EXTRACTING DATA FROM LINE CHARTS IN SCANNED MEDICAL DOCUMENTS

A THESIS SUBMITTED TO AUCKLAND UNIVERSITY OF TECHNOLOGY
IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF COMPUTER AND INFORMATION SCIENCES

Supervisors

Associate Professor Russel Pears

Dr. Muhammad Asif Naeem

Dr. Catherine Crofts

August 2019

By

Kathleen S. de Azevedo

School of Engineering, Computer and Mathematical Sciences

Abstract

Hand-drawn charts contained in printed forms are used to summarize data in a format that can be quickly processed and understood by humans. They differ from computer-generated charts in a few different ways: Firstly, hand-drawn charts are less predictable than computer-generated charts due to the inherent unpredictability of human behaviour; Secondly, they present higher levels of noise as they must be scanned prior to processing, which interferes with the signal. Much of past research has explored the recognition of machine-generated charts, but with no focus on hand-drawn charts in a noisy medium. Therefore, this research develops methods for the recognition of line charts in scanned medical documents. The approach uses geometrical and positional relationships between the elements of the chart to determine the values of its markers, with no human intervention. The experiments were conducted using two distinct datasets: one with 200 machine-generated charts and another with 478 scanned medical form sheets. Experimental evaluation showed a high level of accuracy for the method devised to process the machine-generated dataset. The method applied to the medical form sheets extracted the markers with a low level of error. As future work, the rate of extraction may be improved by making the procedure that detects the region in which the data lines are contained more precise.

Contents

List of Tables

List of Figures

Attestation of Authorship

I hereby declare that this submission is my own work and that, to the best of my knowledge and belief, it contains no material previously published or written by another person nor material which to a substantial extent has been accepted for the qualification of any other degree or diploma of a university or other institution of higher learning.

Signature of student

Acknowledgements

I would like to acknowledge, first and foremost, my three supervisors. My sincere thanks to Prof. Russel Pears, who had the door always open, and who helped me climb all the walls on the way. My thanks to Prof. Muhammad Asif Naeem, whose creativity and advice helped me see beyond dead ends. Last but not least, my thanks to Prof. Catherine Crofts, for her enthusiasm and honesty. This work would not have even started without her. Their guidance, encouragement and critiques were invaluable.
I wish to thank Dr. Joseph Kraft for collating the data set used in this research, and the ethics committee for the trust and the approval to use the data.
Finally, my deep gratitude to Daniel Imamura, who encouraged me throughout the whole project, and to my parents and sister who, despite the distance, greatly supported me in my journey.

Chapter 1

Introduction

Charts are commonly used to concisely present information. They can be found in a variety of documents, such as scientific articles, financial reports and web pages. Utilizing charts to visually represent data brings the following advantages:

- They can make it simpler for humans to understand a large amount of data;
- They can be suitable for revealing relationships and correlations in a dataset;
- They can be read more easily than the raw data that they summarise.

Charts can be of many different types, such as bar charts, line charts and pie charts. The data is usually represented graphically, and supporting information is conveyed through text. A chart can be comprised of several different elements, and among them are axes, markers, labels, the legend and title. Each element has a specific function, as detailed below and illustrated in Figure 1.1.

Axes An axis is used to determine a scale in a chart. A chart can have more than one axis and, when they are present, the labels representing the values are stacked horizontally and/or vertically.

Markers The markers track relationships in the data in graphical form. They are the

bars in the bar charts, the sections in the pie charts and the dots or crosses in the line charts.

Labels The labels can carry textual information about the markers and the axes;

Legend The legend indicates what the different categories of markers represent in the chart.

Title The title is a textual element that briefly explains the purpose of a chart.

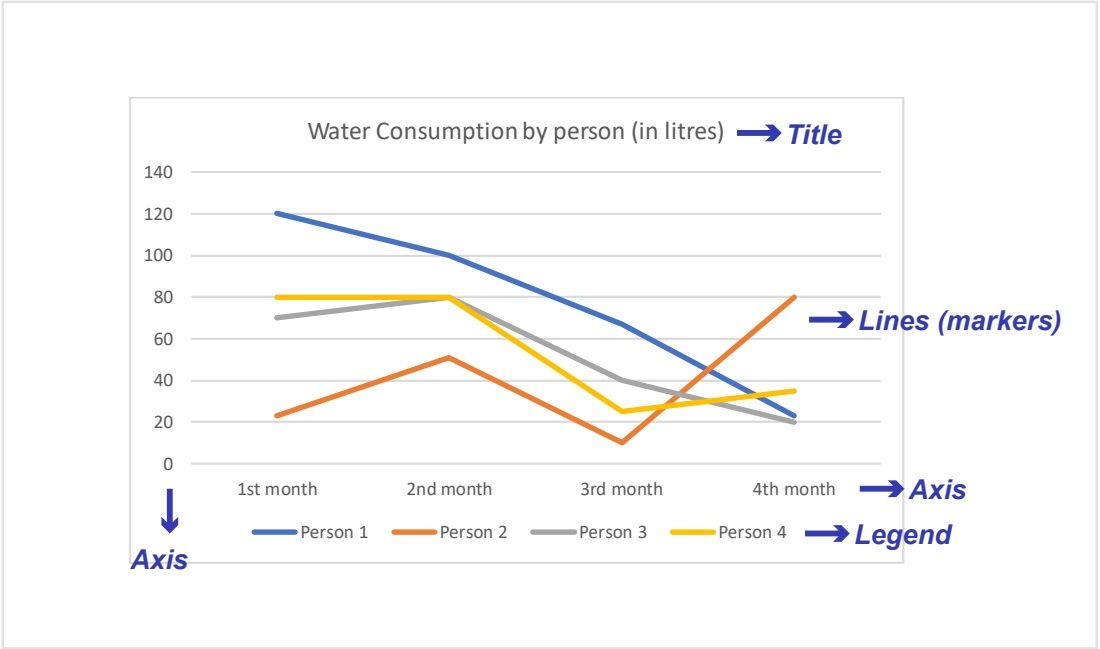


Figure 1.1: A line chart containing a title, legend, markers and axes.

The format used for charts eases human understanding, but it poses a challenge for computers. The task of extracting the underlying data from charts is complex as it requires visual skills inherent to human cognition.

Despite such difficulty, much work has been conducted on ways to extract data from charts computationally. Chart recognition (CR) refers to the tools and techniques used to automatically recognise and understand chart images. The goal of CR is to extract the value of markers contained in a chart by locating and interpreting the elements

present in it. A series of general-purpose tools have been developed to tackle this topic (Jung et al., 2017; Savva et al., 2011). A high level of accuracy has been achieved in the classification of charts by type (pie chart, bar chart, line chart), with figures above 90%. In general, these methods are based on assumptions made about the location and geometrical characteristics of the elements that compose a chart. The methods that do not depend on human interaction have a much lower rate of success, achieving between between 50% and 80%, depending on the noise level present in the underlying images (Al-Zaidy & Giles, 2015; Savva et al., 2011).

Chart recognition strongly relies on another area of research: Optical Character Recognition (OCR). Optical Character Recognition consists of a set of tools and techniques to convert handwritten or machine-generated text contained in digital images to machine-encoded characters. Charts make heavy use of textual content and, as a result, OCR is essential to process them. The performance and accuracy of OCR methods have improved considerably in the past few years due to advancements in deep learning research (Cireşan, Meier & Schmidhuber, 2012).

The main limitation of the current approaches is that they are mostly constrained to charts with low levels of noise. The datasets used in the past research were highly controlled, with samples that do not fit a strict criteria being discarded. Charts with gradients, shades or 3D decorations, or scanned and hand-drawn charts were not included. This means that these methods are not well suited for scenarios in which the charts have been obtained through computer scanning of paper-based images.

The goal of this research is to broaden the scope of the current methods for chart recognition. To accomplish this goal, a dataset of scanned documents with hand-drawn charts will be utilised. This dataset is further detailed in the section below.

1.1 The Dataset

The data set contains 509 records which are notes, exam results and miscellaneous items. 478 of these records are electronically available form sheets with the results of oral glucose tolerance tests (OGTT) with concurrent insulin assay collected between the 1970s and 1990s at St Joseph's Hospital, Chicago IL as part of routine medical practices. The tests tracked the plasma concentrations of insulin, glucose and c-peptide following the ingestion of a liquid containing 100 g glucose.

This dataset has a significant sample size, with all forms containing similar elements and the handwriting of a single person. Each form is comprised of three regions: Header; top chart; and bottom chart. The information to be extracted is encoded as markers in the charts as displayed in Figure 1.2.

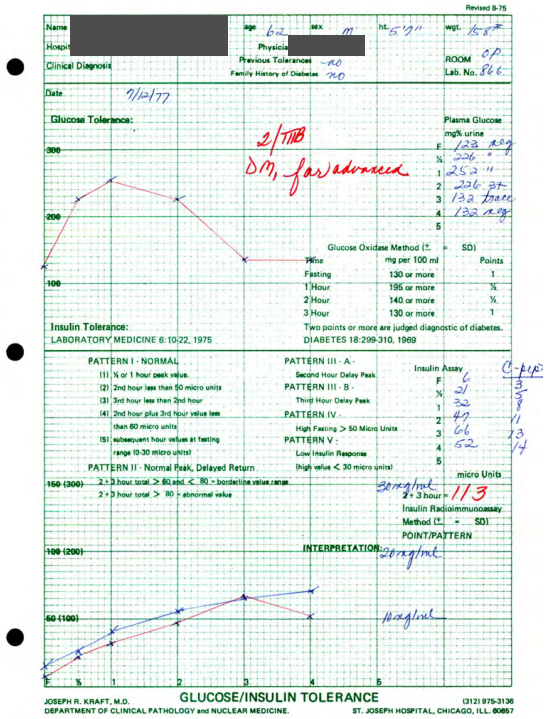


Figure 1.2: A sample form sheet (names redacted).

Scanned forms with mixed content – machine-generated and handwritten – are not uncommon in the medical area. An approach that deals with line charts contained in

this type of document may be generalised to other categories of similar documents.

1.2 Relevance of the Dataset

Dr. Joseph Kraft collated a data set containing the results of around 15,000 oral glucose tolerance tests (OGTT) with concurrent insulin assay between the 1970s and 1990s in Chicago, IL USA. The tests tracked the plasma levels of insulin, glucose and, in some cases, c-peptide right after the ingestion of a glucose drink. 509 artefacts related to these tests were made available as scanned images. These images comprise the dataset used in this research. Ethical approval for its usage was provided by the Ethics Committee of the Auckland University of Technology (18/171).

The metabolic syndrome is a set of conditions that increases the chance of heart disease, stroke and type II diabetes. It is the biggest reason of death and disability in New Zealand according to the Ministry of Health (2018). The sooner the disease is detected, the better the prospects for the patient. Currently, metabolic syndrome is associated with six major symptoms: Insulin resistance, hyperglycaemia, hypertension, low HDL-cholesterol, raised VLDL-triglycerides and visceral obesity. There is an ongoing effort in the research community to find early markers of the disease so that it can be detected and treated early. Of these symptoms, insulin resistance may be the most significant, but the hardest to accurately diagnose (Crofts, Schofield, Zinn, Wheldon & Kraft, 2016).

This data would cost millions of dollars to replicate. C-peptide levels have never been analysed in conjunction with glucose and insulin before, nor is there any data currently published on multiple sampled OGTT with c-peptide analysis. Consequently, inspecting this data set may lead to new insights into early developments of metabolic syndrome.

1.3 Research Motivation

The primary objective of this research is to extract the data markers embedded in the scanned images in order to facilitate data analysis through the use of statistical and machine learning methods. This analysis may shed light on earlier markers of the metabolic syndrome. The secondary objective is to expand chart recognition methods by devising an approach to process hand-drawn scanned charts.

1.4 Research Strategy

This research was conducted in two stages. In the first stage, a dataset comprised of 200 machine-generated charts was used. These charts contained a grid, x and y axes, a data line with markers and text labels. Figure 1.3 displays a sample element of this dataset.

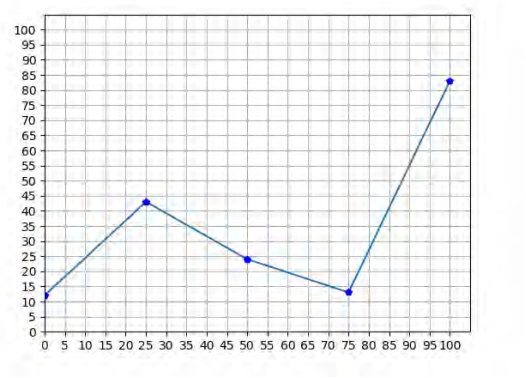


Figure 1.3: Sample chart used in the first experiment.

The second dataset was comprised of the 478 form sheets described in Section 1.1. A general pipeline has been devised to process both scenarios. This pipeline is detailed below and illustrated in Figure 1.4.

Pre-processing In the pre-processing stage, the elements that are not relevant to locating the markers are identified: The background, the grid, the labels and other components other than the markers are targeted. If the images do not have the

same dimensions, they are resized so as to be the same size across the whole dataset.

Marker location In this stage, the markers are located in the image. Their positions are found only in relation to the pixel grid, which does not provide the values in relation to the chart's axes.

Scale conversion In the scale conversion stage, the positions found in the previous stage are converted using the chart scales as a reference. This results in the final values.

