2021

# A Deep Understanding of Structural and Functional Behavior of Tabular and Graphical Modules in Technical Documents

Michail Alexiou
*Wright State University*

# A DEEP UNDERSTANDING OF STRUCTURAL AND FUNCTIONAL BEHAVIOR OF TABULAR AND GRAPHICAL MODULES IN TECHNICAL DOCUMENTS

A Dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy

by

MICHAIL ALEXIOU

Dipl., Technical University of Crete, Greece, 2019

2021

Wright State University

WRIGHT STATE UNIVERSITY

GRADUATE SCHOOL

Dec 8, 2021

I HEREBY RECOMMEND THAT THE DISSERTATION PREPARED UNDER MY SUPERVISION BY MICHAIL ALEXIOU ENTITLED A DEEP UNDERSTANDING OF STRUCTURAL AND FUNCTION BEHAVIOR OF TABULAR AND GRAPHICAL MODULES IN TECHNICAL DOCUMENTS BE ACCEPTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF Doctor of Philosophy.

Nikolaos G. Bourbakis, Ph.D.
Dissertation Director

Yong Pei, Ph.D.
Director, Computer Science and Engineering Ph.D. Program

Barry Milligan, Ph.D.
Vice Provost for Academic Affairs
Dean of Graduate School

Committee on Final Examination:

Nikolaos G. Bourbakis, Ph.D.

Soon M. Chung, Ph.D.

Bin Wang, Ph.D.

Euripides G.M. Petrakis, Ph.D.

George A. Tsihrintzis, Ph.D.

# ABSTRACT

Alexiou, Michail. Ph.D., Department of Computer Science and Engineering, Wright State University, 2021. A Deep Understanding of Structural and Functional Behavior of Tabular and Graphical Modules in Technical Documents.

The rapid increase of published research papers in recent years has escalated the need for automated ways to process and understand them. The successful recognition of the information that is contained in technical documents, depends on the understanding of the document's individual modalities. These modalities include tables, graphics, diagrams and etc. as defined in Bourbakis' pioneering work. However, the depth of understanding is correlated to the efficiency of detection and recognition. In this work, a novel methodology is proposed for automatic processing of and understanding of tables and graphics images in technical document. Previous attempts on tables and graphics understanding retrieve only superficial knowledge such as table contents and axis values. However, the focus on capturing the internal associations and relations between the extracted data from each figure is studied here. The proposed methodology is divided into the following steps: 1) figure detection, 2) figure recognition, 3) figure understanding, by figures we mean tables, graphics and diagrams. More specifically, we evaluate different heuristic and learning methods for classifying table and graphics images as part of the detection module. Table recognition and deep understanding includes the extraction of the knowledge that is illustrated in a table image along with the deeper associations between the table variables. The graphics recognition module follows a clustering based approach in order to recognize middle points. Middle points are 2D points where the direction of the curves changes. They delimit the straight line segments that construct the graphics curves. We use these detected middle points in order to understand various features of each line segment and the associations between them. Additionally, we convert the extracted internal tabular associations and the captured curves' structural and functional behavior into a common and at the same time unique form of representation, which is the Stochastic Petri-net (SPN) graphs. The use of SPN graphs allow for the merging of different document modalities through the functions that describe them, without any prior knowledge about what these functions are. Finally, we achieve a higher level of document understanding through the synergistic merging of the aforementioned SPN graphs that we extract from the table and graphics modalities. We provide results from every step of the document modalities understanding methodologies and the synergistic merging as proof of concept for this research.

# List of Chapters

# List of Figures

## List of Tables

# Acknowledgments

I would like to express my sincere gratitude towards my supervisor, Dr. Nikolaos Bourbakis, for his constant encouragement and guidance during my efforts to complete my PhD. I would also like to thank him for giving me the chance to collaborate with him on his unique research projects as a Graduate Research Assistant and for supporting me through his grant. He has been a true mentor to me, both in my research career and in life and I sincerely hope that we will continue our collaboration beyond the completion of my PhD.

I am also grateful to Dr. Soon Chung, Dr. Bin Wang and Dr. George A. Tsihrintzis for reviewing this work and participating in the examination committee. Especially, I would like to express my sincere gratitude towards Dr. Euripides G. M. Petrakis for his constant mentorship and support during both my undergraduate and graduate years. Through his work and advice he inspired me to pursue a PhD, he gave me the opportunity to join a U.S. university and he laid the foundation for the researcher that I am today. Additionally, I would like to thank my colleagues in the Center of Assistive Research for their meaningful discussions and constant moral support.

Finally, I would like to thank all my friends, especially Kostas, Yiannis, Vasia, Alkis, Panos, Angelos, Sifis and Katerina for always being there for me in times of need. I would also like to express my love and gratitude towards my girlfriend Olga for her patience, love and for showing me the better side of life and myself.

*This dissertation is dedicated to my family who have supported me during my efforts to complete this research. More specifically, I would like to thank my father Stelios, my mother Aleka and my sister Maria, who have given me their unconditional love and psychological support through all these years. Especially, to my dear cousin Lina who through her life experiences has taught me the true meaning of never giving up.*

This dissertation is partially supported by an ONR-BAA grant 2018-2021 and DARPA. Any opinions, findings, and conclusions or recommendations expressed in this material are of those  of the author and do not necessarily reflect the views of the ONR or DARPA.

# 1. Introduction

The rapid increase of published research papers in recent years has escalated the need for automated ways to process them and capture their illustrated information. This need has elevated the field of automatic technical document understanding into a significant research topic. The accurate understanding of technical documents depends on the proper processing of their individual modalities. Modalities such as diagram and graphics images, tables and the natural language text contain important information about the functionality and architecture of the proposed methodologies. However, data analysis must not be bounded to understanding of information that is visually accessible. It should be expanded to accommodate the extraction of the deeper meanings and connections between them. Thus, the retrieval and combination of their internal information can lead to a better understanding of the overall document itself.

As we have already mentioned, each of these individual modalities and their associations with each other provide significant insight about the understanding of the technical document's represented knowledge. With the exception of the natural language text, we consider the rest of the extracted modalities as "images". We present the diagram that describes the aforementioned technical document understanding methodology [75, 76, 86 - 91] in figure 1. This research is focused on developing a methodology where all these modalities can be expressed into the same format (natural language sentences) and a mathematical model. We choose the Generalized Stochastic Petri-net graphs (SPN) as that mathematical model, in order to examine better associations and achieve a deeper understanding of the technical document. We achieve the deep understanding through 3 specific contributions. The first unique contribution is the enrichment of the natural language text part with additional natural language sentences extracted from the aforementioned tabular and graphics "images" of the technical document. The second unique contribution comes from the SPN models of these "images" that compliment the main diagram by generating a simulator for the architecture that the input technical document illustrates. The third unique contribution is the generation of additional new knowledge through the combination of tabular and graphics SPN graphs, which was not available prior to that synergistic merging.

In this research we consider a technical document (TD) as the composition of two parts. The first part contains the natural language text and text captions of images, titles, sections, paragraphs and sentences. The second part includes all the modalities that we consider "images", which are system diagrams (block diagrams), tables, mathematical formulas, graphics, charts and pictures. Thus, each technical document has natural language parts and images parts which we detect and extract through specific software tools.

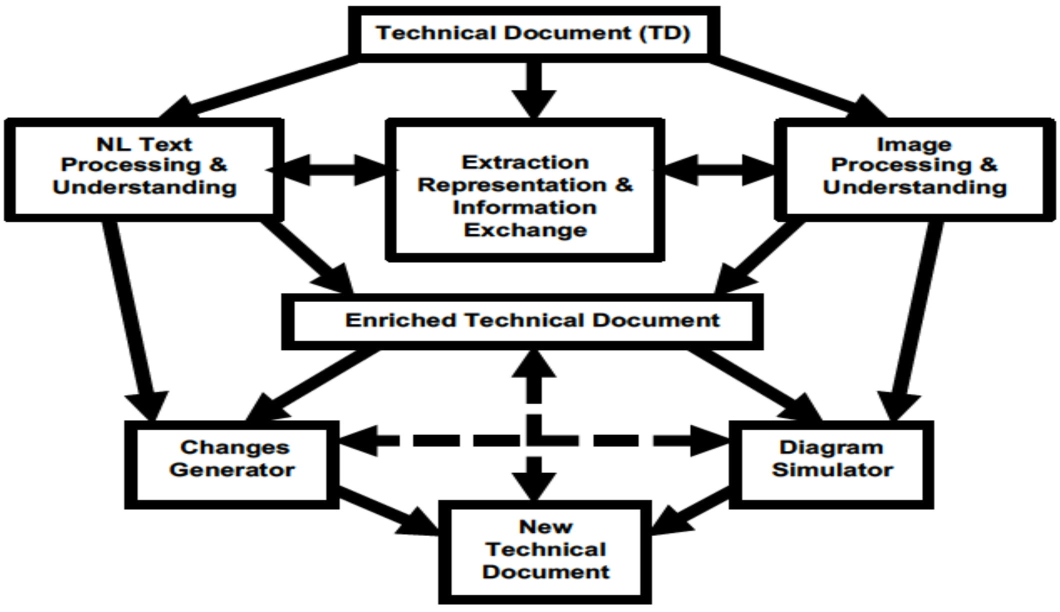

Technical Document (TD)
NL Text Processing & Understanding
Extraction Representation & Information Exchange
Image Processing & Understanding
Enriched Technical Document
Changes Generator
Diagram Simulator
New Technical Document

*Figure 1: The Technical Document Deep Understanding Methodology [75, 76, 86 - 91].*

More specifically, we introduce a novel methodology for the automatic detection, recognition and understanding of tabular information and graphics relations that is contained in technical documents. We define the table structure as the collection of data variables and values, which are arranged in rows and columns. The arrangement of these data must be in a meaningful way that indicates the underlying associations between the variables, in order to be visually recognizable. We define a graphics figure, which contains either curves or bars, as the visual representation of the variations in the values of one or multiple variables with respect to the progression of time. These variables correspond to either different curves or bars. However, the arrangement of these data must be in a meaningful way so that their underlying associations are visually recognizable. We are motivated by the work presented by Bourbakis et. al. [1], where a strategy is discussed for the conversion of different technical document modalities into SPN graphs. The overall goal of our research is to recognize the knowledge that is illustrated in different technical document modalities and combine them through a common representation format, in order to achieve complete and better understanding. Therefore, the SPN graph is selected as the optimal representation format, since it is capable of describing the internal functionality of the processed modalities.

## Chapter Overview

The remainder of the dissertation is organized as follows.

**Chapter 2: Literature Review** details the literature review regarding tabular and graphics images processing. More specifically, in Section 2.1 we examine work related to table detection and recognition. In Section 2.2 we cover the work related to graphics image detection, recognition and understanding. Additionally, in each section we perform a subjective evaluation to determine which studied methodology best aligns with the current research. A different maturity score is assigned to each methodology in order to make it comparable to the others.

**Chapter 3: Processing of Tables Images** examines the overall proposed methodology for the detection, recognition and understanding tabular images in technical documents. Section 3.1 discusses the general table processing methodology as well as its goals and contributions. Section 3.2 provides a comparative evaluation on different table detection methodologies. These methodologies are categorized into rule-based, machine learning based and hybrid. Section 3.3 details the grammar, alphabet and operators of the Pinakas formal language. We have developed this language as the means to map the retrieved tabular information into SPN graphs. Section 3.4 provides the methodology steps for the extraction and recognition of tabular information, such as table variables and values. The proposed methodology covers different types of table structures. Section 3.5 discusses the methodology steps for the understanding of the internal associations and relations between the different table values and variables. Furthermore, it provides the results from the conversion of the retrieved knowledge into other forms of representation. These formats include attributed graphs, natural language sentences and SPN graphs.

**Chapter 4: Processing of Graphics Images** discusses the overall proposed methodology for the detection, recognition and understanding graphics images in technical documents. Section 4.1 presents the general graphics processing methodology and its features. Section 4.2 performs a comparative evaluation on different graphics detection methodologies. These methodologies are separated into rule-based, machine learning based and hybrid categories. Section 4.3 presents the grammar, alphabet and operators of the Kyrtos formal language. We introduce this formal language in order to help us map the extracted graphics relations into SPN graphs. Section 4.4 provides the methodology steps for the extraction and recognition of graphics information, such as axis legends text and the individual line segments the produce the curves. The proposed methodology covers different 3 types of graphics images. Section 4.5 discusses the methodology steps for the understanding of the structural and functional information that are illustrated in a graphics image. We convert these information into natural languages sentences which are later mapped into SPN kernels using the Kyrtos formal langauge.

**Chapter 5: Function Regeneration from Graphics Images** provides the methodology for the recognition of the function that describes the behavior of a given curve. Section 5.1 illustrates the algorithm that recognizes the corresponding funtion type of the curve. Section 5.2 presents the methodology and results of function regeneration for linear curves. Section 5.3 presents the methodology and results for the regeneration polynomial function based curves. Section 5.4 illustrates the methodology and results for the function regeneration of asymptotic curves. Section 5.5 discusses the processing steps and results of the function regeneration methodology in the case of sinusoidal curves. Finally, Section 5.6 covers the processing steps for the approximation of functions that describe curves of arbitrary shape using Cubic Splines.

**Chapter 6: Synergistic Merging of Graphical and Tabular Information** examines the proposed methodology for the combination of extracted tabular and graphical information. More specifically, Section 6.1 presents the overall matching and merging methodology. Section 6.2 presents the results of the table recognition and understanding methodologies for a given table use case. Section 6.3 illustrates the results of the graphics recognition and understanding methodologies for a specified graphics image input. Finally, Secgtion 6.4 presents the results of matching and combining the respective SPN graphs of the 2 modality images. It must be noted that both of these images come from the same technical document and, thus, exists an underlying association between them that needs to be found.

**Chapter 8: Conclusions** summarizes the contributions of this dissertation.

**Chapter 7: Future Work** discusses potential future work in terms of improving the accuracy of the presented recognition and understanding methodologies, optimizing their performance, introducing machine learning to their procedures and examining more advanced and complex data.

# 2. Literature Review

## 2.1 Processing of Tables Images

### 2.1.1 Rule-based Detection of Tables

Laurentini et al. [56] present an algorithm for the identification of tables in document images based on the location of intersecting horizontal and vertical lines, as well as rectangular regions containing text. The algorithm initially detects image connected components that form words. The  words are merged to form phrases based on their horizontal or vertical distance. The algorithm detects lines of continuous black pixels. Whether the pixels are accepted as part of a line depends on a threshold related to the average font size. In this way, the method also handles  skew images. If the formed lines are too close to each other, they are merged into single vertical or horizontal lines. If necessary, certain adjustments are applied to restore lines from errors. Image regions  containing lines are potential regions interest. If the layout formed by the intersecting lines matches with the alignment of the rectangular texts in a certain region, then the intersecting lines are considered to form table cells. The final conclusion depends on whether these formed table cells appear on a certain pattern, in order to ensure to that the region of interest above  form a table structure.

Green et al. [60] proposed a methodology for detecting logical relations between cells that belong to tables in document images. Their model describes  tables based on the connections between table cells. A separate set of rules is applied to distinguish a table from the background image. The algorithm iterates many times in order to adjust confidence weights to vertical and horizontal lines that may form table lines. Further iterations will extract inner cells in each detected table.

The solution by Tupaj et al. [50] applies a sequence of four steps: (a) image segmentation in order to detect regions that may contain tables, (b) Optical Character Recognition (OCR) within areas of interest for text extraction, (c) text analysis for detecting the start and ending of a table and, (d) semantic analysis for extracting table components such as title, cell values etc. The image's deskewing angle is computed and the image is rotated (by the amount of skew) so that, the lines in the image are aligned horizontally and vertically. Horizonyal and vertical lines in the image define sections that may contain tables. The method selects those sections that are most likely to contain tables based on the white space distances and the keywords that have been identified. OCR extracted text is parsed and table boundaries are identified by applying a set of format rules for blank spaces and lines containing multiple text words between any two of them. Vertical table separators are detected based on the orientation of text chunks.

A similar approach by Mandal et al. [65] makes the assumption that the gaps between words in table cells are always larger than those between the words in text areas. Image connected components are extracted first in order to recognize any text words and the lines containing those words. The text words are treated as blobs of black pixels and each line as a set of blobs. The size of the gap between consecutive blobs in a line is obtained from their distance histogram. Based on distance information, words that belong on a text line are combined to form a single text block with no distances them. Otherwise, the method checks for table lines. A table line is identified as one that contains multiple pixel blobs with a certain distance between them. The method also attempts to recapture any missed table lines, by checking the neighborhood of those that are already considered as candidate table lines.

Shamilian et al. [43] present a methodology for detecting tables in images of printed documents based on layout styles defined by the user. More specifically, the user defines important details about the layout of the targeted table using a Graphical User Interface (GUI). These details include frequency of characters, ink template, minimum and maximum characters that may be contained within a table field. The algorithm begins by applying certain image processing operations on the document image such as connected component extraction, skew correction, erasure of any ruling lines and separation of text lines from the rest of the document. The text lines are processed using the defined ink templates to determine whether they belong to a table or not. The location and the total number of tables for a document image is based on the length of ink templates associated with that document page. The method attempts to match the template of the text lines with the predefined ink templates ones. The text lines that match to a template are stored for further processing. Finally, the method retrieves the exact beginning and end of a table using horizontal shifts on the stored text lines.

Jianying et al. [59] propose two algorithms, which in combination can detect tables in a wide range of document types, including images. The first algorithm detects candidate table lines which, when merged, form regions with tables. The method applies a heuristic rule in order to identify image regions with high probability of containing tables and avoid overlays with (possibly erroneous) tables that are detected with less confidence. The second algorithm computes the correlation between words that belong to different text lines based on the spaces between them. If two lines correlate well, they could potentially be part of the same table column. In ASCII documents, this algorithm searches for ASCII words, while in images the algorithm searches again for connected components.

Kieninger et al. [70,71] propose T-Recs, a tool for detecting tables in documents based on the bounding boxes of document words extracted by applying OCR. In the first version of the tool [70], the authors cluster the recognized bounding rectangles of the text words into segments. Words of adjacent text lines are clustered into the same segment, while words of neighboring columns end-up in different clusters. Detection of tables is based on the recognition of multiple column gaps across detected horizontal lines in the segmentation graph. T-Recs achieved high precision for recognition of tables in scientific documents, since these tables obey to predefined templates. Corporate documents might not follow the same structural rule, thus an improved version of T-Recs is presented in [71]. This version, groups text bounding boxes with similar width and height, that might also overlap with each other on the vertical axis. The groups are called column fragments and can potentially span over multiple table cells. The algorithm clusters column fragments instead of bounding boxes. Any clustering errors are corrected using  the y-axis projections of the column fragments.

Neves et al. [61] present a methodology for automatic detection of structures enclosed by vertical and horizontal lines. Their detection relies on identifying intersections between those vertical and horizontal lines. The methodology begins with locating intersections and proceeds by detecting and correcting any errors in the identification process. Identification of intersections is accelerated by applying a set of four rules  that define which pixels patterns are accepted as intersections. These patterns when eroded and combined can produce a total of nine intersection pixel formations. In order to identify a candidate pixel formation as intersection, it is compared against all intersection patterns, even those that come from unification, starting with the highest level pattern.

Wang et al. [40] propose a methodology for automatic detection of tables in images. They divide the problem into the sub-problems of identifying the table and decomposing it. The algorithm begins by classifying the document layout into either single column, double column or mixed column type based on the recognition of blank separators in the document structure. Then, it proceeds to detect horizontal blank blocks and group those blocks and their adjacent words together based on vertical alignment criteria. These groups indicate the candidate regions for containing tables, which are going to be validated using a set of predefined features. Finally, the algorithm computes the vertical projection of the words contained within each table. Individual table columns are extracted using the resulting valleys and peaks as reference points.

Cesarini et al. [72] propose Tabfinder, a tool that detects tables in images of document pages by making use of its corresponding Modified X-Y (MXY) tree representation [73]. The algorithm is trained to identify specific table types, based on certain information extracted by the MXY tree. The method starts with the recursive parsing of the MXY tree for a given page, in order to identify regions enclosed by vertical and horizontal lines. Then, it searches for more vertical and horizontal lines inside those regions and, if two or more lines are detected, these regions are recognized as tables. Otherwise, it searches for perpendicular cuts within them. The mage areas that are most likely to be part of the same table are accepted and merged to form a table. Areas that are less likely to be part of a table are discarded (based on a user-defined threshold). Two methods for computing optimal thesholds are proposed and compared. The first method iterates over  all the possible threshold combinations to find the one that maximizes the table detection accuracy. The second method applies a variation of the Nelder and Mead methodology [74], which is modified in order to replace the value of the threshold with the closest from a set of values. The closest value is the one with the minimum Euclidean distance.

Gatos et al. [38] propose a methodology with 3 main stages for automatic detection of tables in images. The first stage is image pre-processing, the second stage is about spotting the vertical and horizontal lines in the image and, the final stage is the one that detects the tables. Pre-processing begins with the binarization of the image before enhancing it followed by adaptive method, which is useful for the correction of distorted images. Then, the method deduces the text-orientation and corrects the skew by applying a fast Hough transform. Any noisy borders are ereases and the average character height is computed. Character height depends on the rectangle heights of the neighboring connected components in the image. The method then detects lines in the image of the document by following and combining adjacent black pixels to form lines. To accept the lines, they are compared to the previously calculated average character height. Lines accepted are improved after the removal of any image and text regions. These regions are detected after applying a smoothing filter on the image. The table detection process depends on the recognition of intersections between the previously detected horizontal and vertical lines. Once the intersection detection process is complete, the method determines the structure of the table by grouping all the intersections detected and restoring the line brakes where necessary.

The method by Deivalakshmi et. al. [63] also relies on detection of intersections. The method detects horizontal and vertical lines on the binary transformed version of the input document image. The detection of the lines is based on mathematical grouping operations applied on sequences of adjacent black pixels. The algorithm then detects the points of intersection between vertical and horizontal lines by applying the AND operation in pairs of lines. These points of intersection define tables that have ruling lines.

Harit et al. [68] present a technique for detection of tables in document images, based on the recognition of header and trailer patterns from the documents layout. More specifically, the solution analyzes the layout next to the recognized vertical and horizontal separators, in order to detect patterns that indicate existence of tables. It begins by searching for vertical and horizontal separators in the document image, in order to segment the image into individual data blocks, called patches. A user-defined set of rules for recognizing patterns based on the layout on these patches is defined in correlation with the position of the separators that created them. These rules require that different patterns will emerge, when there are differences in the layout features and the contents of regions. These regions are left and right or up and down from the detected separators. After the detection of patterns is completed, the algorithm computes the boundaries for the regions that contain these patterns. If vertical ruling lines exist, they are used for the definition of the left and right boundaries. Otherwise, the left and right boundaries are deduced based on the spatial expansion of the horizontal ruling lines, which border the top and bottom of the table. If there are no ruling lines at all, whitespaces will be identified and used as conceptual separators between columns and rows of tables. The final predictions are based on the analysis of the text components next to separators, as well as the separators themselves.

Jahan et al. [58] propose a methodology for detecting and extracting tables in scanned document pages based on set of rules. The method searches for  all the horizontal text lines in the image using X and Y projections. Then, the algorithm calculates the space between the words in each line. In this way,  plain text lines are distinguished from lines that belong to a table, since a text line has more words and, thus, a greater number of spaces between them, than the lines of a table. The thresholds for the acceptance of table line heights, as well as for the distances between the contents of each cell are based on the heights and spaces of the words in the document image..

The method by Tran et al. [66] is rule-based methodology and relies on recognition of alignment characteristics of connected components. It relies on the assumption that a table will always contain aligned text blocks inside it. If an image region is detected, if it is enclosed by vertical or horizontal lines and, if there are multiple text blocks within this region   aligned vertically and horizontally, then this region is a table. After some standard pre-processing steps and after  line brakes are corrected (using morphological features), the algorithm extracts image connected components and computes their bounding boxes. Then, it searches for regions that can contain tables by detecting bounding boxes that contain other bounding boxes of identified connected components or by detecting only horizontal lines. For the latter case, the algorithm finds an horizontal line and creates a region of interest around it. It then expands this region of interest in a top-down direction, in order to enclose bounding boxes of connected components. If there are multiple text blocks contained within region and the text blocks are aligned vertically with the text blocks of the other lines, then that region is recognized as a table. If any outcome region end-up overlapping with other regions, they are merged to form a single table.

Anh et al. [35] present a solution as a follow-up of their original work [36], which introduced a hybrid methodology for document page segmentation based on the idea of multilevel homogeneity. in order to extract its contents and their structural format. Initially, the image connected components are extracted [37] and classified into text and non-text elements based on predetermined heuristics [36]. The text lines are detected and then deleted keeping their location only. Regions of potential tables are extracted based on the layout features of text lines and non-text elements. These tables are classified into ruling line tables (i.e. color, close, non-close, and parallel) or non-ruling line tables. The detection of color tables relies on consolidating detected rectangular components taking into consideration the (small) distance between them and their size. If the detected components can be arranged into rows and columns, then, the resulting region is a (potential) color table. In order to classify a structure as close or non-close table region, three conditions are checked: (a) the density of the contents within the rectangular components is low, (b) components and text elements do not overlap and (c) there are rectangular components that contain other smaller components. If all of these conditions are satisfied, then the candidate region is classified as a potential close or non-close table region. On the other hand, the regions that contain either only vertical line elements or horizontal line elements are categorized as potential parallel table regions. Finally, for non-ruling line table detection, the method checks for homogeneity in the formed regions inside the text line document. Eventually, table regions are derived as the combination of ruling line regions and,  non-ruling line regions.

### 2.1.2 Machine Learning Based Detection of Tables

Gilani et al. [31] combine image processing and deep learning techniques. Prior to detection, document pages or images are pre-processed to separate text from non-text regions. Images in the output are processed by applying Faster R-CNN [54]. This happens in order to transform them into natural images, since the authors are going to use the Faster R-CNN [54] detection and classification framework. The transformation of an input (binary) image to natural image occurs by applying a combination of distance transformation techniques (i.e. Euclidean distance, linear distance and max distance transformations) on all three channels of an RGB image (red, green, and blue). The processed image is then passed into a fine tuned Zeiler and Fergus model [34], which outputs the feature map. The Region Proposal Network (RPN) of Fast-CNN [54] takes the resulting feature map and parses it through a convolutional layer of its own in order to predict possible regions that contain tables. Both the original feature map and the proposed regions are parsed into fully connected layers. Finally, a Fast R-CNN detection module classifies those regions as tables or not  and returns the bounding boxes of the predicted table regions.

Schreiber et al. [46] propose a deep learning-based approach for the detection of tables in document images and PDF files. Taking advantage of transfer learning, the method employees a fine-tuned pre-trained Faster-RCNN model for spotting tables in documents. Similarly, they apply transfer learning for recognizing the structural features of tables  by fine-tuning a Fully Convolutional Network model for semantic segmentation [47]. Overall, the method relies on Faster R-CNN and suggests two versions of the model to be compared. The first is based on ZFNet [34] and the second one is based on the VGG-16 network [48]. After locating the position of the table in the image, the structural elements (i.e. rows, columns and cells) are identified. The implements an FCN - 2s model, which includes 2 more skip connections. These connections combine characteristics of both the first and the second pooling layers. To make the method scale invariant and capable of predicting scaling factors from a training data set, scale layers are replaced by normalization layers  in the example of [49].

The method by Siddiqui et al. [44] applies a combination of convolutional neural networks (CNNs) using deformable convolution and Faster R-CNN [54]. Traditional CNN approaches rely on extracting images features from their respective layers. Each layer comprises a number of neurons, each receiving a feature map with a specified size as input. This can be problematic for cases of tables in images of varying sizes, because important information can be lost during resizing. Deformable convolution solves this problem by allowing the neurons to adapt their acceptance size, based on the size of the previously given feature map. This type of convolution is achieved through the use of specific offsets, which gain their values from an external convolutional module. The same issue can be detected in the ROI pooling segment as well, causing again loss of valuable information before the final classification. The authors also introduce deformable ROI pooling, for dynamic adjustment of the acceptance limits to the different scales of the received feature maps.

Arif et al. [67] illustrate a deep learning technique for detection tables in document images, based on the different layout and foreground features of the document. The proposed methodology begins with pre-processing of the input image. More specifically, the algorithm searches for numeric characters and text characters in the image. Any numeric data are marked in red and, all text is marked in green. Then, Euclidean distance transformation is applied to the blue channel of the image, in order to enhance the background features of the document. As a result, red and green channels contain foreground information, while the blue channel contains background information. Tables are detected by a Faster R-CNN module. Initially the image is parsed by a pre-trained model like VGG-16 and ZF model (trained on ImageNet [55]) that performs accurate feature detection and outputs the corresponding feature maps. These maps are passed to an RPN module (i.e. typically a CNN), that generates a list of proposals about what regions to search for tables in the document. The detector network is a Fast R-CNN module, that accepts the region proposals and outputs prediction scores and bounding rectangle positions.

Saha et al. [52] present a deep learning-based methodology for detecting graphical objects such as tables, equations and figures in technical document pages that are converted to images. They composed two versions of the model for comparative evaluation. The first version of the model is based on Faster R-CNN[54], while the second one is based on Mask R-CNN[53]. These two algorithms were selected because of their proven good performance. Both models are very similar and they consist of a backbone CNN module and a standard RPN module. The difference is that, Faster R-CNN uses ROI pooling to generate the respective feature maps to be parsed by the fully connected layers for box and class prediction. On the other side, the Mak R-CNN uses ROI align for generating the feature maps for parsing and final prediction. In the case of Mask R-CNN the feature maps are also parsed through a specific module that gives the segmentation mask for each candidate region. The authors exploit the advantages of transferring learning for the training of their models, by deploying the pre-trained VGG-16 and ResNet (trained on ImageNet [25]) network as the basis for their implementation.

Kavasidis et al. [5] propose a method for detecting tables and other graphical objects in document images using a saliency CNN. More specifically, they handle the problem of table detection as a semantic segmentation one, thus they employ the saliency detector to perform the segmentation. Their implementation is based on a VGG-16 network, but with kernels of size 7x3 instead of the traditional ones. In this way, they ensure recognition of tabular features. The method generates specific binary masks, where one mask corresponds to each class of the four author-defined classes. These masks can detect the boundaries of detected document objects, as long as these object belong to one of the set classes. A dilation layer at the end of the detector also ensures the preservation of context between the detected objects. A Conditional Random Field (CRF) model is added as an extra module to the methodology, that accepts the resulting feature maps and help correct any segmentation errors like false unification of what should have been separately recognized tables. Another extra module contains four binary classifiers each trained separately to recognized images that belong to one of the four classes of interest. This step improves the accuracy of the saliency CNN detector.

Riba et al. [69] present a deep learning technique for detecting tables in images of invoice documents based on the recognition of repeated structural patterns using Graph Neural Networks (GNN). They recognize tabular structures using the connection of different document elements. More specifically, the graph vertices represent connected components detected in the image (i.e. text blocks or lines) and the graph edges represent connections between them. The authors train a GNN model to recognize graphs that represent tabular structures in documents using supervised learning. The methodology begins with the analysis of the document image layout to retrieve regions that correspond to either text or graphics (vertical and horizontal lines). These regions are transformed to graph nodes that hold 7-dimensional vectors with a histogram representation of their contents and the positions of their bounding rectangles. Graph edges are placed between different graph nodes based on the connections of their corresponding document regions. The GNN contains Graph Residual Blocks that perform all the transformation on the vectors of the graph node and propagate the corresponding information to other nodes. A Graph Adjacency layer is used to learn the correlation between neighborhood connections and tabular structure recognition by updating the adjacency matrix. The final classifier is used to classify the graph nodes into different classes such as headers, tables etc. Once the categorization is completed, the edge classifier erases the edges between nodes that don't belong to the same class, thus improving the accuracy of the detected tables. The remaining graphs, whose nodes have been classified as tables with a high confidence, are given as final detection predictions.

### 2.1.3 Hybrid Detection of Tables

Kasar et al. [57] propose a method that detects vertical and horizontal lines in an image, extracts specific features of these lines and then parses these features using a Support Vector Machine (SVM). The SVM predicts whether these lines are part of a table structure using only the aforementioned features and without any developer input rule set. Images are pre-processed to enhance certain characteristics (e.g. black and thin lines) and  to correct skew. Then, the method searches for horizontal and vertical lines, and their intersections. The method relies on the assumption that regions with tables exhibit more line intersections of horizontal and vertical lines as opposed to the other document objects (e.g. graphics). Line features extracted are passed to an SVM classifier.

He et al. [62] propose a multi-scale model, which applies a pretrained VGG19 network followed by full-convolutional layers. In this way, the context of each detected document object is maintained. For example, text blocks inside tables are detected as parts of the table and not as individual text objects. Then, the resulting feature maps are parsed by two concurrent modules, one for semantic segmentation and one for contour recognition. Both these modules are networks that have been trained to perform their corresponding tasks, in contrast to other implementations which use dedicated heuristic algorithms. The generated feature maps are parsed by a Conditional Random Field (CRF) to get more optimized versions of the detected segments and contours. CRF has the advantage of taking the context of pixels into account (e.g. color differences and contour characteristics) and, therefore it can correct any errors in the original predictions. The resulting predictions go through a heuristic layer as an extra step to separate the tables from the extracted document segments. This heuristic module includes thresholding of the image and searching for connecting components that form tabular structures. A final CNN layer verifies if the detected regions belong to actual tables with the use of an Intersection over Unition (IoU) function. or not with the use of an Intersection over Union (IoU) function.

Li et al. [64] present a hybrid methodology for detecting tables, graphics and mathematical formulas in document images. Firstly, image components and their vertical projections are detected. Then column and line boundaries are detected by applying a set of heuristics. The lines detected are classified as tables, figures, text etc. by applying a CRF model with two CNNs. The first CNN performs classification based on individual line regions. The second CNN performs classification on each line region and its neighbor line regions. The categorized lines are parsed through a second CRF model, comprising (again) two individual CNNs. The second CRF model merges lines of the same class based on locality criteria. There are cases where multiple figures that share the same region are recognized as a single figure. In order to resolve the issue, the method applies vertical projection to split the central figure region into sub-regions. T sub-regions are merged to form single figures based on some heuristic rules. Finally, a verification CNN module is applied as an extra step to correct any classification errors and increase the overall accuracy of the method.

Huang et al. [42] propose a deep learning methodology for automatic detection of tables in images using a model based on YOLO v3 [11]. YOLO is a popular method that uses CNNs to detect natural objects rather than document objects (e.g. people, cars, building) in images and videos. Therefore, the basic YOLO algorithm has been modified to make it work for table detection. YOLOv3 already has its own anchor proposal system for regions that are most likely to contain tables. However, document objects and natural object are very different, and these anchors may not work well for tables. The method applies  K-Means clustering in the training set of tables, in order to get anchor proposals which are more suitable for detecting tables. The prediction results are analyzed  to improve the precision of detection. More specifically, post-processing removes the extra white margins (if detected) around a table. Then, document features such as page headers and footers are removed. The final predictions have to comply with specific structural rules, in terms of the number of pixels in the table area detected, the size ratios and the distance between the table area and the page borders.

### 2.1.4 Comparative Evaluation of Table Detection Methodologies

We evaluate the aforementioned table detection and recognition methodologies subjectively based on a selection of features presented in table 1. Each of the previously discussed methodologies has been trained and tested on different datasets, therefore a direct evaluation among their corresponding accuracy percentages would be pointless. Therefore, we have selected subjectively the features from table 1 that represent generic functionality in order to be able to compare the methodologies based on their calculated maturity score. Table 2 illustrates the corresponding weights for each feature. The weights for each each feature were determined through a survey of 4 people. More specifically, we have 2 weights for each features based on the user’s and and the developer’s perspective.

**Table I: Selected Evaluation Features**

| **Features** | **Description** |
|---|---|
| Reliability (**F1**) | Methodology produces expected results under normal conditions. |
| Robustness (**F2**) | Methodology produces results under extreme conditions. |
| Simplicity (**F3**) | Easiness of understanding and implementing the methodology. |
| Complexity (**F4**) | Computational complexity of the methodology (e.g. Memory, CPU, and GPU requirements). |
| Product (**F5**) | Potential for commercial deployment of the methodology. |
| Cost (**F6**) | Amount of money needed to use this methodology. |
| Speed (**F7**) | Processing time of the methodology. |
| Scalability (**F8**) | Ability to handle large amounts of data. |
| Portability (**F9**) | Easiness of deployment on various different systems. |
| Accuracy (**F10**) | The accuracy of the prediction results. |
| Efficiency (**F11**) | Methodology can achieve the desired results in an efficient way. |
| Broadness (**F12**) | Being able to apply the methodology in various datasets based on its generalization capability. |
| Further Improvements (**F13**) | Amount of enhancements required in the methodology's design. |
| Extendability (**F14**) | Capability of the methodology to be extended and adopt the new enhancements. |
| Friendliness (**F15**) | The presented solution offers a user-friendly interface. |

**Table II: Weights Associated with Each Feature**

| **Features** | **User Weight ($W_u$)** | **Developer Weight ($W_d$)** |
|---|---|---|
| **F1** | 10 | 10 |
| **F2** | 10 | 9 |
| **F3** | 4 | 10 |
| **F4** | 3 | 10 |
| **F5** | 7 | 10 |
| **F6** | 10 | 5 |
| **F7** | 10 | 10 |
| **F8** | 10 | 9 |
| **F9** | 8 | 10 |
| **F10** | 10 | 10 |
| **F11** | 10 | 10 |
| **F12** | 5 | 9 |
| **F13** | 2 | 9 |
| **F14** | 5 | 8 |
| **F15** | 10 | 8 |

From the available set of features, most of them benefit the overall maturity score. However, certain features such as Complexity, Cost and required Further Improvements counteract the maturity score growth, since they represent unwanted characteristics. Therefore, we design a maturity score formula (1) that represents the effects of the aforementioned characteristics accurately. The results for the maturity score evaluation of rule-based techniques is illustrated in figure 2. Similarly, the results for the machine learning and the hybrid techniques are illustrated in figures 3 and 4 respectively. The results showcase that the methodology presented in [5] achieves the highest maturity score overall. This proves that a properly trained and developed machine learning methodology is sufficient in order to achieve the task of accurate table detection.

$M_k = F_3 + F_{10} + F_{11} + F_{15} + [(F_5 \text{ x } F_9 \text{ x } F_{12} \text{ x } F_{14}) + (F_1 \text{ x } F_2 \text{ x } F_7 \text{ x } F_8)] / (F_4 \text{ x } F_6 \text{ x } F_{13})$ *(1)*

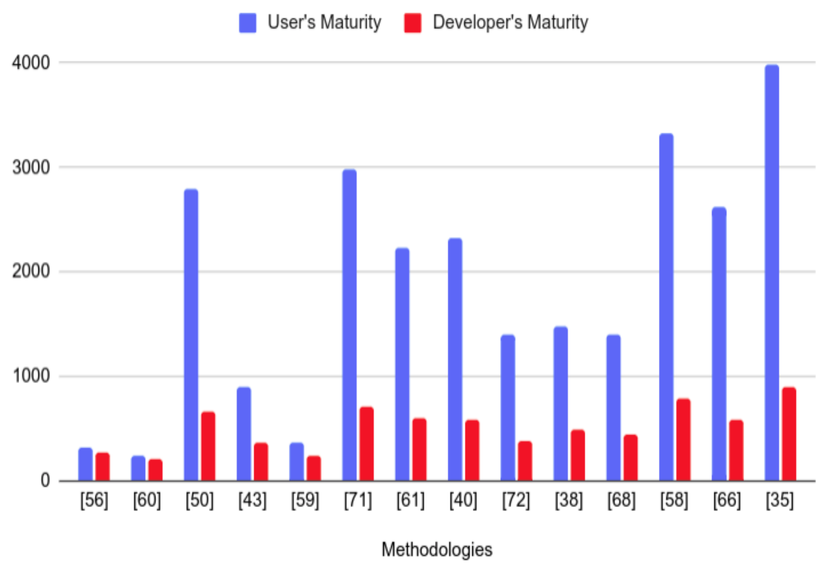


*Figure 2: Rule-based table detection maturity scores for both users and developers weights.*

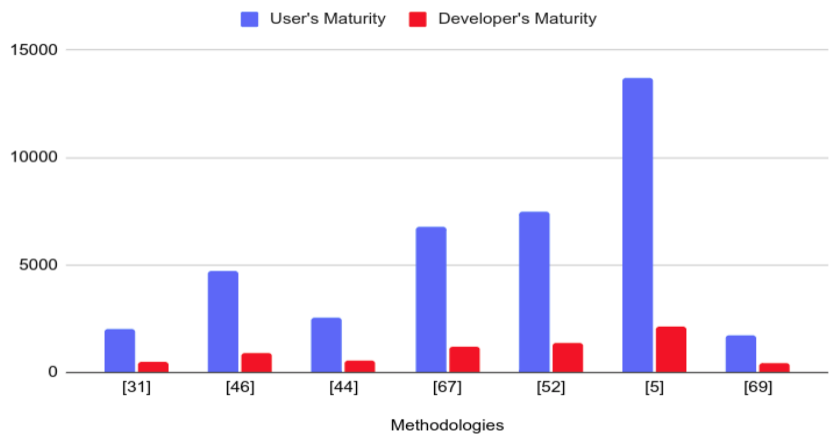


*Figure 3: Machine learning based table detection maturity scores for both users and developers weights.*

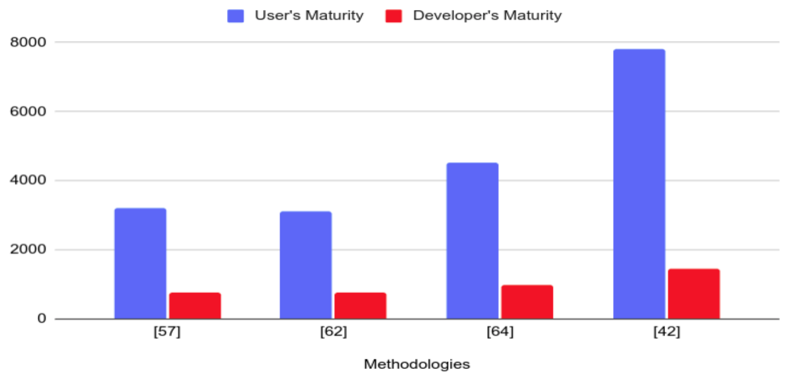


*Figure 4:Hybrid table detection maturity scores for both users and developers weights.*

## 2.2 Processing of Graphics Images

### 2.2.1 Detection of Graphics

Siegel et. al [4] present a methodology consisting of training an object detection model for localization and extraction of figures in academic documents. More specifically, they generate a dataset consisting of document pages that contain figures and charts. The authors have induced ground truth boxes for each of the existing figures of each document page in the training set. They achieve this by making use of the initial LaTeX format of the documents and certain LaTex that result in the drawing of bounding boxes that contain the figures. Then, they feed each labeled document from training set, one page image at a time, into a ResNet-101 model that produces image embeddings. These embeddings are parsed through an OverFeat detection model [28] that finds the drawn bounding boxes containing the figures. The proposed methodology has been deployed as an application on the Cloud and has achieved a score of 96% figure retrieval out of 13 million documents.

Kavasidis et. al. [5] propose detecting graphical figures in technical documents through the use of a saliency CNN. More specifically, they deploy a VGG-16 network with custom kernel sizes in order to ensure the recognition of graphical features. Additionally, they develop binary masks for 4 different heuristically defined classes of objects. They also include an extra Conditional Random Field module that receives the segmented results of the saliency CNN and corrects any existing errors in the feature map. A final model that is trained to categorize detected objects into the 4 author-defined classes is added at the end of the architecture in order to improve the detection accuracy.

Li et. al. [6] introduce PDFigCapX, a tool that extracts graphical figures and their captions from documents using layout characteristics. More specifically, they use the Xpdf parser [7] that separates the input PDF document into natural language text and images of figures. The authors utilize certain keywords such as “Fig.” that indicate the existence of caption text in order to detect its position on the document page layout. Then, they search either below or above the location of the caption text for a figure image. These positions helps associate the images extracted from the Xpdf parser to their corresponding caption text from the document. The proposed methodology is tested specifically on a dataset containing biomedical documents. It achieves a total of 97.3% accuracy score outperforming previous rule-based PDF parsers.

Choudhury et. al. [8] present a strategy for detecting figures in document pages and extracting some primitive metadata. Their work is a continuation of [9] where a methodology is proposed for automatic detection of figures from technical documents using heuristic rulesets on for layout characteristics, their extraction [10] and classification [11] using either rule-based or machine learning based classifiers. More specifically, they initially feed the PDF document through a standard parser that separates text from figures. Then, the images of the figures are parsed through classifiers that categorize them into either line graphs or bar graphs. These classifiers are variations of the standard SVM [31] and Random Forests [32] models. Finally, they use they positions from pixels of the curves in order to deduce each curve's trend.

De et. al. [12] focus exclusively on the detection and extraction of 2-dimensional and 3-dimensional pie chart figures in technical documents. More specifically, they employ a pretrained SSD-Mobilenet [13] which is an R-CNN based model, capable of detecting figures in document page images. The model produces equally well results for the detection of both 2D and 3D charts. Then, they parse the resulting figures through a processing module that applies gradient analysis in order to locate the individual slices in each pie chart.

### 2.2.2 Recognition of Graphics

Chester et. al. [14] describe a visual extraction module that handles simple black and white bar and curve charts. The authors aim to develop a tool that can summarize automatically the information that is illustrated in a given graphics image. The goal is to provide such a tool to visually impaired people. The proposed methodology begins with the extraction of all primary figure components such as text and graphics features. They initially detect potential words and then expand their bounding boxes through dilation in order to capture the rest of the adjacent text. The graphics images elements are recognized as individual continuous entities and characterized by their shapes.  Then, these components of arbitrary shape are merged together based on their positions on the image. If these entities form long vertical lines then the input image must contain bar graphics. If the formed shape has arbitrary changes in its direction then the input image must contain curve graphics. All the recognized information are then expressed in an XML format.

Balaji et. al. [15] present a methodology for the recognition of information that is illustrated in pie and bars charts. The authors initially parse the image through Google's Tesseract OCR [18]. This pretrained model detects every text element in the given chart as well as the graphics legends (axis names, title, etc.). Then, the authors process each type of graphic with a different strategy. Bar charts are considered as connected components. Thus, the authors form regions using these components until they fit a certain criteria. The recognition of pie charts starts by parsing the image through a Canny edge module in order to extract its edges and segments. Then, the authors average the distance between two pixels with different intensities, until they get a good enough circle. Both of these pixels however must be places on the edge.

Al-Zaidy et. al. [17] present an approach for automatic role recognition of the various chart elements. The authors begin with the detection and extraction of text components. This is achieved by considering the text as connected components on which region growing is performed, Then, an OCR is used to read the located area. Graphic components are classified with the use of CNNs into bars, legends, x-axis and y-axis features. Further image processing techniques are used in order to understand the information that is presented in the bars chart of the document. Finally, the authors apply a comparative evaluation between the proposed machine learning strategy and a rule-based one. The proposed methodology outperforms the rule-based approach with regards to data extraction accuracy.

Lu et. al. [19] propose a methodology for automatic recognition of curves with highlighted data points. Their methodology starts with the extraction of data images from the document and their categorizations into specific groups such as images of charts, photographs, etc. More specifically, they extract certain line and text features which are then parsed through a trained SVM [31] model that performs the classification. Then, they parse every image that has been recognized as 2-D plot through an automatic analysis module. After some initial preprocessing, they detect the axis lines using Hough transform. Additionally, they recognize the lines that form the 2-D curves using the Freeman chain coding technique [30]. They combine it with an additional K-Median filter that is able to discard the straight curves of the image. The authors have applied specific heuristics during that procedure in order to detect intersections and connections between the different curves of a 2-D plot. Finally, they propose an algorithm for curve reconstruction based on the information extracted from the recognized graphics image. The same methodology has also been employed in [20], with some additional rules in order to detect more accurately the highlighted data points.

Nair et. al. [21] introduce a methodology automatic analysis of simple plotted lines with slope. They focus on recognizing both solid and partitioned lines of different colors. Their methodology initiates with the detection and extraction of all text elements from the chart image using Google's Tesseract OCR [18]. Then, the employ the Hough transform methodology in order to detect all the vertical and horizontal lines that constitute the axes of the image. Finally, process the plotted lines on the image by tracing their pixels on curve at time. They recognize the directions of the curves by calculating their corresponding slope gradient.

Cliche et. al [22] propose a methodology that aims to reverse engineer any given scatter plot in order to generate the numerical values of its respective data points. The methodology begins with the detection of the areas that contain the axes values of the given chart image. Then, they recognize the actual numerical values of these regions with the use of Google's Tesseract OCR [18]. They proceed by applying DBSCAN [23] once for each axis. The goal is to get the coordinates of all axis ticks through clustering. Then, they use RANSAC regression [16] in order to map the pixel coordinates of their data points into actual chart values. , in order to extract the original table data values. They suggest clustering the scatter data points using DBSCAN [23] once for each axis in order to get the corresponding axis ticks. Then, they associate the data points values with the axis values using the ticks locations.

### 2.2.3 Understanding of Graphics

The authors of [24] [25] recognize the underlying message and semantic meaning of a given bars chart with the use of an inference system. The underlying message can either be about the trend of the bars, the associations between differently colored bars of the same figure and high level messages. However, the authors make the assumption that the input information graphics must already be processed and their metadata must be expressed in XML format. The recognized metadata include axes labels, caption text, bar heights over time, bar colors and associated text or numerical values. Then, they parse that information through the Bayesian inference system which aims to deduce the deeper semantic meaning of the input graphic such as the rising or declining trend of the bars. The goal of the authors is to produce a novel document retrieval methodology, which is based on indexing the documents using the underlying message of their figures rather than abstract text or title.

Greenbacker et. al. [26] deploy the inference system from [24][25] but adapts it for the case of graphics image that contain single curves. More specifically, they detect and isolate the curves on the figure and then partition them into individual line segments using a trained SVM [31] model. Then, they use these generated line segments in order to recognize the trend of the curves. Additionally, the authors create a dataset of summaries for information graphics from human subjects. Then, they use keywords in order to detect correlations between these human based summaries and the inference system's resulting message in order to learn how to deduce deeper meaning.

Finally, Kim et. al. [27] present a methodology for classifying the information that is displayed in information graphics images using a Multimodal neural network. The authors aim to provide a system that will enable impaired people to understand the underlying meaning that is illustrated in graphics images. The information of the graphics are categorized into a set of of 6 predefined classes. These classes include curves of rising trend, decreasing trend, changing trend and etc. The proposed model consists of a Convolutional Neural Network which is later merged with a Bag of Words model into a single Multimodal neural network. The authors evaluate the proposed Multimodal model against a standard CNN. Both networks are trained on an annotated dataset that is generated by surveying a number of human candidates on a set of graphics images. The proposed model achieves a total accuracy score of 74.0% surpassing both the CNN and the human accuracy scores.

