad}}$, is then analyzed by identifying its critical points using the saliency map values (C_l, C_p) at each pixel. We use a threshold on the trace of the tensor $T < 0.01$ for scatter plots to discard weak critical points. The thresholding is tensor-based, as the trace is a tensor invariant. CIE-Lab used for perceptual modeling of color is better suited for tensor voting than the RGB model [20]. Hence, the chart image is converted from the RGB model to the CIE-Lab model prior to the actual tensor field computation.

DBSCAN Clustering We observe that the critical points of chart image computed from tensor field computation form sparse clusters along the boundary of a scatter point as well as it's interior [7]. Since the scatter point interior gives its location, that is the data, and we extract the cluster in the interior. We use density-based clustering, DBSCAN [21], to localize these clusters. We adjust the hyperparameters of DBSCAN clustering to give the best clusters corresponding to specific chart types, *e.g.,* clusters in the corners of bars for a bar chart and clusters near the scatter point centroid for scatter plots. These hyperparameters influence the location of the cluster centroids. The cluster centroid gives the data tuple (x,y) corresponding to the scatter point but in pixel space.

Text Recognition (C5): The data extracted using tensor fields are in pixel space and have to be contextualized to the data space for extracting the data table. Hence, we now combine the data in pixel space with the text information in the image. We perform text detection to get x-axis and y-axis labels and compute the scale factor between the pixel and data spaces. The recognition of other textual elements, namely, plot title, legend, x-axis, and y-axis titles, also plays a crucial role in analyzing chart images.

We use deep-learning-based OCR, namely Character Region Awareness for Text Detection, CRAFT [22] for effective text area detection, including arbitrarily-oriented text. This approach is designed for relatively complex text in images, and it works by exploring each character region and considering the affinity between characters. A CNN designed in a weakly-supervised manner predicts the character region score map and the image's affinity score map. The character region score is used to localize individual characters and affinity scores to group each character to a single instance. So, the instance of text detected is not affected by its orientation and size. From the detected text boxes, its orientation is computed and then rotated.

A unified framework for scene text recognition that fits all variants of scenes called the scene text recognition framework, STR [23] is implemented subsequent to CRAFT. Being a four-stage framework consisting of transformation, feature extraction, sequence modeling, and prediction, STR resembles the combination of computer vision tasks such as object detection and sequence prediction task. Hence, it uses a convolutional recurrent neural network (CRNN) for text recognition. We find that the CRAFT model, along with the STR framework, works efficiently to retrieve labels and titles of the chart image better than other available text recognition tools, *e.g.,* Tesseract OCR [24].

Information Aggregation for Data Extraction (C6): For multi-class scatter plots, interpreting the legend is an essential step. In the absence of legend, we consider the chart to be a simple scatter plot. In the presence of a legend, the number of classes specified in it is used to determine if the chart is a simple or a multi-class scatter plot. We use CIE-Lab color space to identify the colors corresponding to the classes in the legend. The legend's colors are extracted using the morphological operations used in image preprocessing in the legend ROI, as done on the chart canvas in C3.

We use the pixel color at the cluster centroid extracted in C4 to indicate the class corresponding to the scatter point. The actual class label is extracted from the text recognition in the legend in C5. Thus the data table now has a class label along with the (x,y) tuple extracted in the pixel space.

As a final process in ScatterPlotAnalyzer, the data extracted in pixel space is transformed into data space by using the text recognized for axis and corresponding tick labels. The labels add appropriate textual information for the variable along each axis by providing its name and unit value. The unit value of the variables and scatter point location, along with its class information, are used to extract the data table in C6.