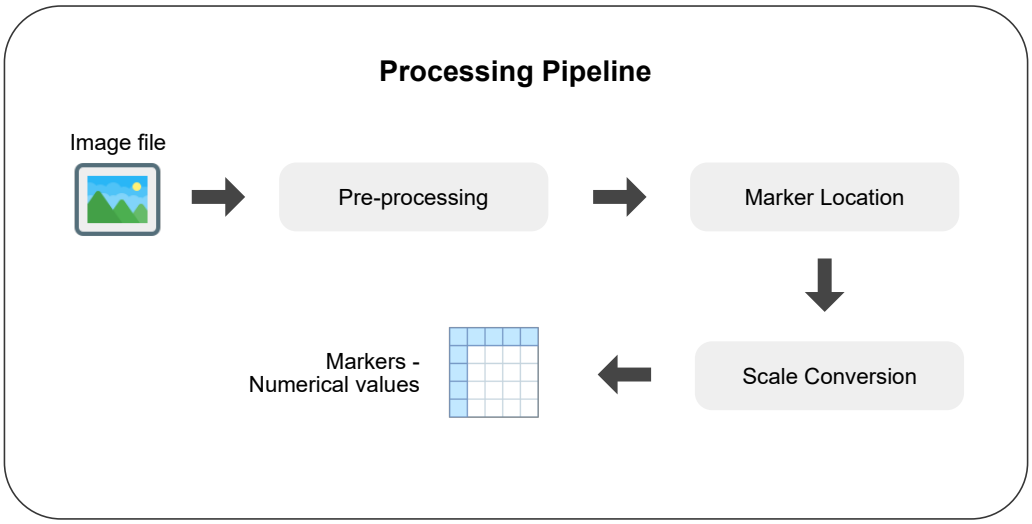


Figure 1.4: General pipeline used in both stages of this research.

Each case study case has different procedures for each of the macro-steps listed above. These procedures are further detailed in the Methodology chapter.

1.5 Research Outline

The remainder of this thesis is divided into four chapters. The second chapter covers the literature review on chart recognition and optical character recognition. Chapter

3 details the methodology used in the experiments. Chapter 4 covers the methods applied to the machine-generated charts, the results of the experiments and analysis. Likewise, chapter 5 covers the methods applied to the scanned documents, the results of the experiments and analysis. Finally, chapter 6 concludes the work with reflections on limitations of the research and some directions for future work.

Chapter 2

Literature Review

2.1 Introduction

Extracting information from digitised documents is a multidisciplinary endeavour. It involves recognizing and interpreting elements that can be of any shape or colour. A digitised document can have illustrations, machine-generated shapes and lines and freehand content. In the sections below, the relevant work previously conducted on chart recognition and optical character recognition is detailed and analysed.

2.2 Chart Recognition

Chart recognition is widely known as the process of locating, identifying and understanding chart components such as data points, data lines, labels, axes, legends and related elements. The information provided in charts is usually concise and dense and, as a consequence, the format is used in a wide array of document types, such as scientific and journalistic articles.

A series of graph digitiser tools allow for the reliable extraction of data points from individual charts. DigitizeIt, GraphClick and UnChart, for instance, provide graphical

user interfaces for the purpose of bulk data extraction from digitised charts (Rakap, Rakap, Evran & Cig, 2016). These tools usually demand intensive manual interaction, though, as the location of data lines, data points and axes have to be provided by the user. Furthermore, charts are generally processed individually, which is an issue for large data sets.

ReVision (Savva et al., 2011) and ChartSense (Jung et al., 2017) are state-of-the-art, general-purpose systems used in the identification and understanding of charts. ReVision uses a three-step pipeline: it first uses a deep learning classifier to detect the chart type among ten distinct categories – bar chart, line chart, pie chart and so on. Once the type has been identified, it then applies techniques tailored to each scenario to extract the values. Finally, it uses the extracted values to generate alternative designs that are potentially easier for humans to understand. So, for example, if the chart is a bar chart, the extracted data will be used to generate a pie chart.

ChartSense is aimed at improving the results obtained by ReVision. ReVision reached a 96% accuracy rate for the chart classification, and extracted the markers of 79% of the bar charts. ChartSense did improve on these numbers with methods tailored for each chart type, but by using a mixed approach which requires manual input from the user. In this method, the user provides the position of base elements through a graphical user interface. For the line chart, the user provides two pieces of information: the interval between markers, which is assumed to be constant across all points, and a location that is known to contain a marker. In both Chartsense and ReVision, charts with 3D effects, noisy samples, samples with gradient colours and scanned images were discarded from the data set.

General purpose tools such as the ones described above usually have chart type recognition as the first step. Current accuracy rates for this step hover above 92% for the most common chart types (Amara, Kaur, Owonibi & Bouaziz, 2017), largely due to the improvements in image classification tasks as a whole (Cireşan et al., 2012).

Older methods relied on the extraction of artificially selected features for classification, whereas the current approaches process images closer to their original form.

Data extraction is at the heart of CR. It consists of locating the markers in an image, (e.g. bars in bar charts, sections in pie charts, line markers in line charts,) and extracting their corresponding numerical values in relation to the chart scale. ChartSense, amongst other tools, utilises a mixed-approach for the extraction (Nair, Sankaran, Nwogu & Govindaraju, 2015). The user must provide a bounding box that surrounds all the data lines, two points in the axis and their relative values, and the interval between points. The algorithm will then extract the values based on the information provided. ReVision does not perform extraction on line charts. The strategy used in the data extraction is fairly constant across the body of knowledge surveyed as the methods repeatedly harness the geometrical features of the images to recognise the values of the markers (Huang, Tan & Leow, 2003). Although deep learning techniques are commonly used in the recognition pipeline, they are usually constrained to the classification of charts based on their types.

Poco and Heer (2017) automatically recovered visual encodings from charts by devising a three-stage pipeline. First, it detects the text elements contained in it, classifies their roles based on their position and geometrical features and decodes the composing characters using a text recognition engine. Next, it uses a standard CNN classifier to detect the chart type. Then, the textual information and the chart type are used to extract information about the axes. This approach stops one step prior to the actual value extraction. Nonetheless, it presents improved performance over ChartSense and ReVision in terms of element detection and identification.

A common method in CR is the segmentation of charts by using text localization and recognition as the textual elements of a chart (axis labels, legend and title, among others) hold relevant information about the data it contains. Existing systems locate text regions in the image, and subsequently pass them on to optical character recognition

engines (Savva et al., 2011; Al-Zaidy & Giles, 2015; Poco, Mayhua & Heer, 2017). The following section further discusses this topic.

2.3 Optical Character Recognition

Optical character recognition – OCR – consists of a set of techniques to identify and transform digitised texts into their machine-encoded equivalent. The text can be handwritten or printed, while the processing can be conducted either online or offline. Online OCR processes characters as they are received, while offline OCR works with the whole text at once in the form of a scanned image or PDF file.

Historically, OCR was approached as a pattern recognition challenge. The process involved a highly pipelined structure with several pre- and post-processing stages, such as deskewing, binarization, denoising, segmentation and feature extraction (Mohammad, Anarase, Shingote & Ghanwat, 2014; Arica & Yarman-Vural, 2001). The limitation of this approach is the lack of flexibility as a series of assumptions are made about the input prior to building the pipeline.

The advancements in deep learning and computer vision over the last ten years made OCR less dependent on pre-processing steps and highly structured pipelines and more dependent on big data sets with annotated samples.

Currently, state-of-the-art approaches to OCR harness deep learning techniques. Three architectures stand out due to their accuracy: EAST (Zhou et al., 2017), CRNN (Zuo et al., 2015) and STN-net (Bartz, Yang & Meinel, 2018).

EAST uses a fully convolutional network (FCN) for text detection. Although this approach does not recognise characters, it detects the regions in the image where text pixels are present. CRNN – convolutional-recurrent neural networks – is an end-to-end architecture that captures words using interconnected neural networks. First, the images pass through convolutional layers to extract a sequence of column features which are,

in turn, passed on to a recurrent layer of the LSTM - long short-term memory type. The LSTM predicts a sequence of characters and return the resulting prediction based on that sequence.

STN-net is a "Semi-Supervised End-to-End Scene Text Recognition". It is similar to CRNN in that it uses interconnected neural networks to predict the words. Nonetheless, it does not use bounding boxes to hint at where the text is for training, which allows it to be trained with a big dataset. Also as with CRNN, it uses a convolutional neural network concatenated with a LSTM. These architectures reached an accuracy of above 95% in the prediction of words and characters.

Tesseract (Smith, 2007) is the OCR engine of choice for experiments which depend on text recognition. It provides multi-language support and works with unicode characters. All the papers surveyed in chart recognition used this tool to process characters. The underlying implementation utilises a LSTM neural network.

Some players in the industry provide cloud Application Programming Interfaces – APIs – for OCR. An API is a set of clearly defined methods of communication among independent computational components (Zimmermann, Stocker, Lübke & Zdun, 2017). These APIs receive an image as the input and return the text detected in it. Google Cloud Vision, Amazon Textract and the Azure Computer Vision API are a few of the most prominent offers available. These services utilise deep learning techniques to perform the recognition.

2.4 Summary

Current OCR methods and techniques, as described above, harness deep learning techniques to achieve the best results. Although the area has seen considerable progress in the past years, it is not yet a solved problem. Modern architectures perform well when applied to benchmark datasets such as MNIST (for handwritten characters), with

error rates below 0.3% (Redmon, Divvala, Girshick & Farhadi, 2016). Nonetheless, in-the-wild scenarios (nature, outdoors) and freehand writing still pose a significant challenge for processing.

The body of work conducted on chart recognition has a few common threads. The data sets used in the experiments do not usually contain scanned or noisy samples. The input is always clearly defined: only solid colours, no gradients, no shadows and no noise. The edge cases are removed from the samples before the experiments are conducted. On top of that, some level of human input is always present, be it to locate the data markers or to identify the axes.

The body of research, to the best of our knowledge, presents a gap for hand-drawn, scanned line charts. This research will seek to fill part of this gap by proposing a method for the extraction of markers from line charts contained in scanned documents.

Chapter 3

Methodology

The study aimed to devise a process to extract data contained in charts from a set of scanned medical documents. These documents are scanned form sheets with mixed content: handwritten and machine-generated elements. The handwritten content is comprised of textual elements and charts, and is the main focus of the extraction process.

The experiment was broken down into two stages. In the first stage, a dataset was set up so techniques and assumptions could be tested in a controlled, noise-free environment. In the second stage, the techniques used in the first stage were refined and applied to the scanned documents. The following two sections detail the first and the second case studies, respectively.

3.1 First Case Study

A dataset with 200 computer-generated charts was used for the first stage. The charts were consistent and presented similar elements – a grid, one data line, five data points and two axes with labels. Each sample was submitted to a processing pipeline so that the pixels that comprise the markers could be located and used to determine the values.

The rationale behind the pipeline is to progressively detect pixels which are not relevant to the end goal so that it can eventually narrow down to the markers. Domain knowledge about how the elements are shaped is used to determine the steps to be performed. Figure 3.1 illustrates the pipeline used, and the steps are outlined in the list below.

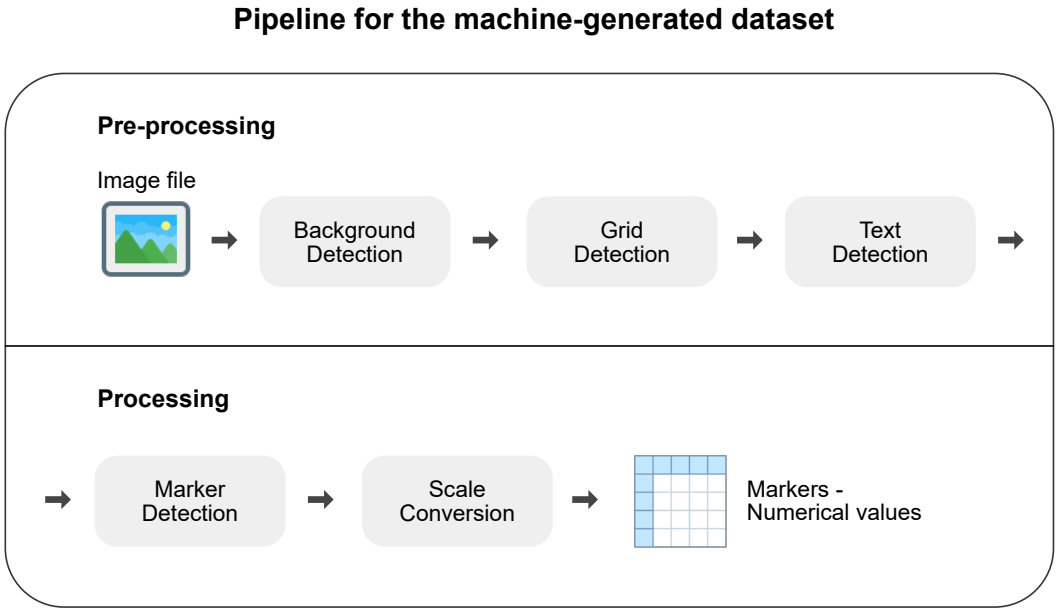


Figure 3.1: Steps in the pipeline for the first case study.

1. Background Detection The most frequently occurring colour is the one used in the background. The first step in the pipeline detects all the pixels in the image that belong to the background by using a colour histogram. These pixels are painted black before further processing is performed. Black is the colour reserved to indicate that a pixel is no longer relevant for processing throughout the pipeline.