### 2.2.4 Comparative Evaluation of Graphics Processing Methodologies

In this section we present a comparative evaluation among graphics processing methodologies similar to the comparative evaluation presented for the table detection methodologies. We employ the same set of subjectively selected features from table 1. The weights of the features are presented in table 2. This time we use a different maturity score formula from the equivalent tables section. More specifically, we compute the average among the scores for each features as their achieved maturity scores. The corresponding formula is illustrated in (2). It must be noted that contrary to the tables section, we cannot compare the maturity scores among different graphics processing fields (i.e. detection, recognition and understanding) since they focus on different aspects of graphics image processing and have different goals. The results for the maturity score evaluation of graphics detection techniques is illustrated in figure 5. Similarly, the results for the recognition and the understanding of graphics techniques are illustrated in figures 6 and 7 respectively. The results showcase that the methodology presented in [5] achieves the highest maturity score overall in graphics detection. The methodology presented in [21] achieves the best results among other graphics recognition methodologies. Finally, the graphics understanding methodology presented in [27] outputs the best performance in its field. We are going to use these 3 best performing methodologies as the basis for our implementations.

$$m_k = \frac{\sum_{i \in F} w_i f_i}{\sum_{i \in F} w_i} \quad (2)$$

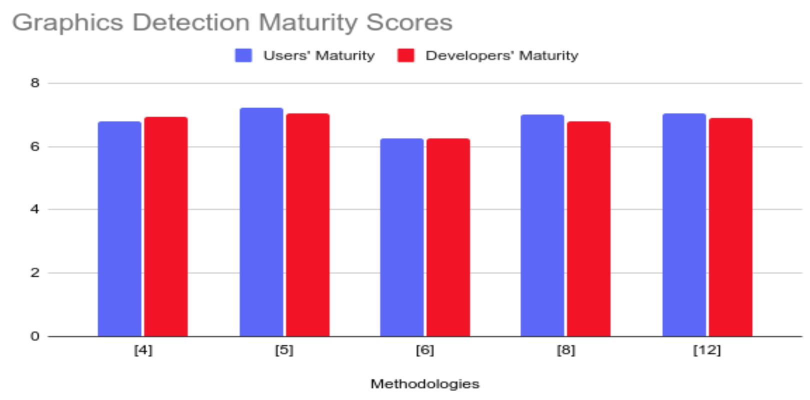


*Figure 5: Graphics detection maturity scores for both users and developers weights.*

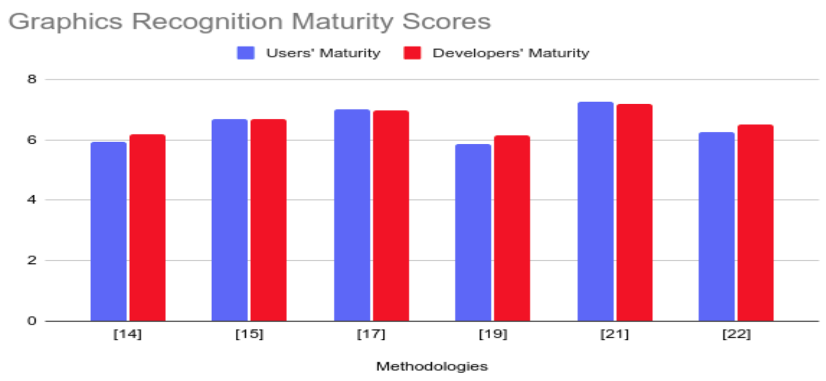
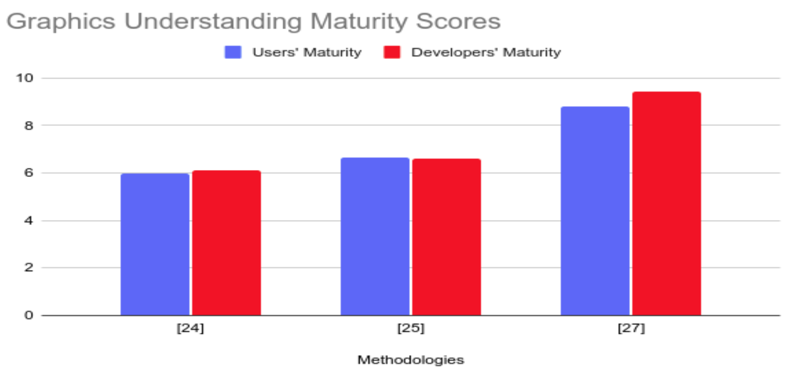


*Figure 6: Graphics recognition maturity scores for both users and developers weights.*



*<u>Figure 7:</u>Graphics understanding maturity scores for both users and developers weights.*

# 3. Processing of Tables Images

## 3.1 Tables Processing Methodology

The literature review section proves that the topic of table detection in document page images has been studied extensively. However, there hasn't been many research efforts towards the cases of table recognition and understanding as of yet. In this study we provide a complete methodology for all 3 cases of tabular processing [3]. More specifically, we aim to achieve a holistic understanding of the document itself, by initially processing each modality individually. The general architecture diagram of the overall system is illustrated in figure 8. The extracted knowledge resulting from the understanding of each modality is represented using an SPN graph. Previous works [75, 76] have merged the SPN graphs resulting from the understanding of natural language text and diagram images. In this study, we focus on the understanding of tables and graphics images.

The proposed methodology begins with the parsing of a technical document in PDF format through the open source PDFBox software [77]. The PDFBox receives any PDF document as input and returns the contained figures and text of its separate pages. The figures can be of different types such as tables, graphics and diagrams which are returned as images. Then, the output images are processed by a detection module which recognizes the modality type of each image. A pre-processing stage precedes the detection module, which deploys a variety of image processing and enhancement techniques. We find that different pre-processing techniques can influence the prediction results of the detection models, either by improving or decreasing their confidence.

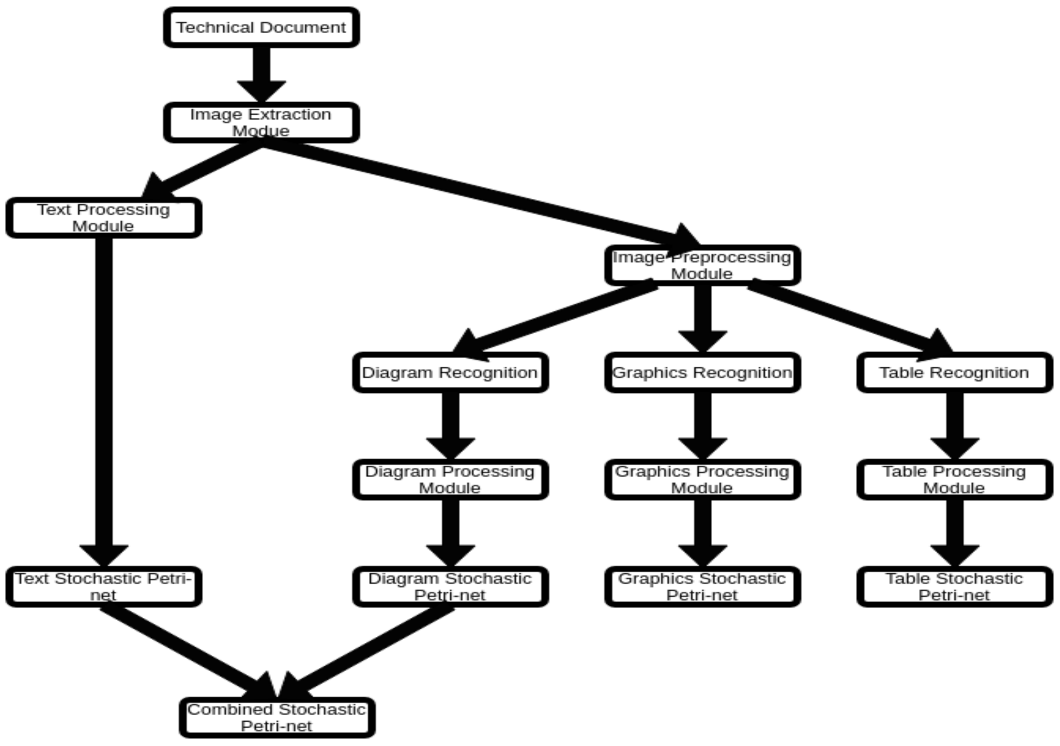

Technical Document
Image Extraction Modue
Text Processing Module
Image Preprocessing Module
Diagram Recognition
Graphics Recognition
Table Recognition
Diagram Processing Module
Graphics Processing Module
Table Processing Module
Text Stochastic Petri-net
Diagram Stochastic Petri-net
Graphics Stochastic Petri-net
Table Stochastic Petri-net
Combined Stochastic Petri-net

*Figure 8: Architecture of the proposed general processing methodology.*

The document modality images are parsed to their respective processing modules based on their predicted category. In the case of tables, the recognition modules analyzes the given image in order to extract title information, tabular variables from rows and columns, as well as the values corresponding to these variables. Finally, the table understanding module receives that extracted information and. The retrieved structural information is represented with attributed graphs, which are then converted to natural language text. The retrieval behavior information is represented into SPN graphs, which are produced from the aforementioned natural language sentences. For this reason, the Pinakas formal language is introduced as a tool that helps us map the natural language text into SPN kernels.

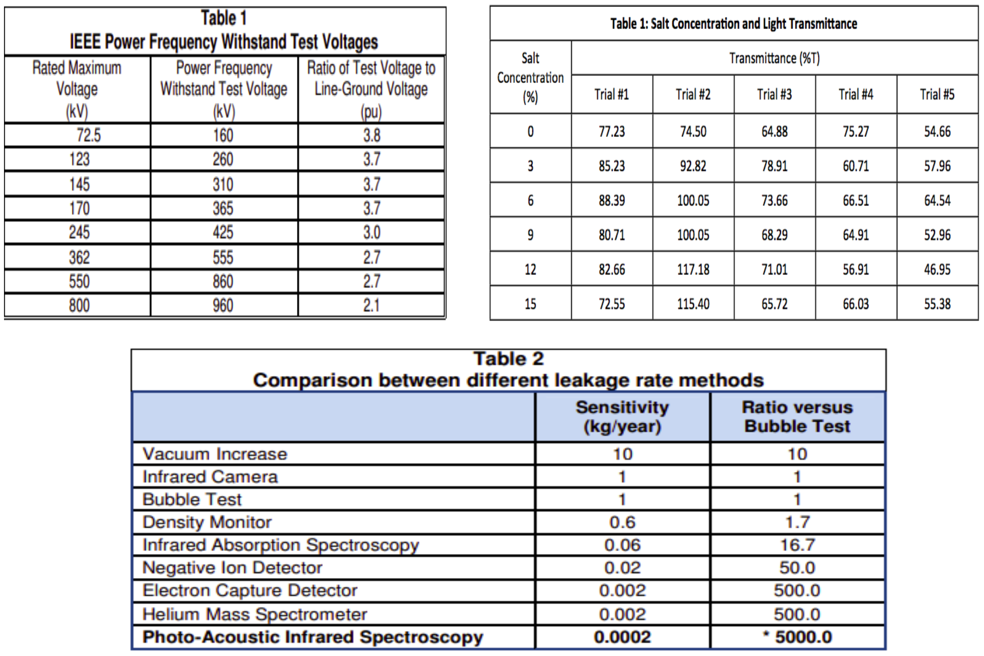


Table 1
IEEE Power Frequency Withstand Test Voltages

| Rated Maximum Voltage (kV) | Power Frequency Withstand Test Voltage (kV) | Ratio of Test Voltage to Line-Ground Voltage (pu) |
|---|---|---|
| 72.5 | 160 | 3.8 |
| 123 | 260 | 3.7 |
| 145 | 310 | 3.7 |
| 170 | 365 | 3.7 |
| 245 | 425 | 3.0 |
| 362 | 555 | 2.7 |
| 550 | 860 | 2.7 |
| 800 | 960 | 2.1 |

Table 1: Salt Concentration and Light Transmittance

| Salt Concentration (%) | Transmittance (%T) | | | | |
|---|---|---|---|---|---|
| | Trial #1 | Trial #2 | Trial #3 | Trial #4 | Trial #5 |
| 0 | 77.23 | 74.50 | 64.88 | 75.27 | 54.66 |
| 3 | 85.23 | 92.82 | 78.91 | 60.71 | 57.96 |
| 6 | 88.39 | 100.05 | 73.66 | 66.51 | 64.54 |
| 9 | 80.71 | 100.05 | 68.29 | 64.91 | 52.96 |
| 12 | 82.66 | 117.18 | 71.01 | 56.91 | 46.95 |
| 15 | 72.55 | 115.40 | 65.72 | 66.03 | 55.38 |

Table 2
Comparison between different leakage rate methods

| | Sensitivity (kg/year) | Ratio versus Bubble Test |
|---|---|---|
| Vacuum Increase | 10 | 10 |
| Infrared Camera | 1 | 1 |
| Bubble Test | 1 | 1 |
| Density Monitor | 0.6 | 1.7 |
| Infrared Absorption Spectroscopy | 0.06 | 16.7 |
| Negative Ion Detector | 0.02 | 50.0 |
| Electron Capture Detector | 0.002 | 500.0 |
| Helium Mass Spectrometer | 0.002 | 500.0 |
| **Photo-Acoustic Infrared Spectroscopy** | **0.0002** | *** 5000.0** |

*Figure 9: Simple Table (left), Extended Table (middle) and Multidimensional Table (right).*

The goal of this research isn't to design an application that covers any possible table use case, but to prove that such an application can exist. Thus, we focus on 3 different types of tables as proof of concept for the presented table processing methodology. Figure 9 illustrates examples from these types of tables, which are simple table (type 1), multidimensional table (type 2) and extended table (type 3).

## 3.2 Detection of Tables

### 3.2.1 Image Pyramids

One of the aforementioned image enhancement techniques that we use is that of creating image pyramids [96]. This pre-processing step aims to reduce the amount of information for each candidate image before parsing them through the different detectors. The intuition behind image pyramids is that the machine is able to capture different features on different scales of the same image, just like a human does. If the machine doesn't have access to multiple scales, then the extracted information might be incomplete. The authors describe many different approaches onhow to construct image pyramids. However, we use the Gaussian pyramids approach for our implementation.

We start by reducing the size of the input image to different resolution scales. At each step of pyramidal reduction we apply the Gaussian smoothing filter. Then, the image is resized back to its original dimensions. This procedure ensures that only essential structural features of the image are preserved. An example of pyramidal processing for a table image is illustrated in figure 10. The example shows that by going too high on the level of pyramidal processing, the image edges can become distorted to the point that the input image is unclassifiable. We conclude through trial and error, that the optimal level for pyramidal reduction in table images is level 3.

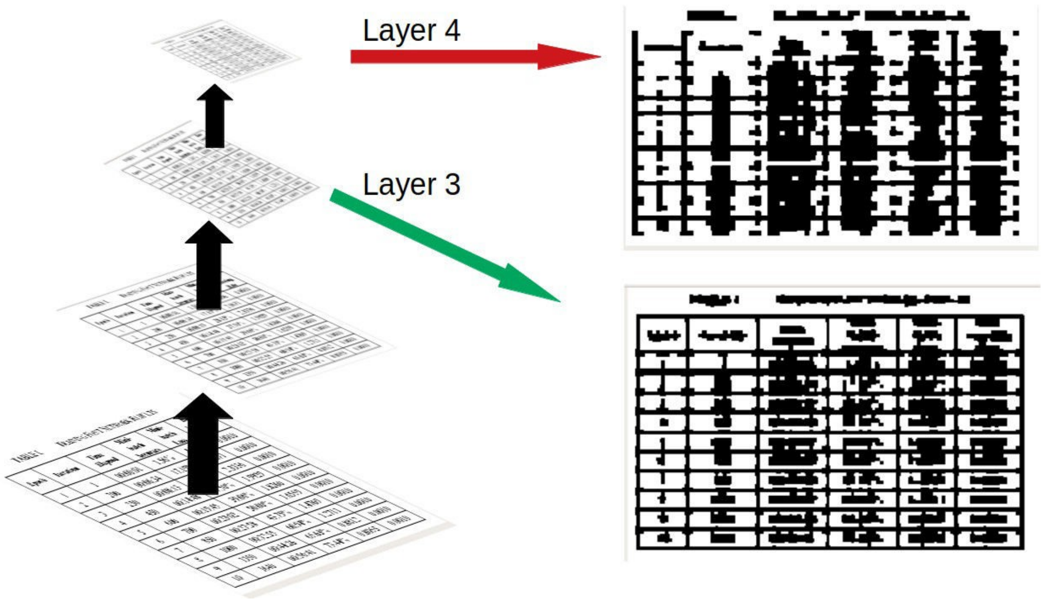


*Figure 10: Example pyramid image reduction for 4 levels*

### 3.2.2 Learning Based Detection

Machine learning has revolutionized the area of image detection and classification. Learning algorithms offer many advantages that range from the ease of implementation to the decoupling of the programmer from the application, due to its "black box" nature. For this applied comparative study, we focus on one-class, binary, and multi-class models. Furthermore, we vary the researched methods by testing for different types of pre-processed input images. More specifically, we implement the one-class classification algorithms of One-Class Support Vector Machine (SVM) [78] and the Isolation Forest [79]. An SVM is supervised learning algorithm for distinguishing input data between acceptable and outliers. Similarly, Isolation forest is unsupervised learning algorithm, which is also used for detecting inconsistent observations. One-class models are more commonly used for anomaly detection in 1- dimensional data consisting of numerical series. Hence, we convert the 2-dimensional images into 1-dimensional arrays using a raster scan. Then, we feed them into the SVM and Isolation Forest algorithms respectively. We train these algorithms purely on positive data. This implies that any other types of images besides tables are considered outliers.

Additionally, we implement and train binary classification and multi-class models with the use of Convolutional Neural Networks (CNN) [80]. CNN models are the state of the art solution for fast and accurate image classification or object detection. A CNN consists of convolution layers, pooling layers, and fully connected layers at the end. For the convolution layer, a sliding kernel of predefined size shifts over the input image, and performs convolutions and preserves only the essential edges trough max or min pooling. Finally, the fully connected layers receive the corresponding output and return a N dimensional array containing the probabilities of the image being a member of class. The N is the number of user defined categories for classification. A binary CNN model is trained on both positive and negative data, and uses the binary cross-entropy as its loss function. On the contrary, multi-class CNN model trains for images from all available classes and uses the categorical cross-entropy as its loss function.

We train two versions for each of the aforementioned algorithms and models, one for binary images and one for pyramidal processed images. The goal is to study the effects that a different pre-processing of data can have in the final predictions. As an example of a CNN architecture, we illustrate the diagram of the binary CNN model that have implemented in figure 11. This model is trained on images that have been initially parsed through pyramidal reduction. The rest of the CNN models that we have trained follow a similar structure.

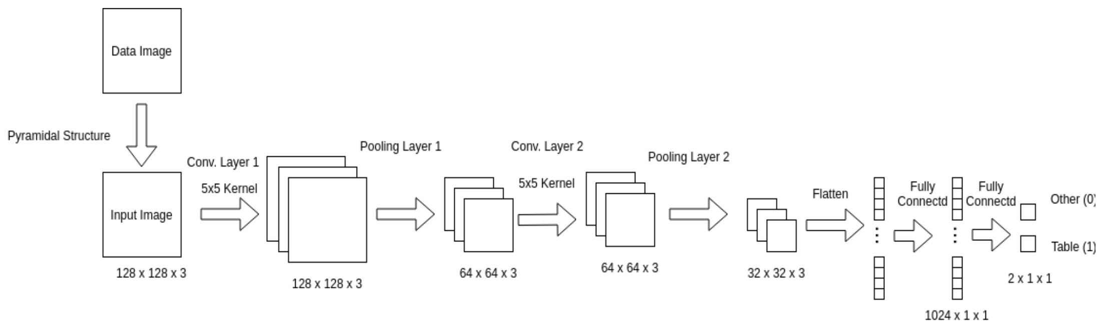


*Figure 11: Example of binary classification model for table images that have been processed with pyramidal reduction.*

### 3.2.3 Hybrid Based Detection

Recent advances in multi-agent applications have introduced the concept of collaborative decision making [81]. The different agent programs negotiate with each other in order to achieve an agreement that benefit all of them. Each agent get to have a vote and at the end they all follow the majority. Different agents may have different weights on specific topics based on their level of expertise. We have implemented a similar methodology which is illustrated in figure 12.

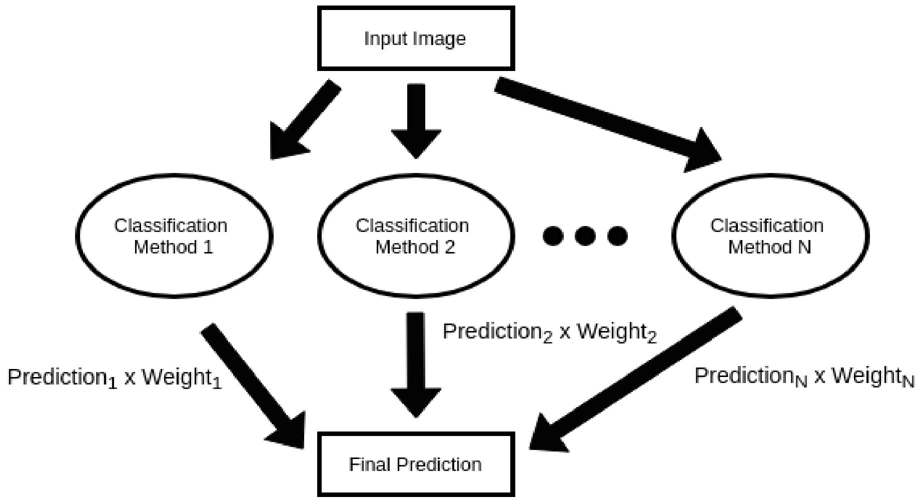


*Figure 12: Architecture of the image classification based on a voting scheme.*

For the current use case, we input every probe image through each trained classification model concurrently and use their prediction results as the votes for the collective decision. We take into account only those models that have showcased an accuracy higher than 50% during the testing phase. The testing score of each model is also used as the weight for each vote. If the majority of the models agree on a specific class, then the image is categorized as its member. Otherwise, the class of the image is deduced based on the prediction of the model that achieves the highest training accuracy. In case that model can't determine a class, then we use the prediction of the second best model and so on until a class can be deduced for input the image. The overall motivation for this approach is to try and compliment each model's individual performance with the advantages that other models may offer.

### 3.2.4 Rule-based Detection

The proposed rule based table detection methodology constitutes a combination of image processing techniques and geometric mathematical formulas. All tabular structures with ruling lines are characterized by table cells that are in alignment and in order. So, we can detect the alignment between the contents of the table cells in order to implicitly deduce that the input image is a table. The programs starts with the application of binarization and Gaussian pyramidal reduction on the input image. The Gaussian pyramidal reduction enables the preservation of the core tabular features and converts the cell contents into masses of black pixels (connected components). Then, we discard all vertical and horizontal lines from the image and locate the centroids of the newly formed pixel regions. We use the position of these centroids as representatives of their corresponding regions, in order to evaluate the alignment between them.

However, it is possible that few of these keypoints diverge from from the average line of alignment. Hence, the evaluation of the alignment between the cell contents is performed by a variation of the unevenness criteria formula that is described in [82]. The authors of this paper present a methodology for the recognition of line segments, with emphasis to segments that showcase anomalies or unevenness. The variable unevenness U for line segment SL is defined as the number i of consecutive pixels with , that follow a different direction di, from the main direction dmain of the line segment SL. More specifically, the authors develop and describe two connected equations in order to tackle this issue. The first equation (3) describes the window that covers the line segment, which contains the unevenness. WU represents the area of the window, HU represents the maximum accepted unevenness and LU represents the length of the line segment containing the unevenness. The second equation (4) defines the unevenness criteria as the division between the height of the window covering the line segment, which contains the unevenness (Wu), and that segment's length. We present both formulas below:

$$Wu = Hu \times Lu \quad (3). \quad and \quad e = Hu/Lu \quad (4)$$

The value of the unevenness criteria variable affects the amount of details in line anomalies that is covered during the procedure. A small unevenness criteria threshold corresponds to a lower tolerance for divergence.

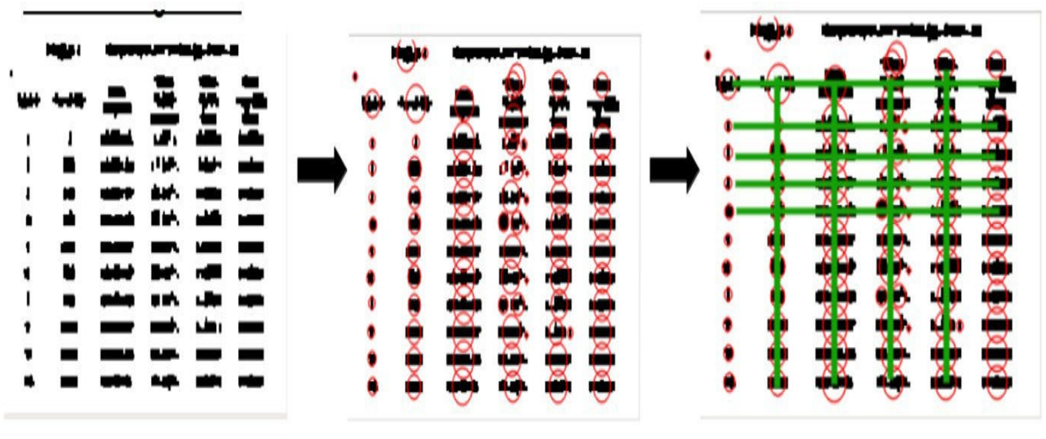
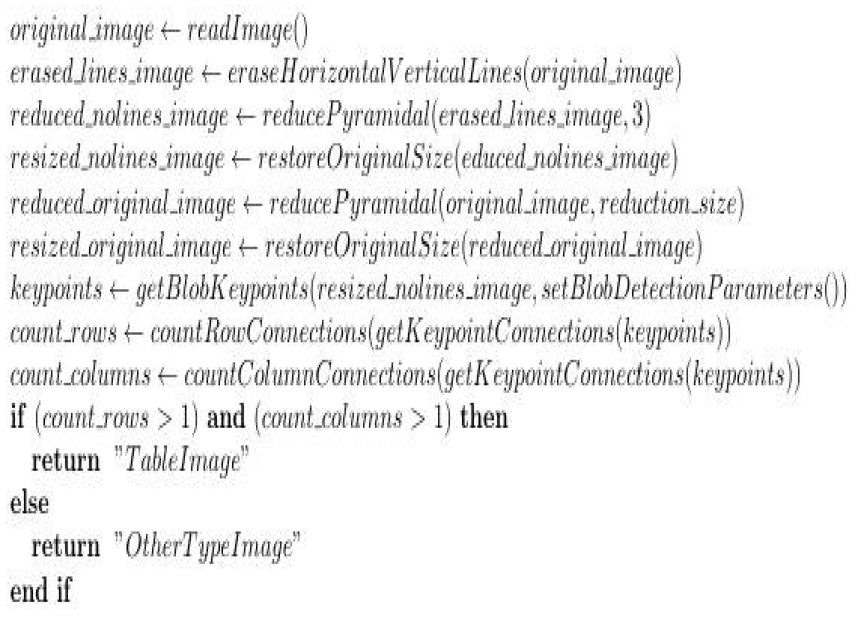


*Figure 13: The process of detection the connected component regions after the pyramidal reduction and the recognition of lines that contain centroids using unevenness.*

Figure 13 illustrates an example of the aforementioned rule-based table recognition process. We deduce the value 0.04 as optimal for the unevenness criteria threshold, through a trial and error process. We search once for alignment between the table columns and once for alignment between table rows. In the first case, we calculate $H_U$ by setting the length of the table as the value for $L_U$. In the second case, we set the table height as the value for $H_U$ and then calculate the limit $L_U$. Then, the algorithm initiates with the detection of the first row and column of pixel regions. In case that many vertical and horizontal areas of size $W_U=H_U x L_U$ include multiple centroids, then we deduce that the given image is a table. The steps of the aforementioned alignment detection algorithm in tables appear in detail on Algorithm I.

**Algorithm I: Rule-based Table Detection**



```
original_image ← readImage()
erased_lines_image ← eraseHorizontalVerticalLines(original_image)
reduced_nolines_image ← reducePyramidal(erased_lines_image, 3)
resized_nolines_image ← restoreOriginalSize(educed_nolines_image)
reduced_original_image ← reducePyramidal(original_image, reduction_size)
resized_original_image ← restoreOriginalSize(reduced_original_image)
keypoints ← getBlobKeypoints(resized_nolines_image, setBlobDetectionParameters())
count_rows ← countRowConnections(getKeypointConnections(keypoints))
count_columns ← countColumnConnections(getKeypointConnections(keypoints))
if (count_rows > 1) and (count_columns > 1) then
  return "TableImage"
else
  return "OtherTypeImage"
end if
```

## 3.2.5 Evaluation Results

We generate both the training and testing datasets by scraping images of tables and graphics from journals published by IEEE. The total number of training images accumulates to 2100. Due to the small scale of the IEEE training set, we employ the K-fold technique. The value K=10 is deduced as optimal through a trial and error testing process. The IEEE test set contains a total number of 100 images, with 52 being tables and 48 being graphics. Finally, we also use a portion of the ICDAR-2013 test set for further evaluation of the studied methodologies. The later dataset contains 53 images of tables and 29 images of graphics.

The tables (3 and 4) showcase the classification outcome scores for the ICDAR-2013 and IEEE test sets respectively. We evaluate the studied methodologies based on the precision, recall and F1-score metrics. TOCSVM corresponds to one-class SVM for tables, PTOCSVM to one-class SVM for pyramidally processed tables, TIF to Isolation Forest for tables, PTIF to Isolation Forest for pyramidally tables, BCNN to binary classification for tables, PBCNN to binary classification for pyramidally processed tables, MCNN to multi-class classification, PMCNN to multi-class classification with pyramidal reduction, RTC to rule-based table classification, HYBRID to the learning methodologies that are combined through the voting system.

**Table III: Classification Scores using ICDAR test set**

| | Accuracy | F1-score | Precision | Recall |
|---|---|---|---|---|
| **TOCSVM** | 0.646 | 0.79 | 0.65 | 1 |
| **PTOCSVM** | 0.24 | 0.09 | 0.20 | 0.06 |
| **TIF** | 0.646 | 0.79 | 0.65 | 1 |
| **PTIF** | 0.292 | 0.15 | 0.33 | 0.09 |
| **BCNN** | 0.67 | 0.72 | 0.81 | 0.64 |
| **PBCNN** | 0.878 | 0.91 | 0.91 | 0.91 |
| **MCNN** | 0.80 | 0.85 | 0.84 | 0.87 |
| **PMCNN** | 0.902 | 0.90 | 0.88 | 0.92 |
| **RTC** | 0.804 | 0.80 | 0.80 | 0.80 |
| **HYBRID** | 0.902 | 0.93 | 0.89 | 0.96 |

**Table IV: Classification Scores using IEEE test set**

| | Accuracy | F1-score | Precision | Recall |
|---|---|---|---|---|
| **TOCSVM** | 0.5 | 0.66 | 0.51 | 0.92 |
| **PTOCSVM** | 0.37 | 0.26 | 0.33 | 0.21 |
| **TIF** | 0.52 | 0.68 | 0.52 | 0.98 |
| **PTIF** | 0.37 | 0.34 | 0.37 | 0.31 |
| **BCNN** | 0.93 | 0.93 | 0.96 | 0.90 |
| **PBCNN** | 0.899 | 0.90 | 0.92 | 0.88 |
| **MCNN** | 0.899 | 0.91 | 0.88 | 0.94 |
| **PMCNN** | 0.91 | 0.91 | 0.92 | 0.90 |
| **RTC** | 0.733 | 0.73 | 0.73 | 0.73 |
| **HYBRID** | 0.92 | 0.92 | 0.94 | 0.90 |

The results illustrate that BCNN model achieves the best accuracy when it comes to the IEEE test set, while the proposed HYBRID model achieves the best accuracy for the ICDAR 2013 test set. Finally, it must be highlighted that the pyramidal reduction helped improve almost all accuracy scores, except from those of the one-class models.

## 3.3 The Pinakas Formal Language

A formal language is capable of representing the internal semantic and syntactic features that are extracted from a natural language sentence. The goal of this research is to recognize the knowledge that is illustrated in different technical document modalities and convert it into a common form of representation. As we have already mentioned, we have chosen the stochastic Petri-net as that common representation form, since it can describe the internal functionality of each technical document modality. Therefore, we develop the Pinakas formal language as the means to map the recognized tabular relations and associations into SPN kernels. The presented work is based on the formal language Glossa presented in [75]. Glossa is a language used for the representation of kernels of natural languages sentences extracted from document text. This methodology proves to be useful for later mapping of those extracted kernels into SPNs. The Glossa language has its own set of alphabet, grammar, and operators, and aims to cover all the different extracted kernel use cases. Similarly, we define our own alphabet, grammar, and operators in an effort to fit every table type in our language.

## 3.3.1 Alphabet

The first step is to define the alphabet for Pinakas language. In order to achieve this we have to consider each variable of the table as an agent and, therefore, associate it with the symbol $A_i$. The symbol $A_i$ now represents an alphanumeric combination that is extracted from the table and is recognized as a table variable name. So, $A_i$ = {x|x is an alphanumeric combination string that constitutes a table variable name or a symbol from group S}. We aim to formulate the connections between different rows of values and the variables that these values correspond to.

We illustrate an example of formulation in figure 14. We use the extended table (type 3) as the case study for this example. $Row_1$(Salt Concentration) is the the value in the first row, that corresponds to variable Salt Concentration. However, an extended table contains columns of subvariables for each variable. Therefore, we have to deal with this issue as well. In this case, Salt Concentration does not have any sub-variables (sub_var_0). In the language of Pinakas the same connection is represented as $VAL_1(SVR0(A_1))$, where $A_1$ is the first variable of the table or the first agent. In the same example, $SVR_0()$ indicates non-existence of any sub-variables, and $VAL_A()$ is an agent's corresponding value for that case. We demonstrate the subvariables of a variable using their position (column number) with respect to its main variable.

More specifically, $SVR_1(A_2)$ represents the first subvariable of agent 2 while $SVR_2(A_2)$ represents the second subvariable of agent 2. This formulation strategy is the most dominant, since it is capable of describing all possible relations between rows, variables, and sub-variables, as well as cover more types of tables in general. Therefore, we adopt it for Pinakas as well. Finally, we consider $VAL_j()$ = {x|x is the jth value of input variable in respect to the table's column}. Also, $SVR_k()$ = {x|x is the kth value of input variable in respect to the table's row}.

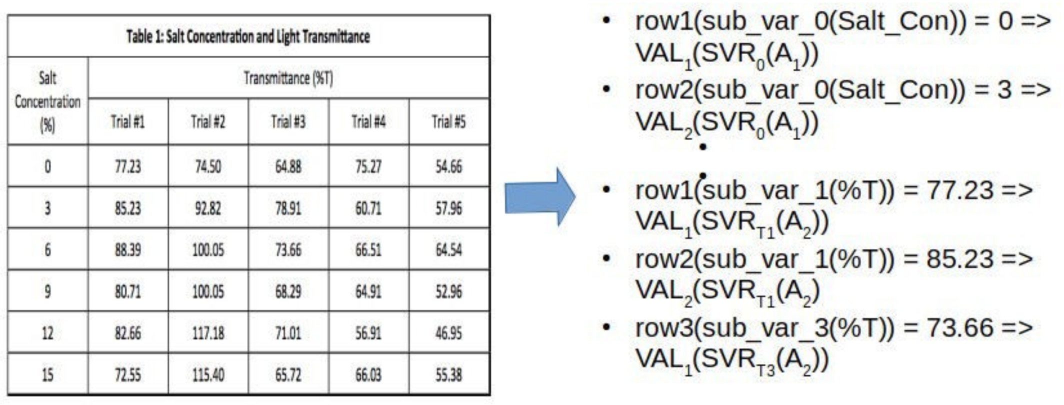


Table 1: Salt Concentration and Light Transmittance

| Salt Concentration (%) | Transmittance (%T) | | | | |
|---|---|---|---|---|---|
| | Trial #1 | Trial #2 | Trial #3 | Trial #4 | Trial #5 |
| 0 | 77.23 | 74.50 | 64.88 | 75.27 | 54.66 |
| 3 | 85.23 | 92.82 | 78.91 | 60.71 | 57.96 |
| 6 | 88.39 | 100.05 | 73.66 | 66.51 | 64.54 |
| 9 | 80.71 | 100.05 | 68.29 | 64.91 | 52.96 |
| 12 | 82.66 | 117.18 | 71.01 | 56.91 | 46.95 |
| 15 | 72.55 | 115.40 | 65.72 | 66.03 | 55.38 |

- row1(sub_var_0(Salt_Con)) = 0 => $VAL_1(SVR_0(A_1))$
- row2(sub_var_0(Salt_Con)) = 3 => $VAL_2(SVR_0(A_1))$
- row1(sub_var_1(%T)) = 77.23 => $VAL_1(SVR_{T1}(A_2))$
- row2(sub_var_1(%T)) = 85.23 => $VAL_2(SVR_{T1}(A_2))$
- row3(sub_var_3(%T)) = 73.66 => $VAL_1(SVR_{T3}(A_2))$

*Figure 14: Formulation Effort Example for Extended Table: This is the way of representation we choose to use in Pinakas for all the connections between variables, sub-variables and rows of values.*

## 3.3.2 Grammar

Now that we have defined the alphabet for Pinakas, the second step is to describe the Grammar G. We define G = (VN, VT, T, F), where:

- VN is the set of non-terminal symbols with VN ={Q, A, V, P, T}
- VT is the set of terminal symbols with VT = {+, -, <, =, @, !, ~}
- T represents the starting symbol of the table
- F is the set of production rules

More specifically, the set of production rules F includes:

- $T \rightarrow [Q] @ [Q] @ \dots @ [Q]$
- $Q \rightarrow (A)(V)(P)$
- $A \rightarrow (A_1 + A_2 + \dots + A_3)\ !\ (A_1 - A_2 - \dots - A_3)\ !\ \dots$

- $A_1 \rightarrow VAL_1(SVR_0(A_1))!VAL_2(SVR_0(A_1))!\ldots!VAL_3(SVR_0(A_1)) \; ! \; \ldots \; ! \; VAL_j(SVR_k(A_1))$

- $V \rightarrow V_1 \; ! \; V_2 \; ! \; V_3 \; ! \; V_4$ because we have only 4 options (is/are greater than or is/are equal to)

- $P \rightarrow (P_1 + P2 + \ldots + P_3) \; ! \; (P_1 - P_2 - \ldots - P_3) \; ! \; \ldots$

- $P_1 \rightarrow VAL_1(SVR_0(P_1)) \; ! \; VAL_2(SVR_0(P_1)) \; ! \; \ldots \; ! \; VAL_3(SVR_0(P_1)) \; ! \; \ldots \; ! \; VAL_j(SVR_k(P_1))$

### 3.3.3 Operators

In the productions rules that we define above, we use some specific symbols. The third and final step is to explain these symbols.

- Operators "+" and "-" represents the "plus" and "minus" operations respectively between agents or patients
- "<" is an operator that represents the "is/are greater than" verb or action
- "=" is an operator that represents the "is/are equal to" verb or action
- The "@" operator represents the "and" connection between different questions (generated and validated relations).
- The "!" operator represents the "or" connection between different possible values of agents, verbs, and patients respectively.
- The "~" operator represents the "not" characterization for questions, agents, or patients.
- [] and () are used to determine the scope of different operations

## 3.4 Recognition of Tables

The methodology for both the recognition and understanding of tables consists of a combination of image and natural language processing techniques, as well as graph representation models. The concept for this methodology is first described by Bourbakis et. al. [1]. More specifically, he proposes a strategy on how to successfully convert the different modalities of a technical document, such as images, graphics, tables, mathematical formulas and algorithms into other forms that describe information. His end goal is to transform all these different modalities into a common form of representation and in that way gain a deeper understanding of the information contained in the document. He presents the stochastic Petri-net graph as the optimal single form of representation. For the use case of tables, he proposes converting them first into an attributed graphs first, in order to maintain their structural information. Then, we can convert the attributed graphs into natural language sentences. Finally, the latter form of representation enables the conversion of the initially extracted tabular information into corresponding SPN graphs.

Initially, the designed tool receives an image of a table as its input. The table image must belong to a technical document and conform to format guidelines from one of the three acceptable table categories. Each table type presents information in its own unique way, so a universal method for successful information extraction is impossible. Any such attempt may lead to unnecessary loss of information. We avoid this issue by classifying the table images first. Then, we follow different processing styles based on the category that the input table is associated with. In order to separate our tables into their three distinctive categories, we have developed a classifying algorithm (Algorithm II). The algorithm decides the category of the received table based on certain structural patterns concerning table cells and rows.

More precisely, the algorithm applies box extraction on the table image in order to get each cell of the table, one row at a time. We maintain only the title box and the consecutive first two rows, because they hold the necessary information about the variables of the table. That information is basically the positions of each box's corners in each row. We use these edge positions in order to deduce how much space in the row is covered by each box. The covered spaced in a row by each box implies the type of the table. In the simple and multidimensional tables (type 1 and 2) for example, all the boxes of the first row have the same delimiters as the boxes of the second row. In the extended table (type 3), some boxes in the first row cover more than one boxes in the second. This indicates the existence of sub-variables in the table, and by extend a table of type 3. However, there is also a difference between type 1 and type 2 tables. In the case the first box in the first row is empty that indicates the existence of 2 dimensions in the table. So the algorithm counts how many cells of the second row does a cell of the first row cover. If it covers more than one, then we automatically have a table of type 3. If each cell from row one covers exactly one cell from row two then we search whether the first box of the first row its empty or not. If it is empty, then we have a type 2 table, otherwise we have a type 1.

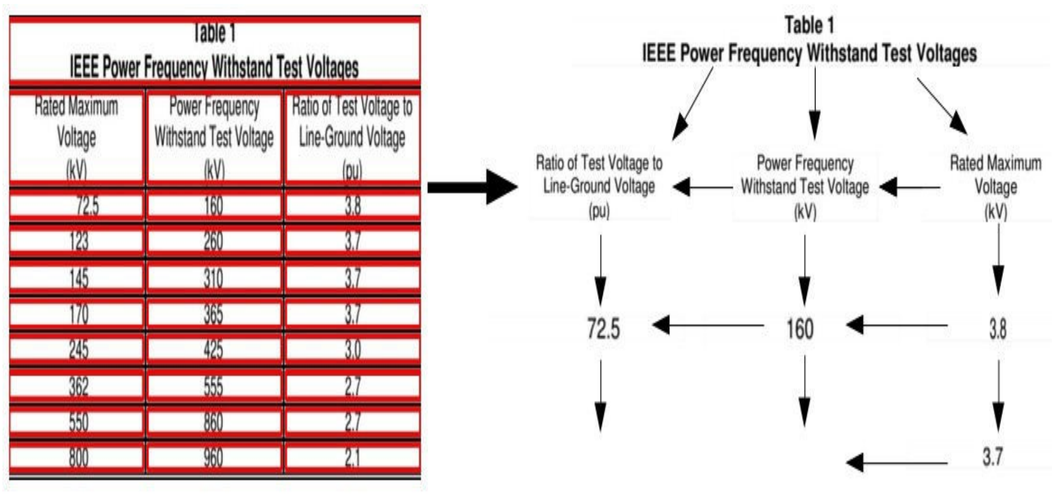


Table 1
IEEE Power Frequency Withstand Test Voltages

| Rated Maximum Voltage (kV) | Power Frequency Withstand Test Voltage (kV) | Ratio of Test Voltage to Line-Ground Voltage (pu) |
|---|---|---|
| 72.5 | 160 | 3.8 |
| 123 | 260 | 3.7 |
| 145 | 310 | 3.7 |
| 170 | 365 | 3.7 |
| 245 | 425 | 3.0 |
| 362 | 555 | 2.7 |
| 550 | 860 | 2.7 |
| 800 | 960 | 2.1 |

Table 1
IEEE Power Frequency Withstand Test Voltages
Ratio of Test Voltage to Line-Ground Voltage (pu)
Power Frequency Withstand Test Voltage (kV)
Rated Maximum Voltage (kV)
72.5
160
3.8
3.7

*Figure 15: Simple table after box recognition, extraction and content association processes are performed.*

After the completion of the classification step, each table follows different processing strategies based on its type, in order to successfully extract the tables information. The tool recognizes the variables and their corresponding values, based on their positions in the table. It stores these values in tuples and links these tuples to their corresponding variable names for further processing in future steps. The general architectures of the processing methodologies for each table type are illustrated in figures 1 6 , 1 7 , and 1 8 respectively. For the extraction and recognition of the table variables and their values. we use a combination of a box extraction the Table OCR [94]. The table cells are extracted following a direction from right to left, starting from the title box of the table and moving downwards.

We take advantage of the order of table cell extraction in order to recognize the number of columns and rows. In addition, it works as an indication about the neighborhood of association between table cells. We illustrate an example of the recognized neighborhoods of association between cells in figure 8. A table cell's neighborhood of association contains the direct left, right, upper and lower table cell. We use the simple table image from figure 1 as the use case for this example.

**Algorithm II: Table Type Classification**

```
$type \leftarrow 0$
$contours \leftarrow getContours(table\_image)$
$title\_box \leftarrow getFirstBox(contours)$
$row\_1\_boxes, row\_2\_boxes \leftarrow getOtherBoxes(contours)$
for $box\_1$ in $row\_1\_boxes$ do
  $count \leftarrow 0$
  if $box\_1$ is within limits of $title\_box$ then
    for $box\_2$ in $row\_2\_boxes$ do
      if $box\_2$ is within limits of $box\_1$ then
        $count \leftarrow count + 1$
      end if
    end for
  end if
  if $count > 1$ then
    $type \leftarrow 3$
    $break$
  end if
end for
if $type$ is 0 then
  $temp\_box \leftarrow getFirstBox(row\_1\_boxes)$
  $content \leftarrow OCR\_tool(temp\_box)$
  if $content$ is empty then
    $type \leftarrow 2$
  else
    $type \leftarrow 1$
  end if
end if
return $type$
```

## 3.5 Understanding of Tables

### 3.5.1 General Understanding Methodology

As we have already mentioned, we have presented a general methodology about the processing and understanding of the table contents in [1]. To the best of our knowledge, there is no other standard methodology for the same task yet. In the current work, we expand on that understanding methodology and include more details about the processing steps for each table type. Figure 16 illustrates the proposed methodology for the recognition and understanding of simple tables (type 1). We aim to emulate the way that humans perceive a table through a program .

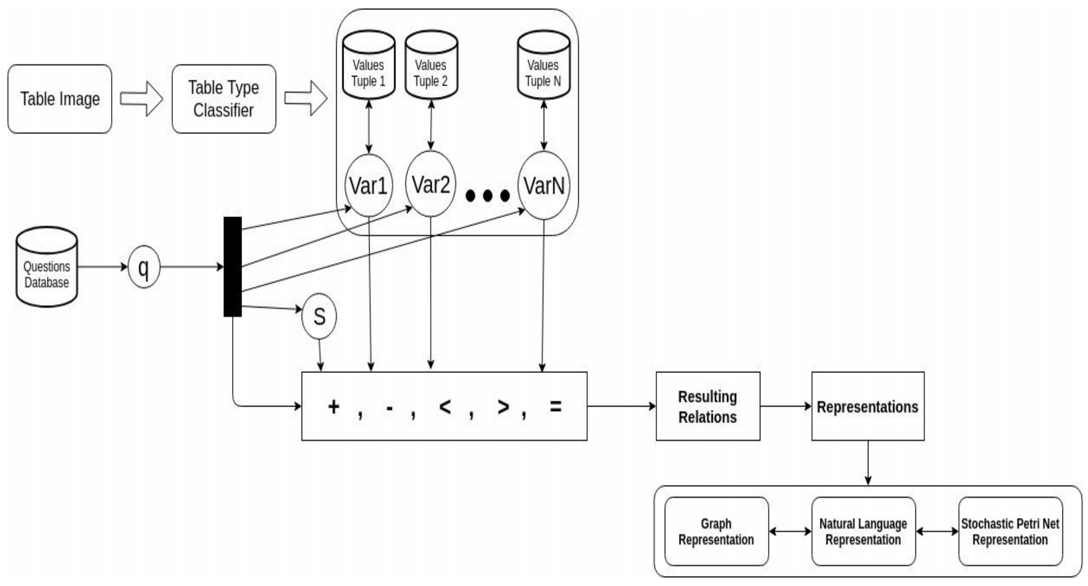


*Figure 16: Simple Table Recognition Methodology*

In order for a human to understand the information that is presented in a given table, he or she asks questions. Examples of such questions could be “Are all the values of the table positive?” or “Which table variable has the greatest value?”. The outcome answers for these questions can determine relationships between the variables of the table. However, a machine cannot be asked a natural language question and return a valid answer. That is why we transform these questions to proper relational queries. For example, in case we have a table with 3 variable X, Y and Z the first question translates into “$0 < X$ and $0 < Y$ and $0 < Z$”. As a second example, the question “Is the summation of the table variable X and Y positive?” is converted into “$0 < X + Y$”. These relational queries reflect the potential associations between table variables. Then, we can validate these relations based on the corresponding values that appear on the image for each table variable.

As it is illustrated in figures 16, 17 and 18, we generate the initial queries them from all the possible combinations of table variables, operational symbols and the S symbol. The S symbol stands for “0” (zero), “ε” (epsilon), “π” (pi) and other mathematical constants that we might want to add in the future. However, some of the produced combinations might be invalid, since they might either not hold up mathematically (semantic error) or they may contain double characters (structural error). Therefore, we filter the integrity of the initial queries by semantic and structural validation algorithms. We explain the validation processes in greater detail below. Each relation has to be verified by both algorithms. Finally, we follow the proposed methodology from [1] and use the verified relations to generate the corresponding attributed graph representations. Then, from the produced graphs we generate the natural language sentences. These sentences are converted into their respective SPNs at the end of the procedure.

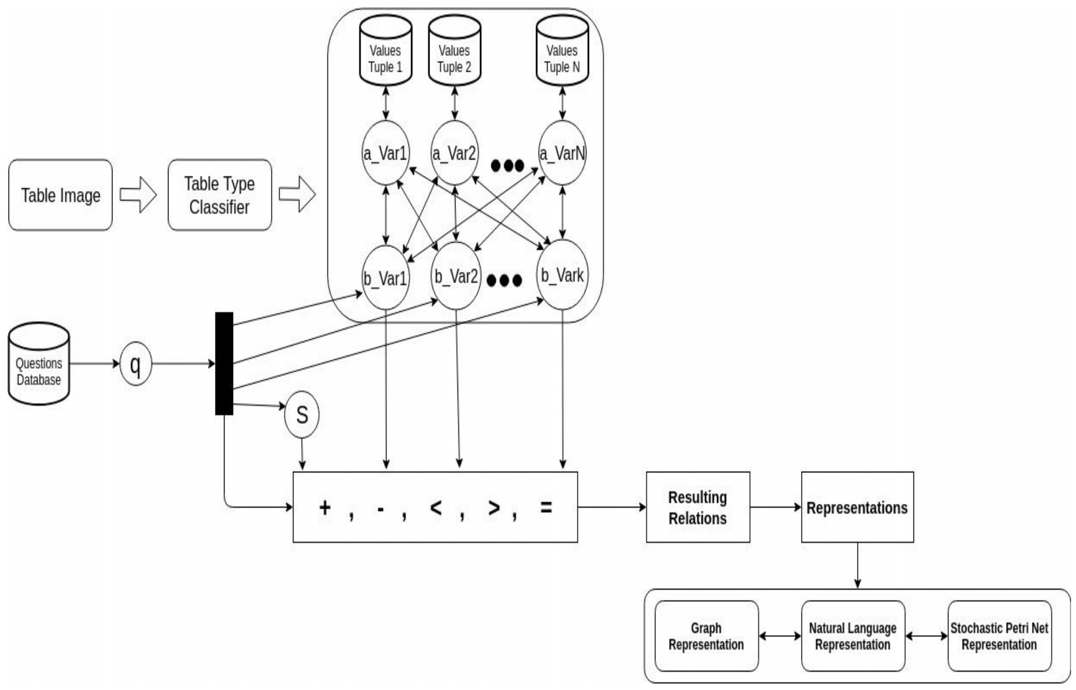


*Figure 17: Multidimensional Table Recognition Methodology*

We present in figures 18 and 19 the methodology diagrams for tables that belong to categories 2 and 3. It is apparent by these diagrams, that the general methodology for these tables remains the same, however, the processing steps are adapted to fit the needs of each table type. These processing steps are derivatives of the Pinakas language described in the previous section. More specifically, in the case of a type 2 table we recognize the row variables of the table. Then, we store the row values into tuples that each corresponds to a specific column variable. This step proves useful for the semantic validation of the generated relations. A similar processing rule applies to an extended table (type 3). However, this time we only store the values of the subcolumn variables. After the association between variables, subvariables and the values of the table is completed, we follow the same processing path as before.