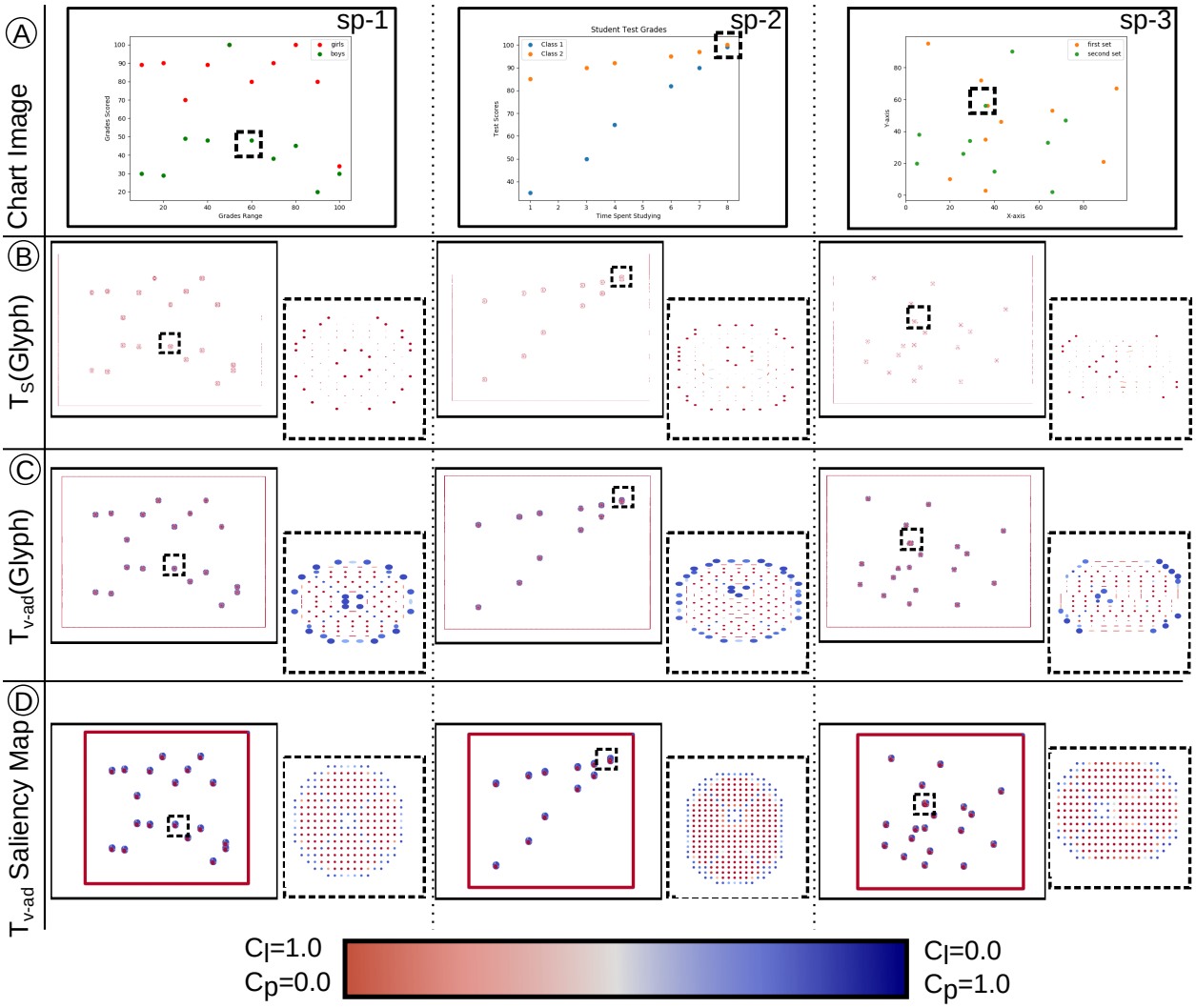

Figure 3: The tensor field computation step in ScatterPlotAnalyzer (C4) uses the images of multi-class scatter plots (A) to compute the structure tensor T_s (B), and the tensor voting field we use, $T_{\text{v-ad}}$ (C). While (B) and (C) show tensor field visualization using ellipsoid glyphs, colored using saliency map, (D) shows the dot map of the saliency map. The color bar shows the coolwarm color mapping used for saliency map of the tensor field. The glyph visualization is demonstrated on a 1:3 subsampled image for clear visualization.

5 Experiments and Results

We have used ∼1000 images from the FigureQA dataset [1] containing bar charts and its subtypes, scatter plots, pie charts, and line charts for training our model. Our model works with ∼90% accuracy.