2. Grid Detection The next element to be detected is the grid. This is accomplished by finding a pattern of colours that is repeated at a regular interval and painting with black all the pixels in the image that match any of the colours in the pattern.

3. **Text (label) Detection** The labels holding the values for the axes are located and their pixels are painted black. Their contents are stored for a later step as they are necessary for creating the scale converters.

4. **Marker Detection** Once the steps above are performed, the only remaining element is the data line with the markers. The lines are thinned so that only the markers remain. The centroid of each marker is used to determine the cartesian coordinate of the marker in relation to the image.

5. **Scale Conversion** The labels detected in the third step are used to derive the actual scales of the chart. In this step, the coordinates of the centroids are converted to the actual scale used in the chart with the converters generated in the previous step, which results in the final values.

The steps above and the dataset are further explained in Chapter 4. The larger the element, the earlier it is detected in the pipeline. The background is the most commonly-occurring element in terms of total pixel coverage in the image and thus its detection is the first step in pre-processing of the image, as indicated above. No assumption is made about how the lines are plotted in regards to dimensions or the algorithm which is used to plot them. The methods utilised for each of the above steps will be elaborated in Chapter 4 that follows.

3.2 Second Case Study

In the second part of the research, the set of scanned medical documents is used. Due to the noise and the less predictable nature of the samples, the methods applied to the first case study had to be altered. The same rationale was maintained; the most commonly occurring elements that contribute the least to the end goal are detected first. Figure 3.2 illustrates the pipeline for this case study, and the steps are outlined in the list below.

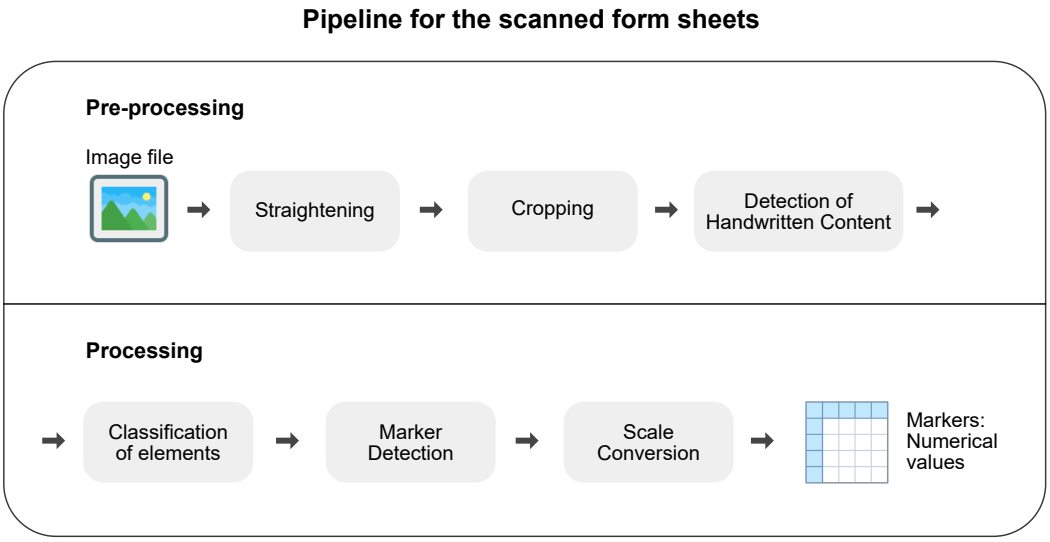


Figure 3.2: Steps in the pipeline for the second case study.

Straightening The form sheets were manually placed in the machine when scanned. As a result, the images were slightly tilted. The first step is to straighten them by skewing the samples so that they become vertically aligned.

Cropping All the information that is relevant to the end result is contained within the grid. Thus, the surrounding margins are removed, and the samples are normalised to be the same dimensions.

Detection of Handwritten Content The handwritten content is considered to be the foreground, whereas everything else is the background. In order to isolate the foreground, the background pixels are detected. This is accomplished by defining ranges of colours that account for the background pixels. Any pixel that is coloured with a value that falls within these ranges is painted black. Black is the colour reserved to indicate that a pixel is no longer relevant for processing.

Location of Elements The resulting image from the previous step is copied and binarised by turning all foreground pixels white. The white regions are dilated, which groups together elements close in proximity.

Classification of Elements The groups created by the dilation in the previous step are classified based on their role. In this step, the regions in which the data lines lie are identified. The data lines are isolated for further processing.

Marker Extraction At this point, only the data lines remain. The lines are thinned, the markers are highlighted and the centroid of every marker is stored to be used in the next step.

Scale Conversion Unlike the first case study, the scale converters are provided as parameters for processing. They are used to reach the final values of the markers in relation to the actual scales contained in the chart.

The steps above and the related dataset are further explained in Chapter 5.

3.2.1 Measuring Marker Extraction Accuracy

A metric was devised to measure the error rate for the methods employed. The metric is dependent on three variables: the ground truth, the predicted value and the scale interval. The ground truth is the actual value of a marker. The predicted value is the value extracted with the methods defined in the research. The scale interval is the range of values available in the axis. The error is calculated by averaging the absolute error of a prediction across a group of samples as it is described in Equation 3.1. This metric was used for both case studies.

$$error = \frac{1}{n} \times \sum_{i=1}^{n} \frac{|groundTruth_i - predictedValue_i|}{scaleInterval} \tag{3.1}$$

This metric has been chosen because it relates the error to the scale used in the chart. If a metric such as MAPE – Mean Absolute Percentage Error – was used, then smaller figures would present larger errors as the error is framed in relation to the value itself.

3.3 Summary

This chapter has outlined the methods used in the experiments. Although both case studies have the same stages from a macro perspective, the details are considerably different. The next two chapters expand on the steps of each pipeline, present the results obtained by applying these methods to the datasets and analyses the figures based on the metrics described for each case study.

Chapter 4

First Case Study

The experiments focused on two distinct scenarios: computer-generated and digitised charts. This first case study describes the methods employed in the dataset containing the machine-generated charts. This scenario is the most predictable due to the nature of the medium: The charts do not present high levels of noise as their characters, lines and shapes follow predictable patterns. For this step, a dataset was collated for the necessary experiments as described in the following sections.

4.1 Experimental Dataset

The dataset consists of 200 images, all 640 pixels wide by 480 pixels high as displayed in Figure 4.1. Each chart is an image, which is comprised of four main elements: The data line, the grid, the y-axis and the x-axis. These elements are positioned on top of a white background.

The data line has five evenly distributed markers representing the data points, which are interconnected by a single line. The line and the markers are of varying colours, and the marker can be shaped either as an inverted triangle, an "X", a square, a cross, or a pentagon (Figure 4.2). The markers and the line do not need to be the same colour.

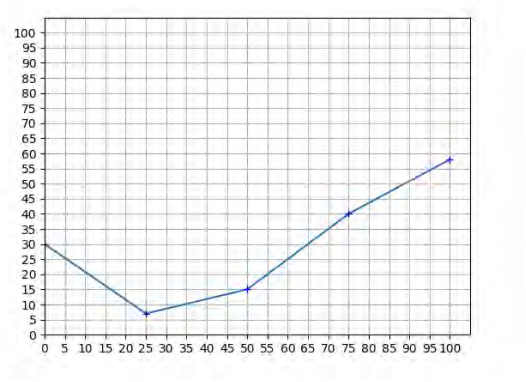


Figure 4.1: A sample chart generated for the first stage.

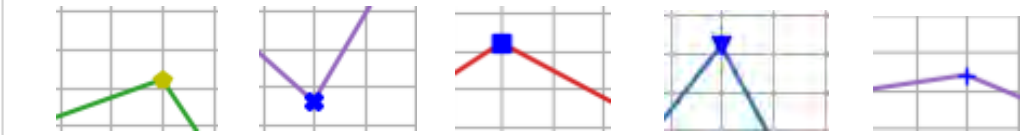


Figure 4.2: The five styles of markers used to generate the dataset. From left to right: pentagon, X-shaped, square, inverted triangle and plus sign.

This means that if the line is blue, the markers could be another colour, e.g. red. The markers do not have the same geometric features and are plotted in varying sizes.

The grid has the same shape and size for all images. Its lines are light gray and spaced at a regular interval. The lines representing y-values are horizontal, while x-values are vertical.

Finally, the x and y axes are horizontally and vertically positioned to the left and bottom of the grid, respectively. Both axes represent values ranging from 0 to 110, with numbers being plotted in increments of five units.

The charts were generated by a script. The colours used and the shape and value of each marker were randomly assigned. The range of possible values for both dimensions was between zero and 100. Both axes show a label for the value "110", but it is only for aesthetic reasons. The distribution of charts based on marker type is displayed in Table 4.1.

Table 4.1: Distribution of charts based on marker type.

Marker Type	Total of items
Inverted Triangle	40
Pentagon	43
"X"	41
Plus Sign	36
Square	40

4.2 Experiment

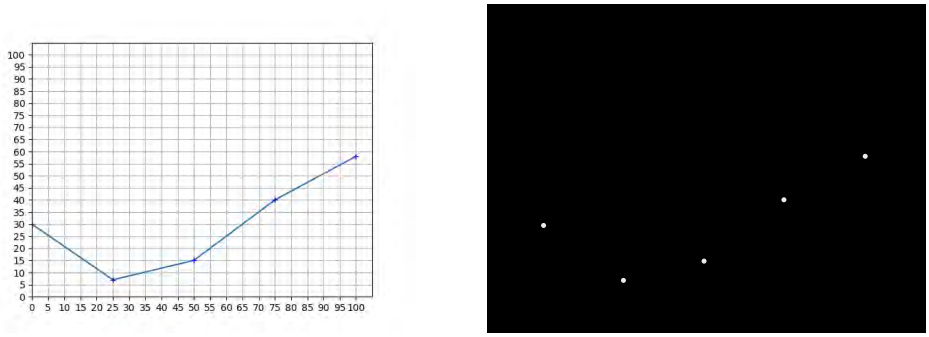
The experiment aimed to extract all the marker values in terms of their Y coordinates and subsequently check the difference between the actual value and what has been detected by the algorithm. To achieve this goal, the process was broken down into a pipeline consisting of five stages: background detection, grid detection, text detection, data line detection, and marker detection.

The irrelevant pixels are progressively identified as a chart is passed through the pipeline. Eventually, the algorithm narrows down to just those pixels representing the markers. Figure 4.3 shows the map which is output in the later stages by this process. In this maps, the colour white is attributed to the areas which are part of a marker, whereas black means that the area is not needed for further processing.

Throughout the process, black is the colour reserved to paint pixels which do not carry relevant information for further processing.

Colour Space

The experiment was conducted using the RGB colour space. A colour space is an organisation of colours, especially for digital image representations. The RGB colour model is an additive model that combines three channels: red, green and blue. These colour are combined and mixed to form a wide array of colours.



(a) Original chart (b) Marker/not marker pixel map

Figure 4.3: Pixels of a sample image classified as either part of a marker or not. White means marker and black means not-marker.

4.2.1 Background Detection

The sample images generated for this case study have solid backgrounds. This means that exactly the same colour is attributed to all background pixels. Although the background takes up most of the space in the samples, it does not convey any useful information. Thus, the first step in the process is to identify and mark all background pixels in order to isolate them from pixels containing markers.

Three main steps are performed to arrive at the desired result. First, a histogram is created containing the frequency for all colours in the image. The histogram stores the number of pixels with a given colour across the whole image as shown in Figure 4.4.

Then, the most frequently occurring colour is identified. Finally, if the number of pixels with this colour surpasses a manually provided threshold (for example, 60% of the total number of pixels in the image), all pixels with said colour are marked as being part of the background. Figure 4.5 displays the output of this step.

4.2.2 Grid Detection

The second element to be detected is the grid, and the input to this step is the output image from the background detection step. The grid has geometrical and spatial

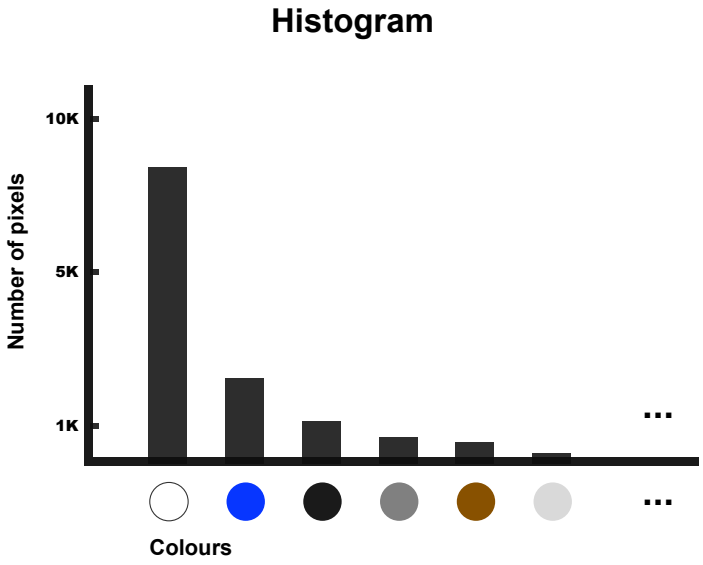
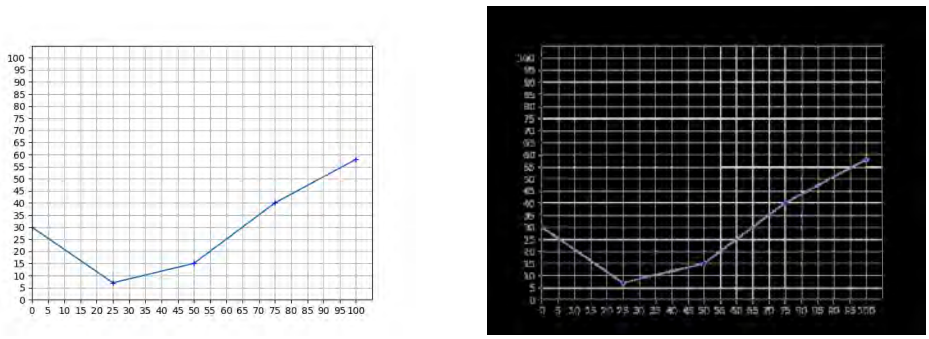


Figure 4.4: A sample of how a colour histogram is represented.