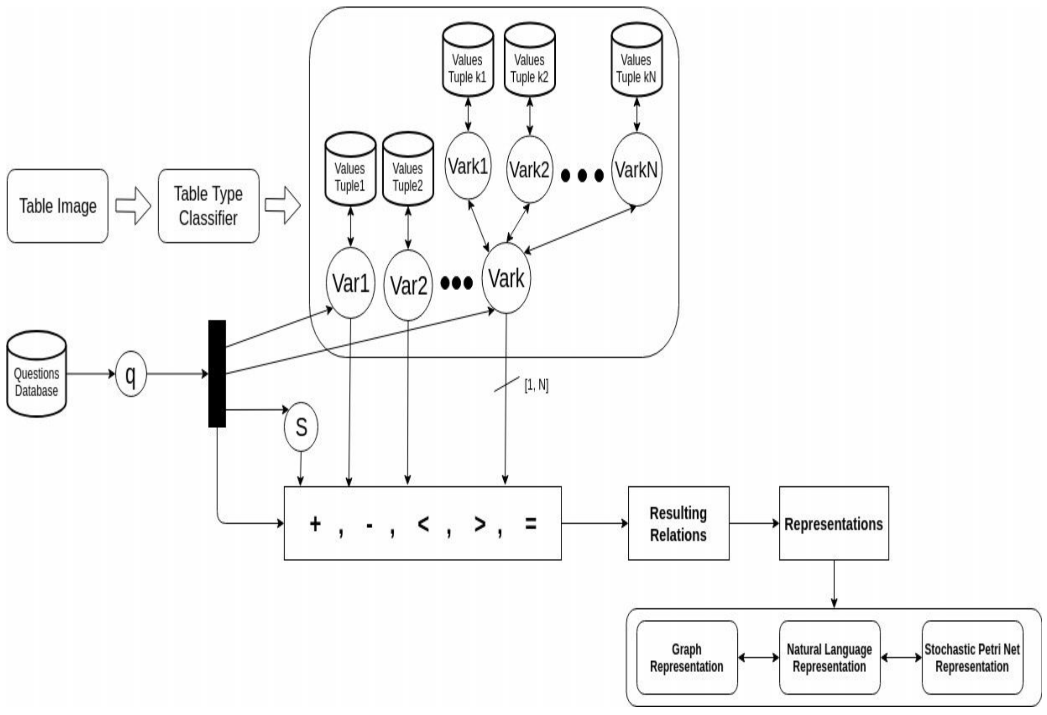


*Figure 18: Extended Table Recognition Methodology*

### 3.5.1.1 Structural Validation Algorithm

Algorithm III showcases the pseudocode for the structural validation algorithm. The algorithm receives a question sequence as input and finds patterns of alternating occurrences between variable symbols and action symbols.

We call variable symbols all of those that are variables of the input table or special symbols that belong to set S. Action symbols are all symbols that indicate operations such as "+" or "-". Relation symbols are "<" and "=", and generally symbols that indicate a relation between agent and patient in the sequence.

The algorithm begins by initializing three flag variables. The "variable_flag" is initialized as True and will turn from False to True every time the next symbol in order of appearance is expected to be a variable. The "action_flag" works in a similar way, although it depends on whether we expect an action symbol or not. The "relation_flag" becomes True only when we find the relation symbol in of the sequence. The algorithm checks every character of the query iteratively, starting from last to first, until the entire sequence is parsed or a mistake in the query's structure is observed. If at any point during this process the appearance of symbols is not in an acceptable order, then the algorithm haults and declares the query as structurally incorrect. If no such problem occurs, the algorithm continues its function until no more characters are left in the sequence. A an acceptable query must always start and end with a variable symbol (variable_flag is False and action_flag is False), and contain always one relation symbol (relation_flag is True) in order to be structurally correct. For example, a query such as " X < Z" is considered as structurally correct, while a sequence such as "X Y < + Z" is not.

**Algorithm III: Structural Validation**

```
$validity\_result \leftarrow False$
$variable\_flag \leftarrow True$
$action\_flag \leftarrow False$
$relation\_flag \leftarrow False$
while $question$ is not empty do
  $symbol \leftarrow popNextCharacter(question)$
  if $(isAgentSymbol(symbol)$ is $True)$ and $(variable\_flag$ is $True$ or $action\_flag$
  is $True)$ then
    if $variable\_flag$ is $True$ then
      $variable\_flag \leftarrow False$
    else
      $action\_flag \leftarrow False$
    end if
  else if $(isActionSymbol(symbol)$ is $True)$ and $(variable\_flag$ is $False$ or
  $action\_flag$ is $False)$ then
    if $variable\_flag$ is $True$ and $relation\_flag$ is $False$ then
      $variable\_flag \leftarrow True$
    else
      $action\_flag \leftarrow True$
    end if
  else if $isRelationSymbol(symbol)$ is $True$ and $variable\_flag$ is $False$
  then
    $relation\_flag \leftarrow True$
    $action\_flag \leftarrow True$
  else
    $variable\_flag \leftarrow True$
    $action\_flag \leftarrow True$
    $break$
  end if
end while
if $relation\_flag$ is $True$ and $variable\_flag$ is $False$ and $action\_flag$ is $False$
then
  $validity\_result \leftarrow True$
end if
return $validity\_result$
```

### 3.5.1.2 Semantic Validation Algorithm

Algorithm IV presents the pseudocode for the semantic validation algorithm for summation queries. Summation queries are those that contain summation operators. The algorithm receives a given relation as its input and parses through its characters individually. For every character it comes across, it checks whether it is a variable, an operation symbol, or the symbol “<”. The occurrence of symbol “<”, indicates that we are at the right side of the relation. That is because, the parsing of the sequence starts from the last character in the left side. If the character is a variable then the algorithm reads the variable’s corresponding value from the value tuples. Then, it appends that value in its corresponding (right or left) summation list. If the character is an operation symbol, then the algorithm sums all the values stored in the list and the list is emptied in order to store new summations. After the calculations of both the left and right summations is completed, the algorithm checks for the validity of the relation. This is achieved by comparing the left and right summation result. The algorithm is repeated for every row of values for the variables in the relation. The relation is considered as verified if and only if it holds true for every row of values. Otherwise, it is not accepted and doesn’t lead to the graph generation step. For the case of subtraction queries the algorithm works in a similar way.

**<u>Algorithm IV: Semantic Validation</u>**

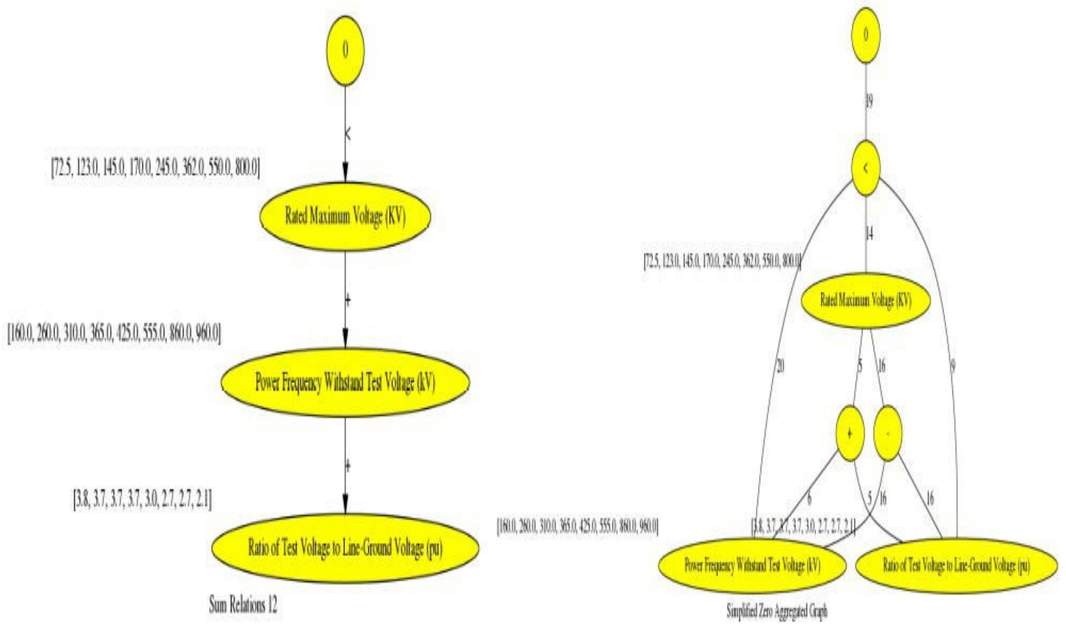
```
$validity\_result \leftarrow True$
Initialize $leftSumList$
Initialize $rightSumList$
$flag \leftarrow False$
for $character$ in $question\_string$ do
  if $character$ is equal to " < " then
    $flag \leftarrow True$
  end if
  if $flag$ is $True$ then
    if $isTableVariable(character)$ is $True$ then
      $temp\_value \leftarrow getVariableValue(character)$
      $rightSumList.append(temp\_value)$
    else if $character$ is equal to " + " then
      $sum \leftarrow addAllListNodes(rightSumList)$
      $clearList(rightSumList)$
      $rightSumList.append(sum)$
    end if
  else
    if $isTableVariable(character)$ is $True$ then
      $temp\_value \leftarrow getVariableValue(character)$
      $leftSumList.append(temp\_value)$
    else if $character$ is equal to " + " then
      $sum \leftarrow addAllListNodes(leftSumList)$
      $clearList(leftSumList)$
      $leftSumList.append(sum)$
    end if
  end if
end for
$right\_sum \leftarrow addAllListNodes(rightSumList)$
$left\_sum \leftarrow addAllListNodes(leftSumList)$
if $left\_sum > right\_sum$ then
  $validity\_result \leftarrow False$
end if
return $validity\_result$
```

## 3.5.2 Conversion of Tables into Attributed Graphs

The idea of using attributed graphs as a tool to represent information has already been described from Bunke et. al. [83]. More specifically, he defines a set of mathematical rules that these graphs have to obey. He uses the attributed graphs in order to present information regarding the data extracted from a circuit diagram.

These data are positions of circuit modules on the board or connections between them, and other information that would be very difficult to represent otherwise. As we have already mentioned on the literature review section, there have also been efforts to use the attributed graphs in order to represent the knowledge that is extracted from database table [97] [98]. However, both of these works consider only the problem of database tables. Database tables have an advantage over table images regarding the processing and understanding of their information, since they maintain the relationships between variables through private and foreign keys.

The proposed methodology generates attributed graphs for each verified tabular relation, directly after its validation process is completed. For the conversion of a verified relation to attributed graph, we use the same strategy with the papers that we mention. That strategy suggests representing the variables as nodes, the values of the variables as attributes of the nodes, and the operators between them as labeled arcs. These arcs connect the nodes that the operators correspond to. The direction of the arcs depends on the priority of the operation between two variables. Furthermore, we can aggregate the attributed graphs of all relations into single attributed graphs. Different styles of aggregation can help us highlight certain relations and conceal others. For example the right graph in figure 18 helps us deduce the importance of the recognized table variable "Rated Maximum Voltage ", since it is the node with the most associations with any other variable. The developed tool performs aggregation on the nodes, and either simple or extensive aggregation on the arcs. Simple arc aggregation unifies all the labeled  arcs based on relation type, while extensive arc aggregation merges all the labeled arcs despite  any differences in label names. Figures 19 illustrates the outcome attributed graphs for the case  of the simple table that is presented in figure 9. The left graph represents a summation relation  and the right graph shows an aggregated version of all the connections. Figure 20 shows the  resulting graph for the same simple table, but this time the nodes have been aggregated. some of  the outcome attributed graphs as given by the methodology here, for the case of the simple table  presented in figure 9.



*<u>Figure 19: </u>An attributed graph of relation "0 < Rated Maximum Voltage + Power Frequency Withstand Test Voltage + Ratio of Test Voltage to Line-Ground*

*Voltage" (left) and the aggregated graph based on arcs from the simple table in Fig.9 (right).*

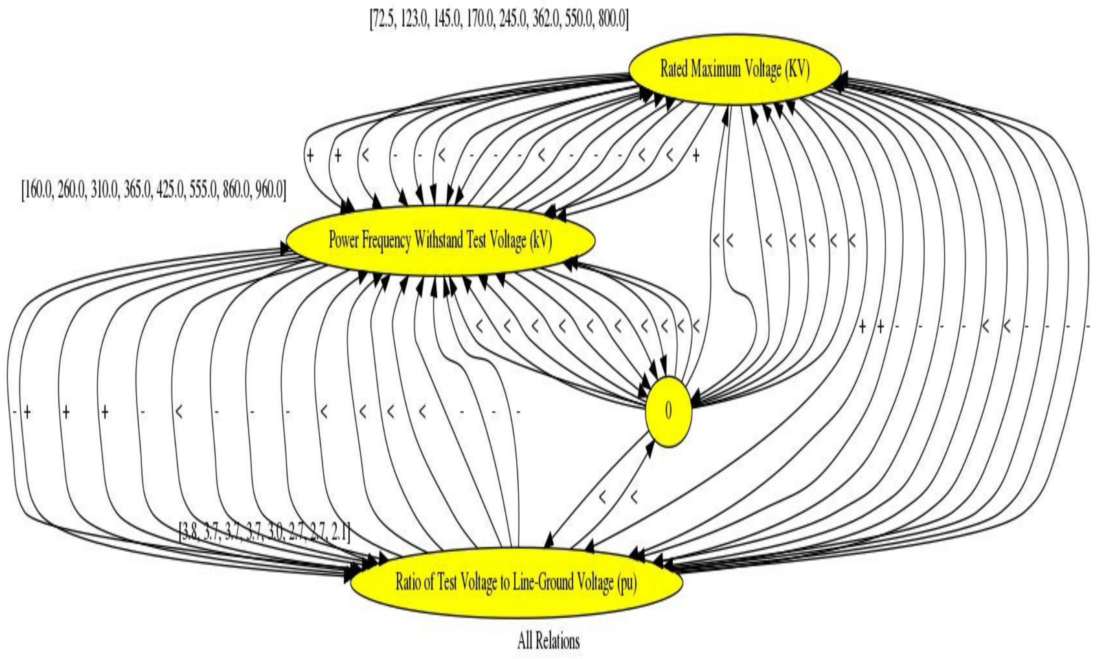


*Figure 20: The attributed graph of all verified relations from the simple table in Fig. 8 based on the names of the nodes.*

Figure 21 showcases attributed graphs generated by relations extracted from the multidimensional table of figure 9. This time we include the nodes that represent the row variables as well. Each row variable is connected to a column variable with an attributed arc. This connection represents the passing of a value or action from between two variables. The label of each arc is the row variable's corresponding value for this specific column variable. That value is extracted from the table image. Figure 22 illustrates an attributed graph for all these relations, with aggregation based on the names of the nodes on each relation.

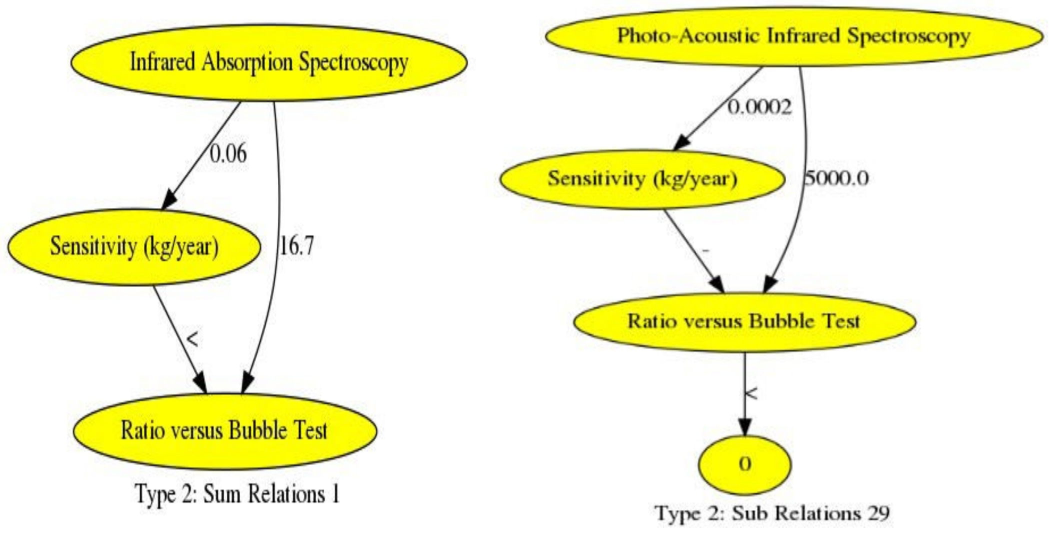


*Figure 21: Example attributes graphs of relations that are extracted from the multidimensional tab l e in Fig. 9.*

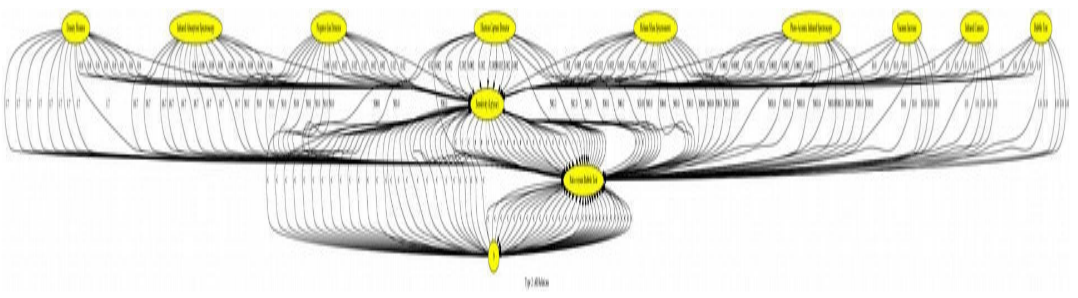
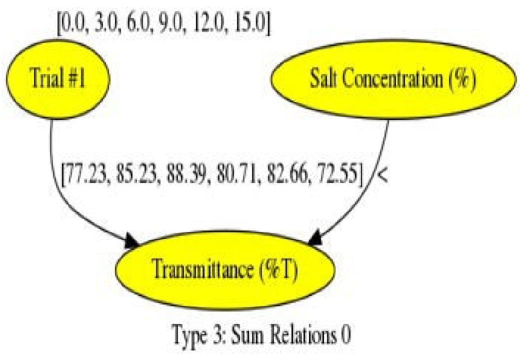
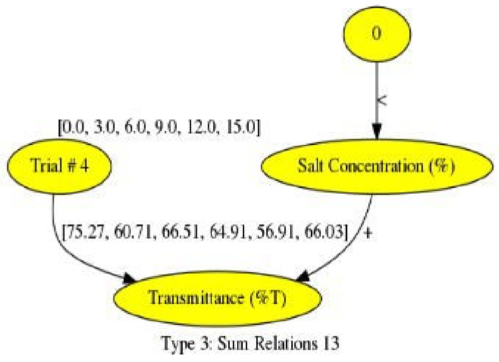


*Figure 22: The attributed graph of all verified relations from the multidimensional table in Fig. 9 based on the names of the nodes.*

Similarly, figures 23 and 24 present separate attributed graphs of validated relations and an attributed graph aggregated on node names respectively. The relations for these graphs were retrieved from the extended table in figure 8. The main difference is the addition of the column subvariable nodes. Each column subvariable is connected with each corresponding column variable through a labeled arc. The value of the arc is a tuple of all the extracted values from that specific column subvariable. It should be highlighted, that the graphs in figure 22 might not be easily comprehensible to humans, but they were never intended to. The goal of this research is to automate the understanding procedure, by emulating the behavior of the humans. Therefore, we only need the program to actually understand the produced graphs.





*<u>Figure 23:</u> Example attributes graphs of relations that are extracted from the extended table in Fig. 9.*

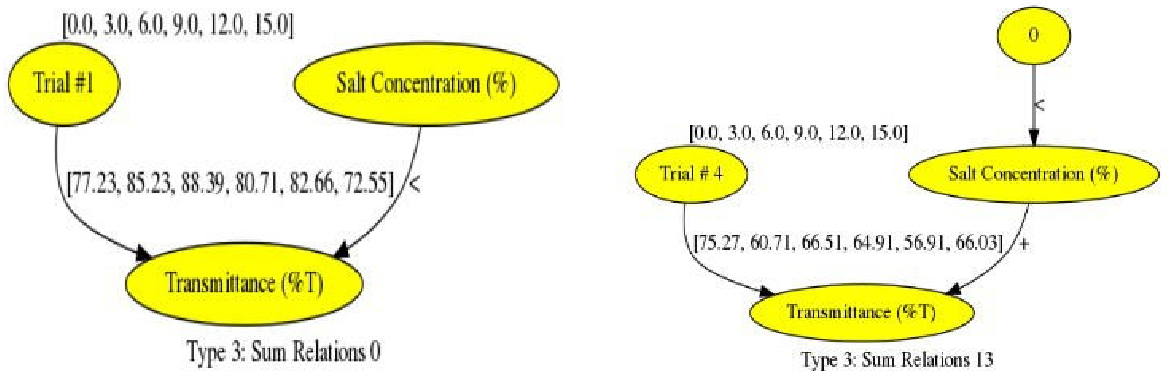


*Figure 24: The attributed graph of all verified relations from the extended table in Fig. 9 based on the names of the nodes.*

### 3.5.3 From Graph to Natural Language Representation

After the successful generation of the attributed graphs, the tool proceeds with their transformation into natural language sentences. These sentences prove to be an important transitional step for the creation of the corresponding stochastic Petri-net graphs. The conversion of each graph relation into a natural language sentence is achieved through a modified version of the structural validation algorithm (Algorithm III). For this specific task, we also introduce the concept of conjunction words. These words fill in the semantic gaps between variables names and action verbs. Conjunction words can either be Agent-to-Patient or Patient-to-Agent words depending on the positions of the characters they refer to in the initial variable relation.

For example, the variable relation “Y + X < Z” is transformed into “The values of Y plus the values of X are greater than the values of Z”. In this example the word “plus” is an Agent-to- Patient conjunction word, added to the sentence in order to reinforce its meaning. More examples of produced natural language sentences are presented below. These specific natural language sentences have been generated from the extracted data of the simple table that is presented in figure 9. They correspond to the attributed graph of figure 20. NLS1 and NLS10 are the equivalent natural language sentences of the relations “Rated Maximum Voltage < Power Frequency Withstand Test” and “ 0 < Rated Maximum Voltage + Ratio of Test Voltage to Line- Ground Voltage” respectively. More specifically, in the case of NLS10, this time the word “plus” is Patient_to_Agent conjunction word. These relations are already verified by our tool, during the process of transforming the extracted table information into attributed graphs. The final natural language sentences represent the extracted internal associations between the variables of the input table.

***NLS1:*** ***“The values of Power Frequency Withstand Test Voltage (kV) are greater than values of Rated Maximum Voltage (KV)”***

***NLS2:*** ***“The values of Rated Maximum Voltage (KV) are greater than values of Ratio of Test Voltage to Line-Ground Voltage (pu)”***

***…***

***NLS5:*** ***“The values of Power Frequency Withstand Test Voltage (kV) are greater than zero”***

…

***NLS10: "The values of Rated Maximum Voltage (KV) plus values of Power Frequency Withstand Test Voltage (kV) are greater than the values of Ratio of Test Voltage to Line-Ground Voltage"***

We illustrate below some example the natural language sentences that are produced from the multidimensional table in figure 9. They correspond to the attributed graph of figure 22. They have a similar structure to the sentences produced from the simple table. The only difference is the addition of the "Row-X" part of the sentence. We need this part to indicate both to the reader as well as to the tool, the values of each row that the relation relation corresponds to. For example, NLS1 represents the results of evaluating the "Ratio versus Bubble Test" variable against the "Sensitivity" variable, when the values of both variables come from row "Density Monitor".

***NLS1: "Row-Density Monitor: The values of Ratio versus Bubble Test are greater than the values of Sensitivity"***

***NLS2: "Row-Infrared Absorption Spectroscopy: The values of Ratio versus Bubble Test are greater than the values of Sensitivity"***

***NLS3: "Row-Negative Ion Detector: The values of Ratio versus Bubble Test are greater than the values of Sensitivity"***

…

***NLS19:*** ***"Row-Density Monitor: The values of Ratio versus Bubble Test are greater than zero" NLS20: "Row-Infrared Absorption Spectroscopy: The values of Ratio versus Bubble Test are greater than zero"***

***NLS21:*** ***"Row-Negative Ion Detector: The values of Ratio versus Bubble Test are greater than zero"***

The natural language sentences that are produced from the extended table of figure 9 follow the same structure as those of the multidimensional table. However, we use "Column-X" instead of "Row-X" in order to highlight the subvariable that the produced sentences correspond to. For example, NLS1 represents the results of evaluating the "Transmittance" variable against the "Salt Concentration" variable, when the former contains the values of the subvairable "Trial #1". These sentences correspond to the attributed graph of figure 24.

***NLS1:*** ***"Column-Trial #1 in Variable- Transmittance (%T):The values of Transmittance (%T) are greater than the values of Salt Concentration (%)"***

***NLS2:*** ***"Column-Trial #2 in Variable- Transmittance (%T): The values of Transmittance (%T) are greater than the values of Salt Concentration (%)"***

***NLS3:*** ***"Column-Trial #3 in Variable- Transmittance (%T): The values of Transmittance (%T) are greater than the values of Salt Concentration (%)"***

**…**

***NLS6:*** ***"Column-Trial #1 in Variable- Transmittance (%T): values of Transmittance (%T) are greater than zero"***

*<u>NLS7: </u>"Column-Trial #2 in Variable- Transmittance (%T): values of Transmittance (%T) are greater than zero"*

**…**

*<u>NLS13: </u>"Column-Trial #3 in Variable- Transmittance (%T): values of Salt Concentration (%) plus values of Transmittance (%T) are greater than zero"*

### 3.5.4 From Natural Language to SPN Representation

The aim of this research is translation of the information and relations that we have extracted from a technical document table, into a common form of representation with the rest of the modalities. The generation of the natural language sentences that describe the relations between the variables of a table, have enabled us to produce the corresponding stochastic Petri-net graphs. For the conversion of the natural language sentences to SPNs, we use the methodology that is employed in [75, 76] combined with the Pinakas formal language. The proposed methodology suggests that every natural language sentence can be described by the AVP format (agent → verb(action) → patient). This means that a sentence can be reduced to agents, verbs, and patients. We consider as agents the nouns that perform an action. Similarly, the patients are the nouns affected by actions. This approach extracts the AVP kernels of natural language sentences and converts them into stochastic Petri-net graphs, with agents and patients represented by input and output places respectively, and verbs represented by transitions.

The illustrated natural language sentence examples of the previous section, the verbs are either the “are greater than” or “are equal to”. These sequences represent the equivalent relations of the same name. A transition is always generated for the verb. The agents and the patients are all the variable names on the right and left side of the verb respectively. The developed tool initially recognizes the variables as well as their respective conjunction words in the natural language sentence. Then, it generates transitions for every conjunction word, and places for every variable. It also generates places for the results of relations implied by the set of either Agent-to-Patient or Patient-to-Agent conjunction words. These places are categorized into “add”, “sub” or plain result places depending on the type of relations between variables they represent. These relations can either be addition, subtraction or no relation at all. They are also divided into results type “1” and “2” based on their positions in the sentence. Result places type “1” are located left of the verb (Agent_to_Patient), indicating relation between agents variables, while results type “2” are positioned on the other side of the verb (Patient_to_Agent), indicating relation between patient variables. For example “add result 1” represents the result of an addition relation between the agents of the sentence. All variable places are input places, while result places can be both input and output places, working as a link between all relations. Figure 25 illustrates the generated stochastic Petri-net graph of the natural language sentence that correspond to the simple table in figure 9. The resulting SPN includes information about all the recognized addition and subtraction relations, as well as which values are in ascending or descending order. The latter semantic association can prove useful for the correlation of different modalities in future work.

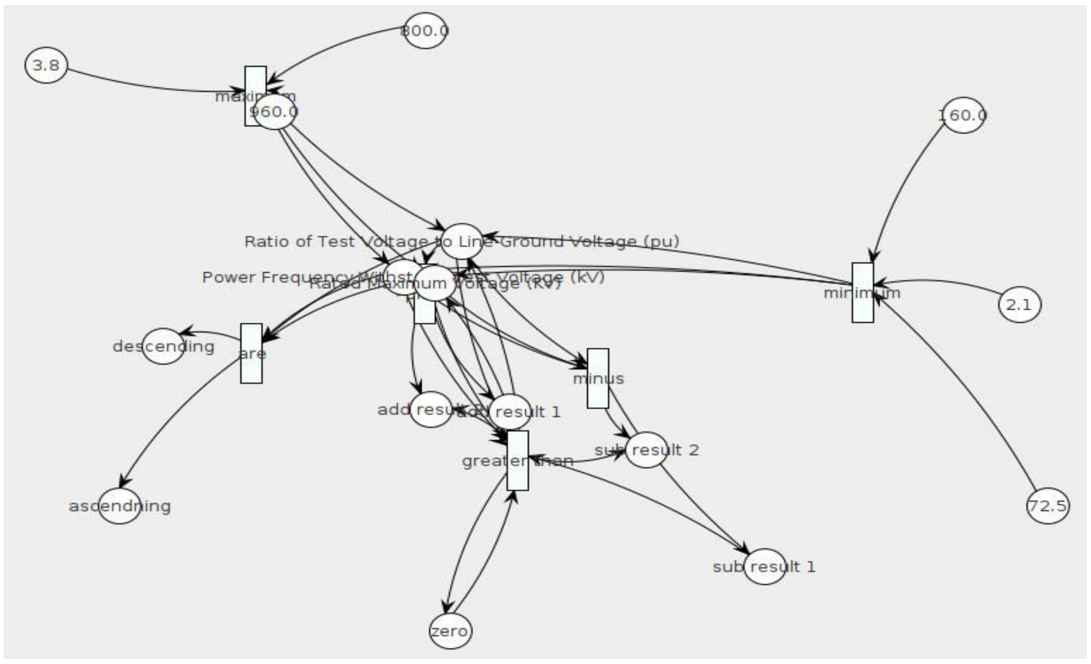


*Figure 25: The SPN graph that is generated from the sentences that correspond to the simple table in figure 9.*

Furthermore, figures 26 and 27 illustrate the resulting SPN graphs for the multidimensional and extended tables of figure 1 respectively. Figure 26 contains additional information about the connection of the row variables to the column variables of the table image. to this method, a natural language sentence can be reduced to its agent words, verb words and patient words. Similarly, figure 27 contains additional information about the connection of the table variables with their respective subvariables. Both SPNs include the aforementioned ascending and descending semantic knowledge.

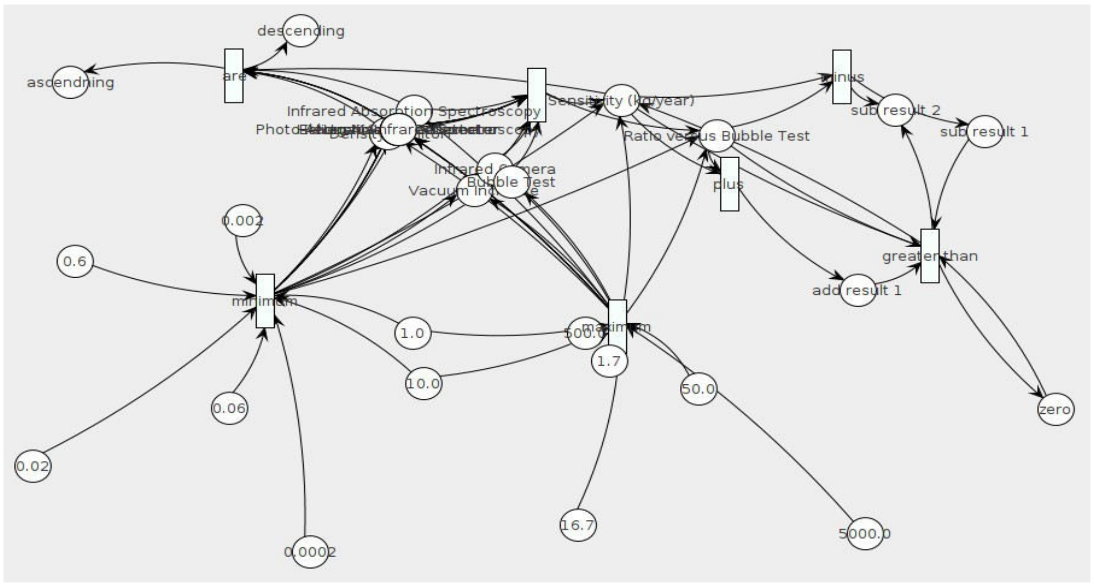


*Figure 26: The SPN graph that is generated from the sentences that correspond to the multidimensional table in figure 9.*

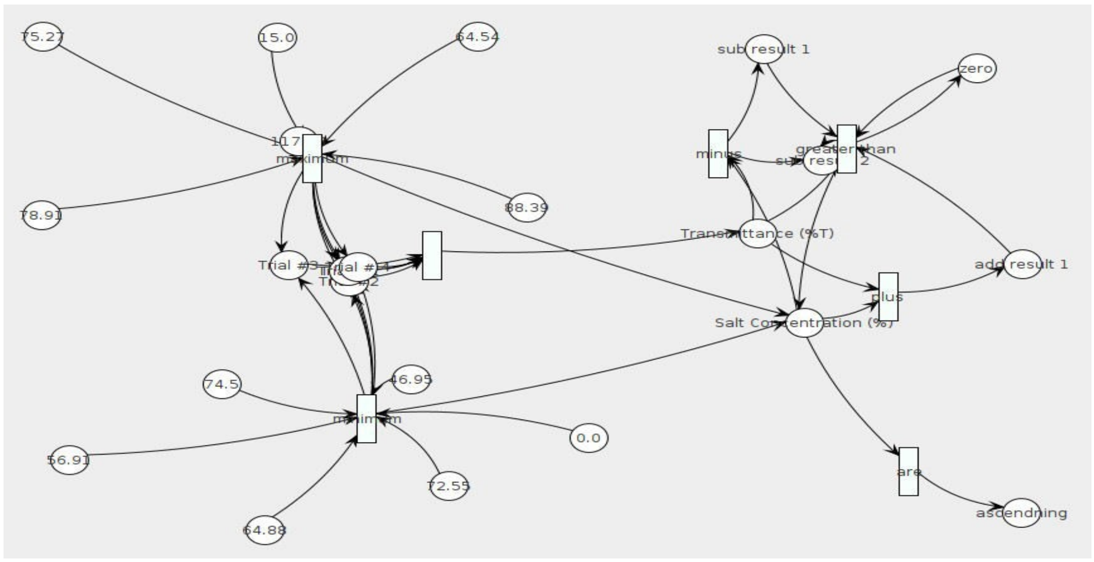


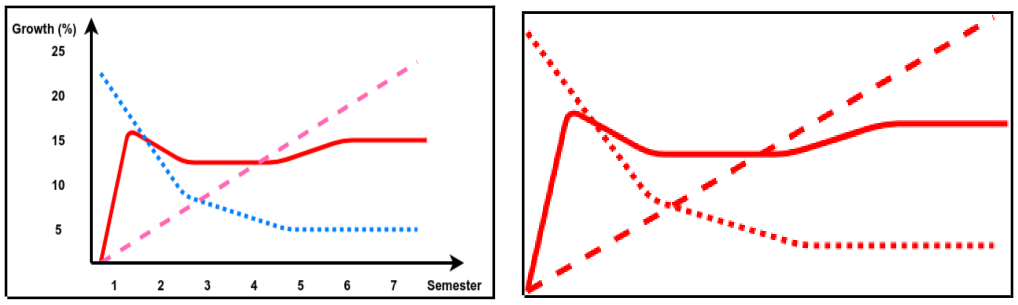
*Figure 27: The SPN graph that is generated from the sentences that correspond to the extended table in figure 9.*

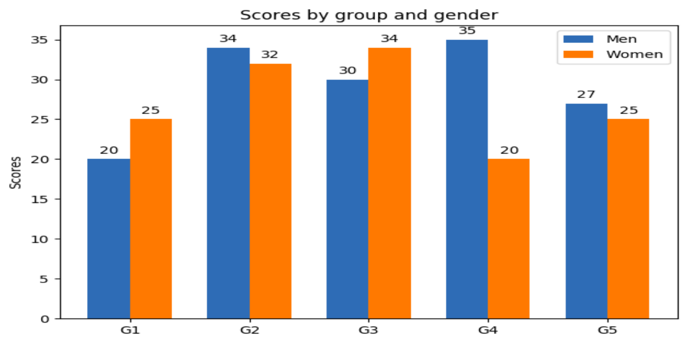
# 4. Processing of Graphics Images

## 4.1 Graphics Processing Methodology

The literature review section proves that the fields of graphics detection, recognition and understanding have been addressed using many different approaches. However, previous recognition applications have been limited to capturing only the information that is easily and visibly accessible such as legend text extraction. Similarly, previous graphics understanding works focus on producing basic summarization of the illustrated information, without capturing the internal semantics. In this study we focus on understanding the deeper information that are not visually accessible in a graphics image and extract the underlying behavior that describes its curves and bars. We use the same architecture diagram that describes the overall system in figure 8. The extracted knowledge resulting from the understanding of graphics imagse is represented using an SPN graph.

After the modality image is extracted from the input document and it has been classified as graphics, then it is parsed to the graphics recognition module. This module analyzes a given graphics image in order to isolate its curves or bars from the axis. In case the graphics image contains curves, then the recognition methodology proceeds with the detection each curve's middle points. These are the 2-dimensional (2D) points where the direction of a curve pivots. The deduced middle points are used for the construction of the straight line segments that describe the curves. In case the input image contains bars, then we use the top points are their respective middle points.





*<u>Figure 28:</u> Mixed curves of different colors (left), partitioned curves of the same color (right) and bars chart (bottom).*

Finally, the produced straight line segments in both curves and bars are used in the graphics understanding module in order to deduced structural and functional information. The deduced curve behavior and functionality is represented using SPN graphs, which are produced from the aforementioned natural language sentences. For this reason, the Kyrtos formal language is introduced as a tool that helps us map the natural language text into SPN kernels.

The goal of this research isn’t to design an application that can analyze any possible graphics type, but to prove that such an application can exist. Thus, we focus on 3 different types of graphics images as proof of concept for the presented graphics processing methodology. Figure 28 illustrates examples from these types of tables, which are partitioned or continuous (mixed) curves of different colors (type 1), partitioned curves of the same color (type 2) and bars graphics (type 3).

## 4.2 Detection of Graphics

### 4.2.1 Image Pyramids

We apply the pyramidal reduction technique in order to study their effects on the prediction accuracy results. Image pyramids [96] reduce the amount of information that is illustrated in each input image and preserves only the essential structural features. More specifically, we deploy the Gaussian pyramids approach into our methodology. We start by reducing the size of the input image to different levels. At each iteration of the procedure, we apply a Gaussian noise filter. Then, the image is scaled back to its original size. We illustrate an example of pyramidal processing for a graphic image in figure 29. It is noticeable that if pyramidal reduction is applied too much, then there can be significant loss of information. Through trial and error, we deduce that the best level for pyramidal reduction of graphics images is level 3.

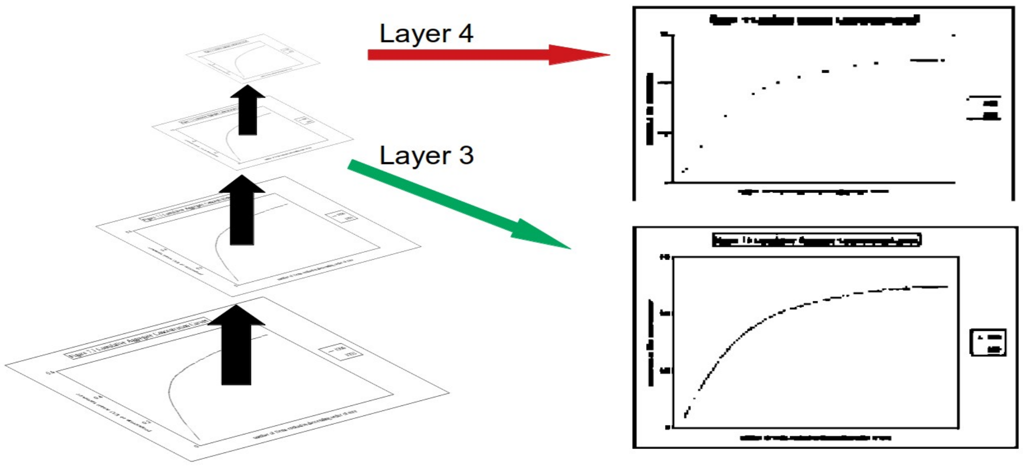


*Figure 29: Example pyramid image reduction for 4 levels*

### 4.2.2 Learning Based Detection

Image detection using deep neural networks have increased their popularity in years. This preference can be attributed to the multiple advantages that neural networks have to offer such as ease of implementation. In the current comparative study, we employ one-class, binary, and multi-class models. Moreover, we vary the studied techniques by including different types of preprocessing. More specifically, we implement the one-class classification algorithms of One-Class Support Vector Machine (SVM) [78] and the Isolation Forest [79]. An SVM is supervised learning algorithm for distinguishing input data between acceptable and outliers. Similarly, Isolation forest is unsupervised learning algorithm, which is also used for detecting inconsistent observations. One-class models are more commonly used for anomaly detection in 1- dimensional data consisting of numerical series. Hence, we convert the

2-dimensional images into 1-dimensional arrays using a raster scan. Then, we feed them into the SVM and Isolation Forest algorithms respectively. We train these algorithms purely on positive data. This implies that any other types of images besides tables are considered outliers.

Furthermore, we train binary classification and multi-class models with the use of Convolutional Neural Networks (CNN) [80]. CNN models are the state of the art solution for fast and accurate image classification. A CNN consists of convolution layers, pooling layers, and fully connected layers at the end. For the convolution layer, a sliding kernel of predefined size shifts over the input image, and performs convolutions on a single neighborhood of pixels at a time. Then, a pooling layer accepts the resulting image and preserves only the essential edges trough max or min pooling. Finally, the fully connected layers receive the corresponding output and return a N dimensional array containing the probabilities of the image being a member of class. The N is the number of user defined categories for classification. A binary CNN model is trained on both positive and negative data, and uses the binary cross-entropy as its loss function. On the contrary, multi-class CNN model trains for images from all available classes and uses the categorical cross-entropy as its loss function.

We train two versions for each of the aforementioned models and algorithms. The first version receives pyramidal reduced images as input. The second version accepts binary images without any additional processing. We illustrate the diagram of the binary CNN model that we have implemented in figure 30. This model is trained on images that have been initially parsed through pyramidal reduction. The rest of the CNN models that we deploy showcase similar structure.

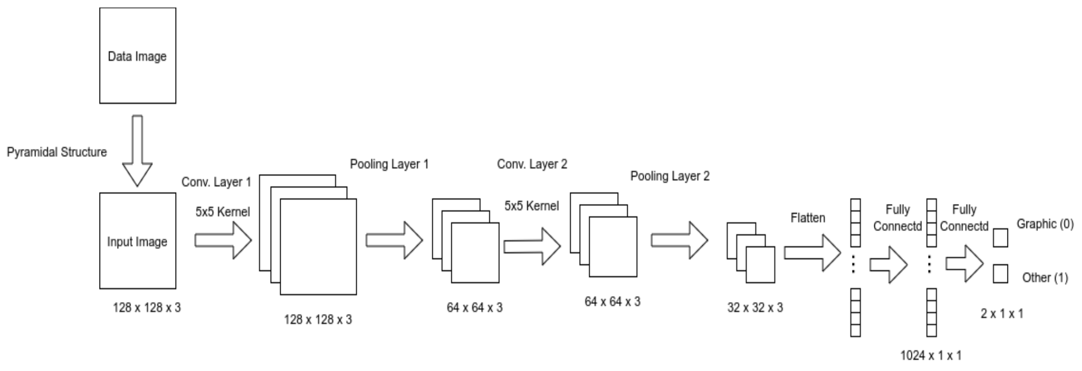


*Figure 30: Example of binary classification model for graphics images that have been processed with pyramidal reduction.*

### 4.2.3 Rule-based Detection

The proposed rule based graphics detection methodology is a combination of geometric mathematical formulas and image processing techniques. The programs initiates with the application of binarization and Gaussian pyramidal reduction on the input image. Gaussian pyramidal reduction enables the preservation of the repeated structural characteristics and converts them into black pixel chunks (connected components). A common structural feature for each graphic is that all curves are contained either inside a box or inside an axis. Next to these axis lines, there are the corresponding names of each axis line.

After the application of pyramidal reduction, the names of each axis line is converted into chunks of black pixels. However, the centroids of these chunks maintain the central positions of their corresponding to the original text words. So, we detect the chunks of black pixels that are positioned next to the axis, and calculate their respective centroids positions. The text names of the axis lines should be positioned in their middle with respect to each axis position. Thus, if we extend vertical lines from the centroids of the black pixel chunks, then these lines should be perpendicular with each other. Additionally, their intersection point should be relatively close to the center of the box or the axis that delimits the curves. The proposed rule-based classification method detects if such an intersecting point exists in a given image. If it does, then the given image is assigned to the graphics category. Figure 31 illustrates an example of our proposed methodology.

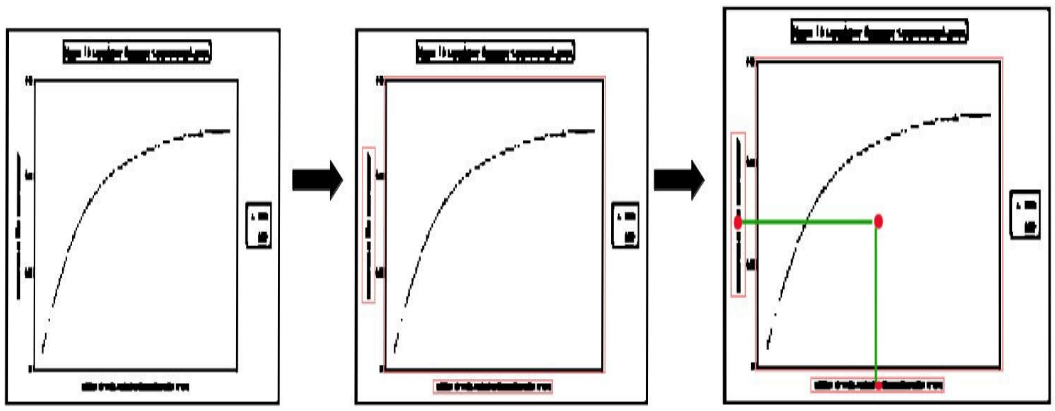


*Figure 31: An example of the rule-based detection methodology for graphics based on structural features.*

### 4.2.4 Evaluation Results

The training and testing datasets are generated from the scraping of tables and graphics images in IEEE journals. The total number of training images accumulates to 2100. Furthermore, we employ the K-fold technique, due to the small size of the training dataset. We deduce the value K=10 as optimal through a trial and error testing procedure. The IEEE test set accumulates to a total number of 100 images 48 of which are graphics and 52 are tables. Finally, we also use a part of the ICDAR-2013 test set for additional evaluation of the studied methodologies. The latter dataset contains 53 tables images and 29 graphics images.

The tables (5 and 6) showcase the classification outcome scores for the IEEE and ICDAR-2013 test sets respectively. We evaluate the studied methodologies based on the precision, recall and F1-score metrics. GOCSVM corresponds to one-class SVM for graphics, PGOCSVM to one-class SVM for pyramidally processed tables, GIF to Isolation Forest for graphics, PGIF to Isolation Forest for pyramidally processed graphics, GBCNN to binary classification for graphics, GPBCNN to binary classification for pyramidally processed graphics, MCNN to multi-class classification, PMCNN to multi-class classification with pyramidal reduction, RGC to rule-based graphics classification, HYBRID to the learning methodologies that are combined through the voting system. It must be noted that the HYBRID methodology follows a similar approach to that from chapter 3, only this time we use the aforementioned Graphics learning methodologies.

The results of the IEEE dataset indicate that pyramidal preprocessing has a negative impact on the total classification accuracy for the recognition of graphics images. We notice that is decreases the achieved accuracy of the pyramidal based models, when compared to the original models. More specifically, the binary CNN model achieves the highest accuracy for the IEEE test set. In contrast, the ICDAR-2013 test results highlight the multi-class CNN model which is trained on pyramidally reduced images.

**Table V: Graphics Classification Scores using IEEE test set**

| | Accuracy | F1-score | Precision | Recall |
|---|---|---|---|---|
| **GOCSVM** | 0.68 | 0.75 | 0.60 | 0.98 |
| **PGOCSVM** | 0.26 | 0.38 | 0.32 | 0.48 |
| **GIF** | 0.67 | 0.74 | 0.60 | 0.96 |
| **PGIF** | 0.27 | 0.39 | 0.32 | 0.48 |
| **GBCNN** | 0.93 | 0.93 | 0.90 | 0.96 |
| **GPBCNN** | 0.90 | 0.90 | 0.88 | 0.92 |
| **MCNN** | 0.90 | 0.89 | 0.93 | 0.85 |
| **PMCNN** | 0.91 | 0.91 | 0.90 | 0.92 |
| **RGC** | 0.43 | 0.43 | 0.43 | 0.43 |
| **HYBRID** | 0.90 | 0.90 | 0.90 | 0.94 |

**Table VI: Graphics Classification Scores using ICDAR test set**

| | Accuracy | F1-score | Precision | Recall |
|---|---|---|---|---|
| **GOCSVM** | 0.39 | 0.54 | 0.37 | 1 |
| **PGOCSVM** | 0.768 | 0.54 | 0.92 | 0.38 |
| **GIF** | 0.424 | 0.54 | 0.37 | 1 |
| **PGIF** | 0.756 | 0.52 | 0.85 | 0.38 |
| **GBCNN** | 0.671 | 0.61 | 0.53 | 0.72 |
| **GPBCNN** | 0.83 | 0.93 | 0.93 | 0.83 |
| **MCNN** | 0.804 | 0.71 | 0.74 | 0.69 |
| **PMCNN** | 0.902 | 0.85 | 0.96 | 0.76 |
| **RGC** | 0.62 | 0.62 | 0.62 | 0.62 |
| **HYBRID** | 0.793 | 0.70 | 0.71 | 0.69 |

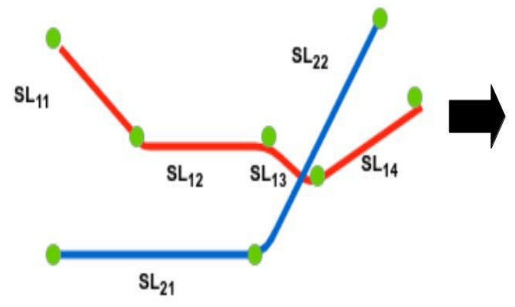
## 4.3 The Kyrtos Formal Language

In general, formal languages are deployed for the representation of internal semantic and syntactic features that are recognized from a natural language sentence. In this paper, we aim to retrieve the knowledge that is illustrated in different technical document modalities and convert it into a common form of representation. As we have already mentioned, that common representation format is the stochastic Petri-net, because of its capabilities to describe the internal functionality of each technical document modality. Therefore, we develop the Kyrtos formal language as the means to map the retrieved associations of each curve's line segments into SPN kernels. The presented work is based on the formal language Glossa [75]. Glossa is a language used for the representation of kernels of natural languages sentences that are extracted from a document. This methodology proves to be useful for the mapping of those extracted kernels into SPNs. The Glossa language has its own set of alphabet, grammar, and operators and aims to formulate all the possible different relations. Similarly, we define our own alphabet, grammar, and operators in an effort to fit every graphics type in our language.

### 4.3.1 Alphabet

We begin with the definition of the alphabet for the Kyrtos language. In order to achieve this we must consider each curve of the graphics image as its own entity. Each curve consists of straight line segments that connect with each other for an angle of specific degrees. For the case of bar graphics, we can form conceptual curves based on the top middle points of each bar. The symbol $SL_{ij}$ now represents the jth straight line segment for the ith curve of the graphics image. So, $C_i$ = {x|x is the ith curve of the input graphics image GR}. Additionally, $SL_{ij}$ = {x|x is a straight line segment that constitutes part of the curve Ci from the input graphics image GR}. Finally, $Q_{ij}$ = {x|x is the associations of between the straight line segments of the ith curve and the jth curves respectively from the input graphics image GR}. We aim to formulate the connections between different line segments of the same curve. We also want to formulate the associations between straight line segments of different curves.

We illustrate an example of formulation in figure 32. This simple graphics image GR is described by the structural information of its curves $C_1$ and $C_2$, as well as their associations with each other. These associations are represented in the form of intersection and parallelism relations between their corresponding straight line segments. For example, "$SL_{13} >< SL_{22}$" corresponds to the intersection between straight line segments $SL_{13}$ and $SL_{22}$ from curves $C_1$ and $C_2$ respectively. Furthermore, "$SL_{11} \sim SL_{12}$" illustrates the connection that is formed between segments $SL_{11}$ and $SL_{12}$ from $C_1$. Finally, $VAL_{CON}()$ represents the value of the particular connection which the the degrees of the formed angle. Similarly, $VAL_{PAR}()$ represents the value of the common slope between parallel segments and $VAL_{INT}()$ represent the location of the intersection point between the segments of the corresponding curves.