We perform experiments to analyze the performance of ScatterPlotAnalyzer that answer the following questions:

- Is the choice of the tensor field from tensor voting, $T_{\text{v-ad}}$, better than the conventionally used structure tensor T_s for data extraction?
- How accurate is the data reconstruction for both simple and multi-class scatter plots?

We consider the structure tensor T_s and tensor voting field $T_{\text{v-ad}}$ for different multi-class scatter plots by studying the shape of the tensor using ellipsoid glyphs [25]. Ellipsoid glyphs in three-dimensional space reduce to elliptical glyphs in two-dimensional, as in our case. The elliptical glyphs are drawn with the major and minor axes of the ellipses oriented along the major and minor eigenvectors of the tensor. The eigenvalues of the tensor give the major and minor lengths of axes of the ellipse. Thus, the degenerate points with almost equal eigenvalues are closer to the circular

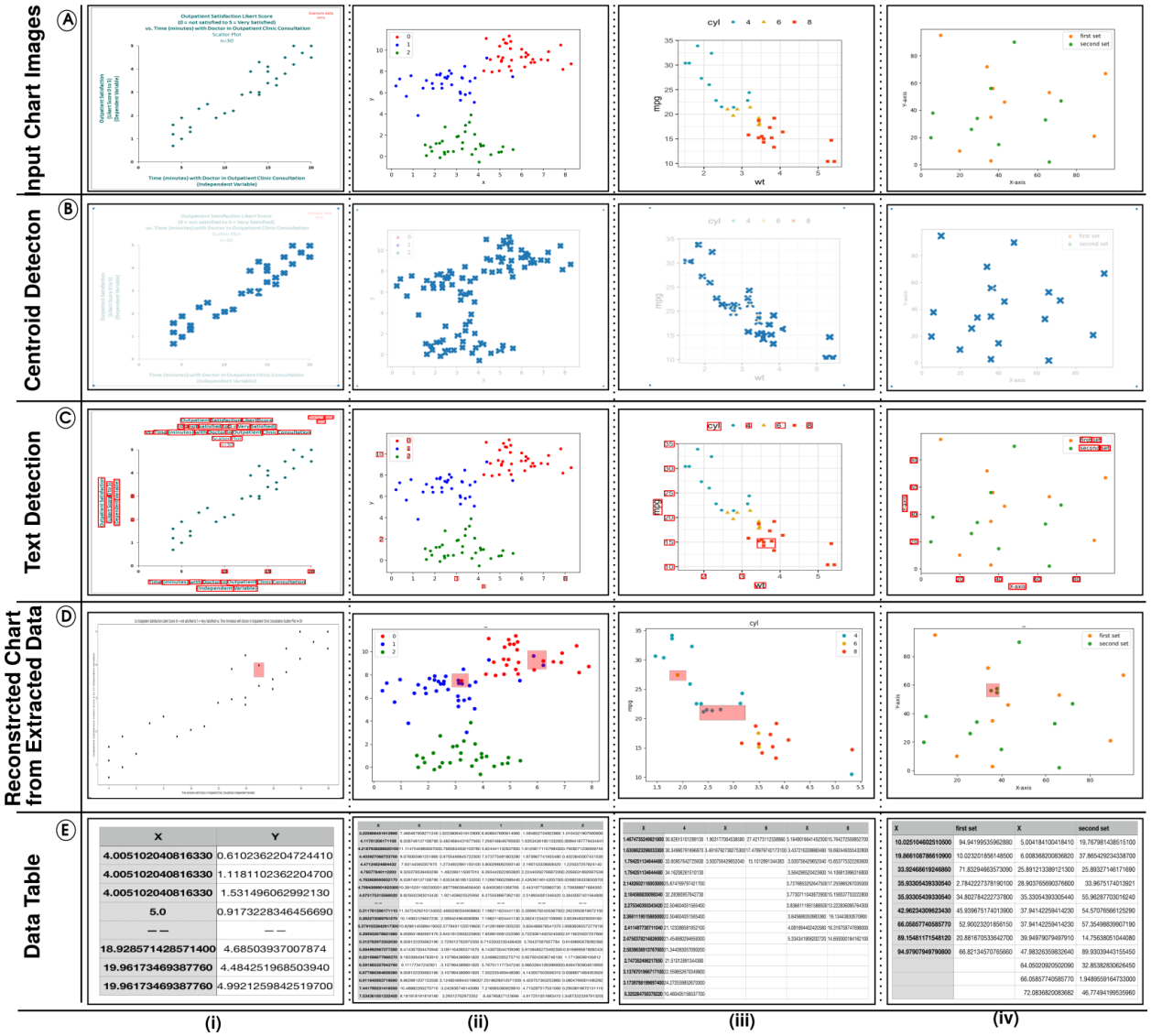

Figure 4: The outcomes of the key steps in our ScatterPlotAnalyzer, namely, tensor field computation (C4), text detection (C5), and data extraction (C6), implemented on the source input chart images. The dataset includes available source images of (i) a simple scatter plot, and (ii-iii) multi-class scatter plots, and an image of multi-class scatter plot created by plotting available data in (iv).