(a) Original chart (b) Background pixels in black

Figure 4.5: The most frequent pixel colour is utilised to detect the background

characteristics that follow a certain pattern as it is composed of horizontal and vertical lines which are evenly distributed. Thus, its composing pixels can be detected by finding a set of colours that is repeated at regular intervals.

This stage is broken down into three steps. First, a central sub-region of the image is selected for reasons that will be explained in the following paragraphs. The algorithm has been configured to select approximately 40% of the central portion of the image.

Then, a number of random pixels – which is set to five – are selected within this bounding box. Figure 4.6 shows the selected portion in red and the selected pixels as green crosses.

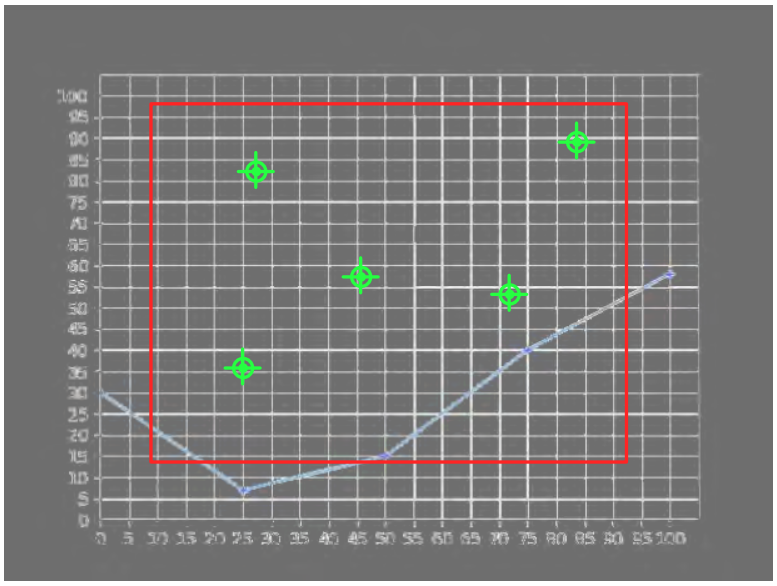


Figure 4.6: The bounding box and the randomly selected pixels.

An image is nothing more than a 2-dimensional stacking of pixels as rows and columns. The next step is to select the full row and column in which each randomly selected pixel resides, including the pixel itself. Figure 4.7 displays in bright pink which rows and columns are selected.

Next, each row and column is inspected for a repeating, regular pattern of colours. The bounding box from the first step is positioned at the center of the image for two reasons. Firstly, it is based on the assumption that the chart will be roughly centered on that position. Secondly, it lowers the chance of selecting a pixel which is outside the chart area, which would potentially return columns and rows with exclusively background pixels. Figure 4.8 illustrates what a row or column with a repeating pattern can look like when inspected from up close.

The goal is to detect the colours of the pixels that belong to the grid. In order to do that, each selected column or row – ten in total – is fully scanned, and a dictionary

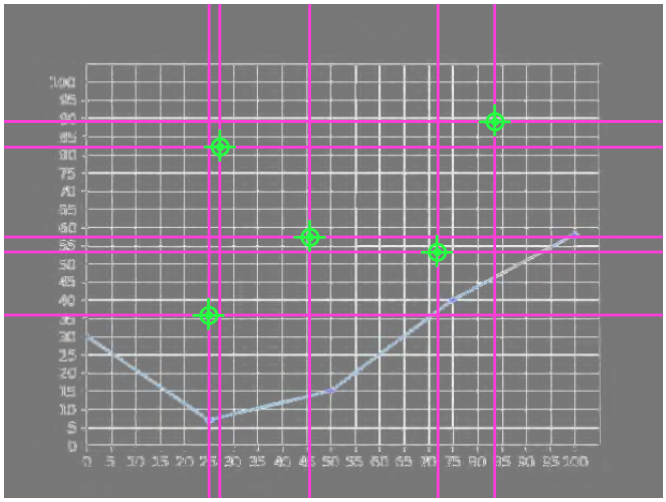


Figure 4.7: Rows and columns selected by the algorithm.

Figure 4.8: An illustrative example of a section of a selected row or column.

is created with the patterns found. Black pixels, as previously mentioned, are ignored. The dictionary stores the patterns as keys, and a list containing the starting index of each occurrence of this pattern is associated with each key. Figure 4.9 displays a sample populated dictionary.

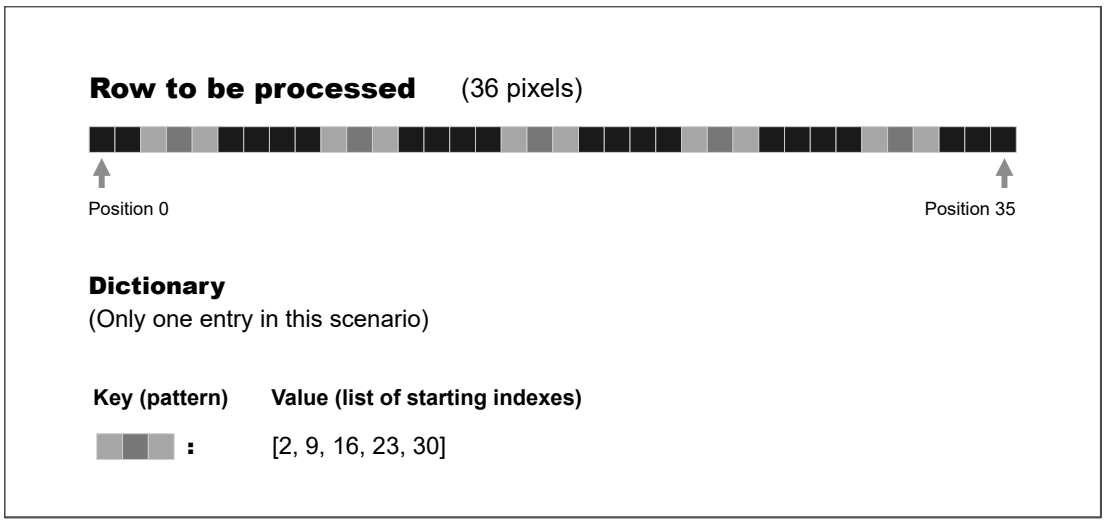


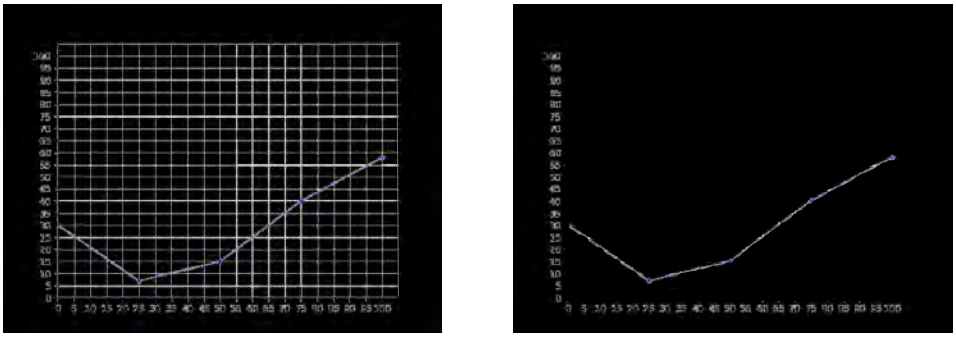
Figure 4.9: Dictionary with patterns as keys and starting indexes as values.

In order to check whether the distances between the occurrences of a single pattern are even, the list of starting indexes is analysed. Two conditions are defined to consider a pattern part of a grid:

- The list has to have at least three starting indexes (the pattern repeats itself at least three times);
- The standard deviation of the distance between all pairs of consecutive indexes is below five pixels. This threshold was selected after extensive experimentation on the set of selected charts.

The standard deviation is used because some tolerance level is necessary. The grid will not be perfectly distributed in all instances; it may have a few pixels in variation due to the way raster algorithms may draw anti-aliased lines. In addition to that, the data line may interfere with the pattern, covering or partially covering one occurrence of it. If the criteria to select the pattern was stricter than this, nothing would be selected.

Once the patterns have been selected, all matching colours in the image are marked with black based on the colours belonging to the selection. As shown in Figure 4.10, this step removed the background grid from the image leaving the axes, the chart line and the data markers intact.



(a) Input image (b) Image after the grid is removed

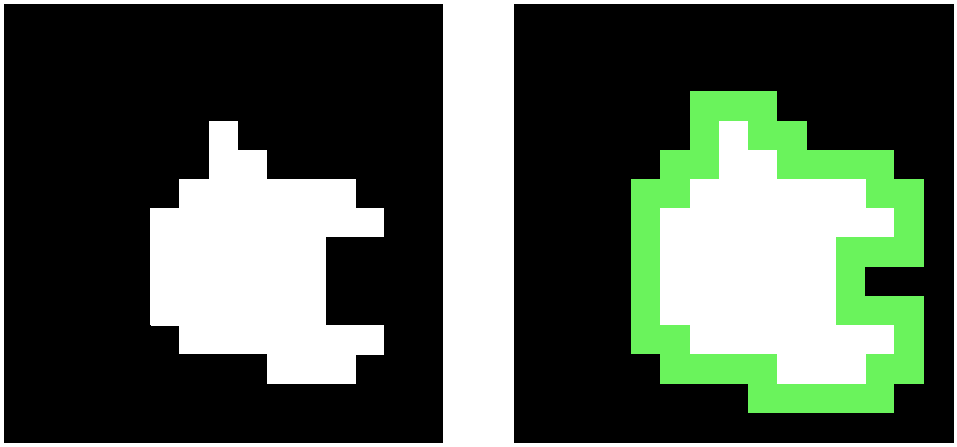
Figure 4.10: Resulting image from the grid detection step.

4.2.3 Text Detection

The goal of this stage is to detect the position of the numbers along the axes, store said positions for later use and remove the numbers so that only the data line is available for the next phase of the pipeline.

This stage is divided into three steps. First, the image is binarised by having all the pixels that are not black painted in white. This means that the image is now only represented by black and white pixels.

Then, a technique called contour detection is applied using OpenCV (Bradski, 2000). OpenCV is a compuer vision library that provides a series of routines for image processing. A contour is a line around all continuous pixels with the same colour in an image. In a binary image, the contours will be traced around either black or white regions. A contour is represented by a list of X and Y coordinates that define the whole trajectory of the line (Figure 4.11). All the boundary points of the contour are identified.



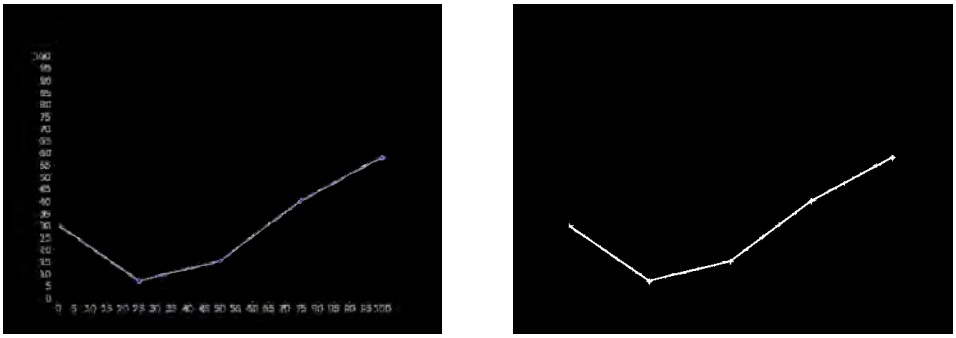
(a) An image with a continuous portion of white pixels.

(b) The contour traced around it.

Figure 4.11: A contour is a curve around points with the same colour.

Finally, the size of the contours are used to find the pixels which are part of the text regions. As previously mentioned, only the numbers and the data line are present at this stage. Every digit has a small contour around it, whereas the data line has a

large contour. All the pixels that are within contours that fit into a small rectangle of a predefined size – 80x50 in this case – are painted black. These contours are then stored for later use as they represent the digits for the axes, which will be needed to find the values of the markers (Figure 4.12).



(a) Input image (b) Image after the numbers are removed

Figure 4.12: Resulting image from the text detection step.