- $GR \rightarrow [C_1]@\,[C_2]@\,[(SL_{12} = SL_{21})\ @\ (SL_{13} >< SL_{22})]$
- $C_1 \rightarrow\ [(SL_{11} \sim SL_{12})\ @\ (SL_{12} \sim SL_{13})\ @\ \ldots]$
- $C_2 \rightarrow\ [(SL_{21} \sim SL_{22})]$
- $VAL_{CON}(SL_{21} \sim SL_{22}) = 120^{o}$
- $VAL_{PAR}(SL_{12} = SL_{21}) = 0$

*Figure 32: Formulation Example for Simple Curves.*

### 4.3.2 Grammar

Now that we have defined the alphabet for Kyrtos, we have to describe its grammar G. We define G = (VN, VT, GR, F), where:

- VN is the set of non-terminal symbols with VN ={Q, SL, C, GR}
- VT is the set of terminal symbols with VT = {V, @, !}
- GR represents the starting symbol of the graphics image
- F is the set of production rules

More specifically, the set of production rules F includes:

- $GR \rightarrow [C_i]$ @ … @ $[C_K]$ @ $[Q_{i\,i+1}]$ @ ... @ $[Q_{KN}]$
- $Q_{iN} \rightarrow [(SL_{i\,j} = SL_{K\,N})$ @ ... @ $(SL_{i\,j+1} >< SL_{K\,N-2})$ @ … ]
- $C_i \rightarrow [(SL_{i\,j} \sim SL_{i\,j+1})$ @ $(SL_{i\,j+1} \sim SL_{i\,j+2})$ @ … @ $(SL_{i\,N-1} \sim SL_{i\,N})]$

### 4.3.3 Operators

In the productions rules that we define above, we use some specific symbols. We explain these symbols below:

- “><” is an operator that represents intersection between line segments of two different curves.
- “=” is an operator that represents parallelism between line segments of two different curves.

- "~" is an operator that represents connection between line segments of the same curve.
- The "@" operator represents the "and" connection between different generated relations.
- The "!" operator represents the "or" connection between different generated relations for the same curve. However, only of these relations can hold true. We use this operator when there are multiple interpretations for the same graphics image. Examples of such use case are the partitioned graphics images of the same color.
- [] and () are used to determine the scope of different operations.

## 4.4 Recognition of Graphics

### 4.4.1 Recognition of Colored Partitioned and Continuous Curves

#### 4.4.1.1 Preprocessing of Input Graphics Image

The methodology initiates with the recognition of the axis and extraction of the subimage that is contained within its boundaries. We aim to extract all the necessary information from the subimage and then associate these results with the original graphics image. In order to accomplish that goal, we must first understand what type of axis is illustrated in the graphics image. Figure 33 showcases the possible axis types that we could come across.

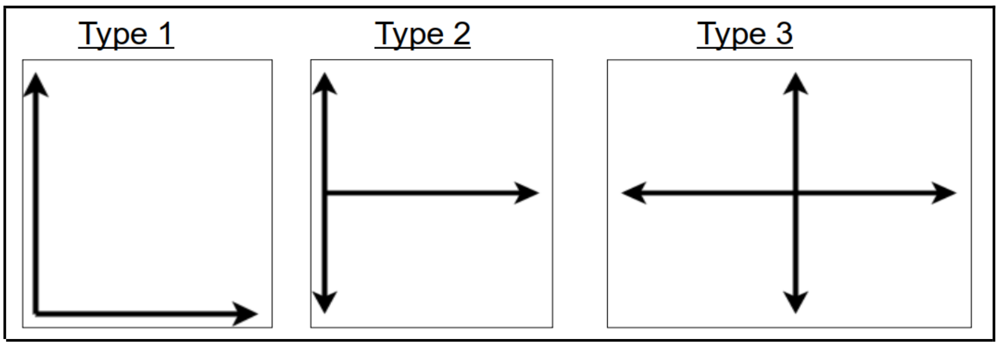


*Figure 33: Possible Axis Types*

There is an abundance of existing image processing techniques for recognizing pixels that form edges or lines, and the distinction of objects in images from their background color. These techniques include standard binarization, segmentation, etc. However, these approaches lead to significant loss of information, when the input image is a graphic. Problems such as missing edges, miscoloring of line segments, confusion between curves and grid during recognition consist possible outcomes. In some cases there can be complete loss of entire curves in the graphics image during binarization. In order to avoid these issues, we have developed an approach based on a combination of image enhancing techniques and the HSV color space.

More specifically, we apply the contrast and sharpen filters on the original image, before transforming their results into the HSV color space. Both techniques enhance different aspects of the graphics image. They also allow for the recognition of any curves despite their color. This is an attribute that is missing from the methodologies that we discuss in the literature review section. Contrast is the difference in light reflected by the objects in an image, making the objects easy to distinguish from one another. To enhance the contrast factor in an image, means to enhance that difference between objects' reflected light. So we its respective result in order to recognize the grid in the subimage that contains the curves. In addition, the sharpening technique for colored images is similar to the skeletonization technique for binary images. We use its respective result in order to recognize graphic lines. An example of this aforementioned methodology is illustrated in figure 34. It must be noted that there was no background grid in the input graphics subimage and therefore the grid result is empty. We can successfully distinguish the grid from the graphic lines, by grouping all pixels of the input subimage based on their color. Then, we recognize the colors that belong to the lines and parse their corresponding pixels with a kernel. The background pixels always form the greatest group, since they cover the most space in the image. The rest of the colors are either part of the background grid (if such a grid exists) or the curves.

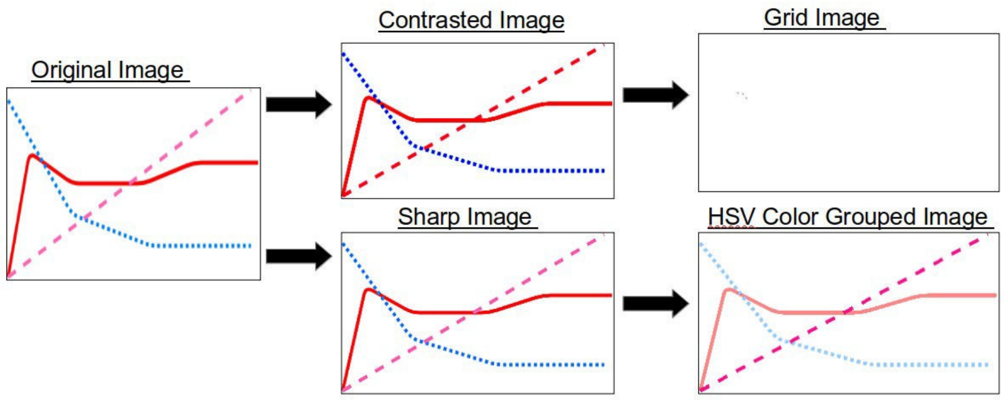


*Figure 34: The results of contrast, sharp and HSV homogeneity filters.*

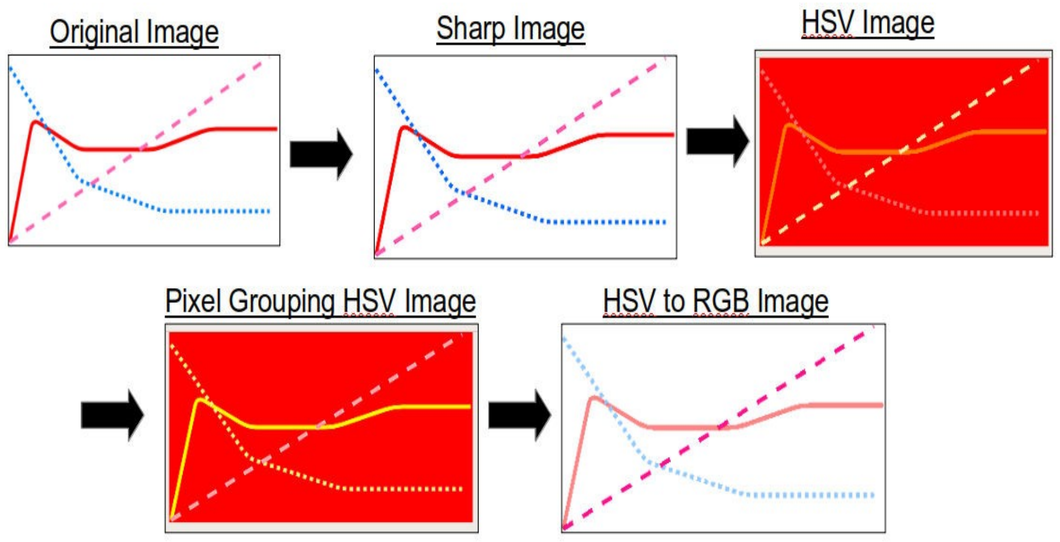


*Figure 35: The results of transforming the colorspace of a graphic to HSV, applying the HSV homogeneity filter and then transforming back to RGB.*

A colored graphics image appears in the human eye as if it contains a specific set of distinctive colors (the background color, the color of the lines, and the color of the grid). However, in reality the different colors that are contained in that same image can be over 100. That is because there are actually different shades of the same color for neighboring groups of pixels. We bypass this problem by transforming the colorspace of the image from Red-Green-Blue (RGB) to Hue-Saturation-Value (HSV). Below we present the rules we apply for color grouping the pixels based on their values of H and S. More detailed grouping can be achieved if we include rules for the V variable as well. We refer to that HSV based color grouping as the HSV homogeneity filter.

The rules for the HSV homogeneity filter are:

- *If $0^O$ <= H < $60^O$, then the color group includes black, white, silver, gray, red and maroon*
- *If $60^O$<= H < $120^O$, then the color group includes yellow and olive*
- *If $120^O$ <= H < $180^O$, then the color group includes lime and green*
- *If $180^O$ <= H < $240^O$, then the color group includes cyan and teal*
- *If $240^O$ <= H < $300^O$, then the color group includes blue and cyan*
- *If $300^O$ <= H < $360^O$, then the color group includes magenta and purple*
- *If S >= 28%, then we have pure colors*
- *If S < 28%, then we have white, black or gray shades*
- *If 20% < V < 70%, then we have gray*
- *If V > 70%, then we have black*

We proceed by changing the value of the pixels to that of the dominant color from their respective groups. Then we transform the image back to RGB colorspace, and commence the curve segment recognition. An example of the results from the HSV transformation and the following pixel color grouping is showcased in figure 35. It is evident by the defined HSV rules, that colors like black, white and gray are going to be categorized into the same group. This creates a problem when it comes to recognizing a potential grid, since after the application of the HSV homogeneity filter, the grid has the same color as the background (the dominant color of its group).

Therefore, we keep a second instance of the input graphics image, to which we apply the contrast enhancement filter. This step turns the dark colors of the image even darker, making them more distinguishable from the background color. Using the Value variable from HSV, we detect the gray and black pixels in the image that constitute the grid of the graphic. We, then, transform the image back to RGB maintaining only the grid of the image.

Based on that grid result, we extract all vertical and horizontal line that constitute the axes lines. If the center of the horizontal line lies on the right side of the vertical center with respect to the x-axis, then we have a type 1 axis. If the center of the vertical axis has the same y position as the center of the horizontal line, plus or minus an error, then we have type 2. Finally, if the centers of vertical and horizontal lines intersect, then we have a type 3 axis. It is important to recognize the type axis, in order to know which graphics subimage to extract.

Afterwards, we use the previous outcome image that contains only the axis information, in order to recognize the axis values and variables. Using the PyTessaract OCR [85], we are able to detect the bounding boxes of the text words around the axes. We also store the boxes centers for future use during the association procedure. We follow a left to right downwards direction, so that the first element that we recognize is the name of the y axis. If the second element corresponds to an alphanumeric value, the this is its unit of measurement. Otherwise, if it is a real value, the it is a value of the axis. Similarly, we recognize if the last 2 elements are either the name of the x-axis and it unit of measurement of just the name of the x-axis. There rest of the elements are recognized as values. Based on the positions of their centers with respect to the graphics image, we are able to distinguish y-axis elements from x-axis elements.

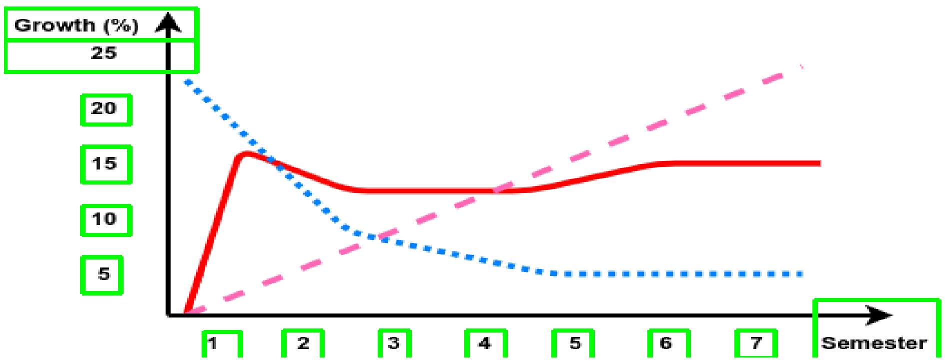


*Figure 36: PyTesseract Axis Values Recognition Results.*

#### 4.4.1.2 Recognition of Curve Straight Line Segments

A curve in a given graphics image is described by the straight line segments that it consists of. Therefore, we must recognize those individual line segments before we can understand that curve. The resulting sharpened image that has been parsed through the HSV homogeneity filter contains lines of single color. This means that the pixels that form a specific curve also have the same color. So the main trail of each graphics curve is now traceable, while we also maintain their original color to help us distinguish them from one another. The next step is to detect the unevenness in each curve's trail. This helps us recognize the different straight line segments that constitute that curve.

We adopt the concept of detecting unevenness in straight line segments into our methodology from Bourbakis et. al. [82]. The authors propose a solution to the issue of line recognition with random anomalies occurring in the line. They refer to these anomalies as unevenness. More specifically, they define unevenness U for line segment SL as the number i of consecutive pixels with , that follow a different direction $d_i$, from the main direction $d_{main}$ of the line segment SL. They develop and describe two interconnected equations in order to tackle this issue. The first equation (3) describes the window that covers the line segment, which contains the unevenness. $W_U$ represents the area of the window, $H_U$ represents the maximum accepted unevenness and $L_U$ represents the length of the line segment containing the unevenness. The second equation (4) defines the unevenness criteria as the division between the height of the window covering the line segment, which contains the unevenness ($W_U$), and that segment's length. We the slope formula below:

$\lambda = (y_1 - y_0) / (x_1 - x_0)$ (5)

The first formula describes a window that covers the line segment, where the unevenness appears. In that formula, $W_U$ represents the area of the window, $H_U$ represents the maximum accepted unevenness and $L_U$ represents the length of the line segment containing the unevenness. The second formula describes the unevenness criteria as the division between the height of the window covering line segment, which contains the unevenness, and that line segment's length. The unevenness criteria is typically a very small number. The smaller the value of the unevenness criteria, the smaller the acceptance limit for the unevenness.

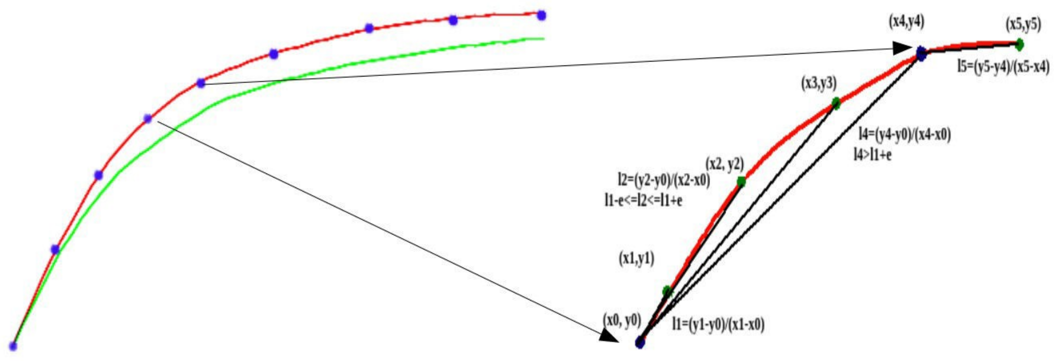


*Figure 37: Example of how the proposed methodology recognizes changes in direction using the slope equation and the unevenness criteria variable.*

The straight line segment recognition is achieved by calculating the slope that is formed between each pixel in the curve’s body. The equation for slope λ between 2 points ($x_0$, $y_0$) and ($x_1$, $y_1$) is illustrated in (5). If the result of the slope equation surpasses a defined limit, then we have detected a new change in the direction of the pixels. That limit is the value of the slope in the previous pixel, for which we detected the change in direction, plus or minus the value of the aforementioned unevenness criteria. Through a trial and error process we deduce that the optimal unevenness criteria threshold depends on the resolution of each graphics image. So we adjust that threshold accordingly by setting as $H_U$ the height of the subimage and $L_U$ the length of the graphics subimage. An example of the aforementioned procedure is illustrated in figure 37. The position of the pixel where the direction change was detected is stored in memory for further use, and the point is marked. Now, the new slope limit is updated with the value of the slope that was calculated for that point. This procedure continues until no more pixels of the same color are found. Then the program repeats itself for the remainder of the recognized colors, except from the background color.

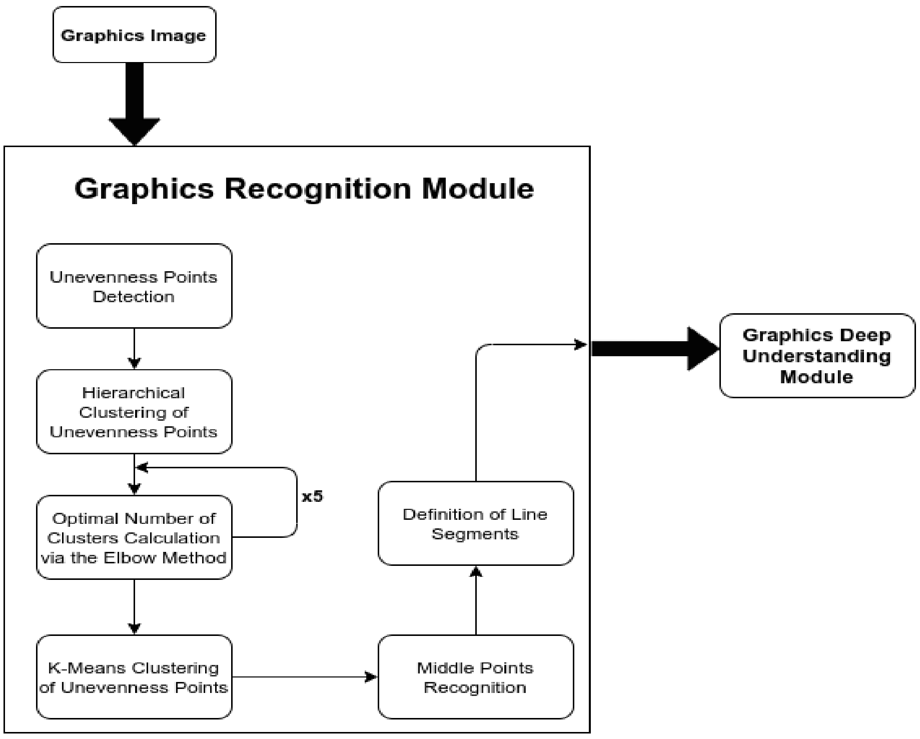

Graphics Image
Graphics Recognition Module
Unevenness Points Detection
Hierarchical Clustering of Unevenness Points
Optimal Number of Clusters Calculation via the Elbow Method
x5
K-Means Clustering of Unevenness Points
Middle Points Recognition
Definition of Line Segments
Graphics Deep Understanding Module

*Figure 38: Proposed Graphics Recognition Methodology.*

The unevenness detection methodology works well for use-cases where the curves in the graphics subimage have a width of one or two pixels. However, in cases where the width is larger, the program might locate multiple unevenness points on the same segment where the unevenness is detected. This is because it parses multiple times through the same segment in order to cover the entire line width. We resolve this issue by performing a clustering technique on the recognized 2-D unevenness points. Then, we keep the middle points of each formed cluster. We use those as the respective start and end points of the straight line segments that constitute the entire graphics curve. It must be noted that by initiating the recognition methodology with the unevenness criteria analysis, we were able to maintain the direction of the lines and disregard any small anomalies, which makes the recognition easier. However, if the process is initiated with the clustering step, then the program takes into account all the pixels of the line, including those that constitute anomalies.

A curve consists of any number of pixels and contains any number of groups of unevenness points. This makes it very difficult to define a specific number of clusters and perform clustering using the standard K-Means methodology. The unpredictability of the required number of clusters leads us to hierarchical clustering instead. Hierarchical clustering is a technique that groups data points into formed clusters, based on a specified distance metric between them. For more accuracy we apply hierarchical clustering to the points of each graphics line individually. Through a trial and error process, we conclude that a distance metric of 3 pixels returns highly detailed clusters that contain an excess of unevenness points than what is needed (larger number of clusters). In contrast, a distance metric of 4 pixels returns somewhat good clusters, that cover the basic idea of the curve (smaller number of clusters). An example of results from both hierarchical clustering procedures are illustrated in figure 39. The graphics subimage of the first graphics type from figure 28 is the input for this example.

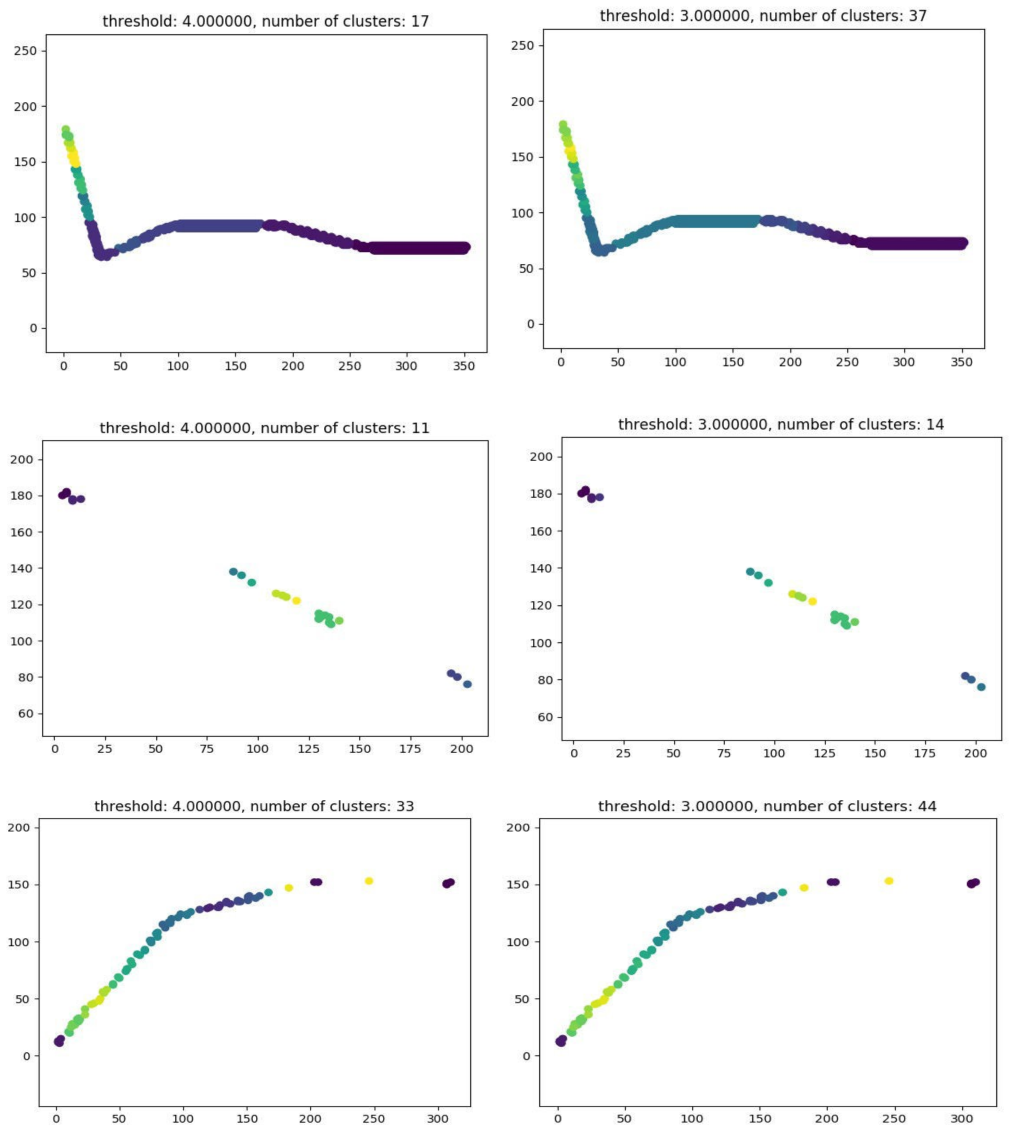
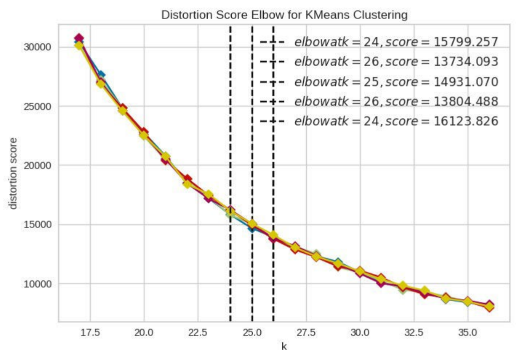
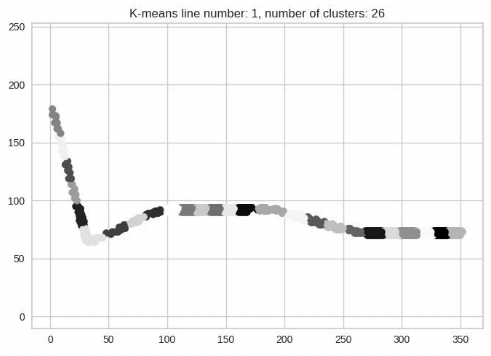


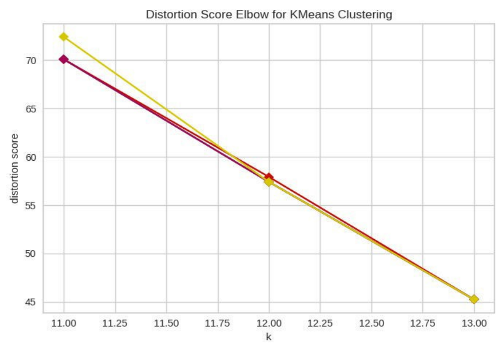
*Figure 39: The results of clustering the detected unevenness points for threshold of 4 pixel distance (left column) and of 3 pixel distance (right column).*

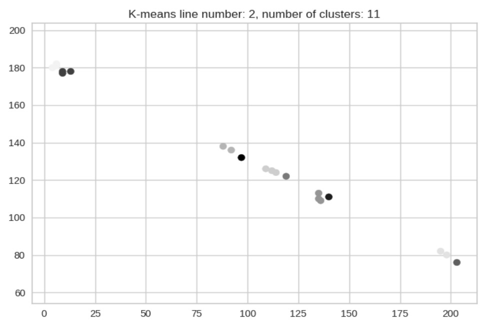
We run two instances of the hierarchical clustering, one for each of the two distance metric values. We later use the resulting cluster numbers as upper and lower limits for a K- Means clustering process. More specifically, the K-Means algorithm performs clustering on the recognized unevenness points of each curve in the graphics image, for every value between its lower and upper cluster number limits. The resulting distortion scores are passed into an elbow calculating function that determines the distortion score for each clustering iteration. The position of the elbow corresponds to the optimal clusters of numbers for that curve. However, we notice that by running the K-Means algorithm multiple times for the same cluster limits, it can return different elbow results. That is due to different initial centroids positions during each run. We deal with this problem by running the K-Means algorithm five times for each graphics curve, and then keep the elbow with the lowest distortion score for each line on average. This empirical solution is based on the fact that a lower distortion score corresponds to higher number of clusters and by extension higher accuracy

Finally, we perform K-Means clustering on the unevenness data points using the optimal cluster number that we have deduced for each curve. The results of this clustering step are presented in figure 40. On the right column of the figure we illustrate the results of the elbow method, when the input image is the type 1 graphics from figure 28. The elbow method indicates 26 and 33 as the optimal number of clusters for the red and the blue curves (curves 1 and 3) respectively. However, no optimal cluster number can be deduced for the magenta line (curve 2). This is due to the low number of unevenness points that by extension produces a low number of clusters. In such cases we keep the lowest number of clusters calculated during the hierarchical clustering phase, and based on that we perform the K-Means clustering. This solution leads to an accurate result, since a lower number of clusters intuitively corresponds to the preservation of the essential curve information. Then, we calculate the position of the middle point of each cluster. A cluster's middle point is different from a cluster's centroid, since the middle point lies exactly in the middle of the cluster, with respect to the other unevenness points in the cluster and the x- axis. We use these middle points as the basis for the line segments that constitute the curves of the graphic.









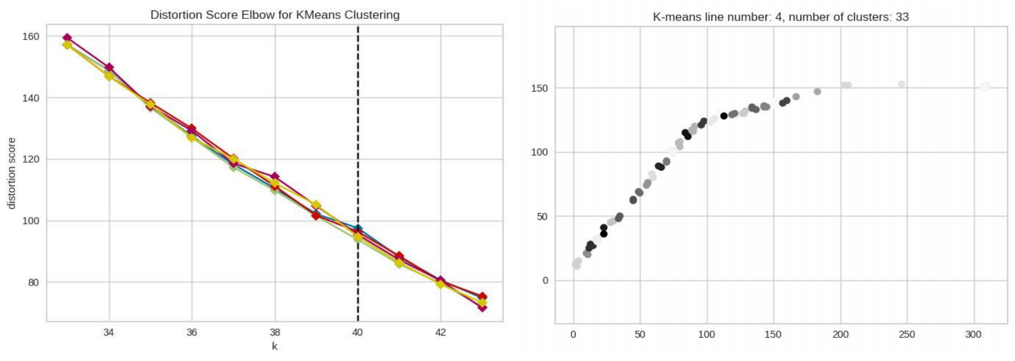


*Figure 40: The results of the elbow methodology (left column) and K-means clustering using the elbow values (right column).*

### 4.4.1.3 Recognition of Noise in Results

Another issue which arises after these processing steps is that of noise pixels. These pixels share the same color with either one of the graphics curves even though their positions are unrelated. They are preserved through the clustering which is an issue, since they form clusters of their own. Eventually, this noise propagates to the generated straight line segments which pass through the positions of the noise pixels. We distinguish a potential noise pixel or a group of noise pixels from curve pixels by creating creating a window around them. The window is delimited by the potential group's most left, most right, most up and most bottom pixels. Then, we extend that window based the recognized size of the line width. A diagram describing this noise correction methodology is presented in figure 40. If there are any pixels of the same color, within the region covered by the line width extension (green area), then we consider the group of pixels as part of the curve. If not, then these pixels are categorized as noise and we discard them.

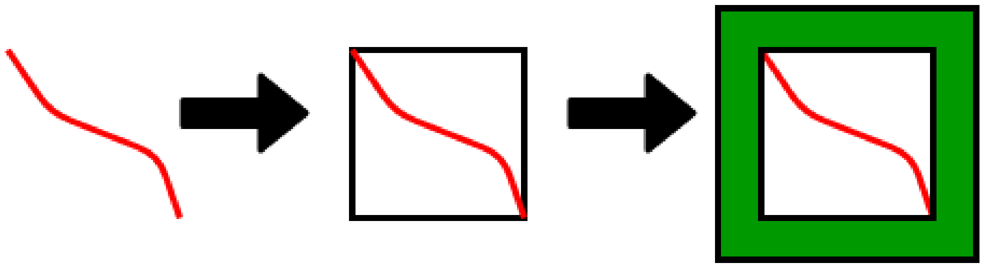


*Figure 41: Example of windowing a line to detect if it is noise.*

We compute the width of each graphics line, by detecting their corresponding starting point first. Then, the program parses through the lines of pixels beneath and upwards from the starting point of each line, in a left to right order. It is searching for the first and final occurrences of pixels with the RGB values of their respective graphics curves. Then, it averages the distance of pixels between the first and final occurrences in order to get each line's width. We search in three rows upwards from the starting point and three rows beneath it, in order to get an average result (figure 42).

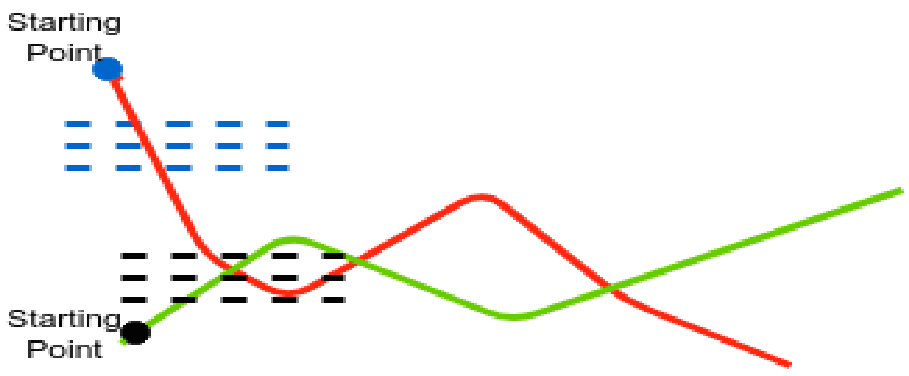


*Figure 42: Example of the line width recognition method.*

### 4.4.2 Recognition of Similarly Colored Partitioned Curves

#### 4.4.2.1 The Feedback Methodology

In order to recognize and understand such curves, we establish that each curve must consist of different shapes. We begin with the utilization of a spiral algorithm, which recognizes the individual pixels that form shapes. We group these recognized shapes based on their distance with each other, as well as their sizes and number of sides. The order or processing steps is essential, since all the curves have the same color, and therefore its difficult for the tool to distinguish one curve from another. Therefore, the individual shapes are recognized and grouped together into curves according to the aforementioned heuristics. When all recognized shapes are categorized to their respective curves, the program assigns them a different color, based on the curve that they belong to. Basically, we convert the type 2 graphics image into a type 1 for which the previously described graphics recognition and understanding methodology is applicable.

A separate feedback module is deployed. It checks the validity of the initial pixel coloring results. Figure 43 illustrates the methodology including the feedback module. More specifically, the feedback module analyzes the colored image and searches for disputable regions of pixels. The tool's decisions are based on the detection of backtracking, while it traces over the direction of each curve's outline. The detected disputable regions are categorized as gray areas by the module. Then, the disputer curves which have potential claim over these regions are identified. Finally, new versions of the colored images are generated, with the disputed regions being assigned the color of their disputer curve. This methodology generates different SPN graphs for each resulting image of the feedback module.

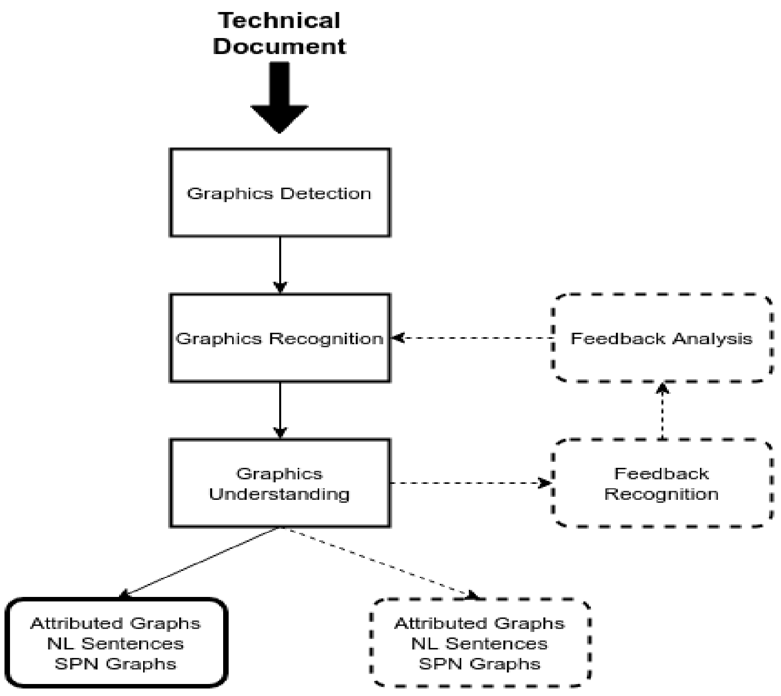


*Figure 43: General methodology including the feedback module.*

#### 4.4.2.2 Initial Recognition Results

The methodology begins with the recognition of the various shapes that constitute the curves. We achieve through the use of a spiral algorithm in the graphics image. It should be noted that we tested multiple established methodologies for their extraction such as Hough Line transform, edge detection algorithms, or the OpenCV blob detector. However, none of the aforementioned techniques provide a delicate solution for the recognition  of detailed information. The proposed methodology begins with the application of the HSV homogeneity filter in order to assign the most dominant pixel color to the curves for easier processing. Then, the spiral algorithm is initiated, starting from the relative center of the image and following a counter clockwise direction around it. The algorithm accepts only the pixels that have the same RGB value with the recognized curve color, keeps track of their positions and stores them into lists. These lists represent recognized curve subsegments. When the program comes across a new pixel, it evaluates its position with respect to already recognized subsegments. Then, the new pixel is assigned to the group that with a distance of 1 pixel or less. In case no such group exists, then a new list is initiated in order to store that pixel, forming a new subsegment.

Due to the order of parsing the pixels, some subsegments that should be united are falsely recognized as different groups. So the initial merging rule of 1 pixel distance is reapplied to the all groups iteratively, until no more groups can be combined. We demonstrate an example of this entire procedure with its results in figure 44. The upper left image is the original input image, and the upper right is the result of the HSV homogeneous representation process. The bottom left image contains an example of the image's pixels being parsed by the spiral algorithm starting from the center of the image. Finally, the bottom left image showcases the result of all the recognized curve subsegments. Each group has been assigned a different color or shade (e.g. different colors of green, or magenta) in order to be recognized as its own entity.

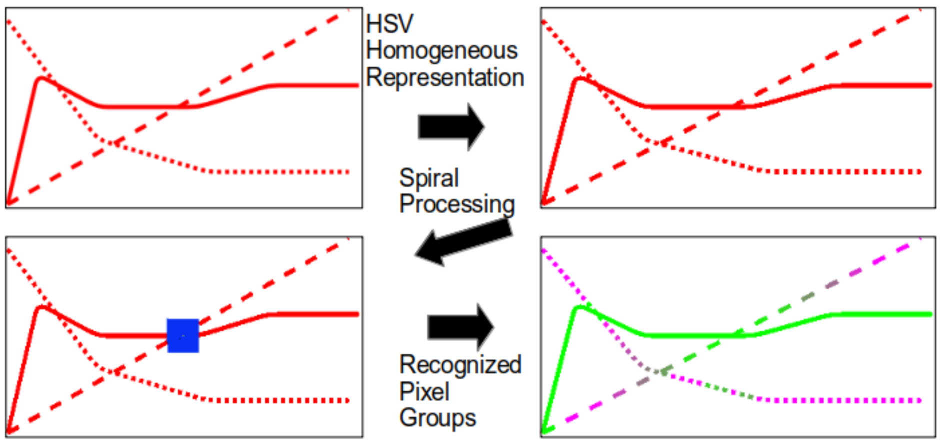


*Figure 44: Results of pixel shapes recognition from the spiral algorithm.*

Then, the methodology proceeds with the calculation of the sizes, number of sides and the centroid of each recognized subsegment. The calculation of these metrics is vital in order to accurately categorize similar shapes into the correct curves. However, due to the application of the HSV homogeneity filter, many outlier pixels appear where shades used to be in the original image subsegments. This makes the detection of number of sides for each shape a challenging task. The proposed solution consists of detecting the outline of each subsegment and then calculating the number of sides. During that calculation, we disregard any outlier pixels that don't follow the direction of the current side. An example of the aforementioned approach is presented in figure 45.

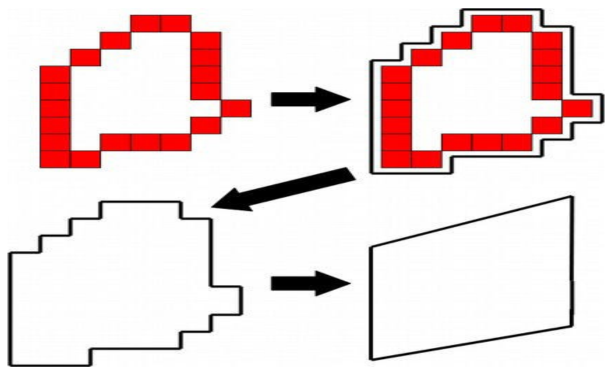


*Figure 45: Proposed solution for number of sides calculation, in case of degraded shapes.*

The next step of the procedure is to combine the formed pixel shapes into curves. More specifically, the first layer of the combination algorithm classifies the recognized shapes into overarching curves based on similar pixel masses and number of sides. This procedure uses a heuristic error value, which has been deduced through trial and error. In figure 46 we showcase the result from the first layer of classification for the recognized subsegments into curves. The unclassified shapes are highlighted with a green outline, while the rest of the curves are highlighted using different colors. A second layer of classification assigns any unclassified shapes into the already formed curves or recognizes new curves. This assignment is based on the position of each subsegment's centroids along with their number of sides. Figure 47 illustrates the results of the second classification layer. We notice in these results that some regions of pixels have been classified incorrectly. We solve this issue with the deployment of the proposed feedback module. Finally, the tool color the categorized pixel segments using different colors for each detected curve. By differentiating the color of the curves, we reduce the problem of same color curves recognition and understanding to that of different color curves recognition and understanding. Figure 47 illustrates an example of the final coloring results. The curves have been assigned the colors of red, green and blue respectively, as a tribute to the RGB colorspace. However, it must noted that there are more colors available for the software to choose from, in case the are more than three curves in the input image.

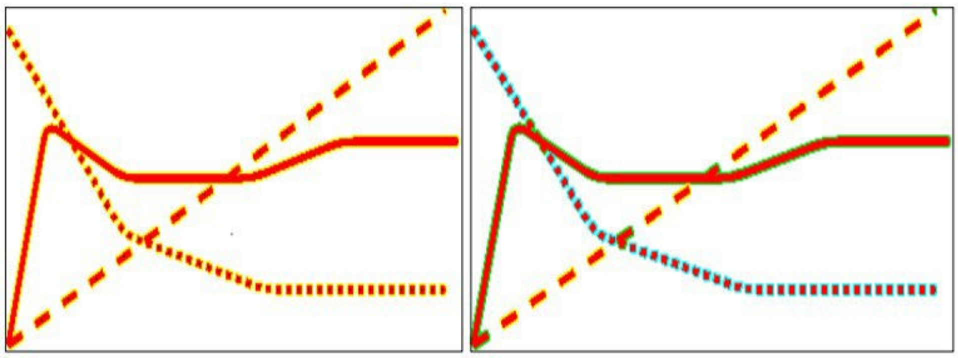


*Figure 46: Left image shows the recognized outline of each shape. Right image shows the results of the first classification layer.*

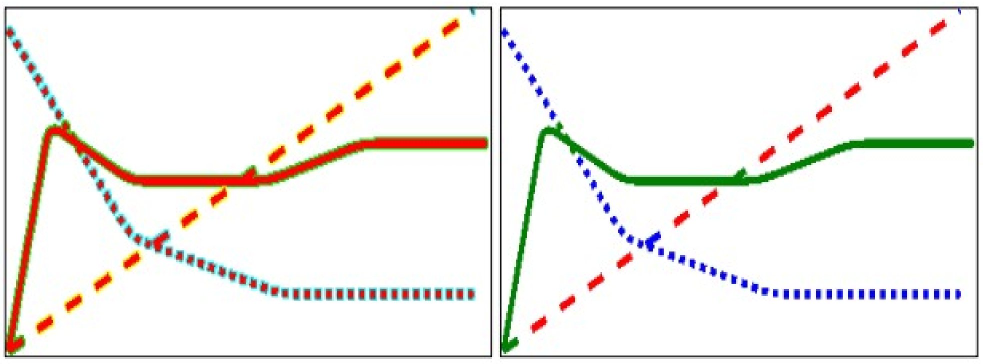


*Figure 47: Left image shows the final classification result. Right image shows the coloring result for the detected curves.*

The graphics recognition module accepts the colored images as input and follows all the processing steps that are described in the previous sections. More specifically, we proceed with the search for unevenness points in each curve. Then, we use hierarchical clustering based on the distance metric. In figure 48 we demonstrate the results of hierarchical clustering for the cases of 4 and 3 pixel distance. We use the elbow cluster validity measurement in order to calculate the optimal number of clusters for each curve. Finally, the K-Means algorithm is applied for the deduced optimal number of clusters. We use the K-Means results in order to retrieve the middle points that construct the line segments for each curve. We notice that for curve 3 no elbow can be detected. This is due to the low number of unevenness points that by extension produces a low number of clusters. In such cases we keep the highest number of clusters calculated during the hierarchical clustering phase and apply the K-Means based on that. This solution leads to an accurate result, since a higher number of clusters intuitively corresponds to a higher number of details for the curve. We present the corresponding results of the elbow method for each curve along with the outcome of the associated K-Means clustering results in figure 49.

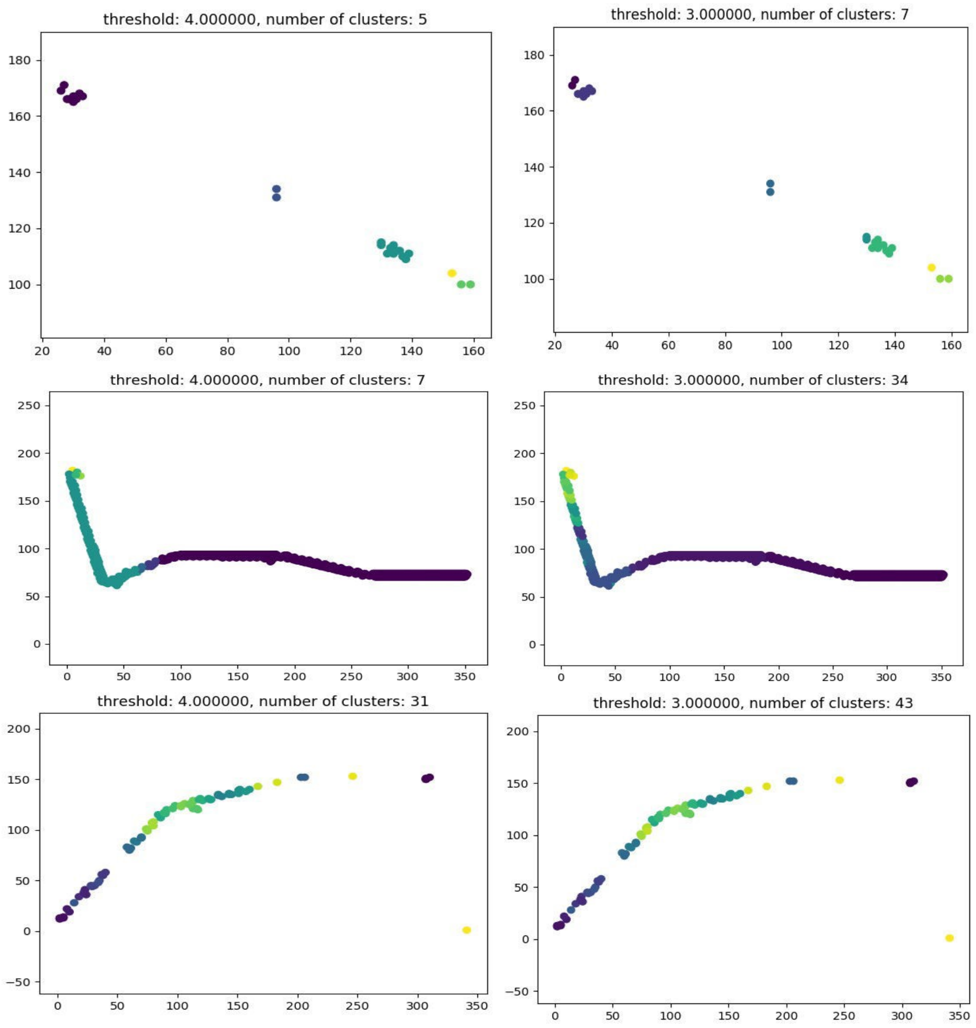


*Figure 48: Hierarchical clustering results for threshold of 4 pixel distance (left column) and of 3 pixel distance (right column).*

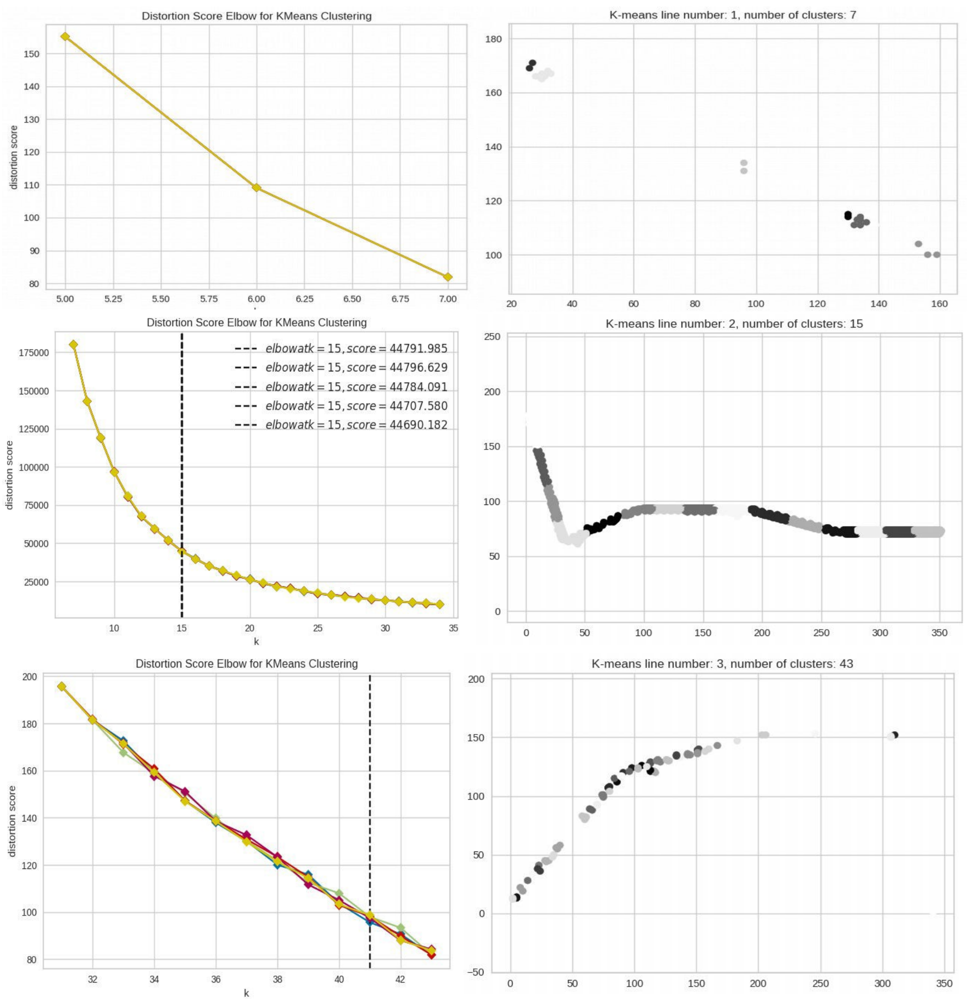


*Figure 49: Elbow results (left column) and K-Means clustering results (right column).*

### 4.4.2.3 Feedback Recognition Results

In the previous coloring result (figure 47), there are some noticeable regions of pixels that are assigned to the wrong curve. This is due to fact that all the curves in the original image have the same color. So, after the HSV homogeneous representation filter is applied, certain pixels of different curves that intersect with each other, end up merged. As a solution to this problem, we have developed a separate feedback module, that is presented in figure 43. This module searches for any disputable regions of pixels within the image. We call this process feedback recognition. If it finds any, then it triggers a separate process called feedback analysis. During feedback analysis, the tool investigates the disputable regions and finds the curves, to which they might potentially belong. It must be noted, that the feedback module is integrated for both partitioned curve graphics as well as continuous curve graphics.