shape. Our results in Figure 3 show that the degenerate points are stronger when using the tensor voting field as opposed to the structure tensor. We can conclude that the degenerate points are strengthened by performing tensor voting on gradient tensor and subsequent anisotropic diffusion.

In terms of the accuracy of the data table given by ScatterPlotAnalyzer, we analyze the qualitative and quantitative results. Figure 4 shows the different steps in ScatterPlotAnalyzer for both simple and multi-class 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* or type-II errors, as has been reported in our previous work [7]. For the extracted points, the data extracted is mostly accurate. Our experiments include a simple scatter plot without legend, a 3-class scatter plot, and 2-class scatter plots, shown in Figure 4 (i)-(iv). We have highlighted the type-II errors in row D in Figure 4 in translucent red highlights.

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

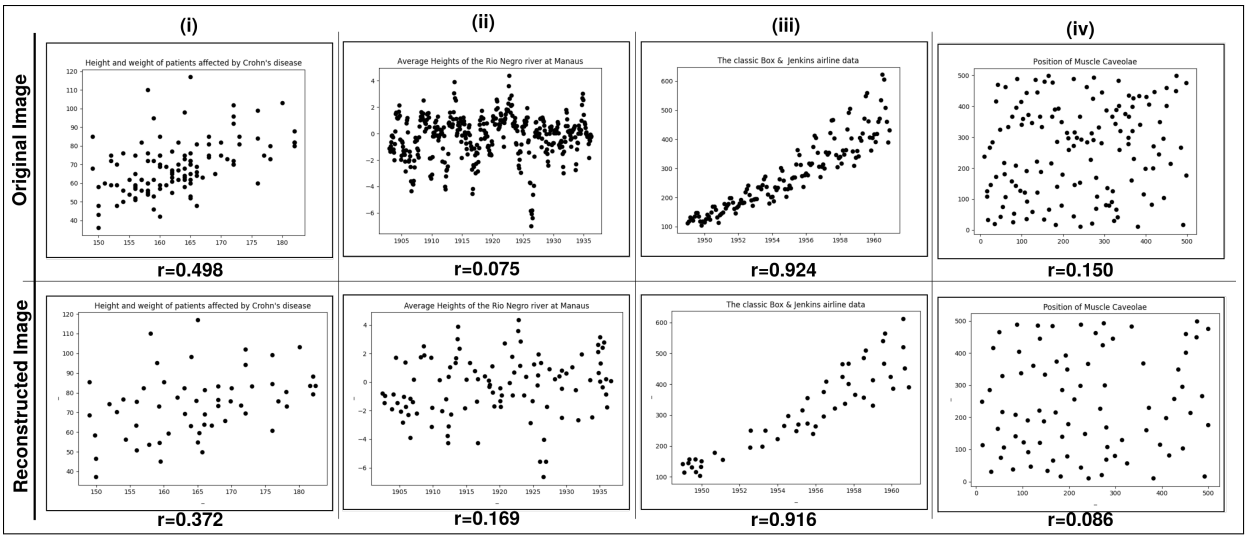

Figure 5: Correlation coefficient (r) values in both original and ScatterPlotAnalyzer reconstructed images of simple scatter plots.