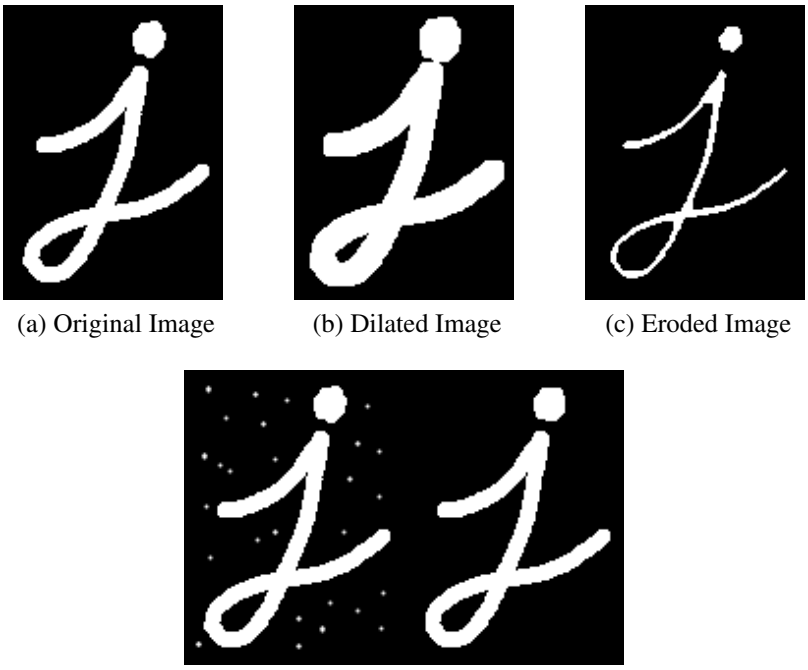
4.2.4 Marker Location

The goal of this stage is to find a central value for each marker. The input to this step is the image with the data line only, and the flow is broken down into three main steps.

First, an opening operation is applied to the image in order to remove the data line and keep the markers only. The opening operation is a type of morphological transformation. A morphological transformation is usually performed on binary images, and it changes the shapes contained in them (Heijmans, 1994).

The opening operation is the combination of two sub-operations: erosion followed by dilation. Both of them work similarly: a window of a predetermined size slides through the image, always having the central pixel as the main reference. For erosion, if one of the pixels under the window is not white, the central pixel is turned black (eroded). For the dilation, if at least one of the pixels under the window is white, the central pixel is turned white (dilated). The operation is performed on every single pixel

of the image. Figure 4.13 illustrates the result of applying the erosion, dilation and opening operations to the binary sample of a handwritten "J" character.



(a) Original Image (b) Dilated Image (c) Eroded Image

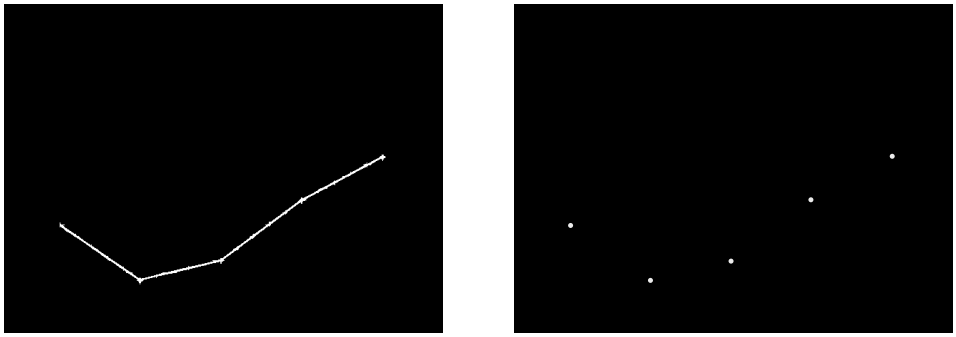
(d) Image with noise (left) and after opening operation (right).

Figure 4.13: Morphological Operations.

The opening operation erodes the data lines and highlights the regions where the markers are. This happens due to how these objects are shaped: the lines are thin and the markers are denser. The size of the window used on the dataset was 3x3. This window size (Figure 4.14) was obtained after extensive experimentation.

Once only the pixels that belong to the markers are highlighted in the image, a two-step operation is performed to define their locations. First, a window slides through the image to detect the approximate location of the markers. The positions in which the window finds white pixels are marked as matches.

The size of the window that slides through the whole image is 5x7. By this stage, the whole image is black except for the markers, which are white. If more than 80% of the pixels within the window are white, the location is added to a list of matches. Both the

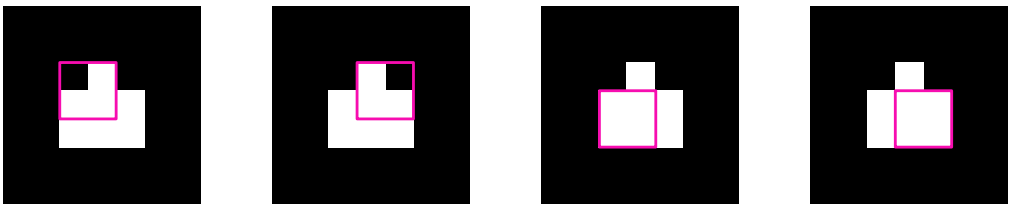


(a) Image with data line only (b) Image with the markers only

Figure 4.14: Result of the opening operation.

window size and the percentage have been determined after thorough experimentation.

Figure 4.15 illustrates all the matches found by a 4x4 window scanning a 7x7 sample image. For illustrative purposes, the window size and image size are different from the ones in the actual dataset. This window follows the rule that at least 75% of the pixels within the boundary have to be white.



(a) First match (b) Second match (c) Third match (d) Fourth match

Figure 4.15: This sample scenario has exactly 4 matches for a 2x2 window. The rule used to determine a match is that 75% of the area within the window should be white.

The matches are then triangulated to determine the final values. After the whole image has been inspected, the list of matches is separated into clusters. A single marker will generate several matches, and the clusters are created based on the position of each match along the x-axis. The number of clusters is equal to the number of markers in the image.

Finally, a process of triangulation is performed on each cluster. The center of each cluster is determined by the mean of the centers of all the matches. Figure 4.16 shows

in green what the final value would be for the scenario presented on Figure 4.15, with the center of each match represented in pink.

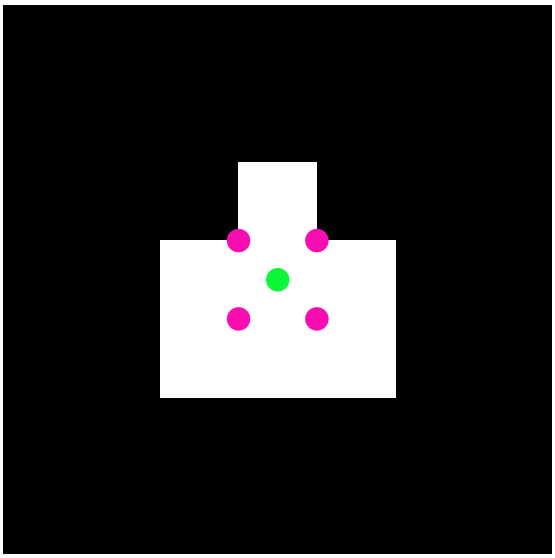


Figure 4.16: Triangulation.

The output from this step is a list with the X and Y pixel coordinates for each marker in the chart.

4.2.5 Scale Conversion

This is the last phase in the pipeline. The exact locations for the markers have been detected, but it is only in relation to the pixel grid. As the final step, the pixel values are converted using the scales extracted from both axes.

In order to accomplish this final task, the steps in the list below are performed. These actions are further detailed in the following sub-sections.

1. Retrieve the list of contours generated in the text detection stage;
2. Separate this list into two sub-lists: contours that belong to the y-axis and contours that belong to the x-axis;
3. Merge the neighbouring contours in both lists so that individual digits become a single number;

4. Use the OCR engine Tesseract to recognise the values of each merged group;
5. Generate the scale converters for both axes;
6. Find the final values for the markers by using the converters.

Retrieval of the List of Contours

In the text detection stage, a list of small contours was stored for later use. These contours represent the positions of all the digits in the image. They are not sorted in any relevant order, and the digits have no reference back to their original full numbers. The list is retrieved to be further processed.

Separation Between x and y Axes

The first step is to separate the contours as belonging to either the x or the y axis. In order to do that, the bounding boxes of the contours are used. This means that, instead of the list of points that comprise the contour, the rectangle that surrounds it is used. This rectangle is parallel to the edges of the image, and it exposes four properties: x and y (the coordinates of the top-leftmost pixel), and the width and height (in pixels). Figure 4.17 represents the bounding rectangle of a contour. The rectangle coordinates are taken in relation to the image grid, which has its origin at the top-left corner of the image.

The contours that surround numbers which belong to the x-axis have similar y-values. This happens because they are organised as a row. If a large amount of contours with similar y-values is detected, then it is likely that they belong to the x-axis.

A dictionary is created in order to find the contours that belong to the x-axis, with the contours being grouped by their y-coordinate. The dictionary key is a range, which is based on the y value. The logic to build this dictionary is depicted by the algorithm outlined in Figure 4.18.

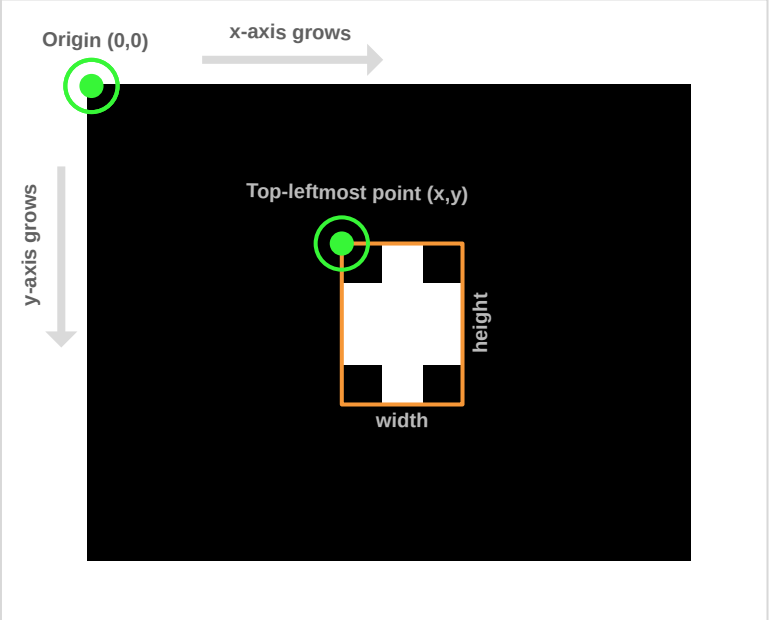


Figure 4.17: A sample contour surrounded by its bounding box positioned in the Cartesian plan.

```
dictionary <- empty
for each contour:
    box <- bounding box of the contour
    if Y of box fits in a range from the
    dictionary:
        add contour to the entry
    else:
        add new key to dictionary (Y - 5, Y + 5)
        add the contour to this new entry
```

Figure 4.18: The algorithm used to create the dictionary containing the label contours grouped according to their respective y-coordinates.

When the dictionary is fully built, the entry with the largest list of contours will contain the locations to all the digits which belong to the x-axis. Once the digits that belong to one axis are known, then, by exclusion, the digits that belong to the other axes are also known. Figure 4.19 exemplifies what the dictionary structure would be for a chart with 10 values distributed between the two axes.

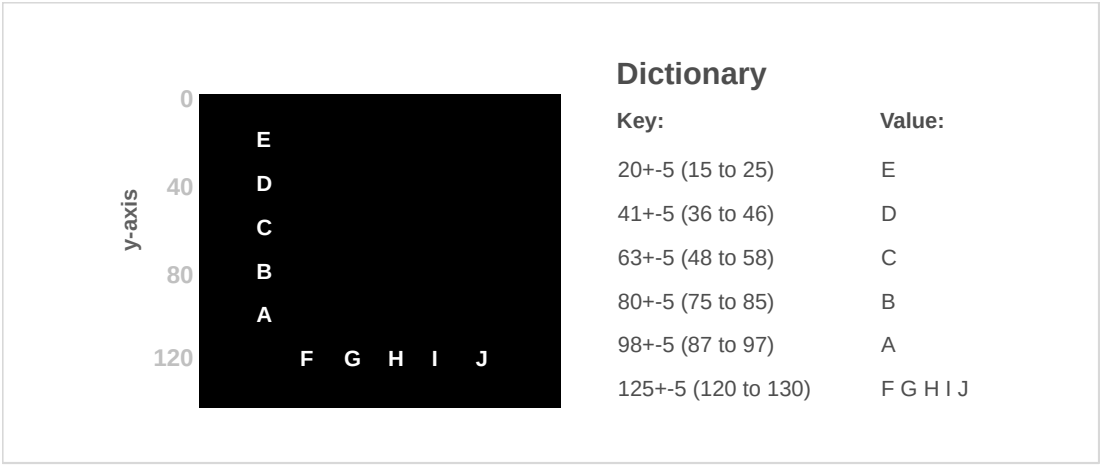


Figure 4.19: The dictionary used to divide contours by axis. The keys are based on the y value of the bounding box for each contour.

Merge Neighbouring Contours

Once the contours have been categorised as either from the y- or x-axis, the neighbouring digits are merged to form the original numbers. Just like in the previous step, the bounding boxes are used instead of the contours.

The two groups are processed separately. First, the list with contours from the x-axis are ordered by their x values. This puts the digits in the order they appear from left to right. Then, the list is scanned to check the distance between adjacent digits. If the distance is below a certain threshold, they are grouped together.

The digits for the y-axis group are treated differently. They are grouped based on their y coordinate, with digits with similar y values being merged. This is performed by using the same dictionary strategy applied to divide the contours between the two axes. Figure 4.20 illustrates the approach.