The feedback recognition procedure starts with the tracing of the outline for each curve. If the curve consists of individual shapes, then it recognizes their respective outlines. After the outlines have been detected, it proceeds to parse through the recognized pixels of the curve body or partition outlines. The parsing begins always from the most left pixel of the outline and continues in a successive manner, always checking for any changes in direction. Multiple successive changes of direction in a sort amount of parsed pixel indicate backtracking. Backtracking means that the general direction of the curve's outline changes to a different state, and then it changes back to its original direction soon after. Incidents of backtracking imply detection of abnormalities, which by extend indicate disputable regions. On the other side, if a change in direction has happened, without any other change for more than ten pixels, then we consider this as a general change of direction in the curve. In that case there are no disputable regions. The threshold number of ten pixel has been determined through a trial and error process. Figure 50 showcases the detected outline of one example curve. Figure 51 illustrates the backtracking detection in that curve, as well as the results of recognizing disputable regions.

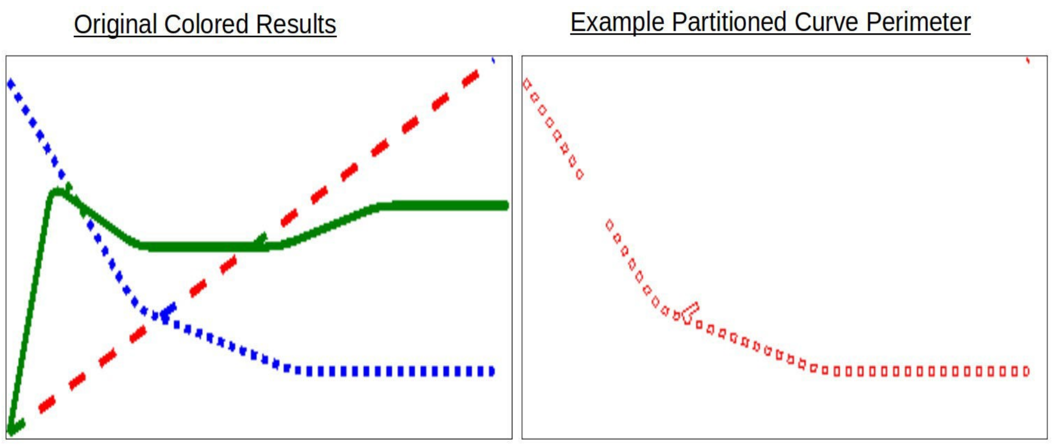


*Figure 50: Left image is the input graphics image (previous coloring result). Right image shows the outline of one of the curves.*

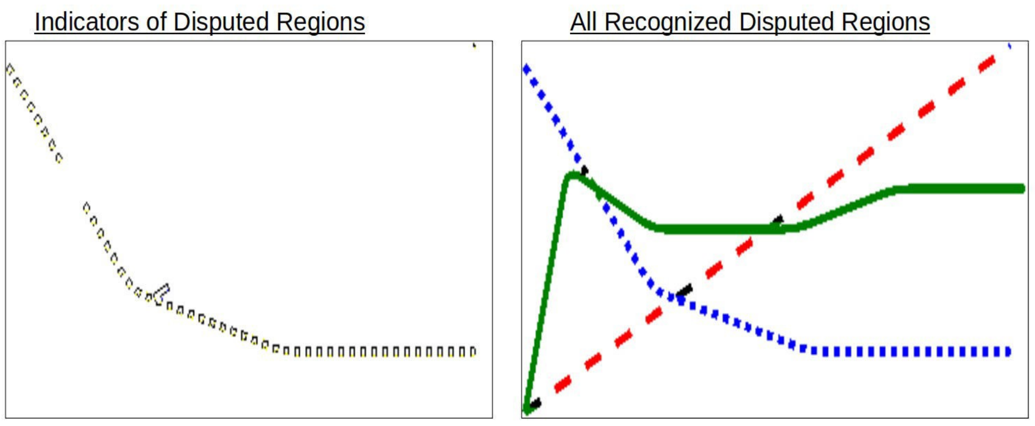


*Figure 51: Left image shows the detection of backtracking pixels. Right image shows the results of recognizing disputable regions.*

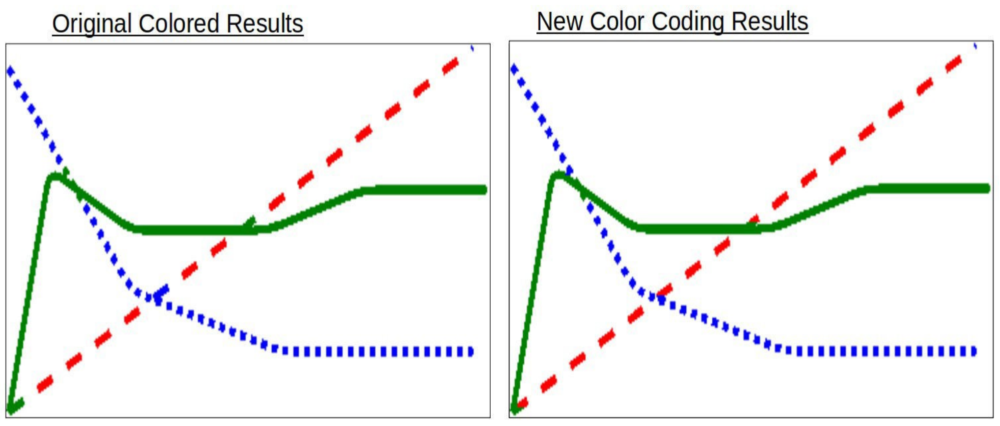
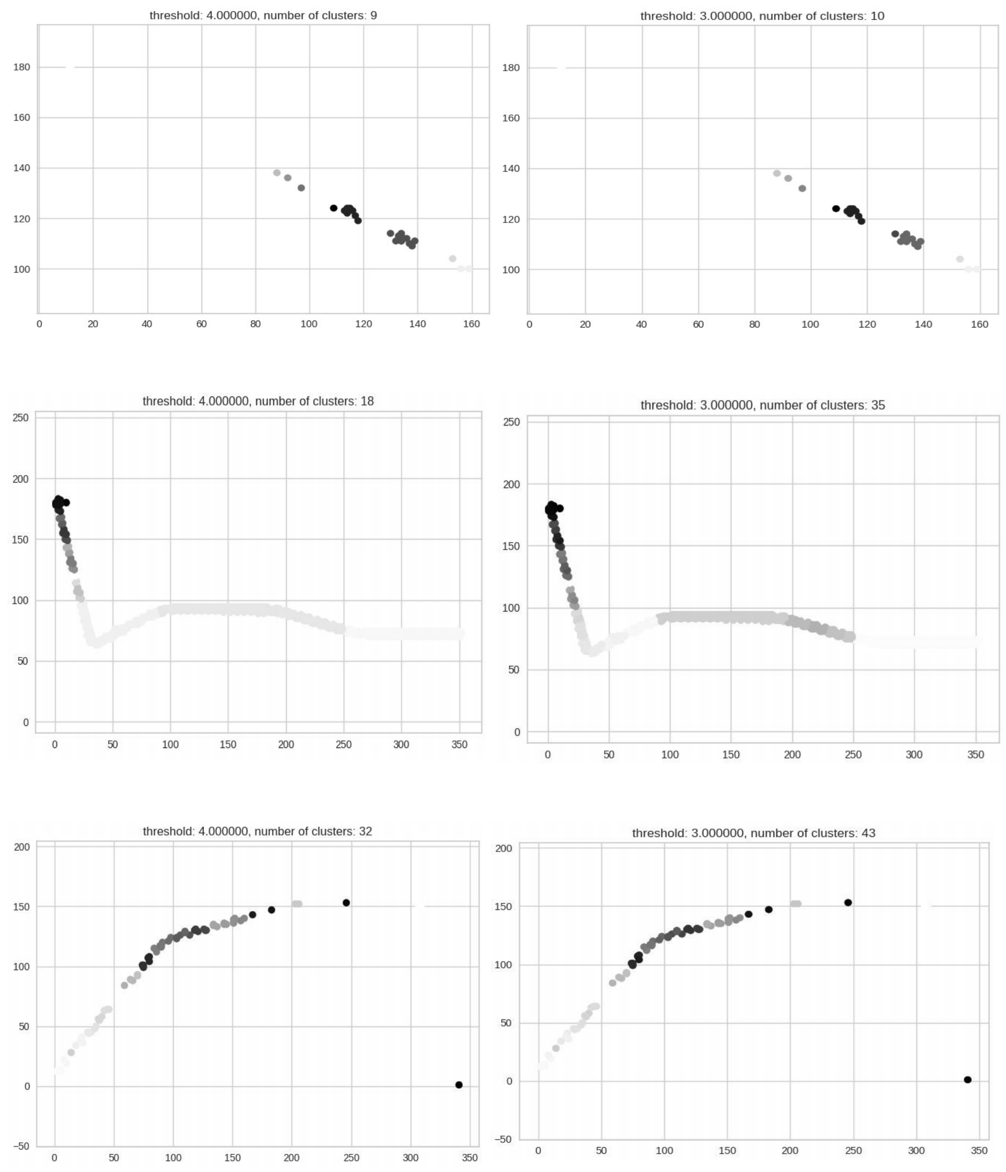


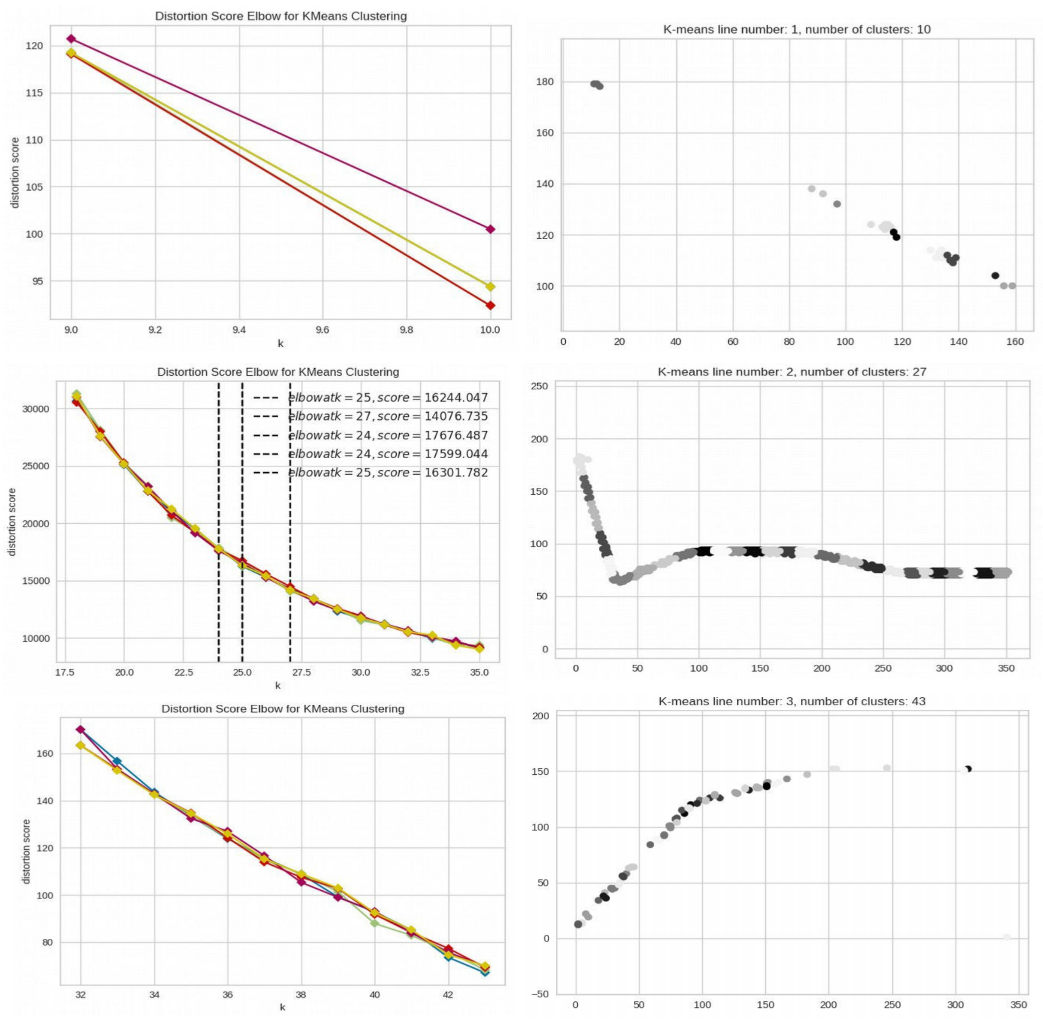
*Figure 52: Left image shows the input graphics image (previous coloring result). Right image shows the result of the feedback module.*

After the disputable regions have been recognized, the resulting image is pushed to the feedback analysis procedure. Feedback analysis processes the given graphics image containing the disputable regions, and calculates the probability of each region belonging to some of the other curves. If the body of a curve passes over a disputable region, then there is a possibility that this region belongs to it. If the centroid of a disputed region is placed before or after the initial or final recognized line segments of a curve respectively and they have similar directions (plus or minus a given unevenness error), then there is a possibility that the disputed region belongs to this curve. The directions are defined based on calculating the corresponding slopes. More specifically, in the first case, we calculate and compare the slope formed by the centroid of the disputed region and starting point of the first line segment of the curve, with the slope of the same first segment. We perform the necessary calculations accordingly for the second case. Figure 52 illustrates the final coloring results of the disputed regions. The feedback module uses these probabilities in order to assign new colors to the disputed regions. The new color of each region is based on the curve it belongs to. So far the methodology doesn't offer a definitive answer for the origins of the disputed regions. Instead it produces the aforementioned probabilities. So, it can potentially produce multiple different versions of coloring for the input graphics image, based on all these probabilities. We have developed a separate probabilistic model, based on the Bayesian Theorem [93] that can solve this problem, which we discuss in subsection 4.4.2.4.

The rest of the procedure follows the established processing steps of the the graphics recognition and understanding methodology. More specifically, the tool applies the HSV homogeneous coloring filter to the graphics image, and then detects the unevenness points of each curve. Then it clusters these points in order to find the middle point of each cluster, which describe the curve. Figure 53 demonstrates the results of hierarchical clustering for the cases of 4 and 3 pixel distance. We present the corresponding results of the elbow method for each the clusters of each curve, along with the outcome of the associated K-Means clustering results in figure 54. When no elbow can be detected, we keep the highest number of clusters perform the K-Means clustering.



*<u>Figure 53:</u> Hierarchical clustering results for threshold of 4 pixel distance (left column) and of 3 pixel distance (right column). These results correspond to the output colored image of the feedback module.*



*<u>Figure 54: </u>The results of the elbow methodology (left column) and K-means clustering using the elbow values (right column). These results correspond to the output colored image of the feedback module.*

### 4.4.2.4 Feedback Probabilistic Modeling

In the previous section, we introduce a feedback module, which checks the validity of the initial coloring results for a given image of partitioned curves with the same color. In this section, we attempt to formulate this procedure, using mathematics instead of heuristic rules. We consider this issues as a problem of uncertainty. Similar to other uncertainty problems, such as navigation in a dynamic environment, we must also make decisions about the origins of the disputed pixel regions in the image, when there are not many available information. The proposed solution for such problems of uncertainty relies on the formulas and associations described in the Bayesian Theorem [93]. More specifically, we use this theorem, in order to express the uncertainty of where each disputed region belongs to, through probabilities.

After the combination of the Bayesian equations with the available information, we reach the following formulas:

$$P(bLN / KB) = [P(KB / bLN) \times P(bLN)] / P(KB) \quad (8)$$

$$P(KB) = P(KB / Modality_1) \times P(Modality_1) + \ldots + P(KB / Modality_N) \times P(Modality_N) \quad (9)$$

where

$Modality_1 = Table \wedge Modality_2 = Diagram \wedge Modality_3 = Text \wedge Modality_4 = Formula \wedge Modality_5 = Algorithm$

and etc. So for example:

*$P(KB / Modality_1) \times P(Modality_1) = P(KB / Table) \times P(Table)$*
and so on according to the available modalities.

P(KB/Modality$_i$) is the accuracy of the recognition of the individual modality i, with P(Modality$_i$) being either 0 or 1. Its value depends on whether that modality exists in the input technical document or not. On the contrary, P(bLN) depends on the direction of that curve and the distance of the disputed region to it. More specifically, if that direction of the curve agrees with the disputable region, then we consider it as a potential part of that curve. Then, the probability of that disputed region belonging to that curve depends on its distance from all other curves that it can potentially be part of. The calculation of the final probabilities is not yet feasible. This is because some of them depend on the calculation of the proposed methodologies accuracy for table and graphics images recognition and understanding. We plan on concluding this methodology in future work. Here we make an effort to only define the problem in terms of mathematics and uncertainty.

### 4.4.3 Recognition of Bars

Initially, we detect the axis type illustrated in the input bars chart image. Similarly to the results from the curves graphics, we extract the inner subimage that contains only the bars. Then, we analyze that image independently in order to recognize its structural information and deduce the illustrated behavior of the bars. After the application of the HSV homogeneity filter, we extract the graphics grid which is analyzed in order to detect the axis legends text and their pixel positions. Figure 55 illustrates the extracted grid in the use case of type 3 graphics image from figure 28.

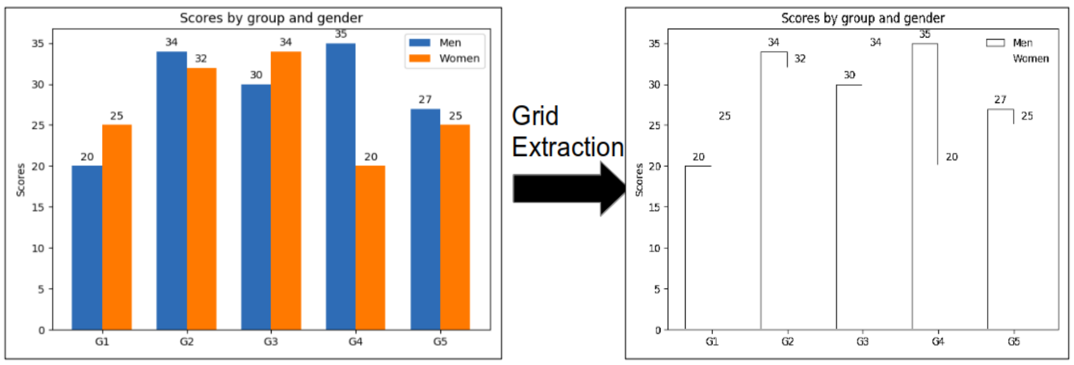


*Figure 55: HSV and grid filter results*

The next step in the proposed bars recognition methodology consists of recognizing the different features of the resulting subimage. These features include the heights and widths of the bars as well as their relative locations in the image. The given subimage can also contain further information about the association of the bars colors to variable names. So we process the given subimage accordingly in order to extract all that information. We begin with the detection of the bar names using the PyTesseract library. More specifically, we locate the bounding boxes of their names and then store the centroid of these boxes. Then, we proceed with the recognition of the associations between colors and variable names. The tool applies the HSV homogeneity filter to the bars image in order to replace all the differently colored pixels with the most dominant color of their group. Then, we binarize and invert the image in order to get all its connected components. The goal is to detect the colors that are associated with previously detected name boxes, so we don't mind that the bars might appear conjoint. Thus, we evaluate their positions with respect to the names bounding boxes centroids. Initially, we find the most common height and most width pixel positions from each name's bounding box. Then, we get the closest lefthanside connected component. Finally, we associate the color of that connected component to each name using their respective centroids. After the association is completed, we erase all that information from the image and proceed with the next steps of the methodology. Figure 56 illustrates the results of the aforementioned processing steps.

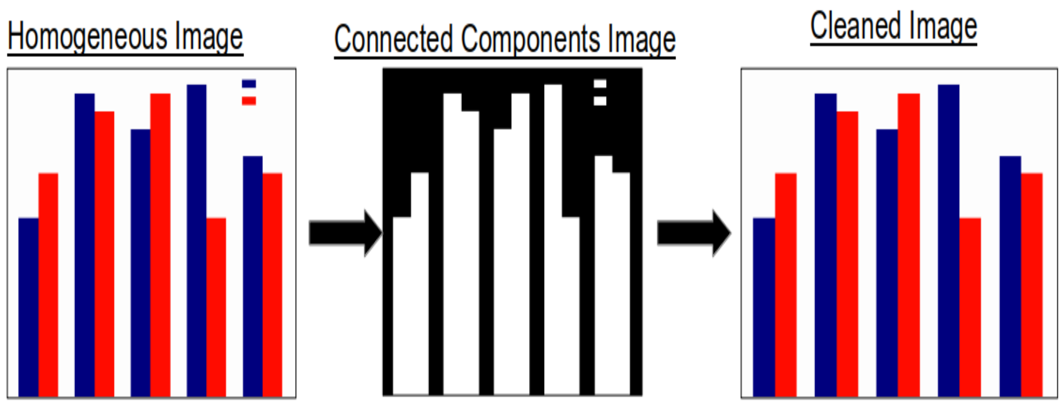


*Figure 56: Detection of color names and association processing steps.*

Then, we proceed with the detection of the highest points from each bar and use them as input for the previously presented curves graphics understanding methodology. Figure 56 illustrates an example of this idea. We begin with the detection of the top corners of each bar in the graphics image. Then, we keep the middle point that is placed between them and use that as the new middle point in order to understand the graphics subimage contents. The conversion of the bar graphics image into a curve graphics image allows for a deeper understanding of the internal associations that are represented in the graphics image.

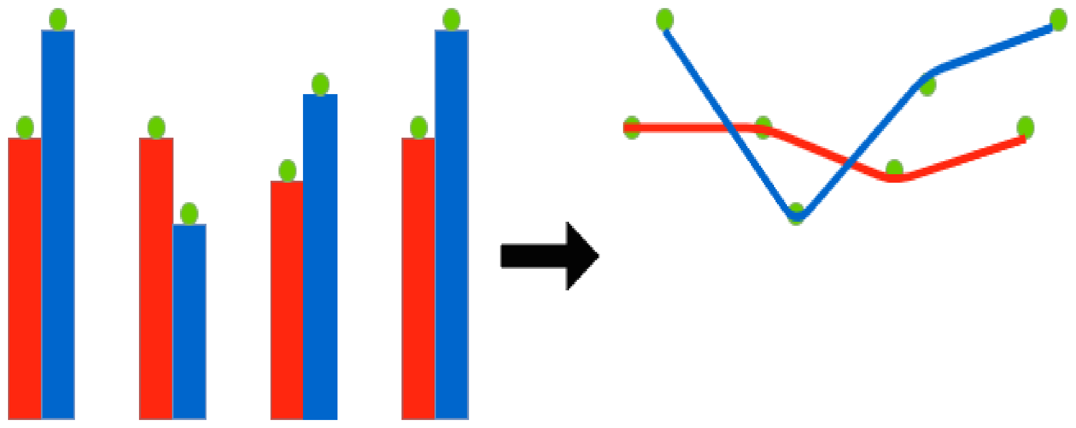


*Figure 57: Detection of bars middle points.*

In figure 57 we demonstrate the results of detecting the corners of each bar in the input subimage. We use the middle point between each two corner pixels of the same color in order to form the resulting straight line segments that form the corresponding curves. Figure 58 illustrates the results of recreated curves superposed over the input bars image. We use the straight line segments in order to extract the different associations between curves. These results also serve as proof that the proposed methodology recognizes accurately input graphics images. The resulting curves are already in their definitive form so there is no need to use any clustering or merging techniques in order to optimize them. Therefore, we use their extracted behavior directly in order to understand the underlying functionality of the initially illustrated bars.

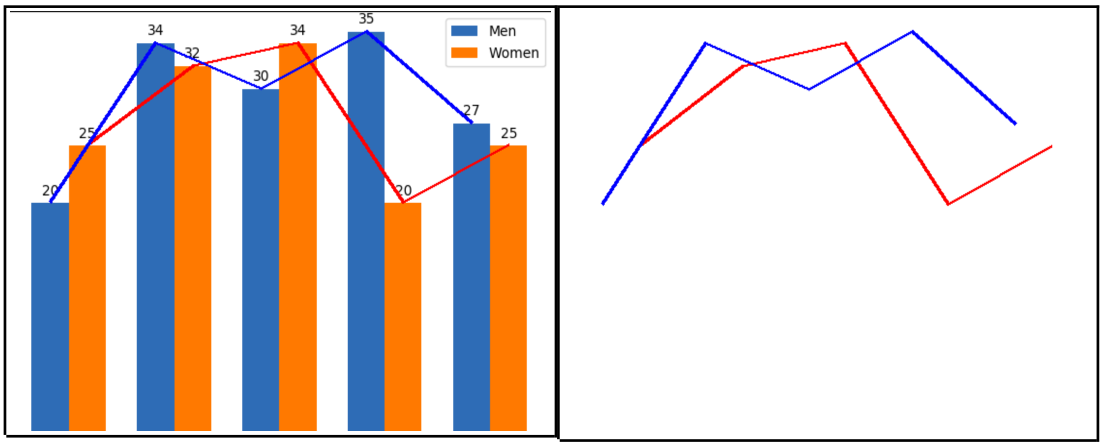


*Figure 58: Curve reconstruction using the recognized line segments (right) and superposed results over the original bar graphics image (left).*

## 4.5 Understanding of Graphics

### 4.5.1 Understanding of Colored Partitioned and Continuous Curves

#### 4.5.1.1 General Graphics Understanding Methodology

We illustrate the proposed graphics understanding methodology in figure 59. To the best of our knowledge, there is no other standard methodology that offers the same level of understanding and analysis for graphics images yet. We expand on previous graphics understanding attempts in an effort to capture a graphics image underlying semantic meaning. In order to accomplish this task, the methodology receives the resulting line segments as input from the recognition module. Then, we use two different merging rules based on the geometrical positions and characteristics of the segments. We use their results in order to generate the corresponding attributed graphs of the structural features and the natural language sentences that describe them. Finally, we convert these sentences into stochastic Petri-net graphs, which describe the functionality of the recognized curves. Although, the results of the second merging rule maintain structural features, they don't include any information about semantic associations between the segments. For this reason we apply a third merging rule. By combining the resulting graphs and natural language sentences of both merging rules, we can deduce an enhanced SPN graph, which maintains both functionality and semantic meaning.

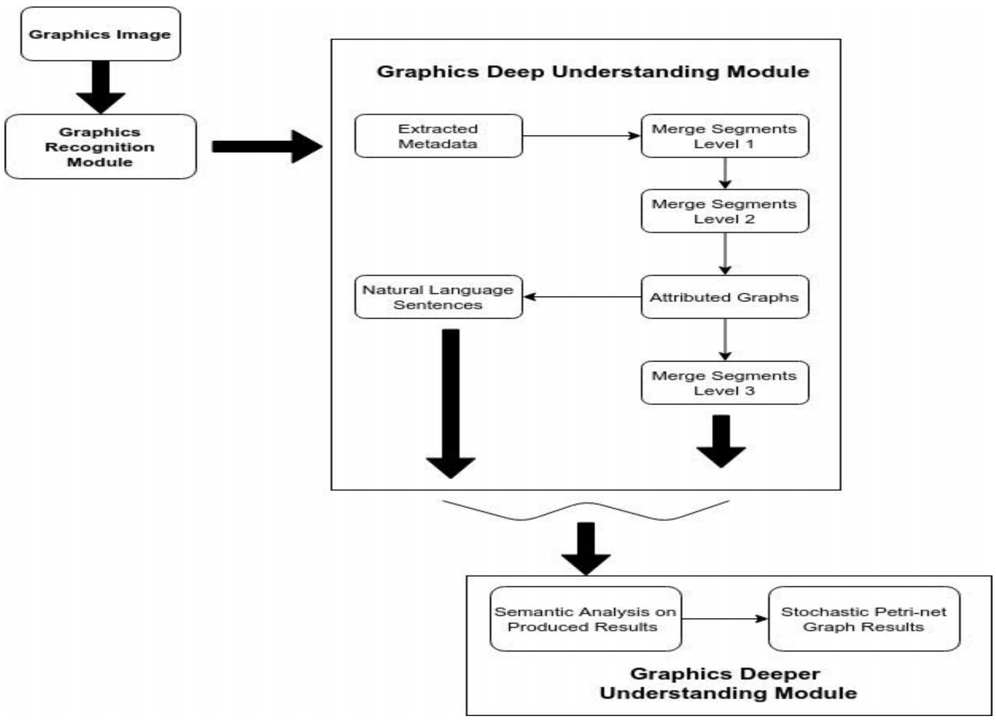

Graphics Image
Graphics Recognition Module
Graphics Deep Understanding Module
Extracted Metadata
Merge Segments Level 1
Merge Segments Level 2
Natural Language Sentences
Attributed Graphs
Merge Segments Level 3
Semantic Analysis on Produced Results
Stochastic Petri-net Graph Results
Graphics Deeper Understanding Module

*Figure 59: Proposed Graphics Understanding Methodology.*

Then, we continue processing the retrieved curves image, with the aforementioned graphics recognition and understanding methodology. We also include an additional module that recognizes the growth rates of each curve that is illustrated in the graphics subimage. This is accomplished by recognizing the individual growth rates of each line segment that constitutes the curves. Additionally, we transfer the recognized middle points into the original image containing the axis information. We achieve that by keeping a matrix containing the surrounding pixels of middle point. This transforms the middle points to a location invariant data points. Finally, we associate the resulting information that we extract from the graphics subimage with the original graphics image, which contains the axis. This is the final processing step in order to gain more accurate and detailed understanding of its underlying information.

#### 4.5.1.2 Merging of Straight Line Segments

We can merge some of these segments based on their geometrical attributes in correspondence to each other. However, if we consider the recognized line segments as straight lines, then many issues arise when calculating the slopes of vertical and horizontal lines respectively. That is because the vertical line has no slope, and the horizontal line has a slope value of always zero. Such special cases make the program unable to calculate the angles between the connected line segments, since the mathematical equations don’t hold true. We avoid this issue by treating the recognized line segments as vectors instead.

We showcase three distinct levels of merging for the recognized line segments. The first level of merging is an effort to erase unnecessary direction changes, that are formed by the recognized line segments. Then, we replace them with a single larger segment, which represents  both of them. We deal with such issues by defining a worst case lemma. When the endpoint of the first segment has a distance of 1 pixel from the starting point of the second segment, with respect to the y-axis, and a distance of less or equal to the detected line width value minus 1 pixel, with respect to the x-axis, then we consider it as the worst case. We present an example of the first level merging rule in figure 60. The results of applying the first level merging rule in the recognized line segments are illustrated in figure 62.

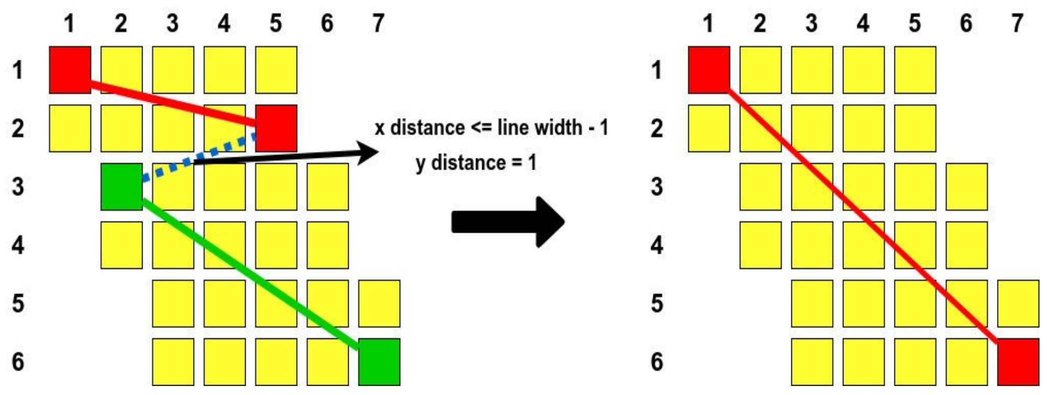


*Figure 60: Example of the first level merging rule.*

The first merging rule produces solid results, however, many overlapping segments might be preserved. Therefore, we define a second level merging rule, that merges line segments based on their slope values. More specifically, it combines line segments whose slope values' difference is less or equal to a defined threshold. As we have already mentioned, we consider each line segment as a vector. We also consider each vector's direction to be the opposite of its counterpart connected segments. So, we calculate the slopes of these line segments as if they were vectors of opposite direction. The value of the threshold is the outcome of the aforementioned worst case lemma. More specifically, the threshold is equal to the the worst case y-axis distance, which is 1, divided by the worst case x-axis distance. The worst case x-axis distance is equal to the detected line width minus 1 pixel. We demonstrate an example of this merging rule and its results when applied to the input graphics image in figures 61 and 62 respectively.

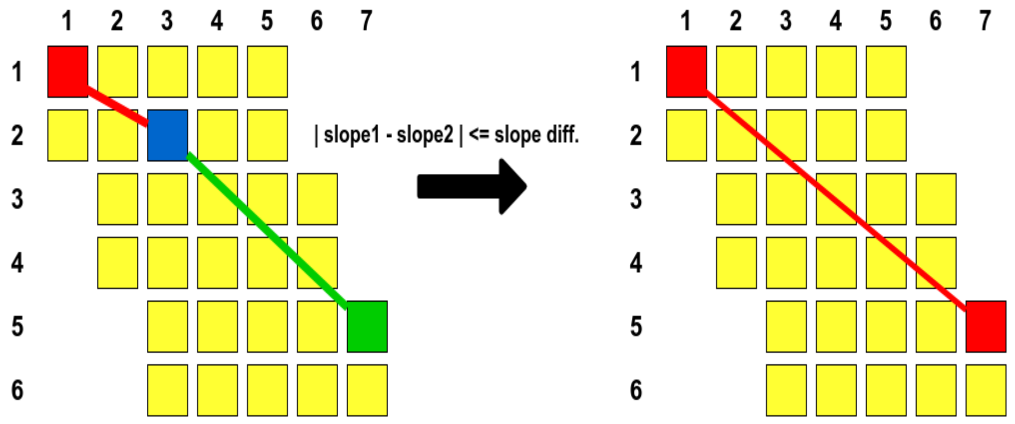


*Figure 61: Example of the second level merging rule.*

The second merging rule returns results with higher accuracy. However, there can still be cases of overlapping segments that should be merged. Therefore, we define a third and final merging rule. The third rule uses the produced attributed graph results that correspond to the second merging rule. Then, it merges their respective line segments based on their common features. More specifically, if two lines are connected and both are parallel to the same third line segment, then they are merged into a single line. The parallelism is concluded based on their attributed graphs. The results of the third merging rule are illustrated in figure 62. It is evident by the results, that while the rule discards almost all unnecessary line segments, it can also lead to potential loss of information. However, this information is associated more with the structural features of the lines, and less with the general sense of directions and proportions between different curves. So we maintain both the results of the second level merging and third level merging rules, since each result offers a different aspect and highlights different attributes of the same graphics curves.

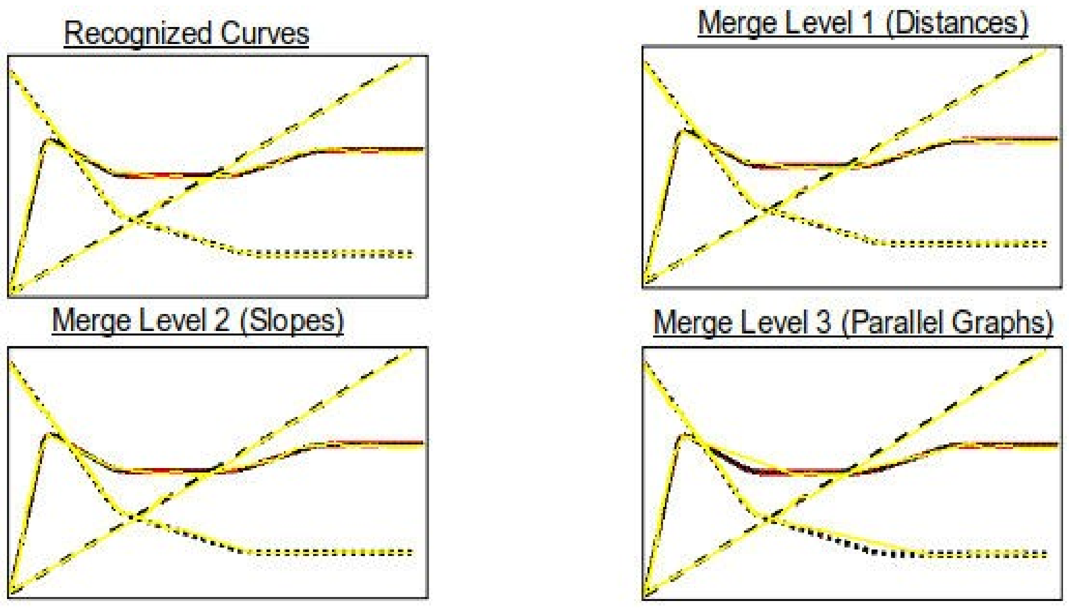


*Figure 62: The recreated curves (yellow) superposed over the original graphics image (black). We include the resulting curves from each merging rule.*

We include color name recognition as part of the proposed understanding methodology of the information that is illustrated in a graphics subimage. The application of the HSV homogeneity filter maintains the original color group (e.g. red, yellow, blue). That is achieved by selecting the most dominant RGB values in each group. These RGB values represents different shades of the most commonly used colors (such as red, blue, green, yellow, magenta, etc.), thus making difficult to recognize their actual names. For example dark red and light red colors are different shades of the same color red. We resolve this issue by maintaining a knowledge base of the RGB values of the most commonly used colors. We then calculate the euclidean distance between the recognized RGB value of the curves and the those values in the knowledge base.

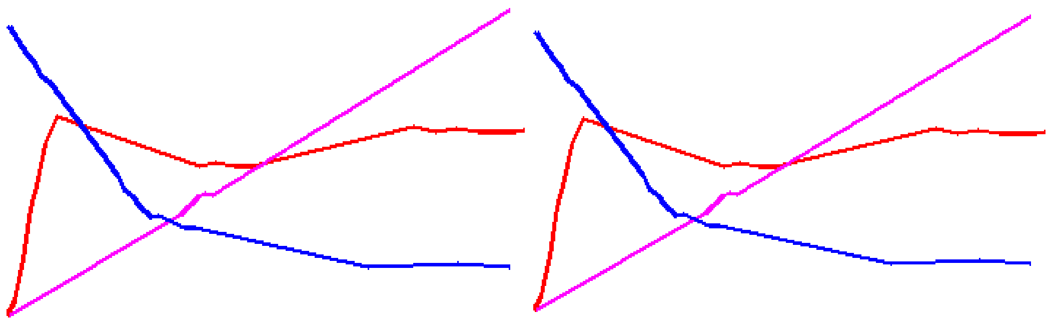


*Figure 63: Colored reconstructed curve after the 2nd merging rule (right) and 3rd merging rule (left) respectively.*

Finally we keep the knowledge base RGB value which corresponds to the minimum distance for each curve and then retrieve their actual names from the knowledge base. Figure 63 illustrates the results of reconstructing the original graphics line, using their corresponding recognized line segments and their respective recognized colors. These results correspond to the type 1 graphics image from figure 28. The left reconstructed graphics image is the result of the second merging rule, and the right image is the result of the third merging rule. We recreate the studied graphics images based on the merging rules outcomes, in order to evaluate the recognition and understanding results of the proposed methodology.

It is evident from these results, that the second rule favors the preservation of structural characteristics. In contrast, the third merging rule highlights the direction and growth rates of the line segments. However, both the recognition and understanding modules are highly accurate for all merging rules.

### 4.5.1.3 Intersection of Straight Line Segments

We consider two different cases of intersections. The first one is the intersection between line segments that form the curves. The second one is the overlapping between line segments of the curves. For the first case, we detect intersections between line segments of different curves by using the equations (6), (7), (8), and (9). Each of the two line segments consists one start point $(x_i, y_i)$ and one endpoint $(x_{i+1}, y_{i+1})$. We can express the position of a random point between them, as a relation of the start point $(x_1, y_1)$ plus the difference $(x_2 - x_1, y_2 - y_1)$ between them, multiplied by a variable T. Similarly we can express a random point for the second curve this time using variable U. Now, we are looking for a common point between these segments by solving the system of equations that is produced. That system of equations is solved by our program for each combination of line segments from different curves. If the results for the variables T and U satisfy the conditions $0 < T < 1$ and $0 < U < 1$, then we have a point of intersection between these segments. Otherwise, we don't. Figure 40 demonstrates the results of detecting intersections using the positions of the recognized line segments as input for the equations. The 2-dimensional intersection points are colored as blue pixels and we contain them inside a blue circle in order to make them distinguishable.

*$(x, y) = (x_1, y_1) + T(x_2 - x_1, y_2 - y_1)$ (6), $(x, y) = (x_3, y_3) + U(x_4 - x_3, y_4 - y_3)$ (7), $T = [x_3 + U(x_4 - x_3) - x_1] / (x_2 - x_1)$ (8), and $U= [(y_4 - y_3)(x_2 - x_1) - (x_4 - x_3)(y_2 - y_1)] / [y_1(x_2 - x_1) - y_3(x_2 - x_1) + (x_3 - x_1)(y_2 - y_1)]$ (9)*

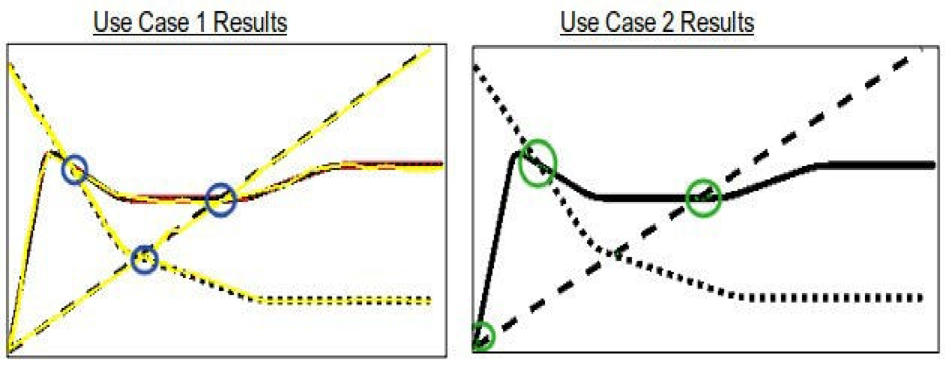


*Figure 64: Intersection points detection (left) and overlapping detection (right).*

It is evident by the results in figure 64, that these equations fail to recognize the intersection at the start of the red and magenta colored lines. That is because this intersection is actually an overlapping. In order to detect overlapping pixels between different lines we employ a different approach. We first define a kernel of 3x3 pixels and create all possible diagonal color combinations, using strictly the recognized colors of the graphics lines. However, we discard the masks that are combinations of only a single color. We then parse through each line of pixels in the image using these masks, and store the positions of the pixels that satisfy them. These pixels are parts of intersections between lines. We can filter out the pixels that correspond to already recognized intersections between line segments by using the previously detected intersection points. More specifically, any pixel that is part of the recognized overlapping points, and also has a distance equal to line width or less, from one of the detected intersection points, is deleted from the list. After this process is done, the only intersecting pixels that are left are part of the overlapping pixel group and imply overlapping between their respective lines. The results of overlapping intersection pixel detection are illustrated in figure 64. We represent them as green pixels within green circles superposed over the black colored curves.

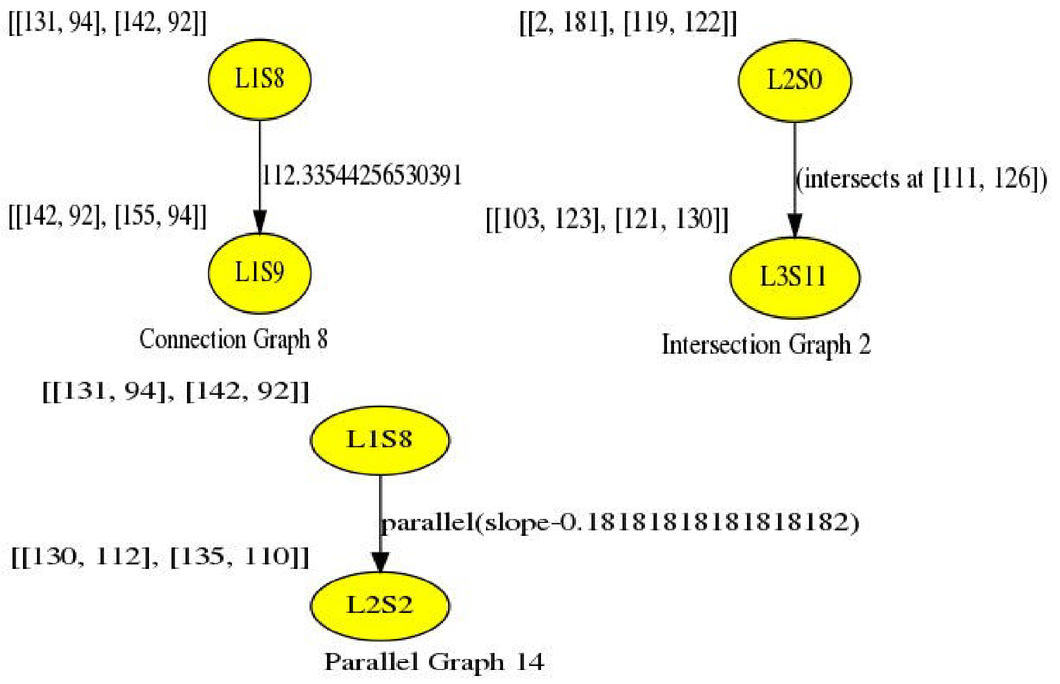


*Figure 65: Connection graph example (upper left), intersection graph example (upper right), parallel graph example (bottom image).*

#### 4.5.1.4 Conversion of Graphics into Attributed Graphs

We represent the information that the program extracts from the graphics subimage into attribute graphs. These information concern the recognized straight line segments that the curves consist of. Additionally, they include other internal associations between the curves, such as which line segments intersect, or which line segments showcase parallelism. The program searches for parallel segments of different lines, based on two rules. The first rule is that the parallel lines must not share any common points. The second rule is that all parallel lines have the same slope. For the case of detecting parallel segments in graphics image, we employ a threshold which is equal to the recognized line width minus 1 pixel. We use that threshold as means to correct any errors that occur during the curve recognition process. These errors include minor differences in computed slope values because of the line width.

The produced attributed graphs correspond to each recognized relation that is extracted from the graphics subimage. The recognized line segments consist the nodes of the graphs. The ($x_0$, $y_0$) and ($x_1$, $y_1$) positions of their respective start and end points constitute the attributes of each node. The label of the arc which connects the two nodes represents their type of relation. This relation can imply a simple connection, parallelism, or intersection. Figure 65 showcases examples for each type of attributed graph we produce so far. The connection graphs represent connections between different line segments of the same curve. The connection graph example of  figure 64 includes the connection's angle degrees as the label of the arc. Parallel and intersection graphs represent parallelism and intersections between different line segments. The parallelism graph illustrates the common slope value as the label of the arc. In contrast, the intersection graph shows the location of the intersection point as the label of the graph arc. These example attributed graphs correspond to the graphics image type 1 from figure 28. All of the illustrated example graphs are produced using the resulting line segments, after the application of the second merging rule (figure 63). Finally, we produce an aggregated version of all the connection graphs for each curve, however, we omit these results for simplicity. In general, we use different styles of graph aggregation which aim to highlight certain relations or conceal others.

#### 4.5.1.5 From Graph to Natural Language Representation

After the successful generation of the attributed graphs, the tool proceeds with their transformation into their corresponding natural language representation. This is achieved with the use of the information that is stored in the nodes and arcs of each graph relation. These sentences prove to be an important transitional step for the creation of the corresponding stochastic Petri-net graphs. For the generation of the natural language sentences, we use the concept of Agent-Verb-Patient (AVP kernels) that is presented in [75]. More specifically, the two nodes in each attributed graph relation are considered the agent (first node) and the patient (last node) of the sentence respectively. The label of the arc corresponds to the verb of the sentence. The potential verbs are “is connected to”, “is parallel to”, “is intersecting with”. Certain conjunction words are also deployed in order to enhance the semantic validity of the produced sentences.

For example the connection attributed graph from 65 is converted into “the straight-line segment L1S8 is connected with (112.33544256530391 degrees) to the straight-line segment L1S9”. We present a few additional examples of generated natural language sentences below. These sentences correspond to the relations that we retrieve from the type 1 graphics image in figure 1. It must be noted that these sentences are generated from the attributed graph results, after the application of the second level merging rule (figure 63).

***<u>NLS0:</u> the curve L2 is color magenta***

**...**

***<u>NLS3:</u> the straight-line segment L1S2 is connected with (13.314701615698201 degrees) to the straight-line segment L1S3***

***<u>NLS4:</u> the straight-line segment L1S3 is connected with (21.583969241769584 degrees) to the straight-line segment L1S4***

**…**

***<u>NLS111:</u> the straight-line segment L2S3 is parallel to the straight-line segment L1S9***

***<u>NLS112:</u> the straight-line segment L2S3 is parallel to the straight-line segment L1S16***

***<u>NLS113:</u> the straight-line segment L2S3 is parallel to the straight-line segment L3S11***

***<u>NLS114:</u> the straight-line segment L2S3 is parallel to the straight-line segment L3S14***

**…**

***<u>NLS423: </u>"The straight-line segment L2S0 is intersecting in coordinates X=111 and Y=126 with the straight-line segment L3S11"***

#### 4.5.1.6 From Natural Language to SPN Representation

The conversion of the natural language sentences to stochastic Petri-net graphs is based on the methodologies described in [75, 76] combined with the Kyrtos formal language. According to this methodology, each natural language sentence can be described by the AVP format (agent → verb(action) → patient). This means that a sentence can be reduced to agents (nouns that perform an action), verbs, and patients (nouns affected by actions). We can extract the AVP kernels of natural language sentences and use them to generate the SPN graphs, with agents and patients representing the input and output places respectively, and verbs representing the transitions.

In the case of graphics natural language sentences, the verbs are either "is parallel to", "is connected to", or "is intersecting with" sequences representing the equivalent associations between line segments. We generate a transitions for each detected verb. The agents and the patients are the line segment names, located on the right and left side of the verb respectively. The program recognizes these names, along with their respective tokens. We consider as tokens all the X and Y coordinates that are included in a natural language sentences. These tokens are used to activate their corresponding transition, generated from the verb of the same sentence. Not all sentences contain such tokens, but only those that require information about location.

Examples of such sentences consists those about connections and intersections. We also generate places that connect to the transition for each recognized agent and patient of the sentence. If the place corresponds to an agent line segment, then it is created as an input place, otherwise it is an output place. Finally, depending on the color that the program recognizes for each curve, we create a colored stochastic Petri-net graph [92] that describes its functionality. Despite including the token of equivalent color to the color name, we also assign that same color to some particular arcs. We only color the arcs that connect the places and transitions, that participate in connection relations. This color assignment represents the course that the color token takes throughout the entire SPN graph, when its transitions are fired.

Figure 66 is the combination of the generated color SPN graphs for all the curves from the same subimage into a single SPN graph. Finally, figure 67 illustrates the combined SPN that describes all the extracted information of the same subimage, including relations about parallelism and intersection. It should be noted, that while the latter SPN graph is difficult to interpret by a human, it still can be understood by the computer. After all, the goal of this research is to automate the graphics image understanding procedure by emulating the human perception.