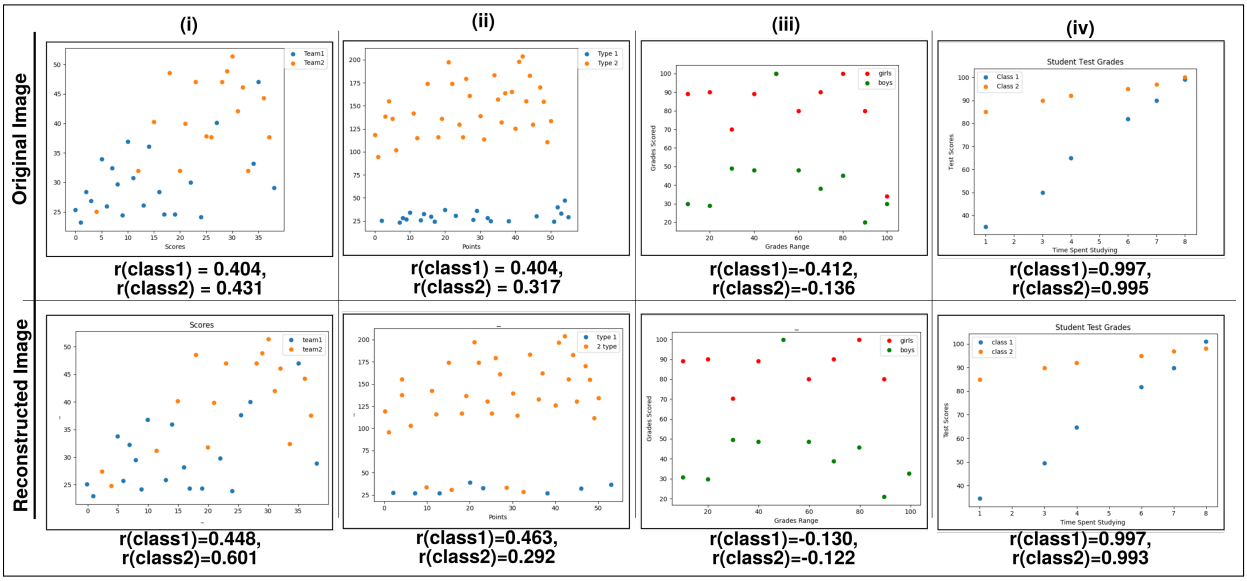

Figure 6: Correlation coefficient (r) values in both original and ScatterPlotAnalyzer reconstructed images of multi-class scatter plots.

the image using ScatterPlotAnalyzer. 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 5 and the same for multi-class scatter plots in Figure 6. 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.

We observe that the quality of the images, irrespective of their source, perform the same using ScatterPlotAnalyzer. The images from the internet in Figure 3 and Figure 4 (i)-(iii) show outputs comparable to the images generated by plotting from available datasets, in Figures 4 (iv), 5, and 6. Thus, our image preprocessing methodology helps in improving the quality of the image to be used with ScatterPlotAnalyzer, even though the images from the internet tend to have lower resolution, noise, and aliasing.

Scatteract reports a maximum of 89.2% success rate, where F_1 score > 0.8 for a plot implies success in data extraction. In our experiments, we do not get F_1 score distribution similar to that of Scatteract, i.e. very low and very high values. Instead, we get a continuous uniform distribution for $F_1 \geq 0.4$. 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. The chart classification accuracy for scatter plots is at 81% in our work, comparable to 86% in ChartSense [11], while the best is at 98% in the method by Choi *et al.* [13]. Our chart classifier, which has been trained and tested with images from the web, requires more pre-training to improve the accuracy rate. Despite the mismatches in quantitative assessment, our results are comparable to those from Scatteract and the method by Choi *et al.* , qualitatively.

Pre-training the CNN is the most time-consuming process in our algorithm. The tensor field computation is also compute intensive. Its implement ion can be made more efficient by using parallel implementation, owing to its embarrassingly parallel characteristic. Sparsification of the degenerate points is required for improving the accuracy of our data extraction process. However, sparsification itself can be improved using an analytical choice of threshold, as opposed to the heuristic we use generically. Thus, the scope of our future work includes improving the overall performance and accuracy of ScatterPlotAnalyzer.