Retrieve Numerical Values from Contours

When the contours for the digits are merged, they can be used to extract the values for the whole numbers. An OCR – Optical Character Recognition – engine called Tesseract (Smith, 2007) is used to retrieve the textual values. These merged contours are used

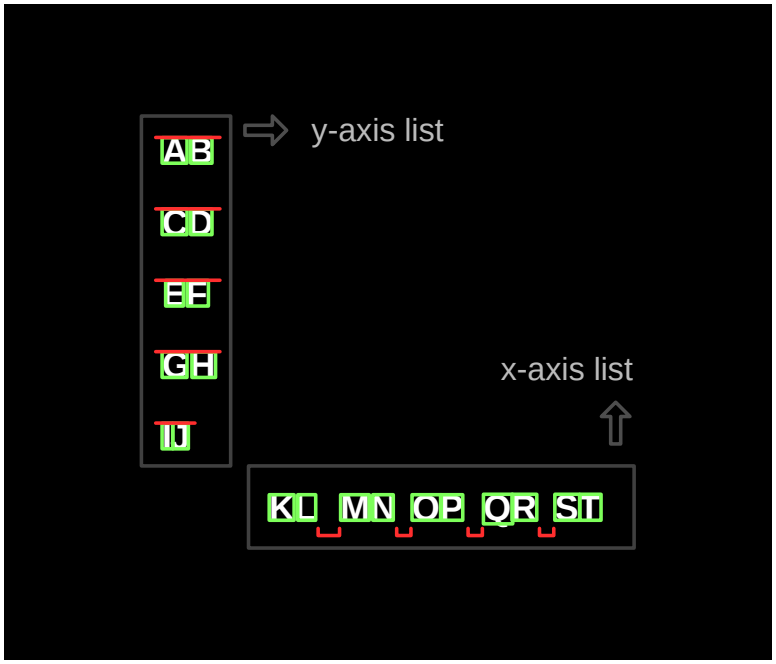


Figure 4.20: The digits on the y-axis are merged based on their Y position (red line), while the digits on the x-axis are merged based on how close they are to each other (red gap).

to cut the labels from the original, unprocessed image, and each piece is recognised individually. The output of this step is a list with the the numerical values for each label.

Create the Scale Converters

The image is represented by a grid of pixels with the origin at the top-left corner. The x dimension grows to the left and the y dimension grows to the bottom. The axes have the origin at an unspecified location, with the y values growing to the top and the x values growing to the left. Figure 4.21 shows the mismatch between the scale of the image grid and the chart axes.

So far, the markers positions are known in relation to the pixel grid. Thus, it is necessary to convert between the pixel scale and the chart scale to extract the final values. First, the scale type for each axis has to be determined. The algorithm identifies two different types: linear and logarithmic. Once the scale type is identified, it is used

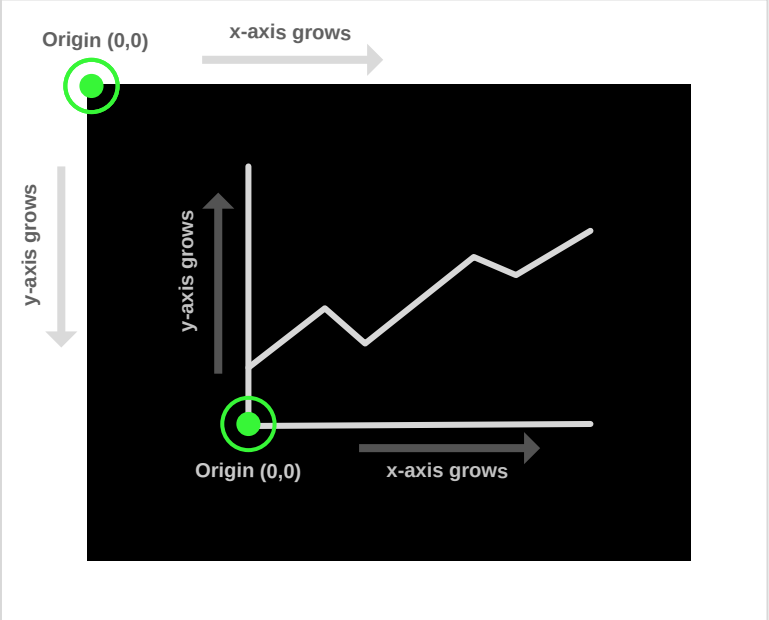


Figure 4.21: The image grid and the axes behave differently.

to create the scale converter, which converts between the pixel value and the chart actual value as detailed in the following paragraphs.

The actual value for each label has been obtained in the text recognition step. In order to identify the type of the scale, the list of labels for the y-axis are ordered based on their numerical values, from lowest to highest. If the second value minus the first value is equal to the third value minus the second value, then it is linear. If the second value divided by the first value is equal to the third value divided by the second, then it is logarithmic. Only these two types of scales were considered in this study.

The dataset used linear scales on both axes, so the converters are tailored to this scenario. In order to build them, two labels are used for each axis. The formula follows the same pattern for both cases, and it uses the lowest and highest labels with their respective numerical and pixel values. The elements used in the converter are as follows:

- YPixelRange: the difference between the highest and lowest values in the y-axis in relation to the pixel grid;
- XPixelRange: the difference between the highest and lowest values in the x-axis

in relation to the pixel grid;

- YActualRange: the difference between the highest and lowest values in the y-axis in relation to the chart scale;
- XActualRange: the difference between the highest and lowest values in the x-axis in relation to the chart scale;
- YPixelMin: the lowest values in the y-axis in relation to the pixel grid;
- XPixelMin: the lowest values in the x-axis in relation to the pixel grid;
- XMarker: the x value of the marker in relation to the pixel grid;
- YMarker: the y value of the marker in relation to the pixel grid.

Each label has an actual value and a pixel value. The actual value refers to the numerical value from the chart, whereas the pixel value refers to the pixel grid. The pixel value for each label is its central pixel. This is a fair assumption as it is usually the case with computer-generated charts. The converter formula for the y-axis is displayed below.

$$finalYValue = \left| \frac{((YMarker - YPixelMin) * YActualRange)}{YPixelRange} \right| \tag{4.1}$$

The converter for the x-axis is similar, but it is not necessary to use the absolute value as both the pixel grid and the chart axis grow in the same direction. This formula is stated below.

$$finalXValue = \frac{((XMarker - XPixelMin) * XActualRange)}{XPixelRange} \tag{4.2}$$

The converters are used for all the markers. Once the conversion is finalised, the end values are reached.

4.3 Results

All the images in the dataset were processed using the algorithm described in the section above. To evaluate the results, the metric introduced and explained in section 3.2.1 was employed. The absolute difference between the actual value and the extracted value is found for the marker and the result is divided by the range in the scale. This process is repeated for all the markers in a group and the arithmetic mean of all results defines the global error for that group.

There were 200 charts in total, with 5 markers in each of them. Every marker has two coordinates – x and y, so adding them all results in 2000 data points (200x5x2). The samples covered a variety of colours for the lines and the markers. Table 4.2 details the results grouped by marker type.

Table 4.2: Error rate for the first use case.

Triangle	X	Pentagon	Square	Plus sign
0.520%	0.398%	0.339%	0.411%	0.344%

The global percentage error was 0.40%. This percentage value refers to the ranges of the scales in the chart. Both axes had a range of 100 units, and 0.40% of 100 is equal to 0.4. Thus, the extracted values were, on average, no farther than 0.4 units from the actual marker value.

Finally, six data points were not detected correctly. The dataset contained 2000 data points, which means a 0.30% rate of false negatives. All the markers that were not identified were using the plus signs to be displayed. No false positives occurred.

4.4 Discussion

The dataset used in this case study was considerably homogeneous. As a result, the error rates for the experiments were low. The background detection process precisely

identified the pixels which were part of the background for all images. The grids were also precisely identified, and the morphological transformations isolated the markers successfully. The character recognition step had a 100% success rate, and the scale conversion worked as described.

Both scales go from 0 to 100, and both span 352 pixels in total. This means that each unit in the chart scale is represented by 3.5 pixels, approximately. The experiment used a dataset with a low density of pixels per unit measured, which results in higher precision. The less pixels per unit, the more precise the algorithm is due to the fact that the pixel grid is used to extract the values.

All the markers that were not detected were plus signs. The false negatives occurred because this style of marker is less dense than the others. This way, the marker can be confused with the line depending on how it is positioned as shown in Figure 4.22. There were 180 plus-sign markers in the dataset, and six were not identified. This means that, for this marker type, the false negative rate was approximately 3%.

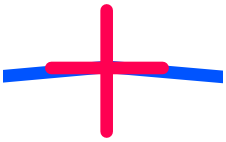


Figure 4.22: Sample case in which the plus sign marker may not be identified as the blue and red lines almost perfectly intersect.

4.4.1 Expected Preconditions for the Case Study

This case study assumes a list of preconditions to work as expected. These conditions are listed below:

- The images provided should have a chart in it, and this chart has to be roughly positioned around the center of the image;

- The chart has to have only the following elements: a grid, a data line with markers and labels for the x and y axes;
- The labels for the axes should not be tilted in any way;
- The images should have a low level of noise;
- The images should have a solid-coloured background;
- The axes should be in either linear or log scale;
- The y axis should be positioned around the center left of the image;
- The x axis should be positioned around the bottom of the image;
- Only one data line should be present;
- The markers should have similar dimensions across all images.

4.4.2 Limitations

The algorithm achieved good results with the designated dataset. Nonetheless, some limitations and areas for future work have been identified.

First, this approach is only suitable to digitised charts with no noise. If the image is of low quality or if it has been exposed to any process that added noise to the signal – through the use of a scanner, for instance –, the efficacy of the processing is hindered. Besides, the algorithm is not tolerant to the presence or absence of unexpected elements in the image beyond those previously defined.

The process of identifying the labels on the axes and recognizing their values depends on many different factors: the font used should be legible, the labels should be parallel to the edges of the image and the neighbouring digits of each label should be closely positioned. Optical recognition of printed characters is a field that has already achieved great results, but a few extreme variations in font type and rotation is enough to confuse an advanced OCR engine.

The algorithm also depends on finding a set of appropriate parameters to process

the whole dataset. The marker detection step depends on the size of the window used in the window sliding process. The same size is used for all the inputs, thus the markers have to be similar in size for all images. The algorithm receives three parameters that should be set according to the characteristics of the dataset, which are: 1) the size of the window that will scan the image looking for markers; 2) the maximum size of a digit and 3) a threshold for the distance which determines whether two digits are close enough to belong to the same label.

Finally, and maybe most importantly, the processing time of this approach is high, with a 640x480 image taking around 40 seconds to be processed in a standard personal laptop, which makes it unsuitable for real-time scenarios. This is a direct consequence of the the window sliding step that is used to find the markers, which consists of a full scan of the image.

Chapter 5

Second Case Study

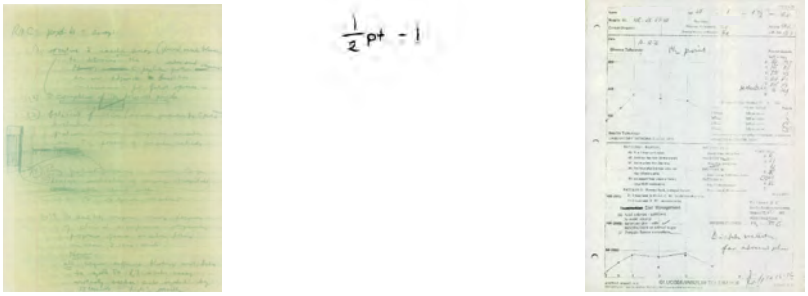
In the second case study, the experiment is conducted on a dataset containing scanned medical documents. The main distinction between the first and the second case studies is the nature of the medium: while the images in the first dataset have no noise, the ones in the second have a considerable amount. Furthermore, the elements present a more unpredictable nature as these charts contain both machine-generated forms and handwritten data. The dataset, experiment and results are described in the following sections.

5.1 Experimental Dataset

The dataset used in this experiment is from the medical domain and contains the results of 478 oral glucose tolerance tests (OGTT) collected in 1977 by Dr Joseph Kraft from St Joseph's Hospital in Chicago IL, USA. The tests tracked the plasma levels of insulin, glucose and c-peptide following the ingestion of a glucose drink. This dataset is unique in its nature because it has a reasonable sample size, with all the forms presenting the handwriting of a single person. Furthermore, they have a consistent pattern, which allows for the consistency of graphical elements.

The data set is currently securely housed by Auckland University of Technology in both physical and electronic formats. The sheets containing the data were made available in electronic format and the relevant information is contained in handwritten charts and text.

The full data set contains 509 records, but some had to be excluded. Thirty one of them were out of scope as they were not form sheets, while three presented problems during the scanning process. Figure 5.1 displays three examples of excluded items.



(a) Notes

(b) Blank sheet with a few notes

(c) Issues with the scanning process

Figure 5.1: Sample of items excluded from the dataset (names redacted).

All the form sheets used in the experiment have similar characteristics. The paper is white, while the form grid and lettering are green. They have three vertically-aligned black circles on the left, made by a paper punch. The grid is divided into three sections: the header, the first chart area and the second chart area.

The header has twelve fields in it. Most of the time, the majority of the fields are empty. Those which are usually filled are the name, age, sex, height, weight and room. The remaining fields are the hospital number, clinical diagnosis, physician, previous tolerance, family history of diabetes and lab number. This section is filled out with blue ink.