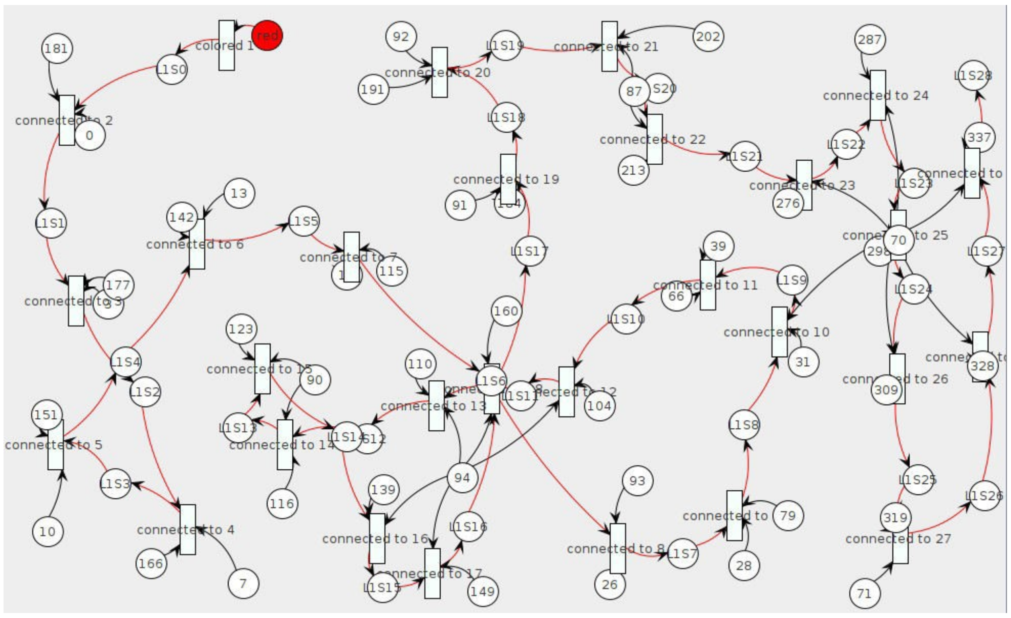


*Figure 66: The SPN graph from the sentences that correspond to the red curve of type 1 graphics image in figure 28.*

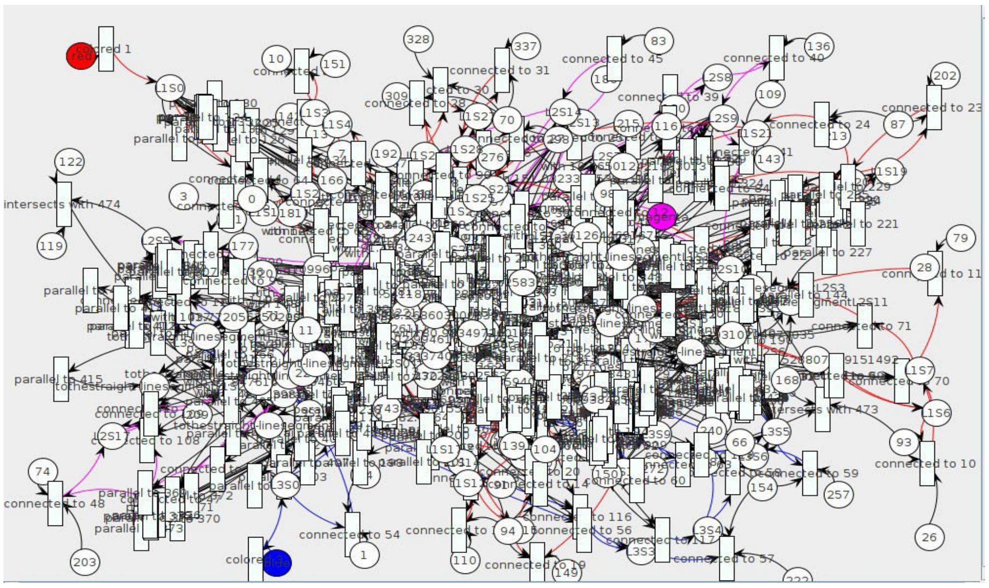


*Figure 67: The SPN graph that is generated from all the sentences that correspond to the type 1 graphics image in figure 28.*

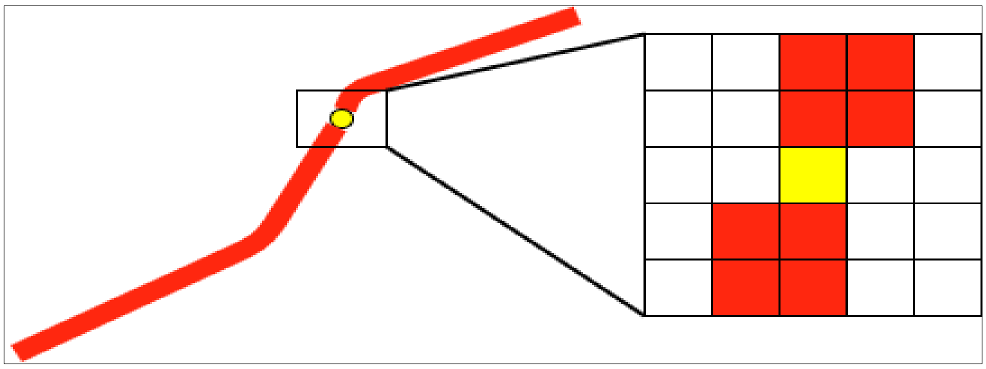


*Figure 68: Example of how the proposed matrix keeps the color of the pixels surrounding the middle point pixel.*

### 4.5.1.7 Curve Growth Understanding

Initially, we associate the recognized middle points from the subimage with their respective counterparts from the original graphics image. This is achieved through the use an 81x81 sized pixel matrix. The optimal size for that matrix was deduced through trial and error. That matrix is set around each middle point pixel and contains the surrounding neighborhood pixels information. Then, we search and locate the area within the original graphics image that matches a given matrix. The designated pixel that matches the position of the middle point from the matrix is the corresponding middle point pixel from the original image. Figure 68 showcases an example of the aforementioned matrix. An example of the association among middle points from the subimage to the original image is illustrated in figure 69.

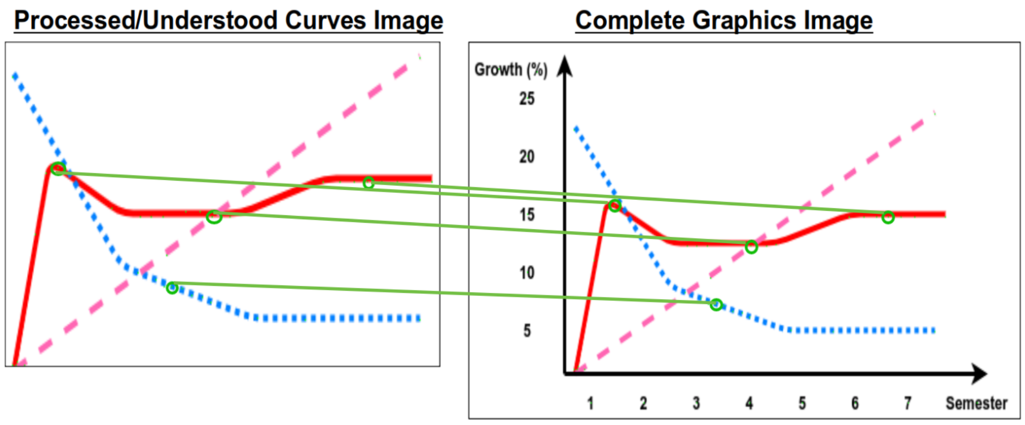


*Figure 69: The results of the middle point pixel association. Left image is the graphics subimage, right image is the original.*

In order to recognize the growth or decay information of each curve we analyze the slopes of their respective line segments. A positive slope indicates growth, while a negative slope represents decay. A steady line is recognized by a slope with 0 value. Further analysis of the slope information among consecutive line segments results in the deduction of more detailed growth conclusions. For example, continuous positive slopes with similar values correspond to linear growth. However, consecutive positive slopes with decreasing values correspond to exponential growth. Finally, we associate the deduced line segment growth results with the information recognized from the axis legends. The association is achieved by cross referencing the starting and ending pixel positions of the segments to the pixel positions of the axis values. The extracted knowledge is represented using natural language sentences. Examples of these sentences are illustrated below. The sentences are later converted into an SPN graph for easier combination with the rest of the deduced SPN functional information. Figure 69 illustrates the resulting SPN graph of the curves' growth understanding from figure 63.

***NLS0:*** ***the curve L3 is color blue***

…

***NLS15:*** ***the straight-line segment L3S4 illustrates linear decay lower than 10 %) during 5 Semester***

***NLS16:*** ***the straight-line segment L3S6 illustrates linear decay lower than 10 %) during 7 Semester***

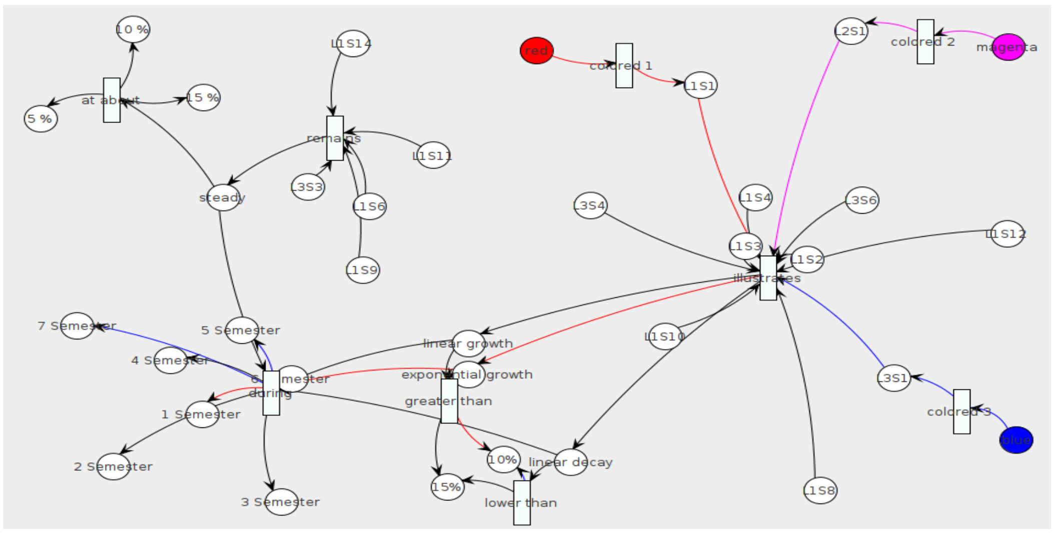


*Figure 70:* *The resulting SPN illustrating the growth rate of the understood curves from type 1 graphics in figure.*

### 4.5.2 Understanding of Similarly Colored Partitioned Curves

#### 4.5.2.1 Initial Understanding Results

We illustrate the results of the recognized straight line segments that form the curves, along with the results for the first merging, second and third merging rules in figure 71. The straight line segments are highlighted as yellow against the black colored original image partitioned curves. In figure 72 we present the reconstructed curves with their respective recognized colors. All of these results correspond to the type 2 graphics image from figure 28.

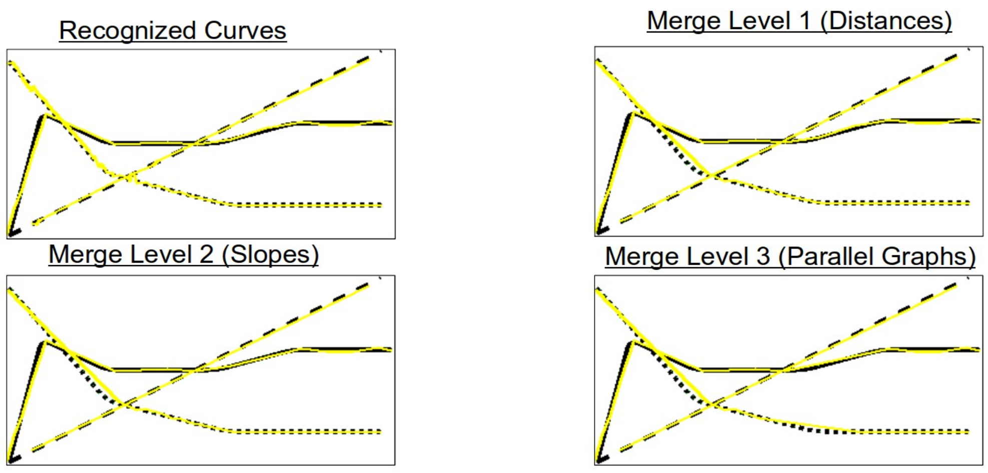


*Figure 71: The results recognized straight line segments before and after the application of the merging rules.*

In figure 73 we present the results of intersection. More specifically, use case 1 results showcase points of intersection between lines, and use case 2 results indicate possible areas of overlapping. It must be noted that due to errors in the coloring process of some regions, we have potential loss of information in the aspect of overlapping recognition. We correct these errors with the deployment of the feedback module. We illustrate the corresponding attributed graph results in figure 74. We include examples of a connection graph for degree of angle, a parallelism graph and an intersection graph.

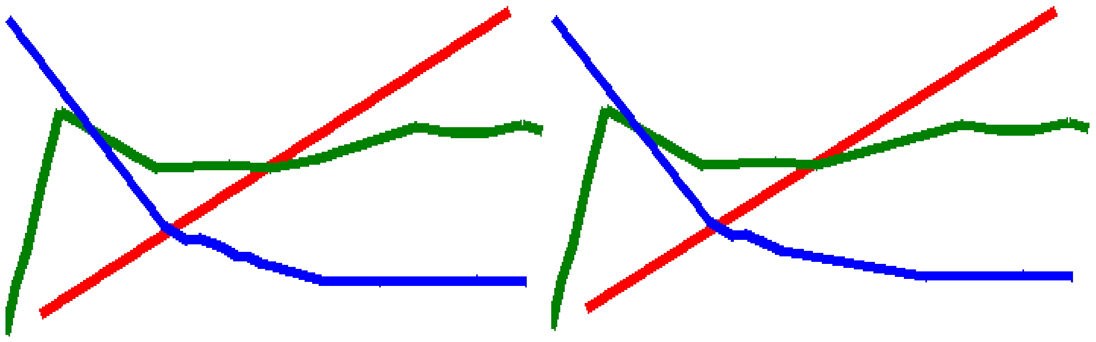


*Figure 72: Colored reconstructed curve after the 2nd merging rule (right) and 3rd merging rule (left) respectively.*

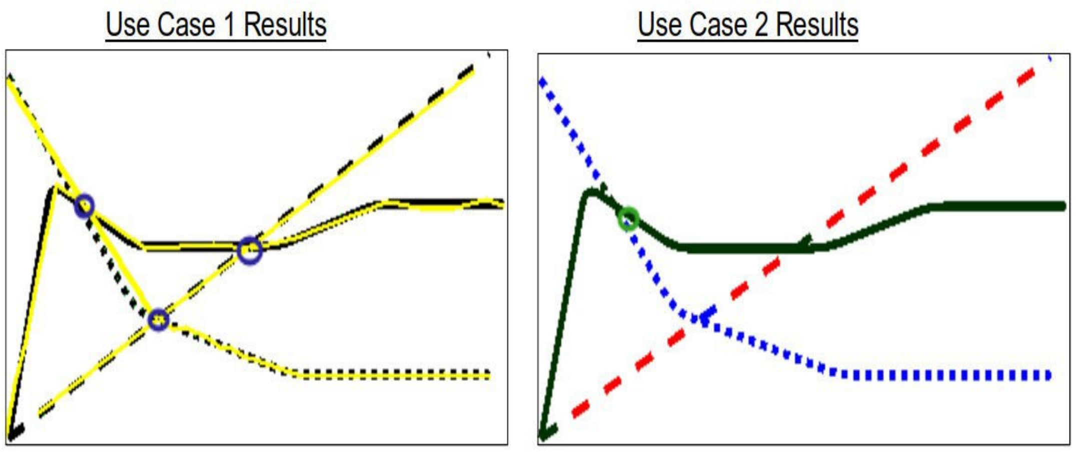
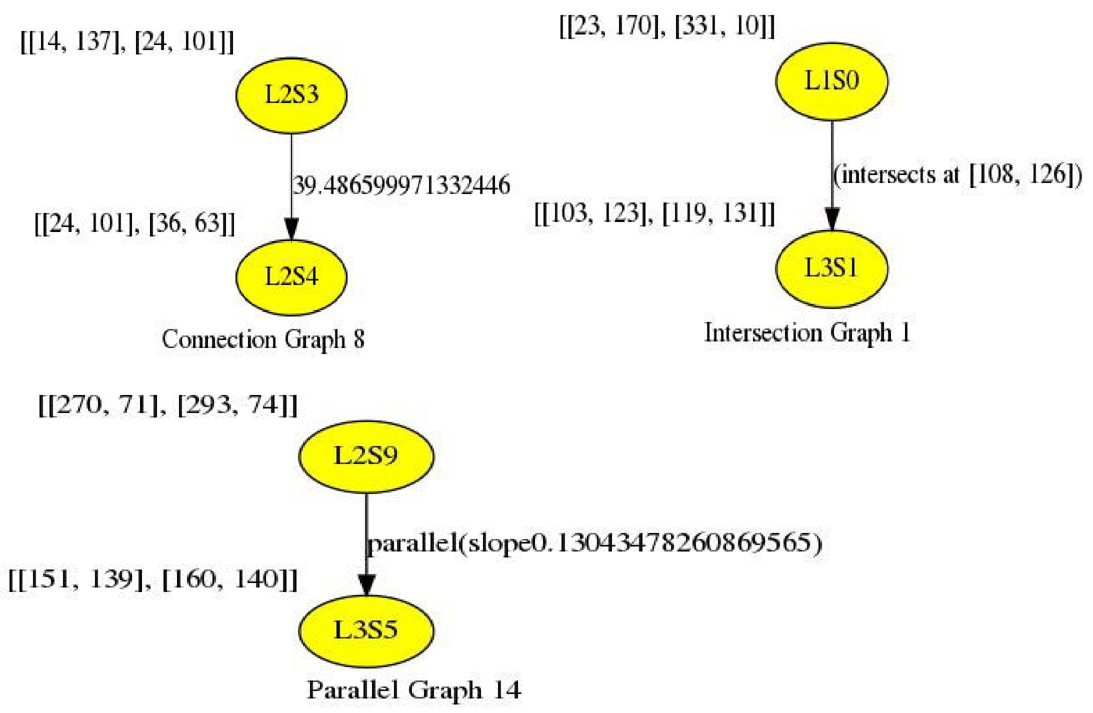


*Figure 73: The results of intersection.*



*<u>Figure 74:</u> Attributed graph results for partitioned graphics input image. A connection graph (upper right), a parallelism graph (bottom image) and an intersection graph (upper left).*

Below we showcase some of the produced natural language sentences that correspond to the attributed graphs retrieved relations from the type 2 graphics image of figure 72.

***<u>NLS0</u>*: *the curve L2 is color green***

…

***<u>NLS8</u>*: *the straight-line segment L2S7 is connected with (117.25110201818276 degrees) to the straight-line segment L2S8***

…

***NLS24*****: the straight-line segment L2S4 is parallel to the straight-line segment L3S3***

***NLS25*****: the straight-line segment L2S4 is parallel to the straight-line segment L3S4***

Finally, we showcase the generated colored stochastic Petri-net graphs of the natural language sentences. Figure 75 illustrate all the extracted information that describe all the curves of the type 2 graphics image from figure 28, combined into a single SPN. The X and Y coordinates of the starting points for each segment serve as extra tokens.

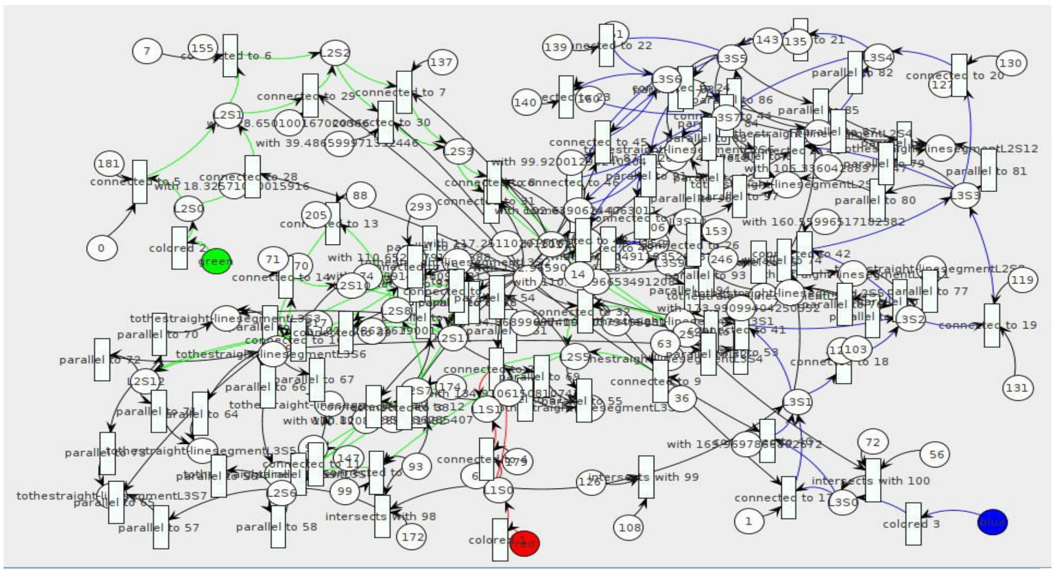


*Figure 75: The SPN result for the extracted information that describe the type 2 graphics image from figure 28.*

### 4.5.2.2 Feedback Understanding Results

We illustrate the results of the recognized straight line segments that form the curves, along with the results for the first, second and third merging rules in figure 76. The straight line segments are highlighted as yellow against the black colored original image partitioned curves. Figures 77 demonstrates the colored reconstruction line results after the application of the second and the third merging rules respectively. These results come after the recognition of the color name for each curve.

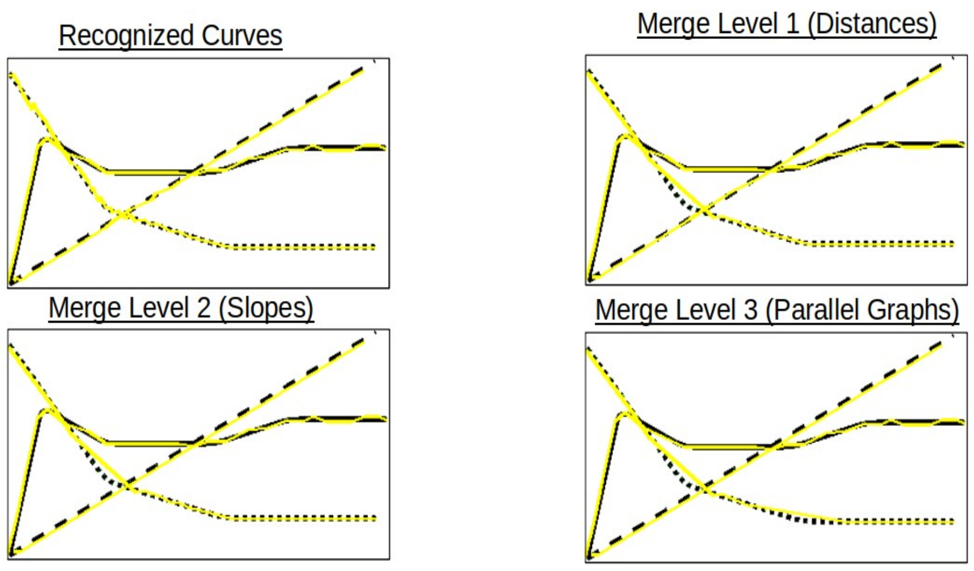


*Figure 76: The results recognized straight line segments before and after the application of the merging rules. These results correspond to the output colored image of the feedback module.*

In figure 78 we illustrate the results of intersection recognition. More specifically, use-case 1 showcases the intersecting points between the curves, while use case 2 illustrates the results of potential overlapping. The results in use case 2 highlight the success of the feedback module, since the feedback analysis procedure restores any previously lost information and color corrects the graphics image.

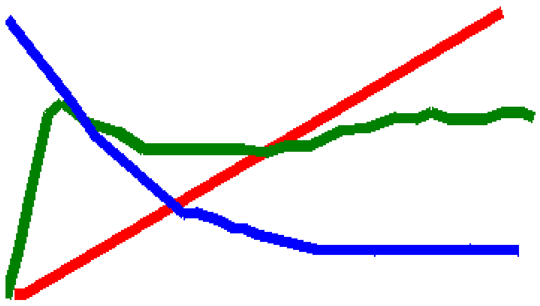


*Figure 77: The results recognized straight line segments before and after the application of the merging rules. These results correspond to the output colored image of the feedback module.*

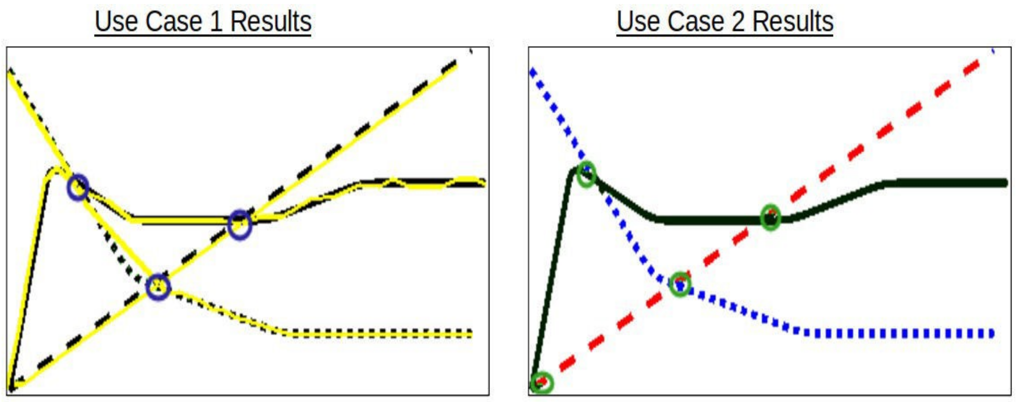


*Figure 78: The results of intersection. These results correspond to the output colored image of the feedback module.*

We illustrate the corresponding attributed graph results in figure 79. We include examples of a connection graph for degree of angle, a parallelism graph and a graph for intersections. Below we give an example of the produced natural language sentences results that correspond to the attributed graphs for the example of the type 2 graphics image from figure 28.

***NLS0:*** ***the curve L2 is color green***

***NLS1:*** ***the straight-line segment L1S0 is connected with (134.82095127186102 degrees) to the straight-line segment L1S1***

***NLS2:*** ***the straight-line segment L2S0 is connected with (20.07711581786 degrees) to the straight-line segment L2S1***

**…**

***NLS111:*** ***the straight-line segment L2S4 is parallel to the straight-line segment L3S2***

Finally, we showcase the generated colored stochastic Petri-net graphs of the natural language sentences. Figure 80 illustrate all the extracted information that describe all the curves of the type 2 graphics image from figure 28, combined into a single SPN.

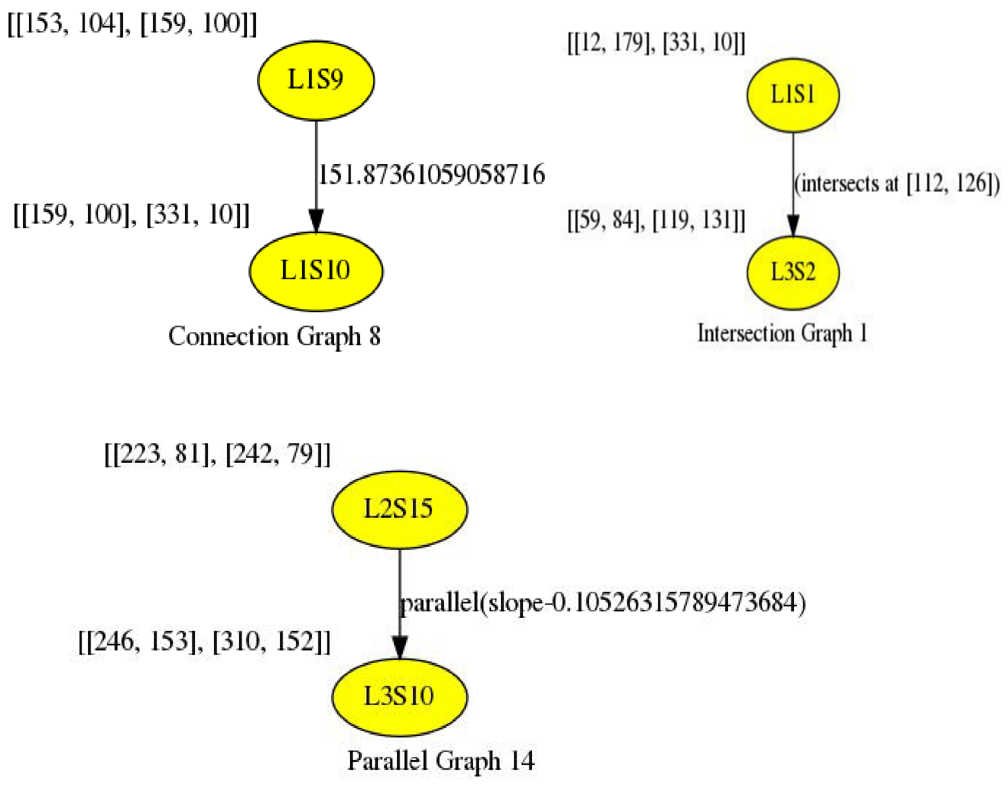


*Figure 7 9 : A connection graph (upper right), a parallelism graph (bottom image) and an intersection graph (upper left). These results correspond to the output colored image of the feedback module.*

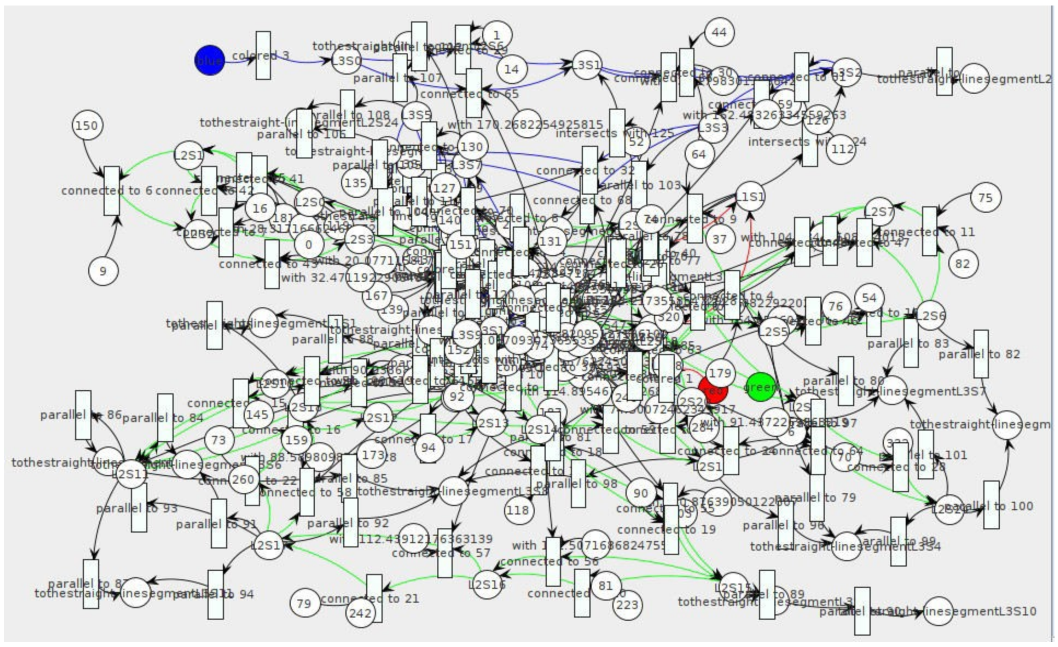


*Figure 80: The SPN result for the extracted information that describe the type 2 graphics image from figure 28. These results correspond to the output colored image of the feedback module.*

### 4.5.3 Understanding of Bars

We use the corresponding recognized middle points in order to get the ratios of heights between the bars of the graphics image. Different bars may have same or different heights which represent their current value. However, the variations of a bar's height with respect to other bars, both from its group and from the other groups, reflect their growth over time. Therefore, we compute the fractions between the ratios of the bars. We convert the extracted information in natural language sentences for easier processing and mapping into SPN graphs. We present examples of these sentences below.

***NLS4:*** ***the bar 1-1 has a size of 275/351 of bar 1-2***

***NLS5:*** ***the bar 1-1 has a size of 275/373 of bar 1-3***

***…***

***NLS47:*** ***the bar 2-8 has a size of 329/296 of bar 2-10***

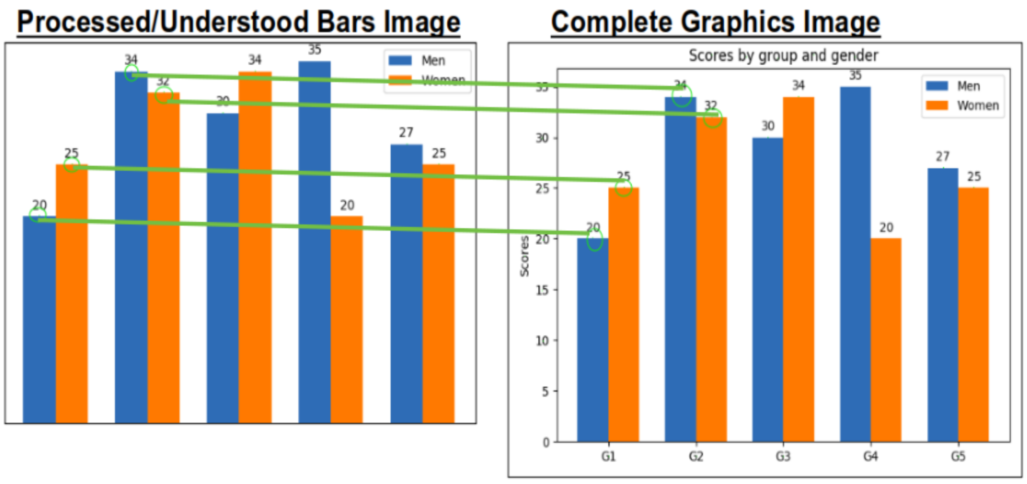


*Figure 81: The results of the middle point pixel association. Left image is the bars subimage, right image is the original bars image.*

Additionally, we store a matrix of size 81x81 around each recognized middle point in order to deduce the location of the middle point in the original bars chart image. We use the produced auxiliary array in order to convert the understood conclusions for the original axis graphics. We also use that auxiliary array to associate the understood growth rates with the values extracted from the axis, in the next section. Figure 81 showcases the association results. It is evident by the results, that the detection of the middle points and their corresponding association is successful.

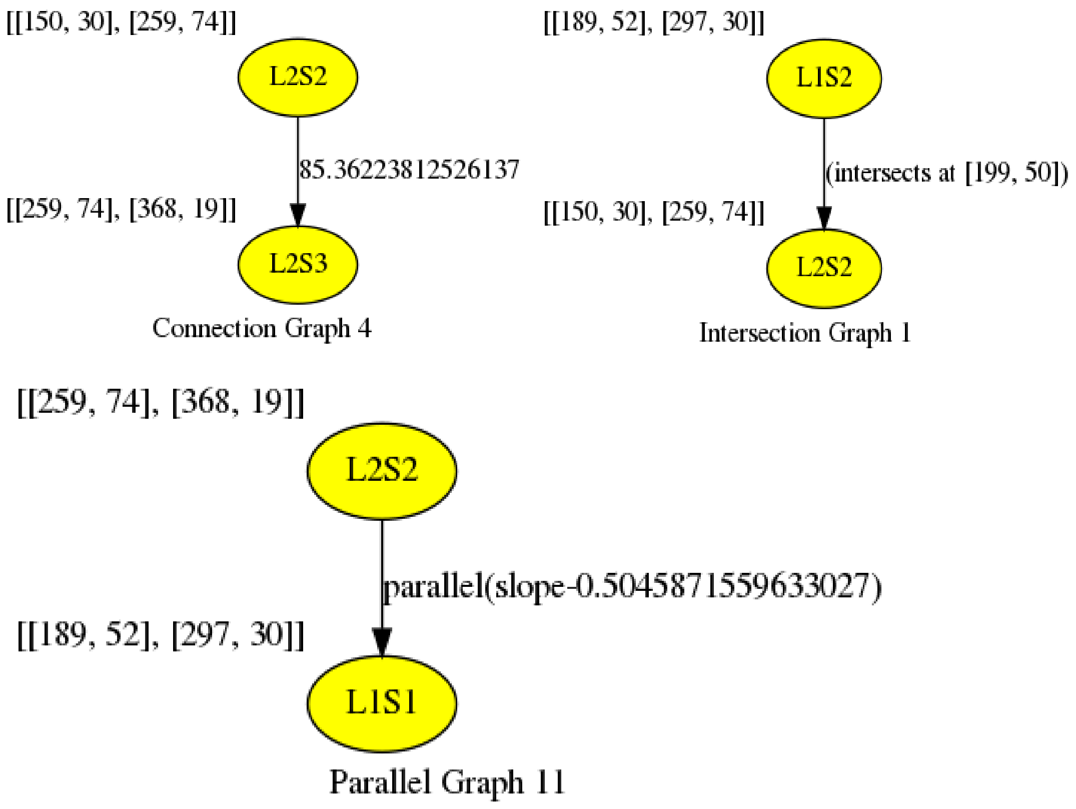


*Figure 82: Upper left image illustrates a connection graph, upper right image illustrates an intersection graph, bottom image illustrate a parallelism graph.*

After the association between the original and the subimage is completed, we apply the same understanding methodology as the one that we introduced for the curves in graphics images. More specifically, we transform the structural information that the program extracts from the new curves image into attribute graphs. These information concern the recognized straight line segments that constitute the curves when they are combined. They also include the points of intersection between the different line segments, as well as any illustrated parallelism. The produced attributed graphs correspond to each recognized relation extracted from the graphics image. The recognized line segments consist the nodes of the graphs, and the X and Y positions of their respective start and endpoints consist the attributes of each node. The label of the arc which connects the two nodes represent their type of relation. These relations might be simple connection, parallelism, or intersection. Figure 82 showcases examples for each type of attributed graph that we can produce. The connection graphs represent connections between different line segments of the same curve, with the arc label being the angle of connection. Parallel graphs shows the parallelism between two different segments with the label of the arc being their common slope value. Finally, intersection graphs represent intersection between different line segments with the label of the arc being the location of the intersection.

Each graph represents a different recognized association and relation between the curves generated from the bars. So, we convert the information that is stored for each relation into their counterpart corresponding natural language representation. These sentences are important since they help map the recognized functionality that is illustrated in the given image into SPN graphs. The generation of the natural language sentences follows the same A-V-P format that we have defined in previous sections of this work. We present a few of the generated natural language sentences that correspond to the input graphics image of the previous examples. It must be noted that these sentences are generated from the attributed graphs that correspond to the information extracted after the second level merging rule is applied.

***<u>NLS1:</u>* the curve L2 is color red**

***<u>NLS2:</u>* Men is color red**

**…**

***<u>NLS6:</u>* the straight-line segment L1S3 with starting point at [297, 30] is connected to straight-line segment L1S4**

**…**

***<u>NLS22:</u>* the straight-line segment L1S2 is intersecting in coordinates X= 199 and Y=50 with the straight-line segment L2S2**

**…**

***<u>NLS69:</u>* the straight-line segment L1S1 shows linear growth until the straight-line segment L1S2**

In the case of graphics natural language sentences, the verbs are "is parallel to", "is connected to" or "is intersecting with" sequences representing the equivalent associations between line segments. We add the verb "has a size of" when the natural language sentence contains information about bars. We generate a transition for each detected verb. The agents and the patients are the line segment or bar names, which are located on the right and left side of the verb respectively. The program recognizes these names, along with their respective tokens. We consider as tokens all the X and Y coordinates information included in a natural language sentences. These tokens are used to activate their corresponding transition, generated from the verb of the same sentence. Not all sentences contains such tokens, but only those that require information about location. Examples of such sentences are those about connections and intersections.

In addition, we generate places that connect to the transition for each recognized agent and patient of the sentence. Finally, depending on the color that the program recognizes for each bar, we create a colored stochastic Petri-net graph that describes its functionality. Despite including the token of equivalent color to the color name, we also assign that same color to some particular arcs. We only color the arcs that connect the places and transitions, that participate in connection relations. This color assignment represents the course that the color token takes throughout the entire SPN graph, when its transitions are fired. Figure 83 illustrates and example of the generated colored SPN graph regarding the connection information of the curves generated from the bars graphics image. Figure 84 presents an example of the generated colored SPN graph regarding the recognized bar size ratios. Finally, figure 85 showcases the resulting colored SPN graphs of all the information that we have extracted from the input bars graphics image. All resulting SPN graphs correspond to the type 3 bar graphics image from figure 28.

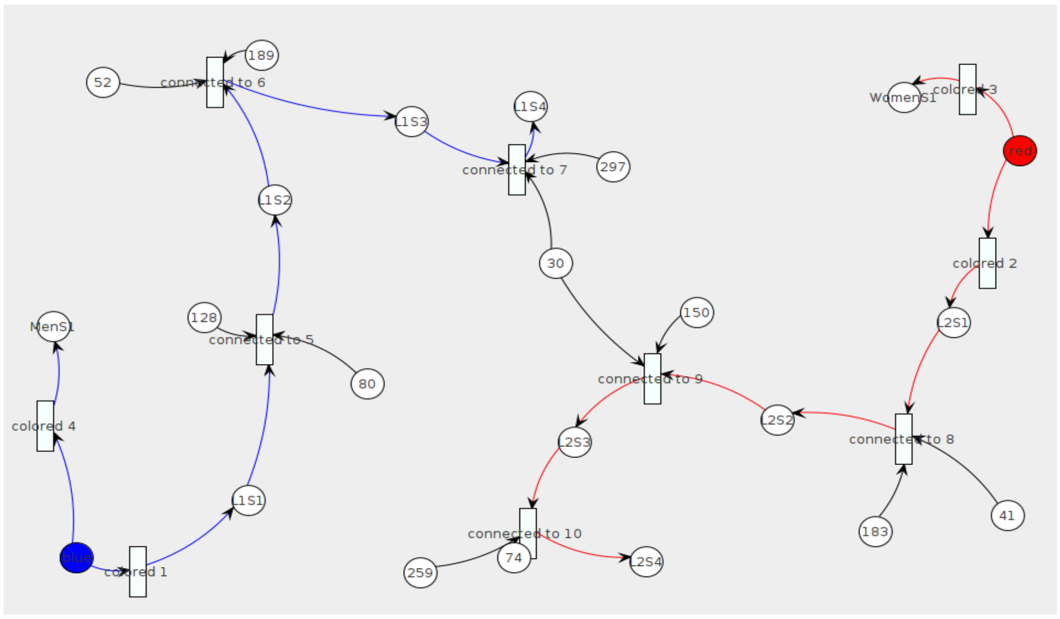


*Figure 83:* *The SPN results of all bars combined.*

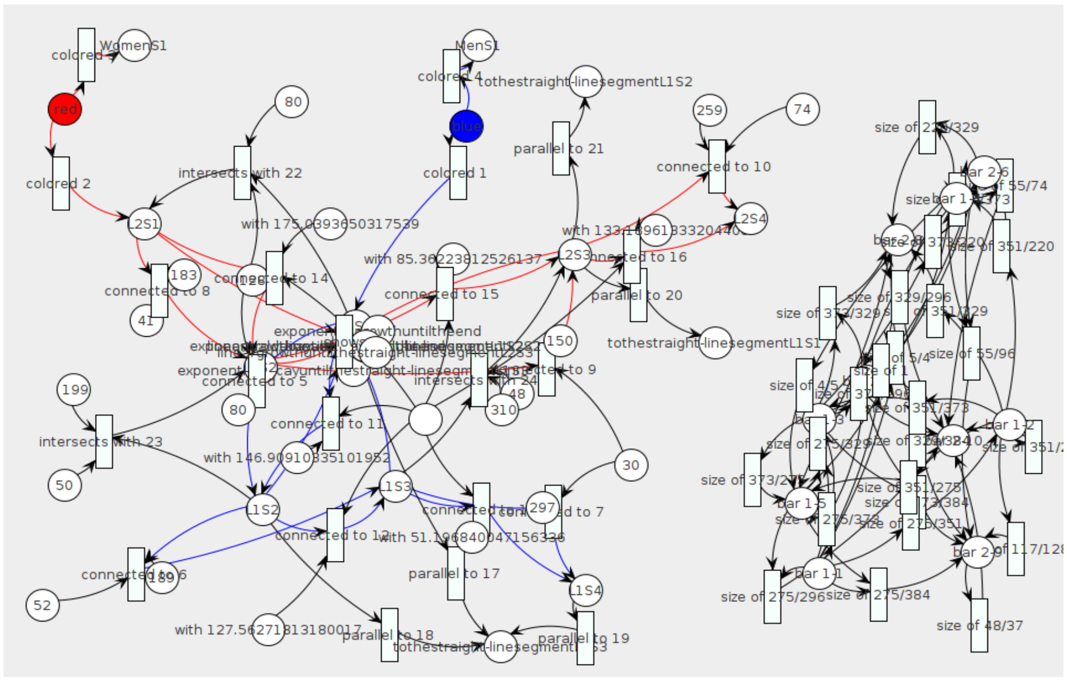


*Figure 84:* *The SPN results for the recognized bar size ratios.*

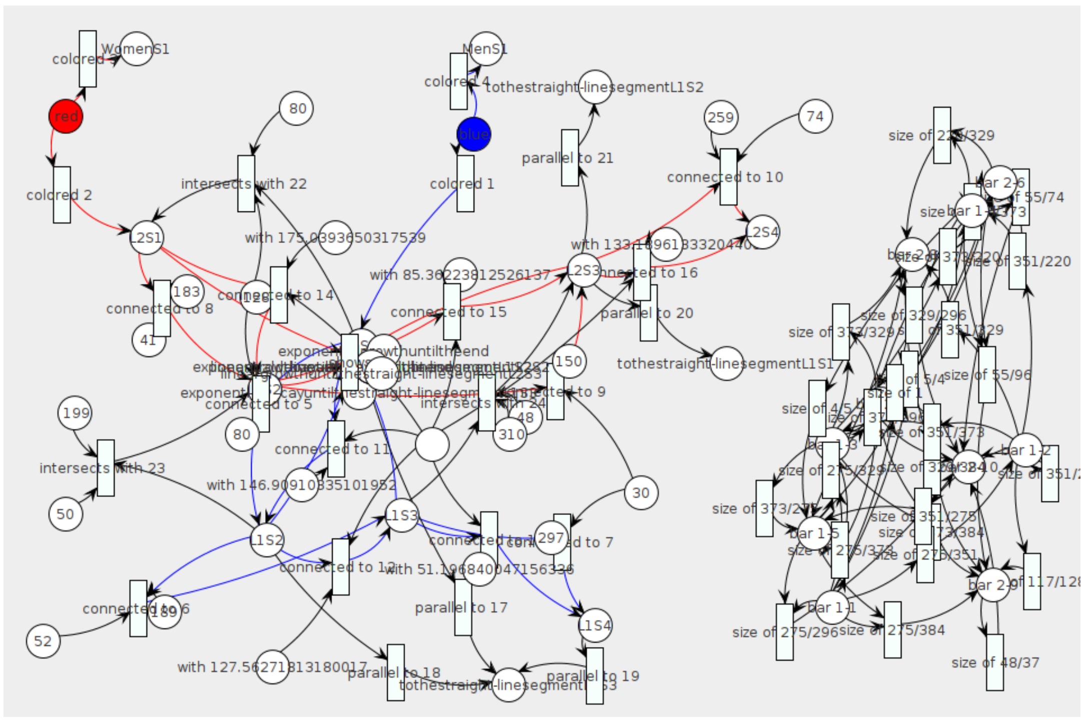


*Figure 85:* *The SPN results for the combined extracted information that describe the graphics image.*

# 5. Function Regeneration from Graphics Images

## 5.1 Curve Type Recognition

The purpose of the current research work has been to understand all the information that is contained within the images of modalities from a technical document. In the case of graphics images, that information contains both the visually available data such as axis text and the embedded functionality of the illustrated curves such as parallelism, intersections and monotonic growth. However, the extraction of that kind of information enables us to generate new and additional knowledge about the curves contained in the graphics image, without necessarily requiring apriori knowledge from the document's text. More specifically, we aim to deduce the functions that describe the curves that are contained within the input graphics image. This is achieved by using the previously extracted knowledge regarding the monotonic behavior of their corresponding line segments. The algorithm 5 is developed in order to categorize the curves into the respective function type that describes them. In case no function type can be deduced, then we introduce an adaptive scheme based on Cubic Splines. It must be noted that since we have 2D pixels as the available dataset, the functions will describe the captured pixel behavior.

The algorithm initiates by receiving a list containing the monotonic characteristics of each curve from a given graphics image. Such characteristics include the deduced growth or decay, linear or exponential for each straight line segment. Then, the algorithm checks for the occurrence number of the words linear, growth and decay in the list. If more than the 2/3 of the list's nodes are characterized as linear, then the curve is probably described by a linear function. The occurrence counting of growth and decay indicates whether the linear function has a positive or negative slope respectively. In case the curve is not linear, then we deduce the function type based on the number of times that the curve changes its direction and therefore its monotonic sign. More specifically, if the curve illustrates initially continuous growth or decay before following a steady pattern, then it is probably described by the asymptotic function. Alternatively, if the curve's direction changes only once from growth to decay and vice versa, then it is probably described by the polynomial function. Finally, multiple changes in the direction of the curve are recognized as the sinusoidal function.

**Algorithm V: Function Type Recognition**

```
$MonotonyList \leftarrow initialize(Curve\_Image)$
$Lcounter \leftarrow countLinear()$
$Gcounter \leftarrow countGrowth()$
$Dcounter \leftarrow countDecay()$
if $Lcounter >= 2/3 * Length(MonotonyList)$ then
  if $Gcounter > Dcounter$ then
    return $"Linear : a * x + b, a < 0"$
  else
    return $"Linear : a * x + b, a > 0"$
  end if
else
  $Ccounter \leftarrow countMonotyChange(MonotonyList)$
  if $Ccounter = 1$ then
    if $MonotonyList.elementAt(0) = "Decay"$ then
      return $"Quadratic : a * x^2 + b * x + c, a > 0"$
    else
      return $"Quadratic : a * x^2 + b * x + c, a < 0"$
    end if
  else
    if $Gcounter >= 2/3 * Length(MonotonyList)$ then
      return $"Asymptotic : 1/(a * x), growth"$
    else if $Dcounter > 2/3 * Length(MonotonyList)$ then
      return $"Asymptotic : 1/(a * x), decay"$
    else
      if $MonotonyList.elementAt(0) = "Decay"$ then
        return $"Sinusoidal : a * sin(b * x + c) + d, a > 0"$
      else
        return $"Sinusoidal : a * sin(b * x + c) + d, a < 0"$
      end if
    end if
  end if
end if
```

## 5.2 Function Regeneration of Linear Curves

We input the graphics image that is illustrated in figure 86 through the aforementioned graphics processing methodology presented in the previous chapter. More specifically, we begin with the application of pyramidal reduction in order to preserve the essential structural information of the image. This procedure converts any text in the axis legends into pixels forming boxes. The resulting image is eroded in order to  enhanced the box formations. Then, we use Table OCR [94] combined with a box detection algorithm in order to recognized the contents of each box, which are the corresponding axis values. Then, the HSV homogeneity filter is applied to the image in order to remove coloring noise and isolate the curve from the grid. The results from each step of this procedure are illustrated in figure 86.

The graphics recognition methodology proceeds with the computation of the unevenness points for each curve. Then, the hierarchical clustering algorithm is applied in order to extract the 2D middle points that form the straight line segments for each curve in the graphics image. More specifically, we apply two versions of the hierarchical clustering algorithm, 1 using a minimum and 1 using a maximum distance threshold among the pixels of each cluster just like in the previous chapter. Then, we compute the optimal cluster number using the elbow metric before using that number in the K-means clustering algorithm, that results in the clusters of each line segment. The 2D middle points are computed from these clusters. Figure 87 illustrates the results each of the aforementioned clustering steps.