6 Conclusions

In this work, we have proposed ScatterPlotAnalyzer, an algorithm for extracting data from images of scatter plots. We have focused on both simple and multi-class scatter plots, where the class information of the scatter points is encoded using color. We have designed ScatterPlotAnalyzer with the central theme of second-tensor fields using tensor voting for geometry extraction. We use a Convolutional Neural Network inspired by VGGNet architecture to classify the scatter plots. We then use human-guided image annotation to extract the canvas containing chart objects, where the interactive annotation makes our algorithm semi-automated. We use image preprocessing to complete the extraction, and scatter points themselves are extracted from the topological analysis of the tensor field. The postprocessing involves the use of deep-learning OCR to localize and detect text. Text is an important ingredient for contextualizing the dataset and converting the extracted data from pixel to the original data space. For identifying the class information from the extracted scatter points, we use information extracted from the legend. Overall, ScatterPlotAnalyzer is an end-to-end algorithm for extracting data from images of scatter plots.

Performance characterization and improving accuracy are organically the next steps to improve ScatterPlotAnalyzer. We require metrics such as the density of scatter points and overlap between multiple classes to determine the accuracy of ScatterPlotAnalyzer implemented on such an image. ScatterPlotAnalyzer is a good proof-of-concept of exploiting image properties for extracting information from chart images. ScatterPlotAnalyzer will be beneficial for aiding other learning approaches as an integrated solution.

ACKNOWLEDGMENTS

This work has been funded by the Machine Intelligence and Robotics Center (MINRO) grant to the International Institute of Information Technology Bangalore (IIITB) by the Government of Karnataka. The authors are grateful for the discussion with T. K. Srikanth, IIITB; Sindhu Mathai of Azim Premji University; Vidhya Y. and Supriya Dey of Vision Empower; Neha Trivedi, XRCVC; Vani, Pushpaja, Kalyani, and Anjana of Braille Resource Center, Matruchayya, that has shaped this work. The authors are thankful for the helpful comments from anonymous reviewers.

References

[1] Samira Ebrahimi Kahou, Vincent Michalski, Adam Atkinson, Ákos Kádár, Adam Trischler, and Yoshua Bengio. FigureQA: An annotated figure dataset for visual reasoning. *arXiv preprint arXiv:1710.07300*, 2017.

[2] Nitesh Methani, Pritha Ganguly, Mitesh M Khapra, and Pratyush Kumar. PlotQA: Reasoning over Scientific Plots. In *The IEEE Winter Conference on Applications of Computer Vision*, pages 1516–1525, 03 2020.

[3] Jorge Poco and Jeffrey Heer. Reverse-Engineering Visualizations: Recovering Visual Encodings from Chart Images. In *Computer Graphics Forum*, volume 36, pages 353–363. Wiley Online Library, 2017.

[4] Komal Dadhich, Siri Chandana Daggubati, and Jaya Sreevalsan-Nair. BarChartAnalyzer: Digitizing Images of Bar Charts. In *(to appear) in the Proceedings of the International Conference on Image Processing and Vision Engineering (IMPROVE 2021)*. INSTICC, 2021.

[5] Manolis Savva, Nicholas Kong, Arti Chhajta, Li Fei-Fei, Maneesh Agrawala, and Jeffrey Heer. Revision: Automated Classification, Analysis and Redesign of Chart Images. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*, pages 393–402. ACM, 2011.

[6] Mathieu Cliche, David Rosenberg, Dhruv Madeka, and Connie Yee. Scatteract: Automated extraction of data from scatter plots. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 135–150. Springer, 2017.

[7] Jaya Sreevalsan-Nair, Komal Dadhich, and Siri Chandana Daggubati. Tensor Fields for Data Extraction from Chart Images: Bar Charts and Scatter Plots. In Ingrid Hotz, Talha Bin Masood, Filip Sadlo, and Julien Tierny, editors, *Topological Methods in Visualization: Theory, Software and Applications (in press)*. Springer-Verlag, and arXiv preprint, 2020.