The first chart area is right below the header and it has four elements: a date, a data line, an optional note and the legend. This chart holds the results for the glucose

exam. The date refers to when the exam was conducted. The data line holds all the measurements and it can have from 5 to 8 points. The legend refers to the measurements and it describes the results textually. Most information is written with blue ink, except for the data line which may be drawn using red ink.

The second chart area is below the first chart and it holds the results of either an insulin assay and a c-peptide test conducted concurrently or the insulin assay conducted twice on different dates. This area has four main elements: two data lines, an optional secondary scale for one of the data lines, and a legend with textual information regarding the data lines. The lines are drawn using distinct ink colours to differentiate between them. Figure 5.2 displays a sample form.

The images were digitised in high resolution and have approximately 4000x2800 pixels. They are not fully vertical due to the limitations of the manual scanning process. This process adds a considerable amount of noise to the images, interfering with the colour of all the pixels.

5.2 Experiment

The goal of the experiment is to extract the data points from both charts. There are a few key differences between the first and the second case studies:

- The image to be processed is larger. While an item from the first dataset had 480x640 pixels, an item from the second has around 4000x2800 pixels;
- This case study takes noise into account as the scanning process influences the colour of every pixel;
- The elements are less homogeneous and predictable. The items that need to be extracted are not similar across images as they were drawn by a human.

Thus, for the reasons listed above, the previous algorithm could not be used for

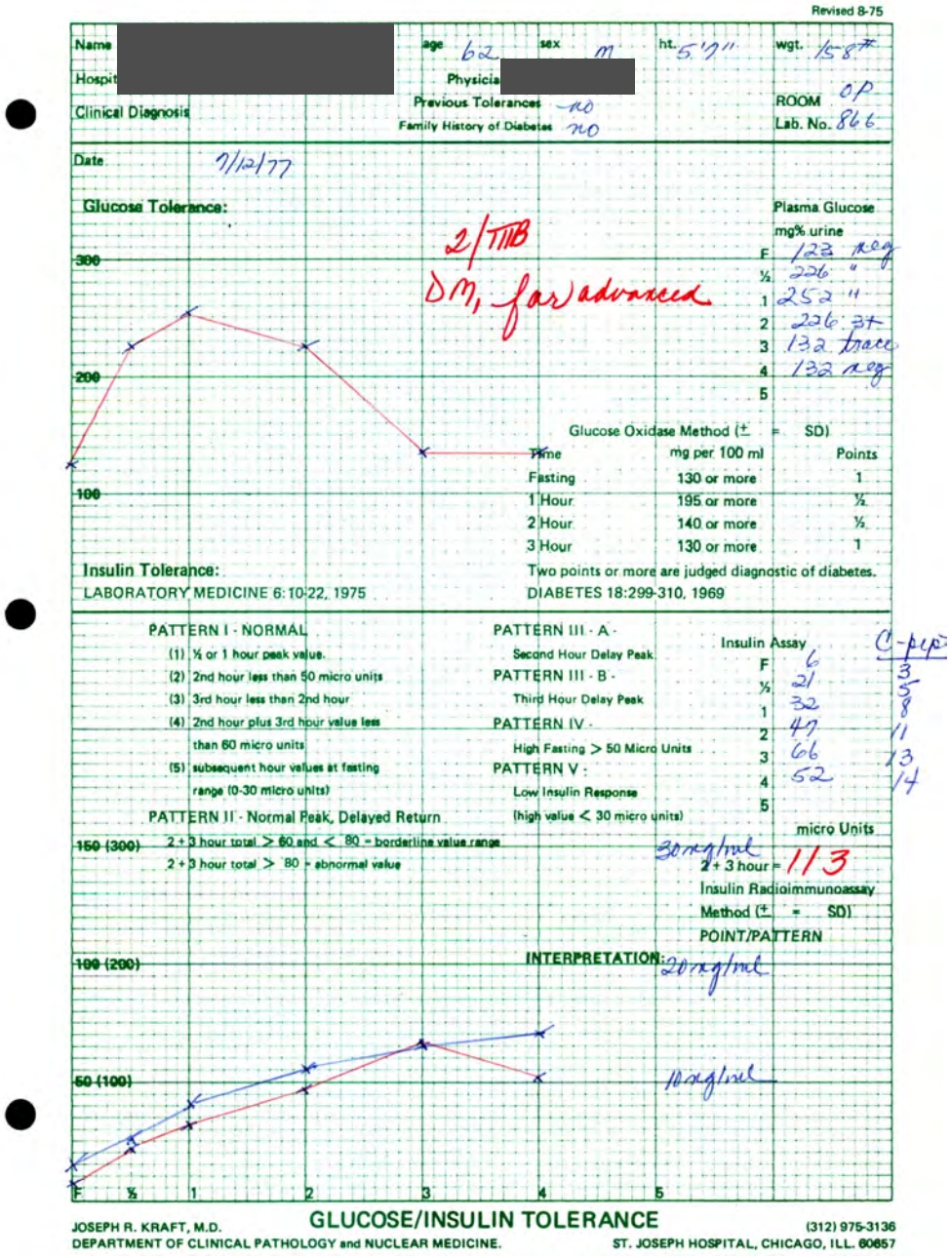


Figure 5.2: A sample form sheet for the second case study (names redacted).

this scenario. Overall, this case study follows a similar logic to the previous one by incrementally detecting pixels which are not relevant to the end goal – detecting the positions of all markers. The experiment is comprised of seven steps, which are listed below:

- Straightening: in order for the marker extraction to work, the input cannot be tilted, which is usually the case for scanned documents. Thus, the image is straightened before any further processing takes place;
- Cropping: the cropping eliminates the margins so that the grid can be used to calculate the markers positions in a later stage;
- Detection of handwritten content: the pixels that comprise the handwritten content are detected and marked for later processing;
- Location of elements: the pixels that belong to the same element (e.g.: data line for the top chart) are grouped together;
- Classification of elements: The elements identified in the previous step are labeled according to predetermined classes;
- Marker extraction: The marker positions are extracted with the pixel grid as a reference;
- Scale conversion: the marker positions are converted based on the actual scale used in the charts.

These stages are described in detail in the following sections.

Colour Space

The experiment was conducted using the HSV colour space. A colour space is an organization of colours, especially for digital representations. The HSV colour space is defined by different combinations of hue, saturation and colour values. This colour space was chosen because it resembles the way the human vision perceives colour, which makes it more intuitive to work with colour ranges and gradations.

5.2.1 Straightening Original Form Sheets

Even though there was an effort to make the images straight during the scanning stage, the digitised versions ended up slightly tilted. This is an issue because it interferes with the extraction of the final values for the markers as the conversion depends on the pixel grid and the actual chart scale being aligned. If they are not aligned, the conversion between the two scales is imprecise. Figure 5.4 illustrates the aforementioned scenario.

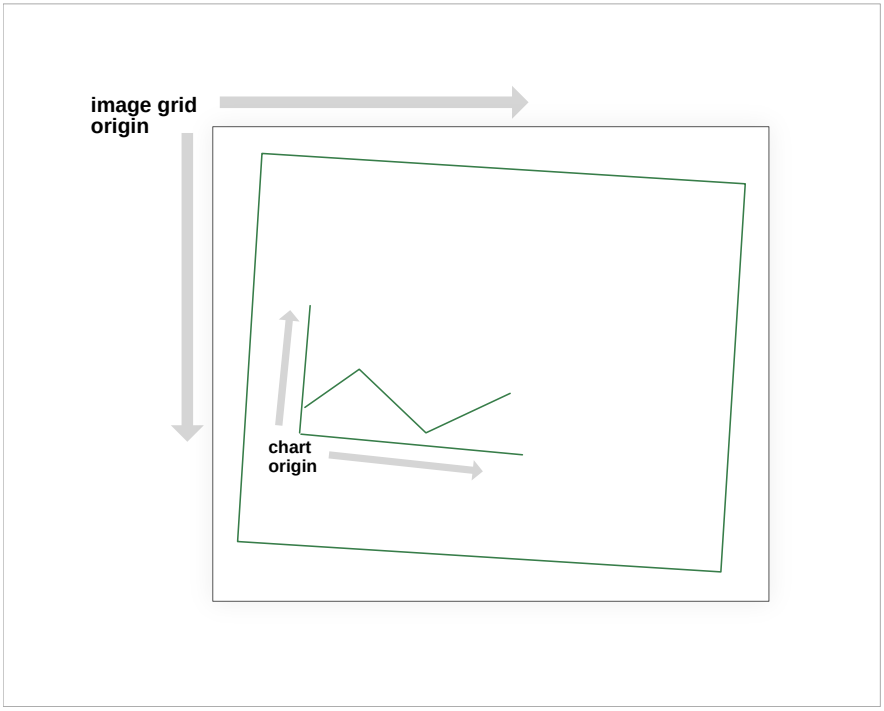


Figure 5.3: Misalignment between the pixel grid and the chart scale.

In order to straighten an image, two steps are taken: first, a small region of the footer that is expected to be constant across all images is selected. Then, the content of this region is used to calculate the angle for the rotation of the whole image back to the straightened position.

Region Selection

The data set was analysed in order to find a region that was fairly constant across all images. The criteria used was that 1) the region should not have any handwritten content in it and 2) it should be present in all samples.

Based on this criteria, the address region in the footer was selected. This selection is shaped as a 116x1114 rectangle. Figure 5.4 displays the portion of the image that is selected.

Figure 5.4: Region selected for the straightening process.

Once the region is selected, its contents are used to determine the rotation angle.

Rotation

The selected region is expected to contain the address from the footer. If the image is tilted, The address will be tilted accordingly. Thus, it is possible to use the rotation of the address to determine the rotation for the whole image.

In order to obtain the rotation angle, a few steps are taken. First, the selected region is converted to grayscale. Then, this grayscale image is binarised using the OTSU thresholding (Otsu, 1979). This type of thresholding is different from the binarization methods used previously. Instead of using a global threshold value for all the pixels, it automatically determines a value based on the histogram of the grayscale image. If the image has two peaks, which means that it has two dominant colours, the threshold will lie in-between them. Figure 5.5 illustrates the OTSU thresholding mechanism.

This approach deals well with noise and it is a perfect fit for the scenario where two elements need to be separated: the foreground and the background. Once the binarization is performed, the background is white and the foreground is black.

Figure 5.5: Address region after OTSU thresholding.

Next, a rectangle is drawn around the address. To accomplish this task, the NOT bitwise operation is applied to the image, turning all foreground pixels – the ones that belong to the characters – white and the remaining ones, black (Figure 5.6).

Figure 5.6: Address region after not operation.

Then, all the white pixels are located, and a minimum area rectangle is drawn around them (Figure 5.7).

Figure 5.7: Text surrounded by minimum area rectangle.

The rectangle will have the same rotation as the text and, consequently, as the whole image. Once the rotation value is identified, it can be used to de-skew the image as illustrated in Figure 5.8.

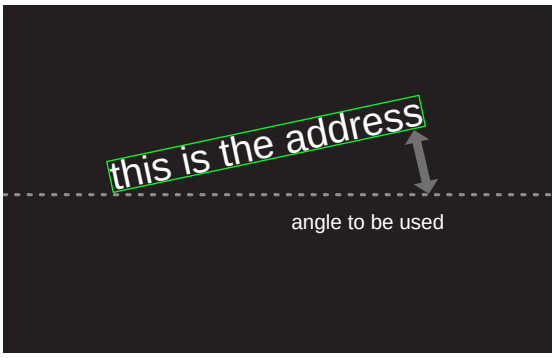


Figure 5.8: The angle used to determine the rotation.

The resulting angle is then utilised to rotate the entire image. The affine operation

is applied, with the center point of each sample and its respective rotation angle as the parameters for the operation. The affine operation is a geometric transformation technique that uses three reference points to make changes to an image. It can rotate, scale, translate and reflect an image, among other operations as illustrated in Figure 5.9. The sample is then rotated and the final output of this stage is a straightened image.

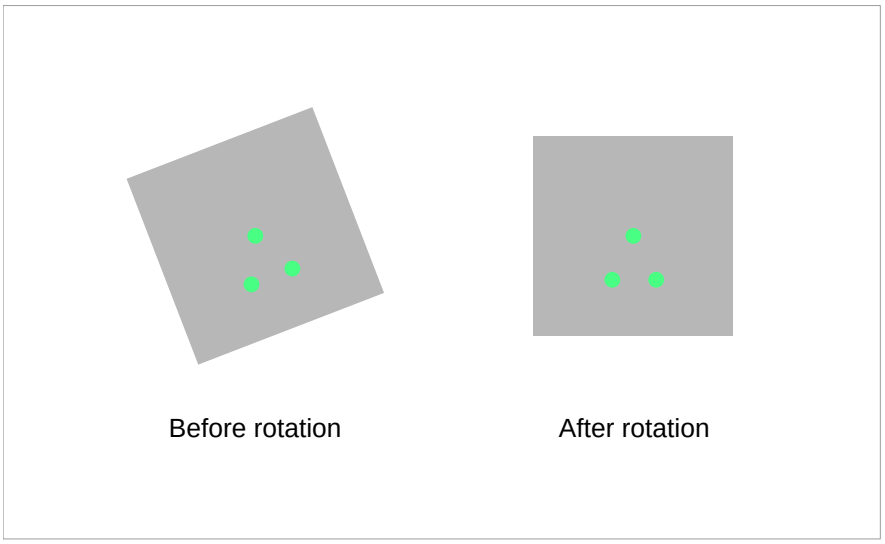


Figure 5.9: Affine operation using three reference points for rotation.