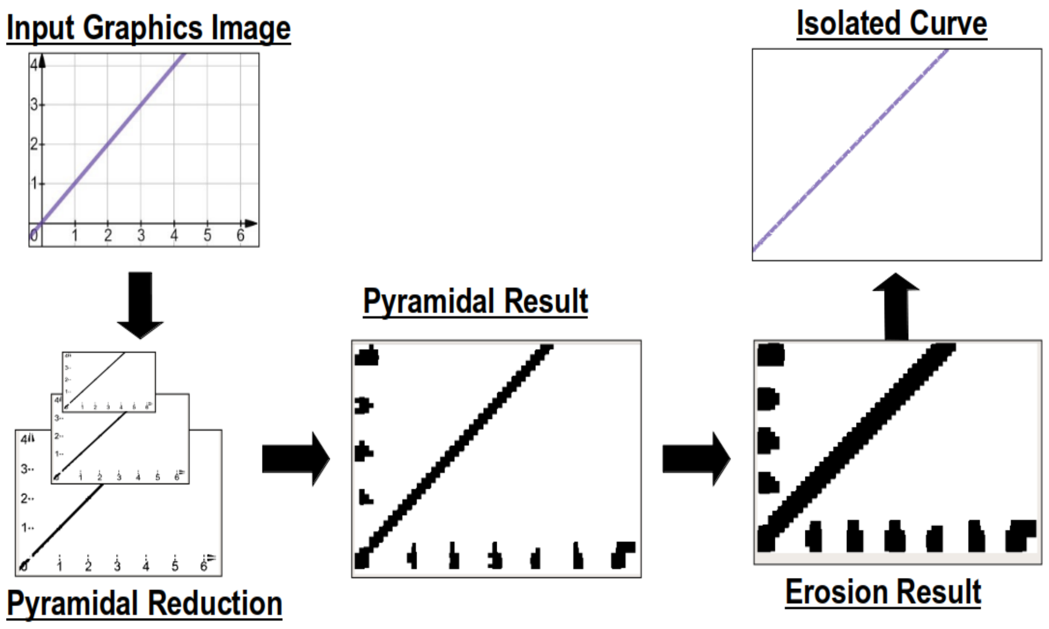


*Figure 86: Linear Curve Preprocessing Results.*

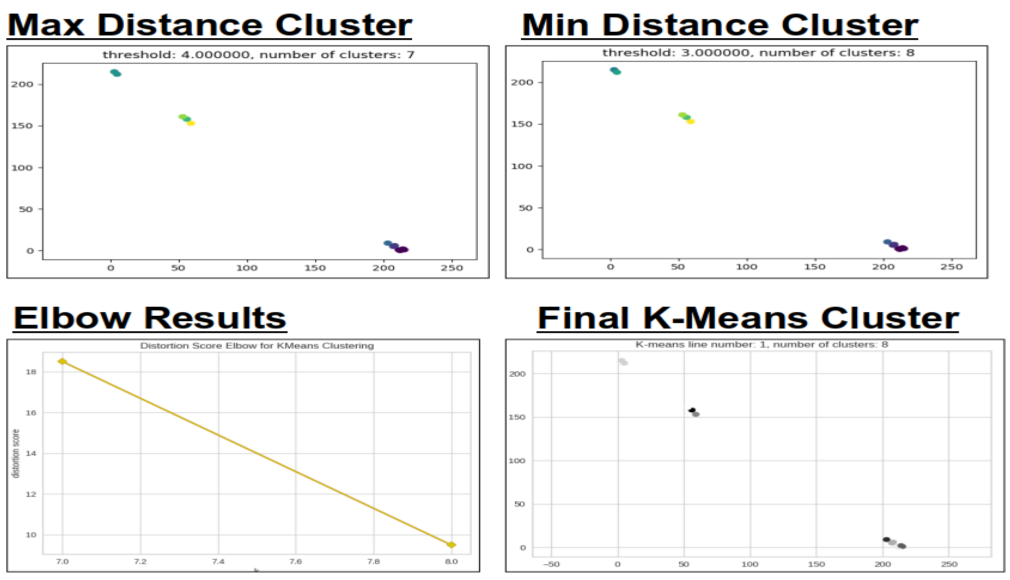


*Figure 87: Linear Curve Clusters Analysis Results.*

The graphics understanding methodology uses the deduced 2D middle points in order to construct the respective straight line segments of the curve. As it is described in the previous chapter, we apply 3 consecutive merging rules in order to remove any unnecessary segments. The result from the recreated curve using the segments remaining after the 3rd merging rule are illustrated in figure 88. Finally, we compute the monotony of each curve using the slopes of its straight line segments and using that information we recognize that the curve is described by a linear function ( y = a*x + b). We solve that equation for every remaining 2D middle point in order to get the best possible approximations for the parameters a and b. The resulting pixel based linear function is y = -1.015 * x + 216.65. It must be noted that the resulting function has a negative slope, due to the fact that we parse the image following a direction from top to bottom. The result of the recreated curve using the deduced pixel based linear function is illustrated in figure 88.

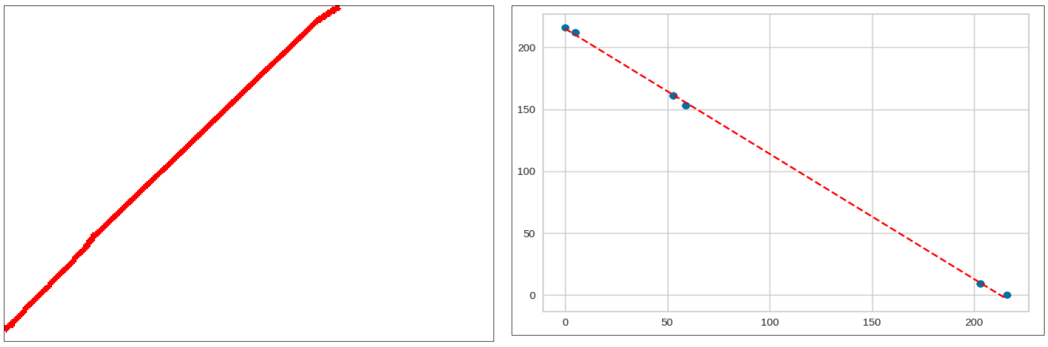


*Figure 88:* *The linear curve recreated from 2D middle points (left image) and from the deduced pixel function (right image).*

The computation of the pixel based function enables the computation of the actual function that is represented in the graphics image. That is due to the recognition of the axis values and their conversion into actual floating numbers as discussed in previous chapters. More specifically, we solve the deduced pixel based linear function using the x positions of the values 2 and 3 from the x-axis. This outputs the y positions of the corresponding y-axis values. These results are 58.2 and 104.93 which correspond to the values 2 and 3 from the y-axis respectively. The results from the calculations of the pixel based function given these values as input are illustrated in figure 89. Then, a system of equations is generated (2 = a * 2 + b and 3 = a * 3 + b) which can be solved to deduce the a and b parameters' values. Therefore, the resulting actual function of the input graphics image is y = 1 * x + 0 = x, which is true.

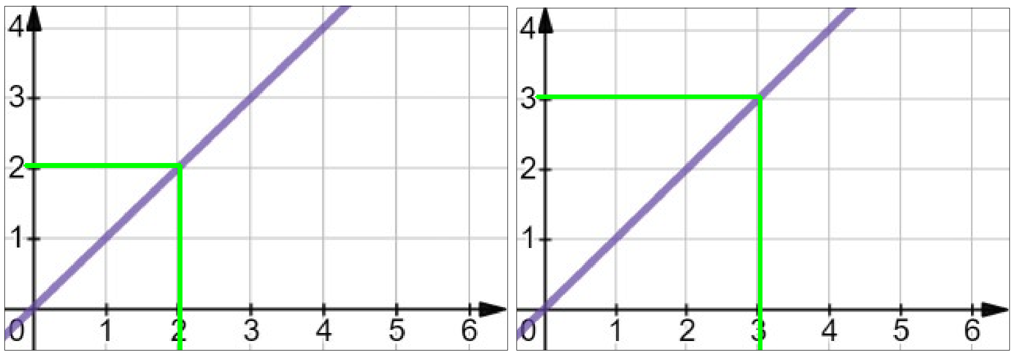


*Figure 89: The y position solutions of function y = -1.015 * x + 216.65 for the x position of the x-axis value 2 (left) and x-axis value 3 (right) respectively.*

## 5.3 Function Regeneration of Polynomial Curves

Similarly to the linear curve, we parse the graphics image that is illustrated in figure 90 through the aforementioned graphics processing methodology presented in the previous chapter. The results from the application of the pyramidal reduction, the erosion and the isolation of the curve using the HSV homogeneity filter are illustrated in figure 90. Then, the graphics recognition methodology proceeds with the computation of the 2D middle points that form the straight line segments for each curve in the graphics image. Figure 91 illustrates the results from the application of the maximum and minimum distance based hierarchical clustering algorithms, the computation of the elbow and the application of the K-means algorithm.

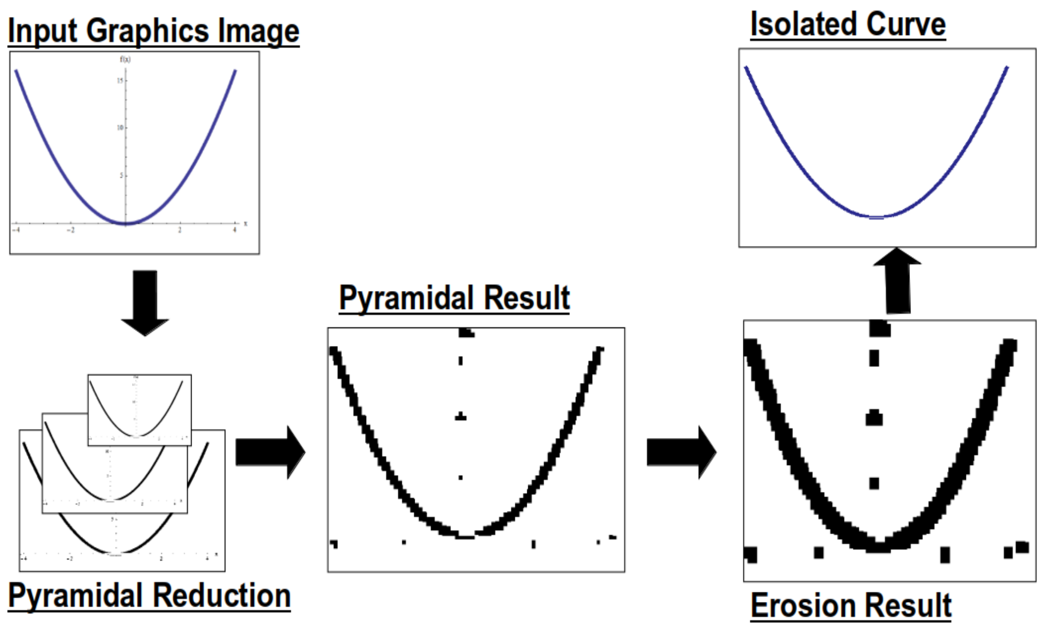
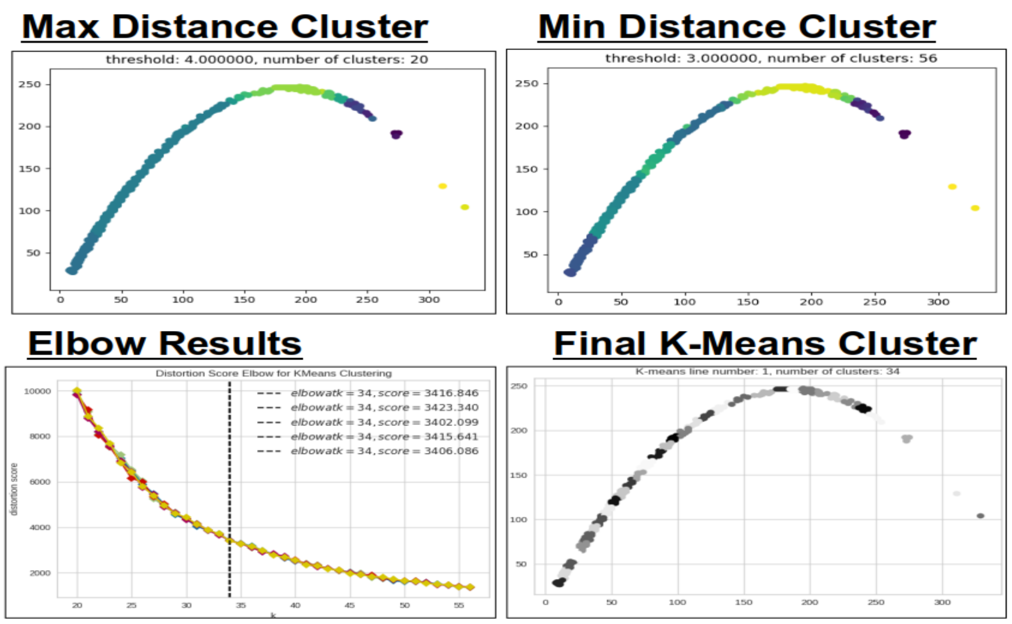


*Figure 90: Polynomial Curve Preprocessing Results.*



*<u>Figure 91:</u> Polynomial Curve Clusters Analysis Results.*

The graphics understanding methodology uses the deduced 2D middle points in order to construct the respective straight line segments of the curve and capture its monotonic information. The result from the recreated curve using the segments remaining after the 3rd merging rule are illustrated in figure 92. Finally, the function type recognition algorithm deduces that the curve is described by a polynomial function (y = a*x ^2 + b*x + c). We solve that equation for every remaining 2D middle point in order to get the best possible approximations for the parameters a, b and c. The resulting pixel based polynomial function is y = 2.56 * x^2 – 0.0069*x + 5.95. The result of the recreated curve using the deduced pixel based polynomial function is illustrated in figure 92.

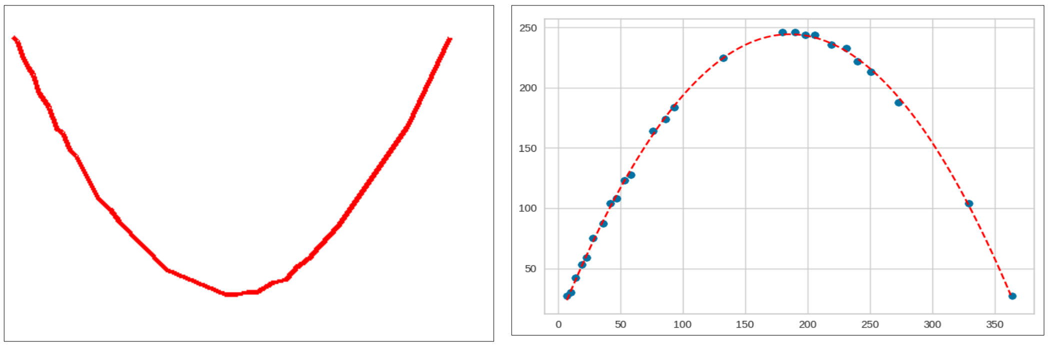


*Figure 92:* *The polynomial curve recreated from 2D middle points (left image) and from the deduced pixel function (right image).*

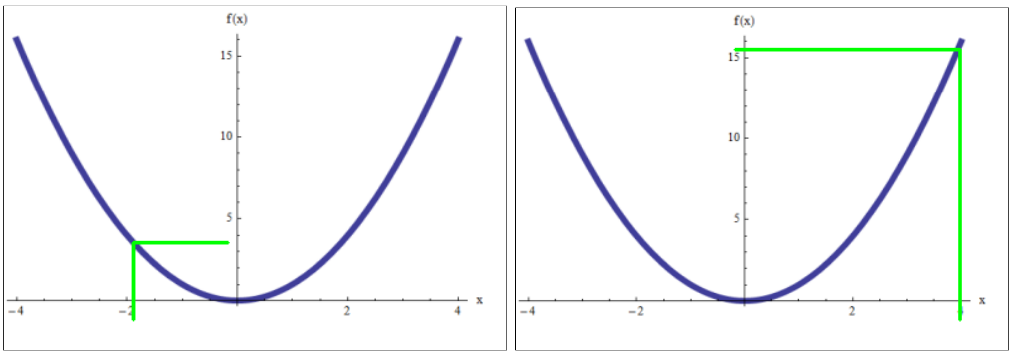


*Figure 93:* *The y position solutions of function y = 2.56 * x^2 – 0.0069*x + 5.95 for the x position of the x-axis value -2 (left) and x-axis value 4 (right) respectively.*

Similarly to the previous section, we solve the deduced pixel based linear function using the x positions of the values -2 and 4 from the x-axis in order to deduce the actual function that is represented in the input graphics image. This outputs the y positions of the corresponding y-axis values. These results are 197.18 and 35.63 which correspond to the values -2 and 4 from the y-axis respectively. The results from the calculations of the pixel based function given these values as input are illustrated in figure 93. Then, a system of equations is generated, which are 3.5 = a*(-2) ^2 + b*(-2) + c and 15.5 = a*4 ^2 + b*4 + c. The third equation of the system is the solution for x=0, which is equal to c = 0. Therefore, the resulting actual function of the input graphics image is y = (15/16)*x^2 + (1/8)*x.

## 5.4 Function Regeneration of Asymptotic Curves

Similarly to the previous curves, we parse the graphics image that is illustrated in figure 94 through the aforementioned graphics processing methodology presented in the previous chapter. The results from the application of the pyramidal reduction, the erosion and the isolation of the curve using the HSV homogeneity filter are illustrated in figure 94. Then, the graphics recognition methodology proceeds with the computation of the 2D middle points that form the straight line segments for each curve in the graphics image. Figure 95 illustrates the results from the application of the maximum and minimum distance based hierarchical clustering algorithms, the computation of the elbow and the application of the K-means algorithm.

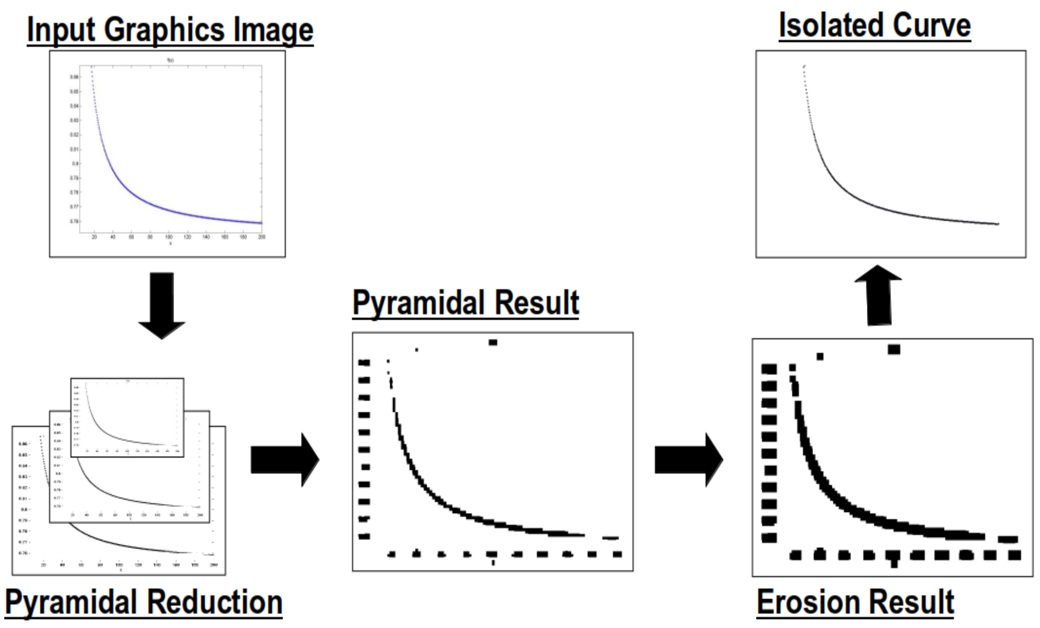


*Figure 94: Asymptotic Curve Preprocessing Results.*

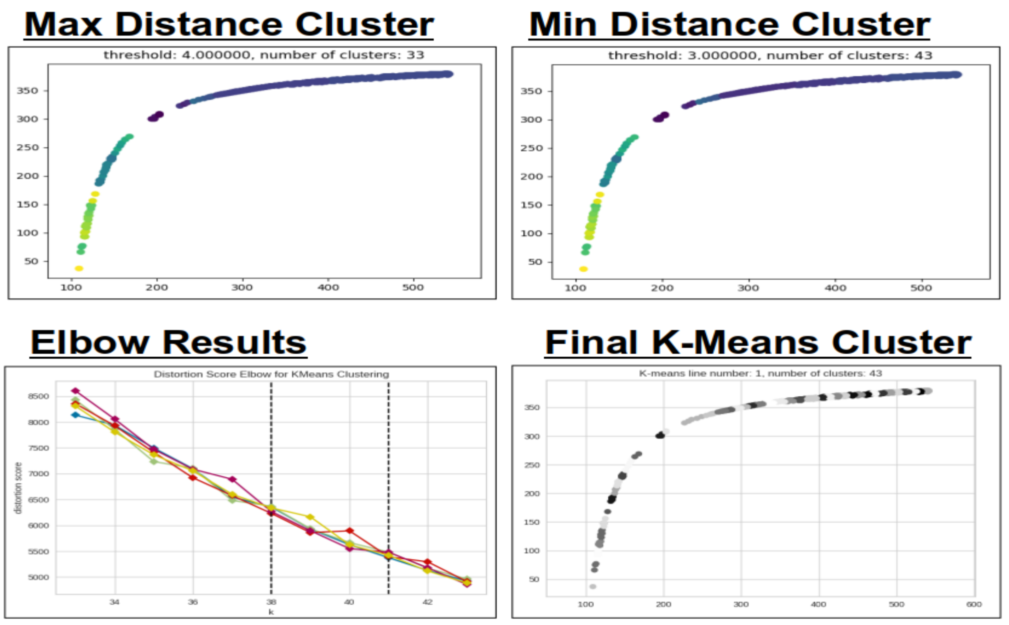
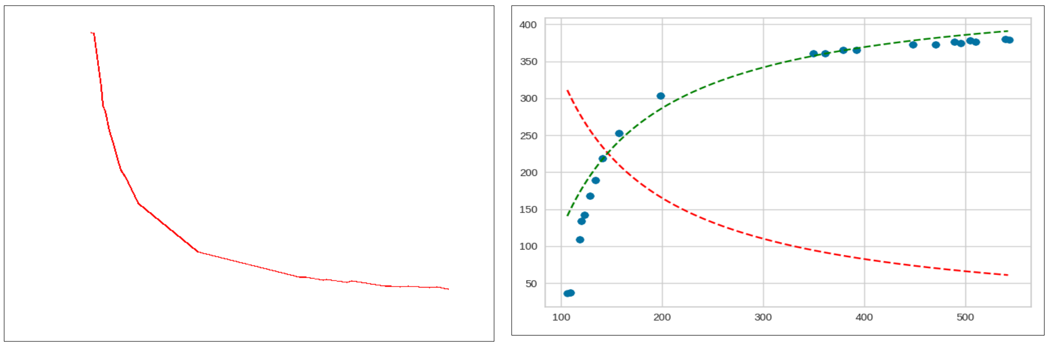


*Figure 95: Asymptotic Curve Clusters Analysis Results.*

The graphics understanding methodology uses the deduced 2D middle points in order to construct the respective straight line segments of the curve and capture its monotonic information. The result from the recreated curve using the segments remaining after the 3rd merging rule are illustrated in figure 96. Finally, the function type recognition algorithm deduces that the curve is described by an asymptotic function (y = 1/( a*x)). We solve that equation for every remaining 2D middle point in order to get the best possible approximations for the parameter a. The resulting pixel based asymptotic function is y = 1 / (0.003 * x). The result of the recreated curve using the deduced pixel based polynomial function is illustrated in figure 96. It must be mentioned that the red partitioned line in these results represents the initially deduced function is inverted due to the parsing direction of the image (top to bottom). Thus, we compute the intersection point between the data points and the red partitioned line and using its location we invert the function once again in order to match the 2D data points.



*<u>Figure 96:</u> The asymptotic curve recreated from 2D middle points (left image) and from the deduced pixel function (right image).*

Similarly to the previous section, we solve the deduced pixel based linear function using the x position of the values 100 from the x-axis in order to deduce the actual function that is represented in the input graphics image. This outputs the y position which corresponds to y-axis value 0.765 as shown in figure 97. Then, we solve the equation 0.765 = 1 / (a*x). Therefore, the resulting actual function of the input graphics image is y = 1 / (0.00013 * x).

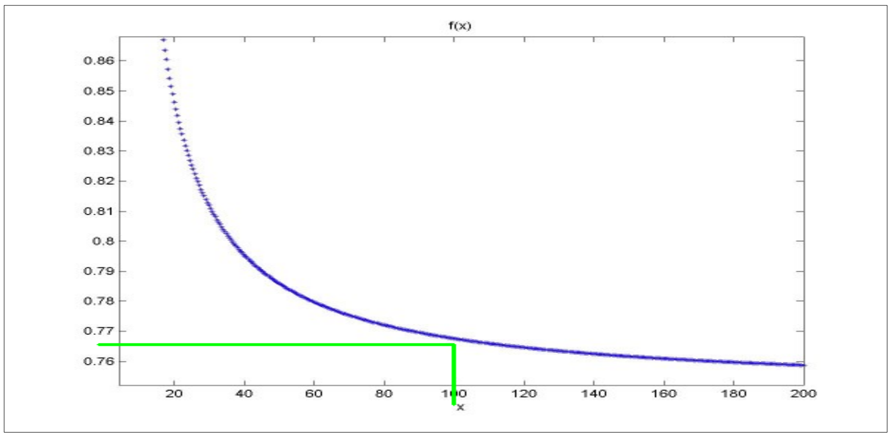


*Figure 97: The y position solutions of function y = 1 / (0.003 * x) for the x position of the x-axis value 100.*

## 5.5 Function Regeneration of Sinusoidal Curves

Similarly to the linear curve, we parse the graphics image that is illustrated in figure 98 through the aforementioned graphics processing methodology presented in the previous chapter. The results from the application of the pyramidal reduction, the erosion and the isolation of the curve using the HSV homogeneity filter are illustrated in figure 98. Then, the graphics recognition methodology proceeds with the computation of the 2D middle points that form the straight line segments for each curve in the graphics image. Figure 99 illustrates the results from the application of the maximum and minimum distance based hierarchical clustering algorithms, the computation of the elbow and the application of the K-means algorithm.

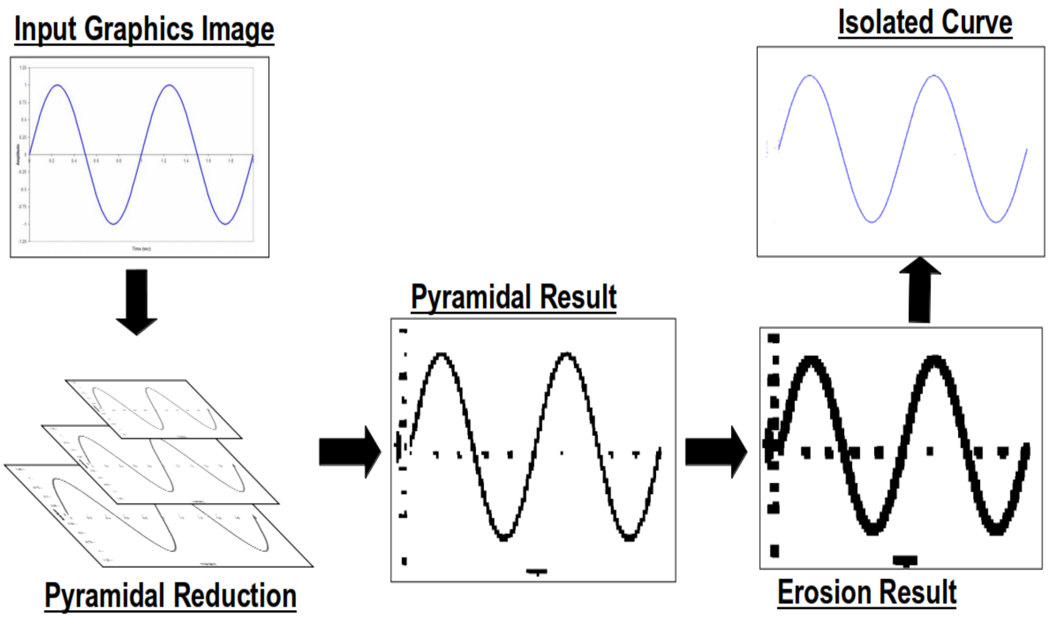


*Figure 98: Sinusoidal Curve Preprocessing Results.*

The graphics understanding methodology uses the deduced 2D middle points in order to construct the respective straight line segments of the curve and capture its monotonic information. The result from the recreated curve using the segments remaining after the 3rd merging rule are illustrated in figure 100. Finally, the function type recognition algorithm deduces that the curve is described by a polynomial function (y = a * sin(b * (x - c))). For this function type solving the equation for every remaining 2D middle will not result in possible approximations for the parameters a, b, c and d because they are essentially different metrics. Therefore, we apply computer vision and image processing techniques in order to find the solutions for the parameters.

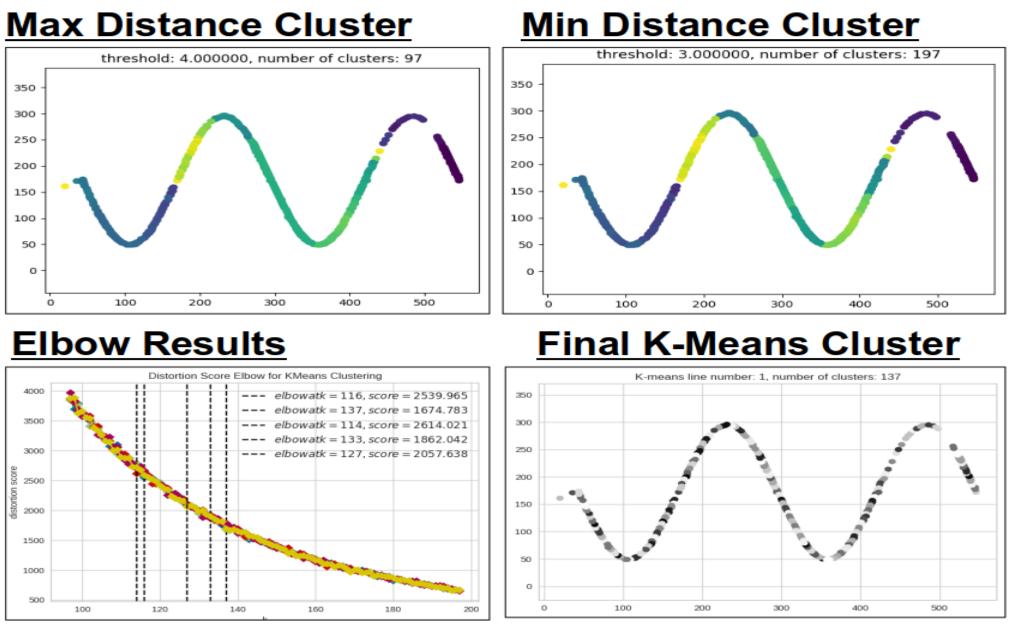


*Figure 99:* *Sinusoidal Curve Clusters Analysis Results.*

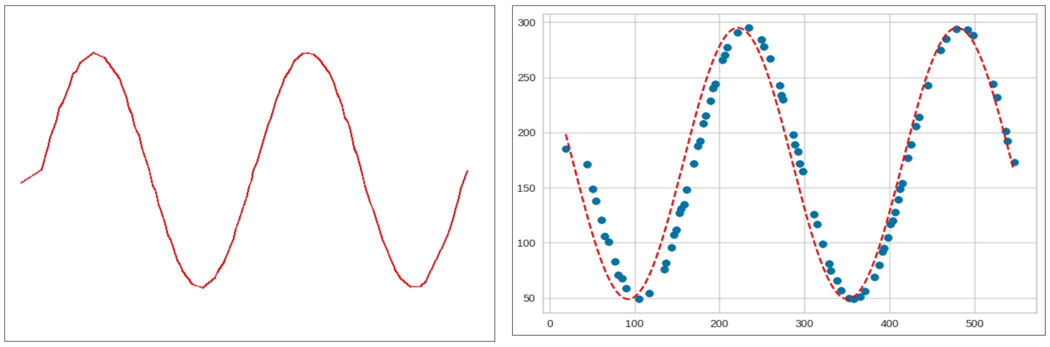


*Figure 100:* *The sinusoidal curve recreated from 2D middle points (left image) and from the deduced pixel function (right image).*

More specifically, parameters a stands for the amplitude of the curve, b stands fro the period, c for the phase shift and d for the midline. We compute the midline d parameter using the maximum and minimum positioned pixels from the curve's middle points in the 2D plain. Similarly, the amplitude a is the distance between the maximum point and the midline. We compute the period b by computing the frequency of the curve first. Then, the phase shift is calculated using the pixel positions of the start of the axis, the first maximum positioned data point of the curve and the intersection of the latter's projection with the midline. The resulting pixel based sinusoidal function is y = 123 * sin(0.02434 * (x - 157)) + 172. The result of the recreated curve using the deduced pixel based polynomial function is illustrated in figure 100.

Similarly to the previous section, we solve the deduced pixel based linear function using the x positions of the values 0.2 and 1.6 from the x-axis in order to deduce the actual function that is represented in the input graphics image. This outputs the y positions of the corresponding y-axis values. These results are 49.23 and 258.129 which correspond to the values 1 and -0.702 from the y-axis respectively. The results from the calculations of the pixel based function given these values as input are illustrated in figure 101. However, in this case we cannot create a system using sinusoidal equations because it's not allowed mathematically. Thus, we associated the previously computed parameters with their corresponding actual counterparts using the actual values from the axis legends. Therefore, the resulting actual function of the input graphics image is y = 1 * sin(3.14 * (x – 1.57)) + 0.

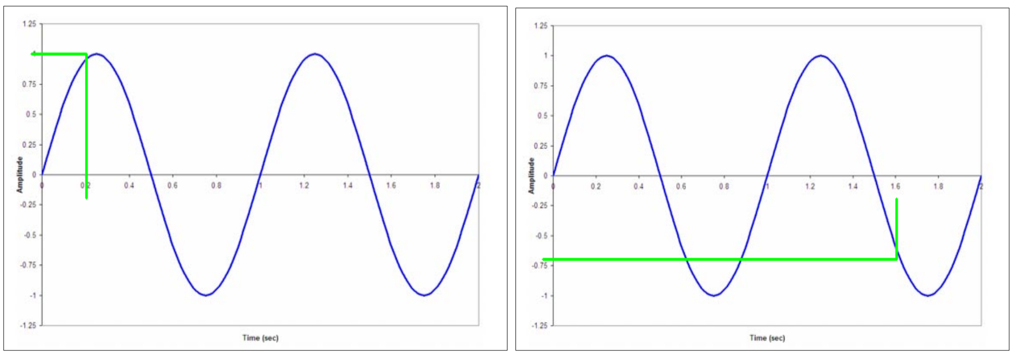


*Figure 101: The y position solutions of function y = 123 * sin(0.02434 * (x - 157)) + 172 for the x position of the x-axis value 0.2 (left) and x-axis value 1.6 (right) respectively.*

## 5.6 Function Regeneration of Arbitrary Curves

In case of very complex complex curves, which cannot be represented accurately by a single function we use a different approach in order to find the corresponding function. More specifically, in this case we use the middle points of the bars that can form conceptual behavioral curves. We use these middle points as the different ranges of values between which search for functions. In addition, we have to ensure that each one functions is also continuous with its prior and posterior function. We deal with this issue by employing the cubic Spline interpolation technique. This methodology requires that the first derivatives of two consecutive line segments are equal for a given x, where x is the value of their common point. So $f'_{i-1,i}(x_i) = f'_{i,i+1}(x_i)$. We also require that their respective second derivatives are also equal $f''_{i-1,i}(x_i) = f''_{i,i+1}(x_i) = k_i$. Only then, we can ensure continuity between consecutive curves, and by extend their functions. This concept is equivalent to maintaining the respective slopes and bending moments equal. Using this lemma we solve the produced equations and arrive into formulas (10) and (11). :

$f_{i,i+1}(x) = (k_i/6)\ [((x - x_{i+1})^3) / (x_i - x_{i+1}) - (x - x_{i+1})(x_i - x_{i+1})] - (k_{i+1}/6)\ [((x - x_{i+1})^3) / (x_i - x_{i+1}) - (x - x_i)(x_i - x_{i+1})] + (y_i(x - x_{i+1}) - y_{i+1}(x - x_i))/(x_i - x_{i+1})$ (10), $k_{i-1}(x_{i-1} - x_i) + 2k_i(x_{i-1} - x_{i+1}) + k_{i+1}\ (x_i - x_{i+1}) = 6[(y_{i-1} - y_i)/(x_{i-1} - x_i) - (y_i - y_{i+1})/(x_i - x_{i+1})]$ (11)

Figure 102 showcases an example of how we perceive the different cubic functions for a given curve, in order to compute the corresponding cubic Spline function. Therefore, a Spline is actually cubic curve consisting of other consecutive cubic curves all combined together. In order to compute the individual functions, we use the recognized middle points as input for the formula (10). In this way, we create a system of linear equations, which we solve using Gaussian elimination. This results in the possible $k_i$ values, which we then input to formula (11), in order to interpolate and get any y-axis values we need.

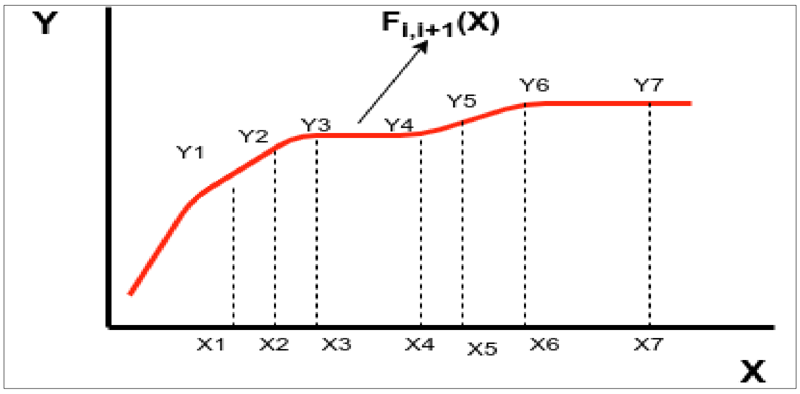


*Figure 102: Cubic Spline example consisting of consecutive cubic functions.*

We use the x-axis values of the pixels as input for the aforementioned formulas. Thus, the interpolant formula (11) outputs the  corresponding y-axis values for any x-axis value. Below we illustrate some examples of the functions that we deduced using cubic Splines for the first curve (red). We consider the values of the recognized middle points in the x-axis of the image as the initial input range points. Furthermore, we print these produced functions a combination of text and pseudocode format. The outcome sentences can later be used as input for other processing modules,

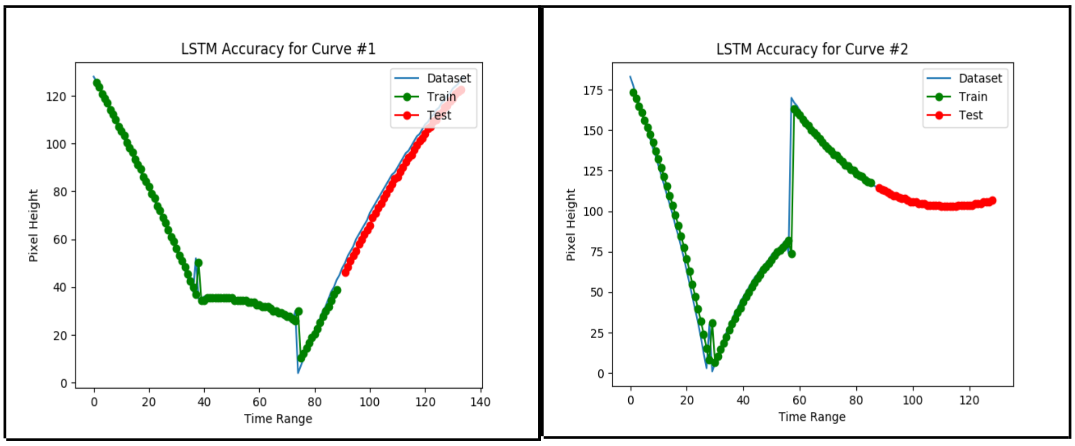
***<u>NLS1:</u> In range 80<=x<=189 the curve corresponds to function 0.004*x**2 - 1.169*x + 203.883***

***<u>NLS2:</u> In range 189<=x<=297 the curve corresponds to function -0.002*x**2 + 0.749*x – 41.632***

***<u>NLS3:</u> In range 297<=x<=515 the curve corresponds to function -0.005*x**2 + 3.597*x – 719.232***

As mentioned above, the extraction of the functions of the curves lead to a greater understanding of their internal functionality and behavior. Therefore, the next logical processing step is to use the collected data and try to predict the future behavior of each curve. So the goal is to try and get that dataset first. In the introduction we propose a reverse engineering methodology for the extraction and understanding of datasets from graphics images. That methodology depends on the initial recognition of the middle points. Figure 103 showcases the results of the cubic Spline interpolation procedure, with the recreated dataset of pixel positions colored blue. It is evident by the results, that the recreated dataset has multiple inaccuracies. We attribute these inaccuracies to the fact that there is no actual pattern behavior that we can recognize in these curves. That specific graphics image is just an example containing random curves without any actual meaning or goal. While cubic Splines are a handy methodology for data interpolation and thus dealing with missing data, it is not recommended to use them for extrapolation purposes. Therefore, we choose to use a standard LSTM neural network, which we train only on the reproduced dataset of y-axis values.

An LSTM neural network is a subtype of recurrent neural networks. It is capable of training in time series data and forecast their next values. We consider the problem of predicting future curve behavior as a time series problem, since a curve can evolve in any direction through the progression of time. More specifically, we train the LSTM on 70% of the interpolated dataset, and test it on the remaining 30%. The training lasts 100 epochs and has a batch size of 1. We consider a different LSTM model for each curve. Figure 103 illustrates the plotted datasets as well as the results of the training and testing prediction for each curve. We use the blue color for the dataset plotting, the green color to represent the training accuracy results, and red color for the testing accuracy results. It is evident by the prediction results, that the future predicted y-axis values almost overlap completely with the original. This proves that the after its training, the LSTM model is capable of producing accurate forecasting results for each curves future behavior. It is also proves the concept that we can reverse engineer a given graphics image and reproduce the dataset that was used to generate it.



*<u>Figure 103:</u> The results of the LSTM forecasting model. Left image corresponds to the first conceptual curve for red bars showing the accuracy results of the training process (green) and validation process (red). Similarly, right image corresponds to the second conceptual curve for the blue bars.*

As further proof for the LSTM model's forecasting accuracy, we have computed the corresponding Root Mean Square Error values. Since the dependent variable in this problem is the value y, it can range anywhere from 0 to the 403 which is the total height of the image. Table 7 illustrates the corresponding RMSE results. A smaller RMSE value with respect to the dependent variable value indicates that the experiment is successful. Thus, the small values illustrated in table 1 also imply that the forecasting of the LSTM model is accurate for both curves.

**Table VII: RMSE results of the LSTM forecasting model**

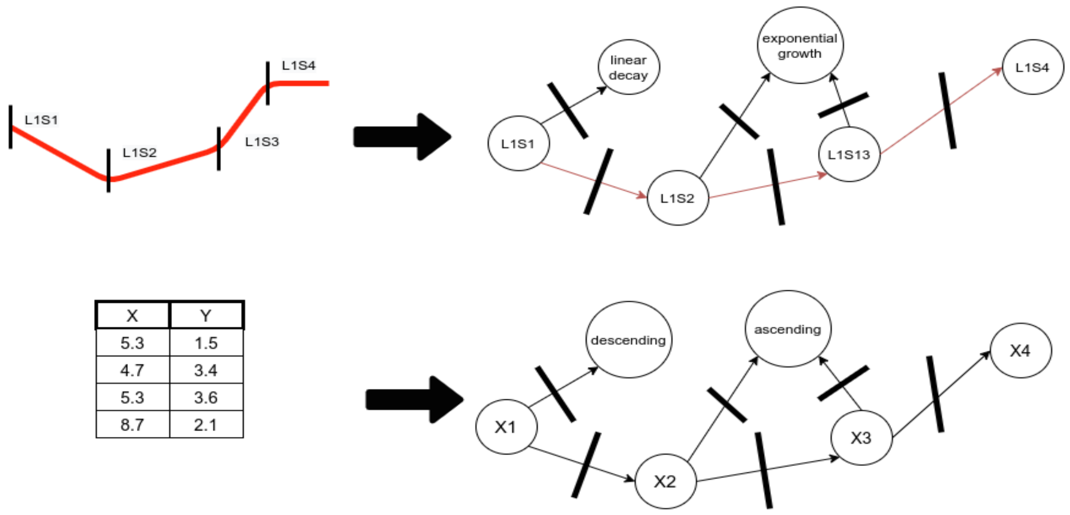
| | Train Accuracy RMSE | Test Accuracy RMSE |
|---|---|---|
| **Curve #1 (blue)** | 3.82 | 3.76 |
| **Curve #2 (red)** | 11.30 | 0.56 |

# 6. Synergistic Merging of Graphical and Tabular Information

## 6.1 Synergistic Merging General Methodology

The purpose of this research has been not only to capture and understand the underlying behavior that is contained in images of tables and graphics, but to also generate new additional knowledge. In the case of the table modality this is achieved through the validation of the generated relations among table variables and relational operators. These validated relations represent the upper and lower limits of the functionality that is illustrated in a given table image. Similarly, in graphics images we also understand the functional information of the curves by extracting their monotonic behavior. Using that information, we are able to generated new knowledge by computing the pixel based and actual value functions that describe them.

However, in case that we are given a table and a graphics image from the same document, we can combine their extracted functionalities and gain additional knowledge as well as a better understanding of the document itself. The presented synergistic merging methodology is achieved because of the previously generated SPN graphs that represent the extracted functionality from each respective modality. More specifically, the use of SPN graphs allows for the mathematical merging of the individual graphics and tables information based on the functions that describe them, however, without the requirement of any prior knowledge about these functions. It must be noted that the synergistic methodology requires that there is an underlying connection that needs to be found between the two given graphics and tables images. Otherwise, it cannot generate any results, since there isn't any associations between the images.



| X | Y |
|---|---|
| 5.3 | 1.5 |
| 4.7 | 3.4 |
| 5.3 | 3.6 |
| 8.7 | 2.1 |

*Figure 104: Examples of produced SPN graphs for tables and graphics.*

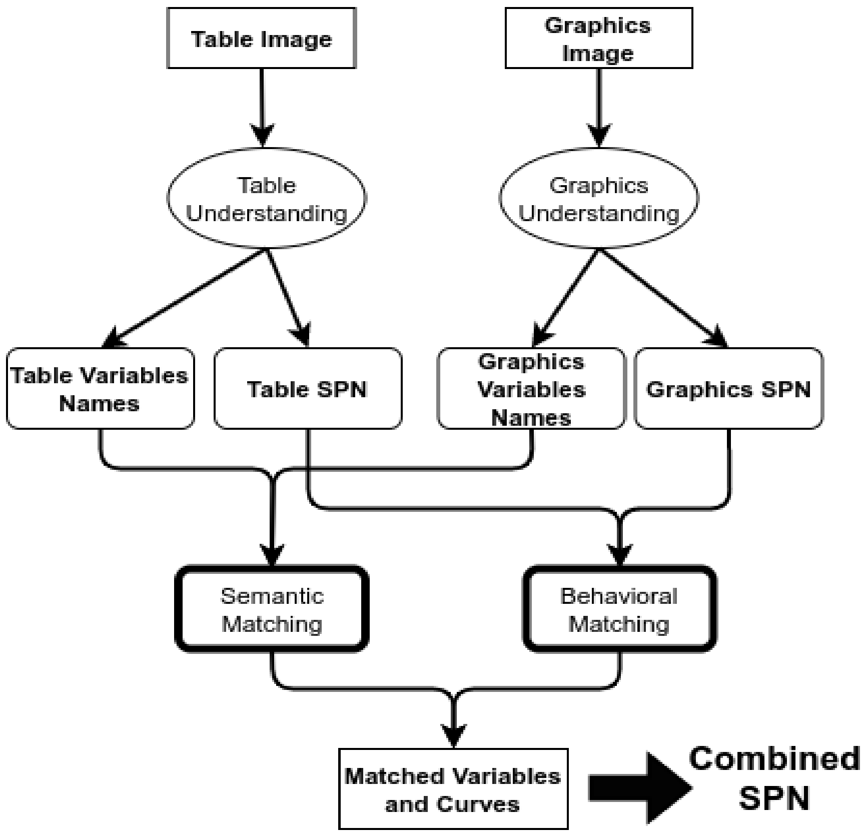


*Figure 105: The architecture of the synergistic merging methodology.*

The overall diagram of the presented synergistic merging methodology is illustrated in figure 105. The methodology receives the extracted results from the two input images (graphics and table) and then performs two types of matching. More specifically, semantic matching is performed using the recognized table variable names and graphics axis names. If there is a match among two variables . Otherwise, the methodology proceeds with the behavioral matching that is performed using the produced SPN graphs. More specifically, the methodology checks for similar patterns among the ascending or descending of table variable values and the growth or decay of the the given curve. Figure 104 demonstrates an example of a given curve and a table along with their corresponding SPN graphs. Since, there aren't any names available for the curve, the match is going to be performed using their extracted functionality. More specifically, the descending information from table variable X is matched with the decay information of the red curve. Similarly, there is a matching among the ascending and growth information from the table variable X and the red curve respectively. Thus, the two are matched based on their extracted behavioral information that are represented in their corresponding SPN graphs.

| Single Phase Power Parameters | | |
|---|---|---|
| Root Mean Square Voltage (Vrms) | Current (Irms) | Peak Voltage (Vpeak) |
| 115 | 5.8 | 162.63 |
| 115 | 5.8 | 0 |
| 115 | 5.8 | -162.63 |
| 208 | 3.3 | 294.16 |
| 208 | 3.3 | -294.16 |
| 230 | 2.9 | 325.27 |
| 230 | 2.9 | -325.27 |

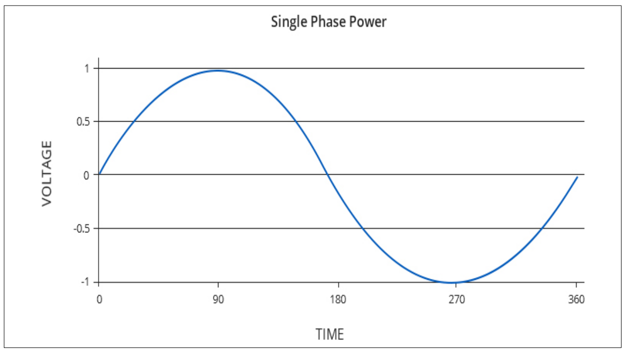
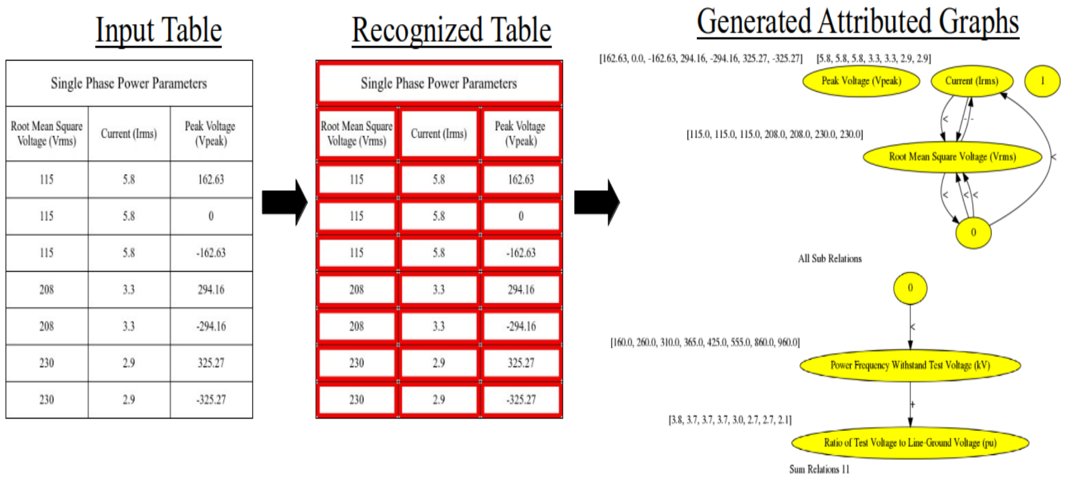


*Figure 106: The test cases for the synergistic merging methodology.*

Figure 106 demonstrates a table of power parameters from a single phase system. These parameters are the Root Mean Square Voltage, the Current and the Peak Voltage. Additionally, figure 106 presents the graphical representation of the AC voltage value in the span of time for the same single phase system.  as proof of concept. These two images are connected with an underlying association, which can be deduced after their individual analysis is performed. We are going to apply the previously presented tables and graphics understanding methodologies respectively to the corresponding images of figure 106. Then, we will use the aforementioned synergistic merging methodology on their results in order to prove the concept of the discussed idea based on an actual use case.