[8] Ankit Rohatgi. Webplotdigitizer, 2011.

[9] Aaron Baucom and Christopher Echanique. ScatterScanner: Data Extraction and Chart Restyling of Scatterplots. 2013.

[10] Leilani Battle, Peitong Duan, Zachery Miranda, Dana Mukusheva, Remco Chang, and Michael Stonebraker. Beagle: Automated Extraction and Interpretation of Visualizations from the Web. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, page 594. ACM, 2018.

[11] Daekyoung Jung, Wonjae Kim, Hyunjoo Song, Jeong-in Hwang, Bongshin Lee, Bohyoung Kim, and Jinwook Seo. ChartSense: Interactive Data Extraction from Chart Images. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, CHI '17, page 67066717, New York, NY, USA, 2017. Association for Computing Machinery.

[12] Noah Siegel, Zachary Horvitz, Roie Levin, Santosh Divvala, and Ali Farhadi. FigureSeer: Parsing result-figures in research papers. In *European Conference on Computer Vision*, pages 664–680. Springer, 2016.

[13] Jinho Choi, Sanghun Jung, Deok Gun Park, Jaegul Choo, and Niklas Elmqvist. Visualizing for the non-visual: Enabling the visually impaired to use visualization. In *Computer Graphics Forum*, volume 38, pages 249–260. Wiley Online Library, 2019.

[14] Gérard Medioni, Chi-Keung Tang, and Mi-Suen Lee. Tensor Voting: Theory and Applications. *Proceedings of RFIA, Paris, France*, 3, 2000.

[15] Jaya Sreevalsan-Nair and Beena Kumari. *Local Geometric Descriptors for Multi-Scale Probabilistic Point Classification of Airborne LiDAR Point Clouds*, pages 175–200. Springer Cham, Mathematics & Visualization, 2017.

[16] Tai-Pang Wu, Sai-Kit Yeung, Jiaya Jia, Chi-Keung Tang, and Gerard Medioni. A Closed-Form Solution to Tensor Voting: Theory and Applications. *arXiv preprint arXiv:1601.04888*, 2016.

[17] Rodrigo Moreno, Luis Pizarro, Bernhard Burgeth, Joachim Weickert, Miguel Angel Garcia, and Domenec Puig. Adaptation of tensor voting to image structure estimation. In *New Developments in the Visualization and Processing of Tensor Fields*, pages 29–50. Springer, 2012.

[18] Shengfa Wang, Tingbo Hou, Shuai Li, Zhixun Su, and Hong Qin. Anisotropic Elliptic PDEs for Feature Classification. *Visualization and Computer Graphics, IEEE Transactions on*, 19(10):1606–1618, 2013.

[19] Tzutalin. Labelimg. `https://github.com/tzutalin/labelImg`, 2015.

[20] Rodrigo Moreno, Miguel Angel Garcia, Domenec Puig, and Carme Julià. Edge-preserving color image denoising through tensor voting. *Computer Vision and Image Understanding*, 115(11):1536–1551, 2011.

[21] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *KDD*, volume 96, pages 226–231, 1996.

[22] Youngmin Baek, Bado Lee, Dongyoon Han, Sangdoo Yun, and Hwalsuk Lee. Character region awareness for text detection. *CoRR*, abs/1904.01941, 2019.

[23] Jeonghun Baek, Geewook Kim, Junyeop Lee, Sungrae Park, Dongyoon Han, Sangdoo Yun, Seong Joon Oh, and Hwalsuk Lee. What Is Wrong With Scene Text Recognition Model Comparisons? Dataset and Model Analysis. *CoRR*, abs/1904.01906:4714–4722, 10 2019.

[24] Ray Smith. An overview of the Tesseract OCR engine. In *Ninth international conference on document analysis and recognition (ICDAR 2007)*, volume 2, pages 629–633. IEEE, 2007.

[25] Gordon Kindlmann. Superquadric Tensor Glyphs. In *Proceedings of the Sixth Joint Eurographics-IEEE TCVG conference on Visualization*, pages 147–154. Eurographics Association, 2004.