## 6.2 Tables Image Processing Results

In chapter 3 we present the overall tables image processing methodology which includes the table recognition and understanding methodologies. More specifically, the recognition methodology receives the table image as input and extract its variables and their corresponding values based on the location of the table cells. Then, the methodology generates all the possible permutations among the table variables and the special symbols from chapter 3. These permutations are validated both structurally and then semantically using the recognized values. The understanding methodology converts the validated relations are into attributed graphs that represent the structural information that is extracted from the table image. The results from each of the aforementioned processing steps for the table image of figure 107 are illustrated in figure 106. Figure 108 also showcases the resulting aggregated attributed graph that contains all the extracted and validated relations among the table variables.



| Single Phase Power Parameters | | |
|---|---|---|
| Root Mean Square Voltage (Vrms) | Current (Irms) | Peak Voltage (Vpeak) |
| 115 | 5.8 | 162.63 |
| 115 | 5.8 | 0 |
| 115 | 5.8 | -162.63 |
| 208 | 3.3 | 294.16 |
| 208 | 3.3 | -294.16 |
| 230 | 2.9 | 325.27 |
| 230 | 2.9 | -325.27 |

| Single Phase Power Parameters | | |
|---|---|---|
| Root Mean Square Voltage (Vrms) | Current (Irms) | Peak Voltage (Vpeak) |
| 115 | 5.8 | 162.63 |
| 115 | 5.8 | 0 |
| 115 | 5.8 | -162.63 |
| 208 | 3.3 | 294.16 |
| 208 | 3.3 | -294.16 |
| 230 | 2.9 | 325.27 |
| 230 | 2.9 | -325.27 |

*Figure 107: The table recognition process for the table of figure 106.*

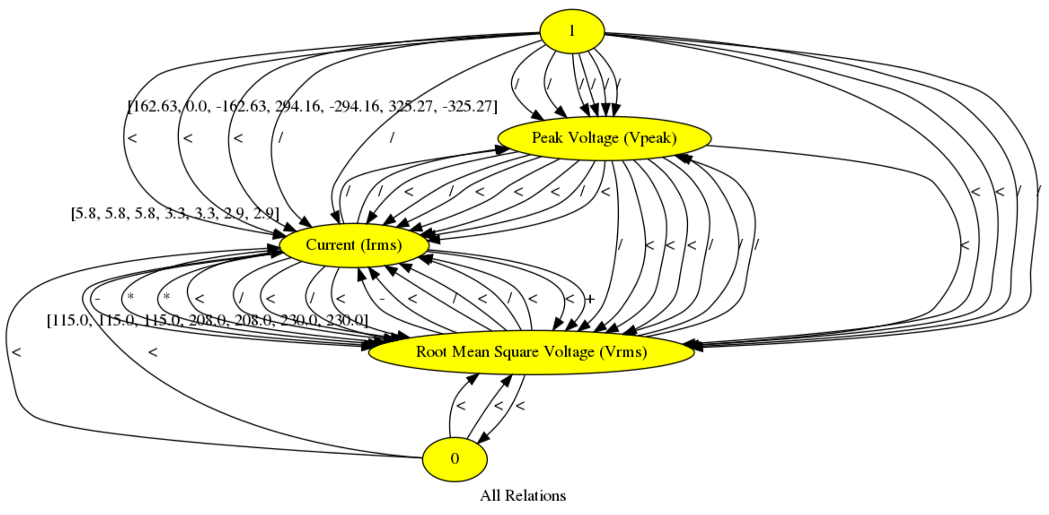


*Figure 108: Aggregated attributed graph for table image of figure 106.*

The aggregated graph from figure 108 is used in order to produce the natural language sentences that describe the information extracted from the given table image. As discussed in chapter 3, these sentences follow a predefined A-V-P format. We illustrate some of the resulting natural language sentences for the table from figure 106 below as an example. The sentence 49 contains the information about the ratio between the Peak Voltage (Vpeak) variable and the Root Mean Square Voltage (Vrms) variable. That ratio is computed to be 1.414. This specific result is important due to the fact that it corresponds to the actual inverse ratio between the theoretical Vrms and Vpeak values, which must always be the square root of 2. It also proves that we generate new knowledge that was not initially available from the given image. We don't capture its semantic meaning, however, that would go beyond the purposes of the current research.  Finally, the natural language sentences are converted into SPN kernels using the Pinakas language introduced in chapter 3. The final SPN graph that represents the understood tabular behavior is illustrated in figure 109.

***<u>NLS5:</u> zero is greater than values of Current (Irms) minus values of Root Mean Square Voltage (Vrms)***

***<u>NLS6:</u> values of Root Mean Square Voltage (Vrms) minus values of Current (Irms) are greater than zero***

**…**

***<u>NLS15:</u> values of Current (Irms) are greater than 1 divided by values of Root Mean Square Voltage (Vrms) divided by values of Peak Voltage (Vpeak)***

**…**

***<u>NLS49:</u> values of Peak Voltage (Vpeak) and values of Root Mean Square Voltage (Vrms) have a ratio of 1.414***

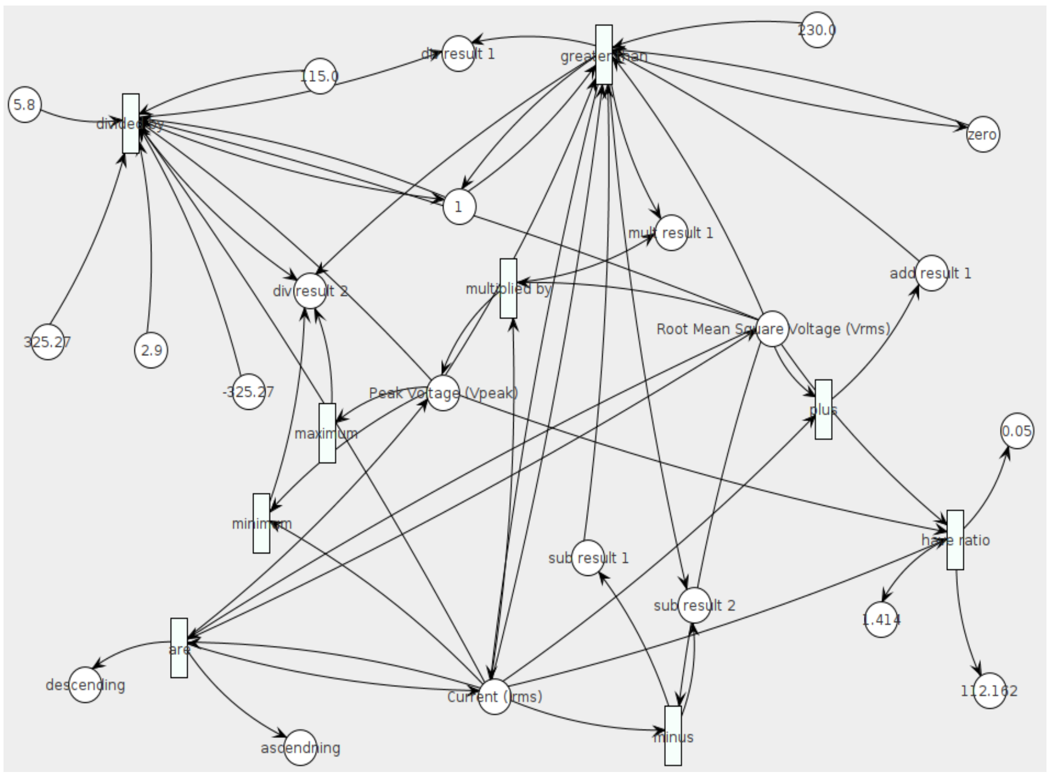


*Figure 109: The SPN graph of the table image from figure 106.*

## 6.3 Graphics Image Processing Results

In chapter 3 we present the overall graphics image processing methodology which includes the table recognition and understanding methodologies. More specifically, the recognition methodology receives the graphics image as input and extract its axis information and the line segments of the illustrated curves. More specifically, the methodology starts with the application of the pyramidal reduction, the erosion and the isolation of the curve using the HSV homogeneity filter. The respective preprocessing results to the graphics image from figure 106 are are illustrated in figure 110. Then, the methodology detects the 2D middle points that constitute the straight line segments for the curve of the input graphics image. Figure 111 illustrates the results from the application of the maximum and minimum distance based hierarchical clustering algorithms, the computation of the elbow and the application of the K-means algorithm.

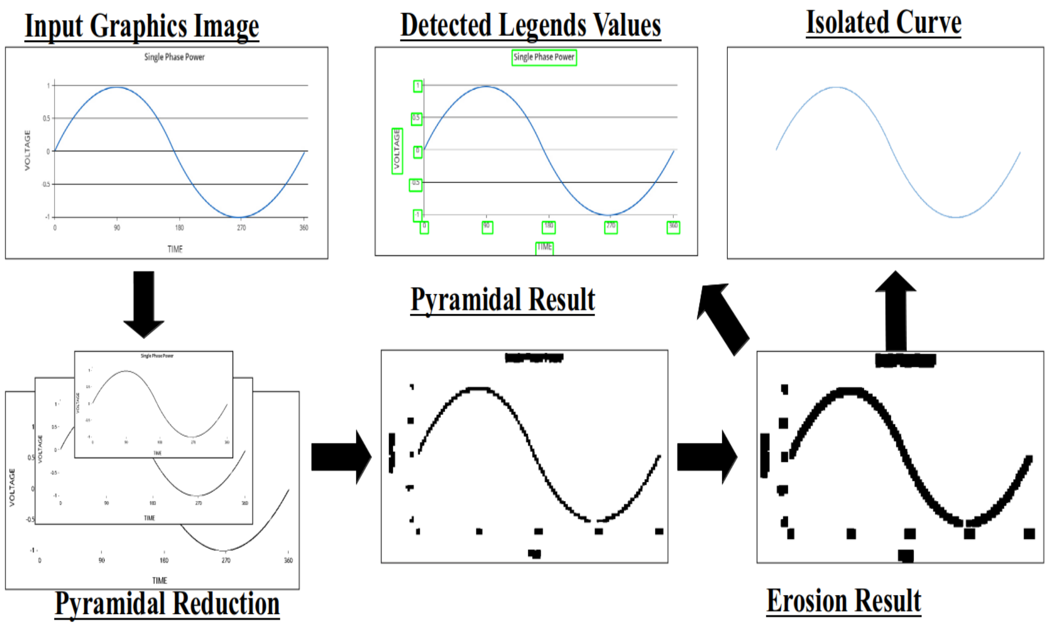


*Figure 110: Preprocessing results of input graphics image from figure 106.*

The graphics understanding methodology uses the deduced 2D middle points in order to construct the respective straight line segments of the curve and capture its monotonic information. The result from the recreated curve using the segments remaining after the 3rd merging rule are illustrated in figure 112. Finally, the function type recognition algorithm deduces that the curve is described by a sinusoidal function (y = a * sin(b * (x - c))). Therefore, we apply computer vision and image processing techniques in order to find the approximations for the parameters a, b, c and d. More specifically, we use the sinusoidal function regeneration methodology which we described in chapter 5. The resulting pixel based sinusoidal function is y = 144 * sin(0.00997 * (x – 3887.32)) + 184. The result of the recreated curve using the deduced pixel based polynomial function is illustrated in figure 112.

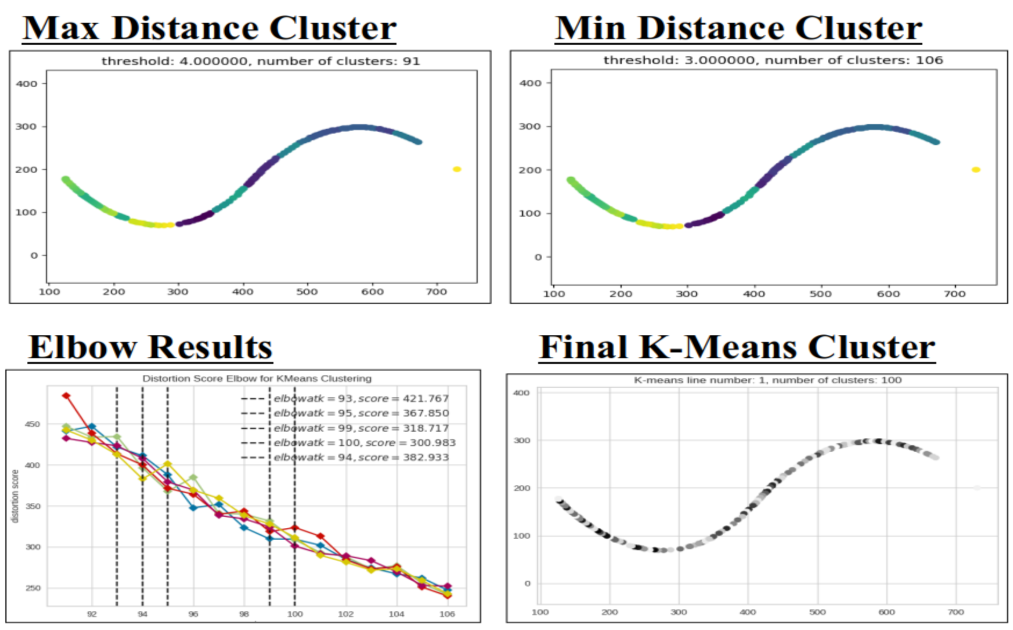


*Figure 111: Clusters analysis results of input graphics image from figure 106.*

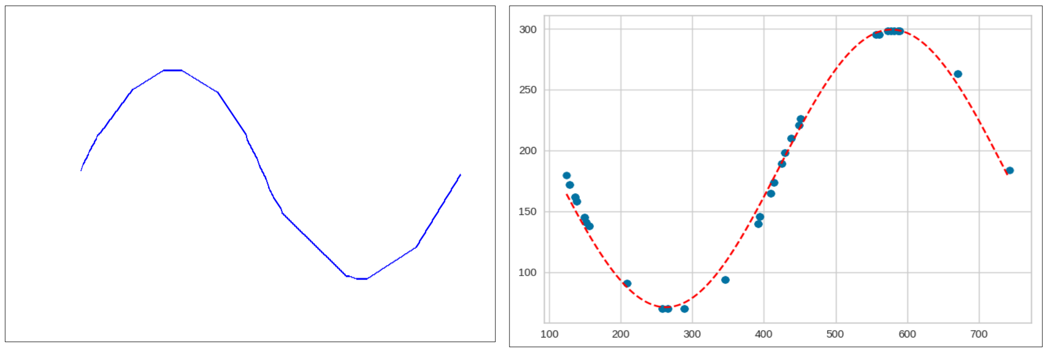


*Figure 112: The sinusoidal curve recreated from 2D middle points (left image) and from the deduced pixel function (right image).*

As proof for the efficiency the pixel based function, we solve it using the x positions of the values 0 and 270 from the x-axis in order to deduce the actual function that is represented in the input graphics image. This outputs the y positions of the corresponding y-axis values which are 0.2 and 1 respectively. The results from the calculations of the pixel based function given these values as input are illustrated in figure 113. Then, we proceed to compute the actual parameters of the function based on the methodology described again in chapter 5 for sinusoidal function regeneration. The resulting actual function of the input graphics image is y = 1 * sin(0.01711 * (x – 1.372)) + 0.

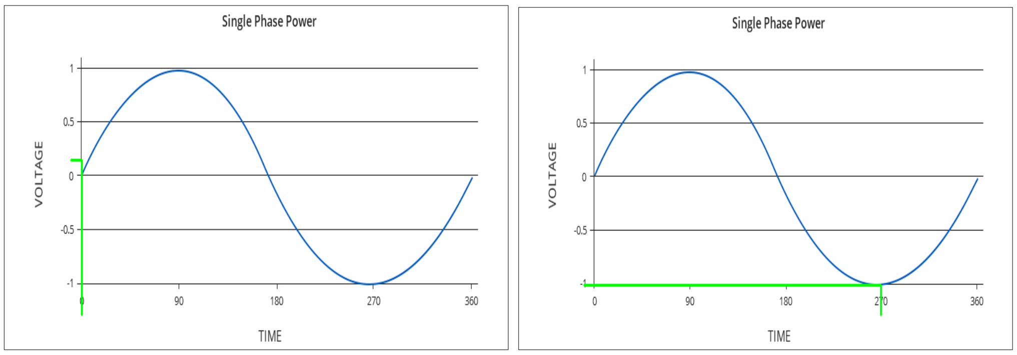


*Figure 113: The y position solutions of function* y = 1 * sin(0.01711 * (x – 1.372)) + 0 *for the x position of the x-axis value 0 (left) and x-axis value 270 (right) respectively.*

Bellow we present a sample of the overall produced natural language sentences that are generated from the extracted information from the graphics image of figure 106. As discussed in chapter 3, these sentences follow a predefined A-V-P format. As demonstrated in sentence 40, we also maintain the information regarding the understood functions that describe a curve's functionality. Finally, the natural language sentences are converted into SPN kernels using the Kyrtos language introduced in chapter 4. The final SPN graph that represents the understood graphics behavior is illustrated in figure 114.

***NLS7: the straight-line segment L1S6 is connected with (147.00220336262106 degrees) to the straight-line segment L1S7***

***NLS8: the straight-line segment L1S7 is connected with (114.83829451548637 degrees) to the straight-line segment L1S8***

**…**

***NLS26: Line segment L1S1 illustrates linear growth greater than 0.5 Single Phase VOLTAGE % during 0 TIME***

**…**

***NLS40: Pixel TIME is input to y = 114.00000 * sin(0.00997 * (x - 3887.32000)) + 184.00000 and outputs Pixel Single Phase VOLTAGE***

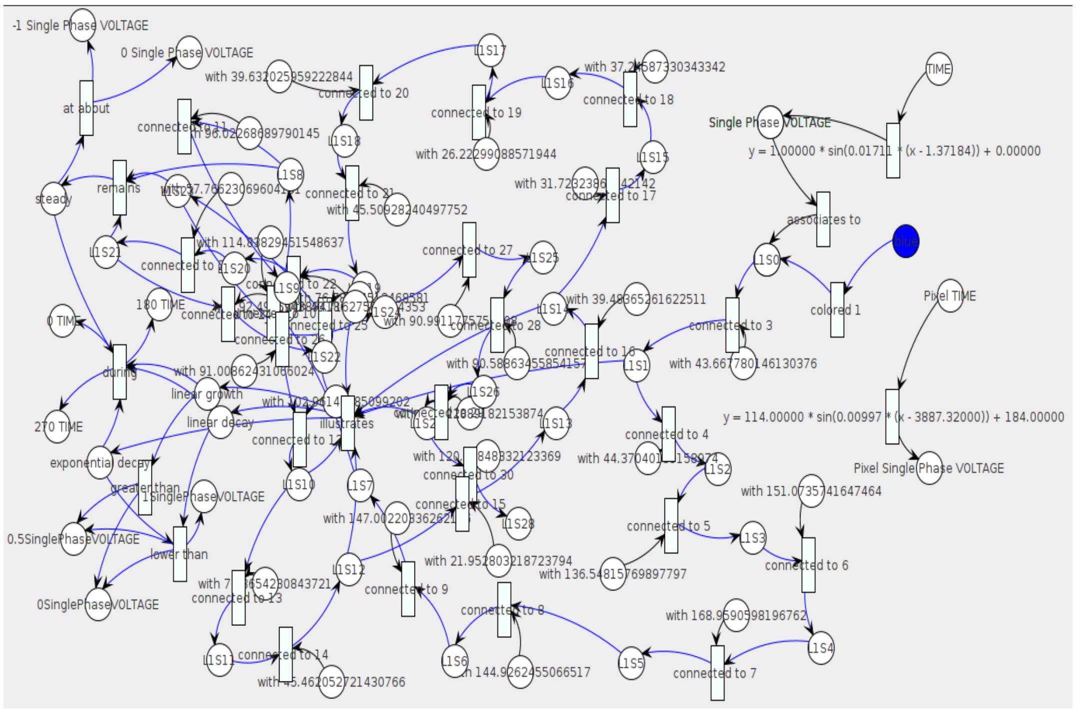
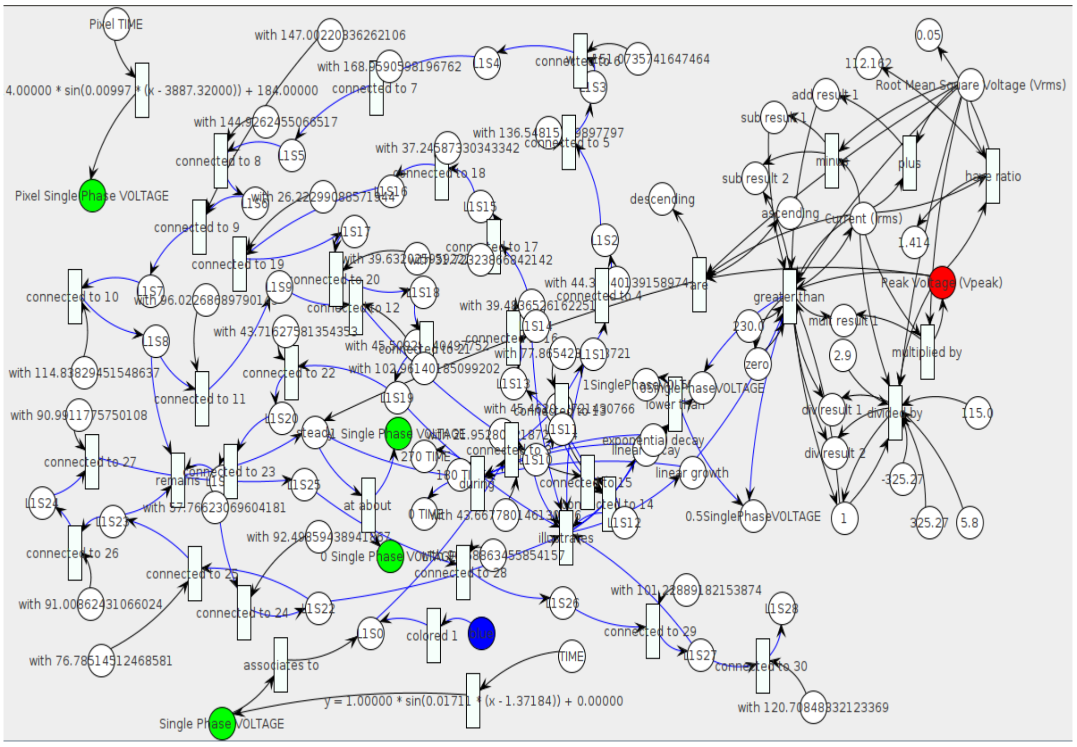


*Figure 114: The corresponding SPN from graphics image of figure 106.*

## 6.4 SPN-based Combination of Information

Finally, we present the results from the synergistic merging of the SPN graphs that correspond to the table and graphics images from figure 106 respectively. As we have mentioned in section 6.1, there isn't a clear correspondence among the table variable names and the axis name. Thus, the synergistic matching methodology proceeds to the step of behavioral matching with the use of the generated SPN graphs. More specifically, the methodology will recognize the correlation between the table variable Vpeak and the curve due to their matching fluctuation in values. This is a correct assumption since the two variable essentially describe the same information. After the matching is completed, the methodology initiates the combination of the two SPN graphs. Figure 115 illustrates the first step of the combination, with the curve's matched named colored green and the table variable's name colored red. Figure 116 shows the next step of the combination procedure, where both names are colored red. Finally, figure 117 illustrates the last step of the process, where the curve's name has been converted into the same name from the table variable.



*<u>Figure 115:</u> First step of merging the SPN graphs*

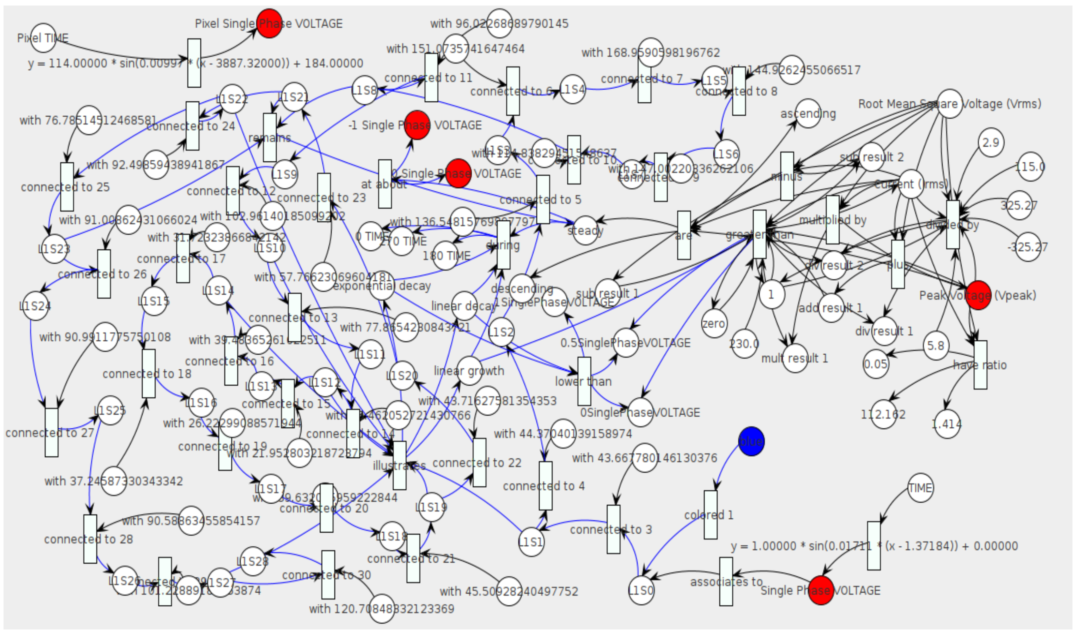


*Figure 116:* *Second step of merging the SPN graphs.*

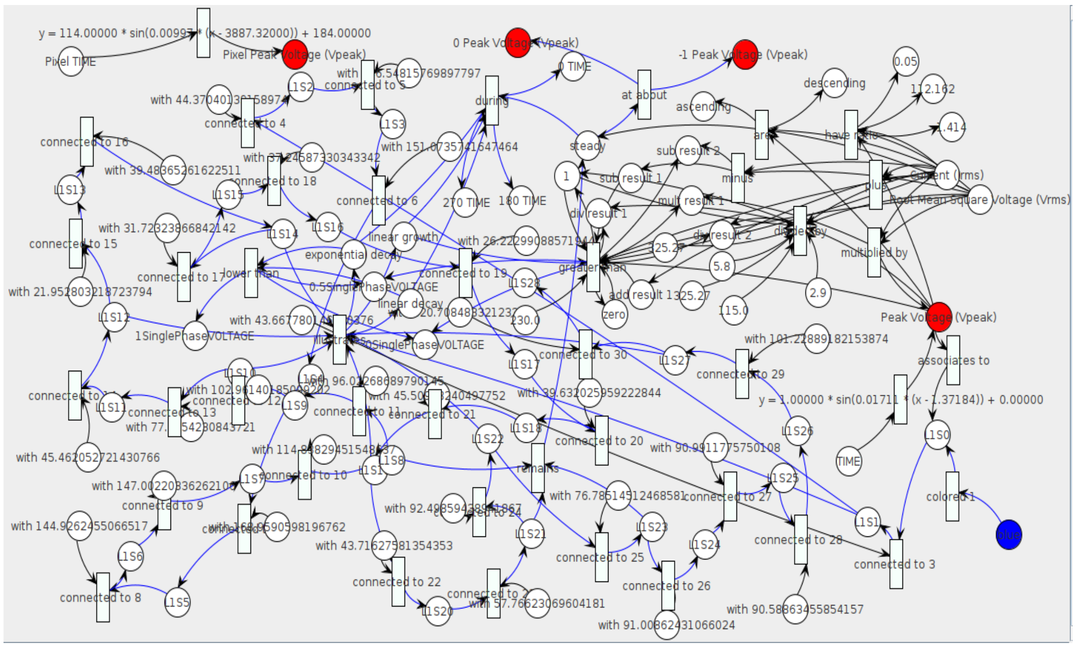


*Figure 117:* *Final step of merging the SPN graphs.*

# 7. Conclusions

In conclusion, we have presented a complete methodology for the detection, recognition and understanding of tables and graphics images in technical documents. The tables processing methodology covers 3 different types of tables which are distinguished based on their structural features. Similarly, the graphics processing methodology can analyze 3 different types of graphics. These are graphics images containing 1) segmented or continuous curves of different color, 2) segmented curves of the same color and 3) bars. The extracted knowledge from both tables and graphics is represented using SPN graphs. Additionally, we present a methodology for the combination of the retrieved information from tables and graphics of the same document based on their SPN graphs. This SPN based synergistic merging method enables us to combine the information of tables and graphics using the functions that describe, without requiring any prior knowledge about them.

The presented method accepts images from modalities of technical documents as input and categorizes them into tables and graphics images. The tables are parsed through a classification algorithm that categorizes them into three different types. Each table type follows a unique processing path in order recognize and understand its contents accurately. Initially, the recognition module locates the table title and the table variables. The tables variables are associated with their corresponding values of on their location and the table type.

Then, we produce all possible permutations among the table variables and specific relational operators. The generated permutations are validated both syntactically and semantically. The validated relations are converted to attributed graphs in order to preserve the structural information of the table. The attribute graph relations are translated into natural language sentences, which are then combined into an SPN graph. The Pinakas formal language is introduced in order to map the natural language sentences into the SPN kernels.

The detected graphics  images are parsed into the graphics recognition module. More specifically, this module extracts the text information of this axis and then isolates the curves from the original input image. The presented methodology proceeds to compute the 2D middle points which delimit the line segments that form each curve. These data points are computed using a combination of clustering techniques iteratively. The graphics understanding methodology filters these 2D middle points through different merging rules in order to preserve only the essential line segments. These line segments are used to deduce both structural and functional information of about the illustrated curves such as intersections, parallelism and monotony. Furthermore, we recognize the type of function that describes each curve's behavior using the aforementioned monotonic information. Different processing steps are performed to compute both the pixel based and the actual value functions, based on their type. Finally, we generate the natural language sentences that contain all the understood structural and functional information. Then, we use the presented Kyrtos formal language in order to map the natural language sentences into the SPN kernels . The produced SPN graph describes the input graphics image underlying functionality.

The purpose of this research has been both to understand the underlying knowledge that is illustrated in images of technical document modalities, as well as to generate new knowledge if possible. The experiment results from chapter 6 prove that the presented methodology achieves this goal, by combining successfully the functional information that are extracted from input table and graphics images of the same document into a synergistic SPN graph. That synergistic SPN graph now holds information from both modalities and can associate them using their matched variable and curve names.

# 8. Future Work

As part future work, the primary goal is to improve both the accuracy and the processing speed of the presented methodologies. More specifically, while the graphics recognition and understanding methodology returns correct results, they could improve in terms of detecting accurately the positions of curve's middle points. The accuracy of table and graphics understanding also depends on the relations that we are validating for. Additional and more complex relations can contribute to the overall accuracy of the extracted functionality.

Furthermore, we have found that the computational complexity of the tabular processing methodology depends directly on the number of table's variables and operator symbols. That is because we generate all the possible permutations among them, before preserving those that are validated. Similarly, the graphics processing methodology's computational complexity depends on the size of the image. That is because we parse all the pixels multiple times in order to get as much accurate results as possible. Therefore, the optimization of both methodologies is important in order to improve the overall speed. Finally, we plan on testing the methodology on more complex tables and graphics in order to expand the methodology to capture more advanced knowledge.

# Bibliography

[1] N. Bourbakis, “Converting Diagrams, Symbols, Formulas, Tables and Graphics into SPN and NL Text sentences for Automatic Deep Understanding of Technical Documents”, Int. IEEE Conf. on ICTAI, Boston, MA, Nov. 2017

[2] Nikolaos G. Bourbakis, J. Sukarno Mertoguno, “A Holistic Approach for Automatic Deep Understanding and Protection of Technical Documents”, Int. J. Artif. Intell. Tools 29(6): 2050007:1-2050007:39 (2020)

[3] M. S. Alexiou and N. Bourbakis, "Automatic Deep Understanding of Tables in Technical Documents," 2020 IEEE 32nd International Conference on Tools with Artificial Intelligence (ICTAI), Baltimore, MD, USA, 2020, pp. 443-451, doi: 10.1109/ICTAI50040.2020.00076.

[4] N. Siegel, N. Lourie, R. Power, and W. Ammar, “Extracting scientific figures with distantly supervised neural networks,” in Proceedings of the 18th ACM/IEEE on joint conference on digital libraries. ACM, 2018, pp. 223–232.

[5] I. Kavasidis, S. Pallazo, C. Spampinato, C. Pino, D. Fiordano, D Giuffrida, and P. Messina, “A Saliency-Based Convolutional Neural Network for Table and Chart Detection in Digitized Documents” In: Ricci E., Rota Bulò S., Snoek C., Lanz O., Messelodi S., Sebe N. (eds) Image Analysis and Processing – ICIAP 2019. ICIAP 2019. Lecture Notes in Computer Science, vol 11752. Springer, Cham

[6] P. Li, X. Jiang, and H. Shatkay, “Figure and caption extraction from biomedical documents,” Bioinformatics, 2019.

[7] https://www.xpdfreader.com/

[8] S. R. Choudhury, S. Wang, and C. L. Giles, “Scalable algorithms for scholarly figure mining and semantics.” in SBD@ SIGMOD, 2016, p. 1

[9] S. Ray Choudhury, P. Mitra, and C. L. Giles, “Automatic extraction of figures from scholarly documents,” in Proceedings of the 2015 ACM Symposium on Document Engineering. ACM, 2015, pp. 47–50.

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

[11] S. Ray Choudhury and C. L. Giles, “An architecture for information extraction from figures in digital libraries,” in Proceedings of the 24th International Conference on World Wide Web. ACM, 2015, pp. 667–672.

[12] P. De, “Automatic data extraction from 2d and 3d pie chart images,” in 2018 IEEE 8th International Advance Computing Conference (IACC). IEEE, 2018, pp. 20–25.

[13] A. G. Howard et al., “MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications,” Apr. 2017.

[14] D. Chester and S. Elzer, “Getting computers to see information graphics so users do not have to,” Foundations of Intelligent Systems, pp. 33–80, 2005.

[15] A. Balaji, T. Ramanathan, and V. Sonathi, “Chart-text: A fully automated chart image descriptor,” arXiv preprint arXiv:1812.10636, 2018.

[16] Fischler, M.A., Bolles, R.C.: Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. Communications of the ACM 24(6), 381–395 (1981)

[17] R. A. Al-Zaidy and C. L. Giles, “A machine learning approach for semantic structuring of scientific charts in scholarly documents,” in AAAI, 2017, pp. 4644–4649

[18] A. Kay, Tesseract: An Open-Source Optical Character Recognition Engine. 2007.

[19] X. Lu, S. Kataria, W. J. Brouwer, J. Z. Wang, P. Mitra, and C. L. Giles, “Automated analysis of images in documents for intelligent document search,” International Journal on Document Analysis and Recognition (IJDAR), vol. 12, no. 2, pp. 65–81, 2009.

[20] S. Kataria, W. Browuer, P. Mitra, and C. L. Giles, “Automatic extraction of data points and text blocks from 2-dimensional plots in digital documents.” in AAAI, vol. 8, 2008, pp. 1169–1174.

[21] R. R. Nair, N. Sankaran, I. Nwogu, and V. Govindaraju, “Automated analysis of line plots in documents,” in Document Analysis and Recognition (ICDAR), 2015 13th International Conference on. IEEE, 2015, pp. 796–800

[22] 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.

[23] Ester, Martin; Kriegel, Hans-Peter; Sander, Jörg; Xu, Xiaowei (1996). Simoudis, Evangelos; Han, Jiawei; Fayyad, Usama M. (eds.). A density-based algorithm for discovering clusters in large spatial databases with noise. Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96). AAAI Press. pp. 226–231.

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

[25] R. Burns, S. Carberry, and S. Elzer Schwartz, “An automated approach for the recognition of intended messages in grouped bar charts,” Computational Intelligence, vol. 35, no. 4, pp. 955–1002, 2019.

[26] C. F. Greenbacker, P. Wu, S. Carberry, K. F. McCoy, and S. Elzer, “Abstractive summarization of line graphs from popular media,” in Proceedings of the Workshop on Automatic Summarization for Different Genres, Media, and Languages.

[27] E. Kim and K. F. McCoy, “Multimodal deep learning using images and text for information graphic classification,” in Proceedings of the 20th International ACM SIGACCESS Conference on Computers and Accessibility. ACM, 2018, pp. 143–148.

[28] Pierre Sermanet, David Eigen, Xiang Zhang, Michael Mathieu, Rob Fergus, and Yann LeCun. 2014. OverFeat: Integrated Recognition, Localization and Detection using Convolutional Networks. In ICLR.

[29] K. Davila, S. Setlur, D. Doermann, U. K. Bhargava and V. Govindaraju, "Chart Mining: A Survey of Methods for Automated Chart Analysis," in IEEE Transactions on Pattern Analysis and Machine Intelligence, doi: 10.1109/TPAMI.2020.2992028.

[30] Freeman, H.: Computer processing of line-drawing images. ACM Comput. Surv. 6(1), 57–97 (1974)

[31] A. Gilani, S. R. Qasim, I. Malik and F. Shafait, "Table Detection Using Deep Learning," 2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR), Kyoto, 2017, pp. 771-776, doi: 10.1109/ICDAR.2017.131.

[32] A. Shahab, “Table ground truth for the UW3 and UNLV datasets,” [Online; accessed 7-April-2017]. [Online].Available: http://www.iapr-tc11.org/mediawiki/index.php title=Table_Ground_Truth_for_the_UW3_and_UNLV_datasets

[33] Abbyy. (2017) OCR SDK engine. [Online]. Available: https://www.abbyy.com/en-eu/ocr-sdk/

[34] Zeiler M.D., Fergus R. (2014) Visualizing and Understanding Convolutional Networks. In: Fleet D., Pajdla T., Schiele B., Tuytelaars T. (eds) Computer Vision – ECCV 2014. ECCV 2014. Lecture Notes in Computer Science, vol 8689. Springer, Cham

[35] T. T. Anh, N. In-Seop and K. Soo-Hyung, "A hybrid method for table detection from document image," 2015 3rd IAPR Asian Conference on Pattern Recognition (ACPR), Kuala Lumpur, 2015, pp. 131-135, doi: 10.1109/ACPR.2015.7486480.

[36] Aly Teppi, Na In and Kim Seong. (2015). Hybrid page segmentation using multilevel homogeneity structure. ACM IMCOM 2015 - Proceedings. 10.1145/2701126.2701138.

[37] Chang, Fu et al. “A linear-time component-labeling algorithm using contour tracing technique.” Comput. Vis. Image Underst. 93 (2004): 206-220.

[38] B. Gatos, D. Danatsas, I. Pratikakis, and S. Perantonis, “Automatic table detection in document images,” Pattern recognition and data mining, pp. 609–618, 2005.

[39] Yefeng Zheng, Changsong Liu, Xiaoqing Ding and Shiyan Pan, "Form frame line detection with directional single-connected chain," Proceedings of Sixth International Conference on Document Analysis and Recognition, Seattle, WA, USA, 2001, pp. 699-703, doi: 10.1109/ICDAR.2001.953880.

[40] Y. Wang, I. Phillip, and R. Haralick, “Automatic table ground truth generation and a background-analysis-based table structure extraction method,” in Document Analysis and Recognition, 2001. Proceedings. Sixth International Conference on. IEEE, 2001, pp. 528–532.

[41] J. Redmon and A. Farhadi, “Yolov3: An incremental improvement,” arXiv preprint arXiv:1804.02767, 2018.

[42] Huang, Y., Yan, Q., Li, Y., Chen, Y., Wang, X., Gao, L., & Tang, Z. (2019). A YOLO-Based Table Detection Method. 2019 International Conference on Document Analysis and Recognition (ICDAR), 813-818.

[43] J. H. Shamilian, H. S. Baird and T. L. Wood, "A retargetable table reader," Proceedings of the Fourth International Conference on Document Analysis and Recognition, Ulm, Germany, 1997, pp. 158-163 vol.1, doi: 10.1109/ICDAR.1997.619833.

[44] Siddiqui, S.A., Malik, M.I., Agne, S., Dengel, A., & Ahmed, S. (2018). DeCNT: Deep Deformable CNN for Table Detection. IEEE Access, 6, 74151-74161.

[45] Tsung-Yi Lin, Piotr Dollar, Ross Girshick, Kaiming He, Bharath Hariharan and Serge Belongie "Feature Pyramid Networks for Object Detection", arXiv preprint arXiv:1612.03144v2, 2017

[46] S. Schreiber, S. Agne, I. Wolf, A. Dengel and S. Ahmed, "DeepDeSRT: Deep Learning for Detection and Structure Recognition of Tables in Document Images," 2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR), Kyoto, 2017, pp. 1162-1167, doi: 10.1109/ICDAR.2017.192.

[47] J. Long, E. Shelhamer and T. Darrell, "Fully convolutional networks for semantic segmentation," 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Boston, MA, 2015, pp. 3431-3440, doi: 10.1109/CVPR.2015.7298965.

[48] Simonyan, K., & Zisserman, A. (2015). "Very Deep Convolutional Networks for Large-Scale Image Recognition." CoRR, abs/1409.1556.

[49] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C. Y. Fu, A. C. Berg (2016) "SSD: Single Shot MultiBox Detector." In: Leibe B., Matas J., Sebe N., Welling M. (eds) Computer Vision – ECCV 2016. ECCV 2016. Lecture Notes in Computer Science, vol 9905. Springer, Cham

[50] S. Tupaj, Z. Shi and D.H. Chang, (1996). "Extracting Tabular Information From Text Files."

[51] http://www.icst.pku.edu.cn/cpdp/ICDAR2017_PODCompetition/index.html

[52] R. Saha, A. Mondal and C. V. Jawahar, "Graphical Object Detection in Document Images," 2019 International Conference on Document Analysis and Recognition (ICDAR), Sydney, Australia, 2019, pp. 51-58, doi: 10.1109/ICDAR.2019.00018.

[53] K. He, G. Gkioxari, P. Dollár and R. Girshick, "Mask R-CNN," 2017 IEEE International Conference on Computer Vision (ICCV), Venice, 2017, pp. 2980-2988, doi: 10.1109/ICCV.2017.322.

[54] S. Ren, K. He, R. Girshick and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 39, no. 6, pp. 1137-1149, 1 June 2017, doi: 10.1109/TPAMI.2016.2577031.

[55] J. Deng, W. Dong, R. Socher, L. Li, Kai Li and Li Fei-Fei, "ImageNet: A large-scale hierarchical image database," 2009 IEEE Conference on Computer Vision and Pattern Recognition, Miami, FL, 2009, pp. 248-255, doi: 10.1109/CVPR.2009.5206848.

[56] A. Laurentini and P. Viada, "Identifying and understanding tabular material in compound documents," Proceedings., 11th IAPR International Conference on Pattern Recognition. Vol.II. Conference B: Pattern Recognition Methodology and Systems, The Hague, Netherlands, 1992, pp. 405-409, doi: 10.1109/ICPR.1992.201803.

[57] T. Kasar, P. Barlas, S. Adam, C. Chatelain and T. Paquet, "Learning to Detect Tables in Scanned Document Images Using Line Information," 2013 12th International Conference on Document Analysis and Recognition, Washington, DC, 2013, pp. 1185-1189, doi: 10.1109/ICDAR.2013.240.

[58] M. A. C. A. Jahan and R. G. Ragel, "Locating tables in scanned documents for reconstructing and republishing," 7th International Conference on Information and Automation for Sustainability, Colombo, 2014, pp. 1-6, doi: 10.1109/ICIAFS.2014.7069552.

[59] Hu Jianying, Kashi Ram, Lopresti Daniel and Wilfong Gordon. (1999). Medium-Independent Table Detection.

[60] E. Green and M. Krishnamoorthy, "Model-based analysis of printed tables," Proceedings of 3rd International Conference on Document Analysis and Recognition, Montreal, Quebec, Canada, 1995, pp. 214-217 vol.1, doi: 10.1109/ICDAR.1995.598979.

[61] L. A. Pereira Neves and J. Facon, "Methodology of automatic extraction of table-form cells," Proceedings 13th Brazilian Symposium on Computer Graphics and Image Processing (Cat. No.PR00878), Gramado, Brazil, 2000, pp. 15-21, doi: 10.1109/SIBGRA.2000.883888.

[62] D. He, S. Cohen, B. Price, D. Kifer and C. L. Giles, "Multi-Scale Multi-Task FCN for Semantic Page Segmentation and Table Detection," 2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR), Kyoto, 2017, pp. 254-261, doi: 10.1109/ICDAR.2017.50.

[63] S. Deivalakshmi, K. Chaitanya and P. Palanisamy, "Detection of table structure and content extraction from scanned documents," 2014 International Conference on Communication and Signal Processing, Melmaruvathur, 2014, pp. 270-274, doi: 10.1109/ICCSP.2014.6949843.

[64] X. Li, F. Yin and C. Liu, "Page Object Detection from PDF Document Images by Deep Structured Prediction and Supervised Clustering," 2018 24th International Conference on Pattern Recognition (ICPR), Beijing, 2018, pp. 3627-3632, doi: 10.1109/ICPR.2018.8546073.

[65] S. Mandal, S. Chowdhury, A. Das and B. Chanda (2006). “Simple and effective table detection system from document images. International Journal on Document Analysis and Recognition.” 8. 172-182. 10.1007/s10032-005-0006-5.

[66] D. Tran, T. Aly, A. Oh, S. Kim and I. Na (2015). “Table Detection from Document Image using Vertical Arrangement of Text Blocks.” International Journal of Contents. 11. 77-85. 10.5392/IJoC.2015.11.4.077.

[67] S. Arif and F. Shafait, "Table Detection in Document Images using Foreground and Background Features," 2018 Digital Image Computing: Techniques and Applications (DICTA), Canberra, Australia, 2018, pp. 1-8, doi: 10.1109/DICTA.2018.8615795.

[68] G. Harit and A. Bansal (2012). “Table detection in document images using header and trailer patterns.” ACM International Conference Proceeding Series. 10.1145/2425333.2425395.

[69] P. Riba, A. Dutta, L. Goldmann, A. Fornés, O. Ramos and J. Lladós, "Table Detection in Invoice Documents by Graph Neural Networks," 2019 International Conference on Document Analysis and Recognition (ICDAR), Sydney, Australia, 2019, pp. 122-127, doi: 10.1109/ICDAR.2019.00028.

[70] T. Kieninger, "Table Structure Recognition Based On Robust Block Segmentation" 1998

[71] T. Kieninger and A. Dengel, "Applying the T-Recs table recognition system to the business letter domain," Proceedings of Sixth International Conference on Document Analysis and Recognition, Seattle, WA, USA, 2001, pp. 518-522, doi: 10.1109/ICDAR.2001.953843.

[72] F. Cesarini, S. Marinai, L. Sarti and G. Soda, "Trainable table location in document images," Object recognition supported by user interaction for service robots, Quebec City, Quebec, Canada, 2002, pp. 236-240 vol.3, doi: 10.1109/ICPR.2002.1047838.

[73] F. Cesarini, M. Gori, S. Marinai and G. Soda, "Structured document segmentation and representation by the modified X-Y tree," Proceedings of the Fifth International Conference on Document Analysis and Recognition. ICDAR '99 (Cat. No.PR00318), Bangalore, India, 1999, pp. 563-566, doi: 10.1109/ICDAR.1999.791850.

[74] Song Mao and T. Kanungo, "Empirical performance evaluation methodology and its application to page segmentation algorithms," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 23, no. 3, pp. 242-256, March 2001, doi: 10.1109/34.910877.

[75] A. Psarologou and N. G. Bourbakis. “Glossa - A Formal Language as a Mapping Mechanism of NL Sentences into SPN State Machine for Actions/Events Association.” Int. J. Artif. Intell. Tools 26 (2017): 1750012:1-1750012:31.

[76] M. S. Alexiou and N. G. Bourbakis, "An Evaluation of Methods on Detecting, Recognizing and Understanding Graphics Images in Technical Documents," 2021 IEEE 33rd International Conference on Tools with Artificial Intelligence (ICTAI), 2021, pp. 568-574, doi: 10.1109/ICTAI52525.2021.00091.

[77] "Apache PDFBox | A Java PDF Library." [Online]. Available: https://pdfbox.apache.org/ [24] E. Adelson, C. Anderson, J. Bergen, P. Burt, J. Ogden. (1983). "Pyramid Methods in Image Processing". RCA Eng.. 29.

[78] C. Cortes and V. Vapnik, "Support-vector networks," Mach Learn, vol. 20, no. 3, pp. 273– 297, Sep. 1995

[79] F. T. Liu, K. M. Ting and Z. Zhou, "Isolation Forest," 2008 Eighth IEEE Int. Conf. on Data Mining, Pisa, 2008, pp. 413-422, doi: 10.1109/ICDM.2008.17.

[80] LeCun Y., Haffner P., Bottou L., Bengio Y. (1999) Object Recognition with Gradient-Based Learning. In: Shape, Contour and Grouping in Computer Vision. Lecture Notes in Computer Science, vol 1681. Springer, Berlin, Heidelberg

[81] C. H. Yu, J. Werfel and R. Nagpal, "Collective decision-making in multi-agent systems by implicit leadership", AAMAS (2010)

[82] N. G. Bourbakis and D. Goldman, "Recognition of line segments with unevenness used in OCR and fingerprints", Int. Journal Engineering Applications of Artificial Intelligence (Volume 12, Issue 3, June 1999, Pages 273-279)

[83] H. Bunke, "Attributed Programmed Graph Grammars and Their Application to Schematic Diagram Interpretation," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. PAMI-4, no. 6, pp. 574-582, Nov. 1982, doi: 10.1109/TPAMI.1982.4767310.

[84]https://www.chegg.com/homework-help/questions-and-answers/b-graph-following-data-use-symbols-necessary-table-1-salt-concentration-light-transmittanc-q35334661

[85] A. Kay, Tesseract: An Open-Source Optical Character Recognition Engine. 2007.

[86] Bourbakis, Nikolaos & Mertoguno, Sukarno. (2020). A Holistic Approach for Automatic Deep Understanding and Protection of Technical Documents. International Journal on Artificial Intelligence Tools. 29. 10.1142/S0218213020500074.

[87] N. Bourbakis and B.Manaris, An SPN based methodology for document understanding, IEEE Conf. Tools with AI-98, Nov. 1998, pp.10-15

[88] J. Gattiker and N. Bourbakis, An SPN scheme for functional and structural representation of knowledge, Int. Conf. SEKE, June1995, MD, pp. 47-53.

[89] N. Bourbakis and M. Mills, Converting images into NL text sentences for representing & associating events, Int. Journal on Semantic Computing, vol. 6,

No.3, pp. 353-370, 2012

[90] J. Gattiker, S. Mertoguno and N. Bourbakis, A multimedia based SPN approach for reverse engineering of digital circuits, Int. Journal Knowledge Data and Intelligent Engineering Systems, vol.2, 234-245, 1997

[91] N. G. Bourbakis, “Retrieving images using regions attributed LG graphs,” Dept. ECE, George Mason Univ., Fairfax, VA, GMU-ECE-TR-1987.

[92]G. Chiola, P. Chen, G. Balbo and S. Bruell, "An Example of Modeling and Evaluation of a Concurrent Program Using Colored Stochastic Petri Nets: Lamport's Fast Mutual Exclusion Algorithm" in IEEE Transactions on Parallel & Distributed Systems, vol. 3, no. 02, pp. 221-240, 1992. doi: 10.1109/71.127262

[93] https://en.wikipedia.org/wiki/Bayes%27_theorem#Statement_of_theorem

[94] https://ocr.space/tablerecognition

[95] R. Zanibbi, D. Blostein, J. Cordy. (2004). “A survey of table recognition”. IJDAR. 7. 1-16. 10.1007/s10032-004-0120-9.

[96] E. Adelson, C. Anderson, J. Bergen, P. Burt, J. Ogden. (1983). “Pyramid Methods in Image Processing”. RCA Eng.. 29.

[97] Z. Liu, S. B. Navathe and J. T. Stasko, "Network-based visual analysis of tabular data," 2011 IEEE Conf. on Visual Analytics Science and Technology (VAST), Providence, RI, 2011, pp. 41-50, doi: 10.1109/VAST.2011.6102440.

[98] F. Gilbert and D. Auber, "From Databases to Graph Visualization," 2010 14th Int. Conf. Information Visualization, London, 2010, pp. 128-133, doi: 10.1109/IV.2010.28.