5.2.2 Cropping

The content to be extracted is fully contained within the grid. In this step the straightened image is cropped so that the white margins and the footer are removed. After that, all samples are normalised to the same dimensions. The normalization occurs because the grid is used as a reference to create the scale converters and to extract the markers.

The grid is the largest connected green component in the image. In order to locate its position, three steps are taken. First, the image is binarised, with the green pixels becoming white and the remaining, black. Then, the sample undergoes two morphological operations: closing and opening so that the white regions are more clearly defined (Heijmans, 1994). Finally, all the contours surrounding the white

regions are detected and the biggest one is used to crop the image. The following sections detail each step.

Binarization

The first step is to binarise the image by turning the green pixels into white pixels, and the remaining into black pixels. This is accomplished by defining a HSV colour range that captures the green colour present in the data set. The samples were manually analysed to determine the range to be used.

The range used is defined in the HSV colour space. Each one of the three channels can hold values between 0 and 360. The range used is defined below:

- Hue: 50 - 110.
- Saturation: 0 - 255.
- Value: 0 - 255.

The above range encompasses all greens from darkest to lightest. The pixels that match this range are turned white, whereas the ones that do not match become black. Figure 5.10 displays the resulting binarised sample.

Morphological Operations

The resulting binary image highlights the green pixels in white. The grid shape is clear against the black background, but a closer inspection reveals one issue. The noise of the original image interfered with some pixels that belong to the grid. As a result, the pixels were not identified as green, which generated discontinuities in the grid shape.

Another factor contributed to the discontinuities. The handwritten content was painted black in the binary image, which creates holes and breaks up the grid. In a later step, the detection of the grid location will depend on it being fully connected. Thus, the image must be transformed to get rid of the these small holes.

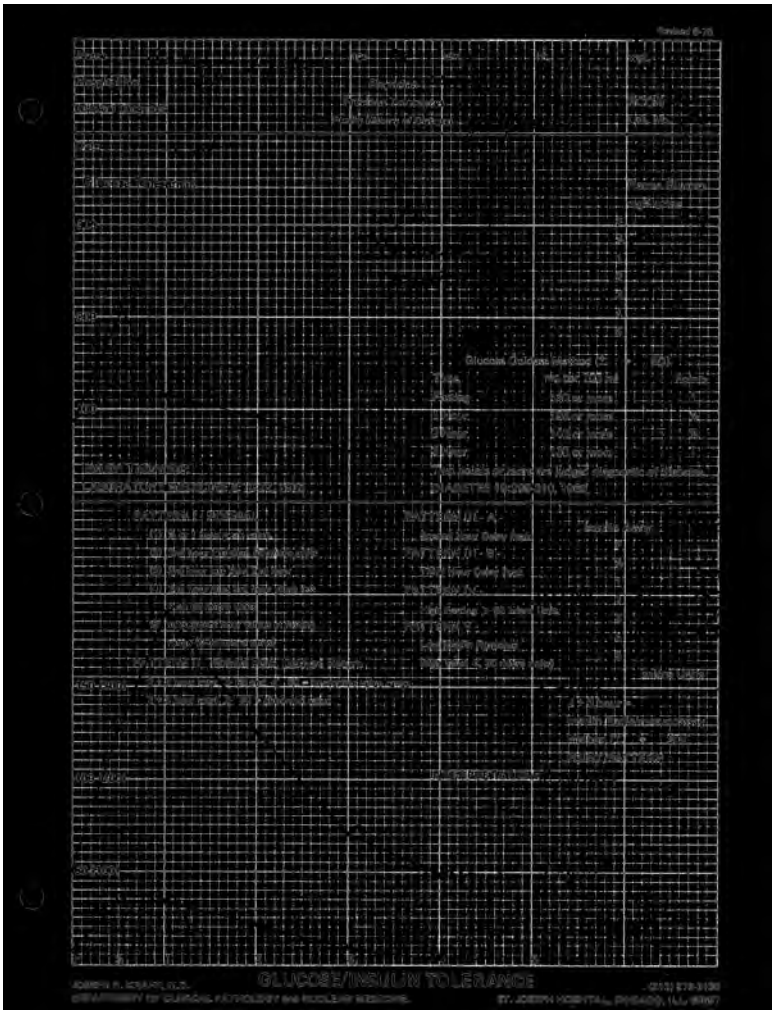


Figure 5.10: Sample sheet after binarization. Green pixels in white and the remaining in black.

Two morphological operations are performed in order to connect all the grid regions: opening followed by closing. These operations were used and explained in the previous case study. The closing operation aims to make the grid lines thicker and connected and it is comprised of two steps. The image is first dilated so that the parts can expand and connect. Once connected, it is eroded so it goes back to its original thickness. This operation has been performed using a kernel of 10x10. Figure 5.11 displays the workings of the closing operation.

The opening operation aims to get rid of some of the remaining noise. It is an

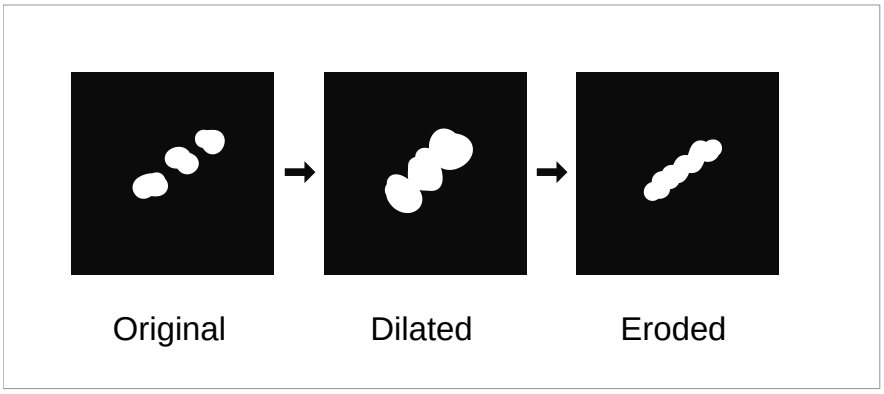


Figure 5.11: Schematic representation of the closing operation. This procedure overcomes discontinuities.

erosion operation followed by dilation which eliminates small white spots. Figure 5.12 displays the workings of the opening operation.

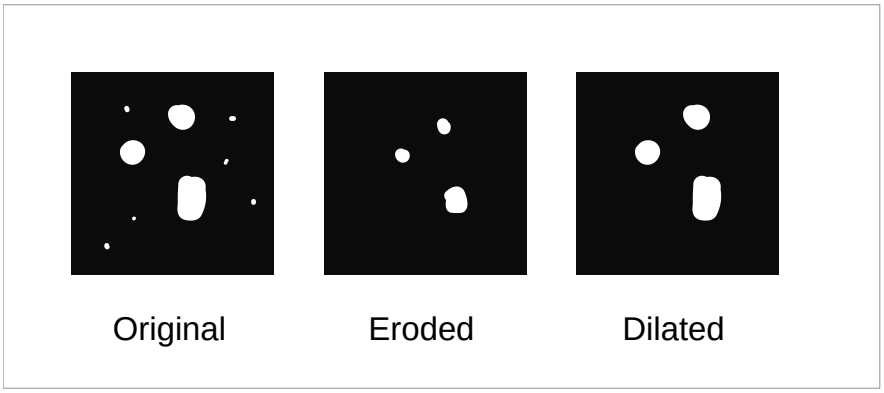


Figure 5.12: Schematic representation of the opening operation. This procedure removes noise on the background.

Contour Detection

In order to find the grid, the biggest connected white component has to be located in the binary image generated by the previous step. The contour detection technique is used to accomplish this goal.

Contour detection was explained and used in the previous case study. A contour can be described as a line surrounding all the continuous pixels which have the same color

or intensity. The biggest contour in the binary image belongs to the grid, so once this contour is located, the grid position is known.

A contour, programmatically speaking, is represented by a list of points and their respective x and y coordinates. These values are used to determine which region of the original image will remain. The image is cropped based on this line, and the pixels which are not inside of it are discarded, which completes the cropping.

Once the steps above have been completed, the cropped image is resized to 4030x2820 pixels. This step is taken so that the image sizes are normalized and the same scale converters are used across the data set. Figure 5.13 displays the resulting cropped sample.

5.2.3 Detection of Handwritten Content

The input to this stage is the cropped image, and the goal is to narrow it down to the pixels carrying the most relevant information for processing; the ones that comprise the handwritten content. This is accomplished by identifying and marking pixels that belong to the background.

The approach taken is similar to the previous case study, but with a significant difference. While the first case study relied on a single colour value to perform this step, the background in this scenario has noise and more than one colour. Due to these two factors, a few colour ranges are used instead of exact values.

In order to determine which colour ranges would be used in the detection process, the data set is manually inspected. To goal is to define ranges that are general enough without the risk of discarding relevant information. Three different ranges are defined:

- White: the white regions correspond to the paper in which the forms were printed;
- Green: the green regions correspond to the grid and the printed characters;

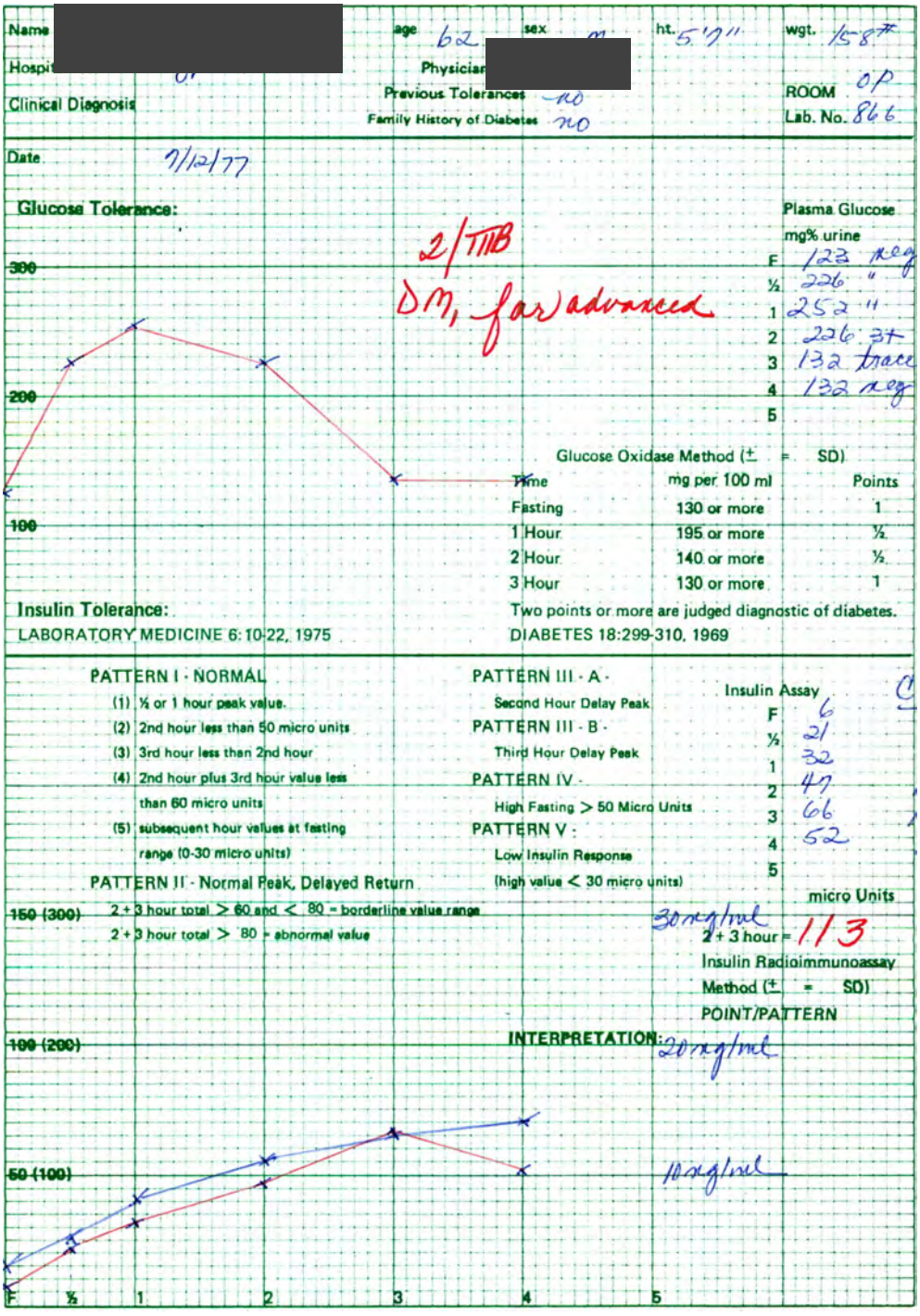


Figure 5.13: Sample image after cropping. The margins and footer have been removed (names redacted).

- Yellow: the documents have been stored for decades, so the yellow range has to be accounted for due to the aging of the material.

The colour space used to define the ranges is HSV. The values used for each channel are detailed in the table 5.1.

Table 5.1: Colour ranges used to detect the background.

	Green	**White**	**Yellow**
Hue	(50-110)	(0-255)	(0-255)
Saturation	(0-5)	(0-255)	(0-255)
Value	(10-50)	(0-255)	(0-255)

Procedure

The pixels with colours that lie within any of the defined ranges are painted black. Black is the colour used throughout the experiment to mark pixels that are not relevant to the end goal.

This approach depends on the colours being fairly homogeneous across the data set, which is the case for this case study. Figure 5.14 displays the result of this step.

5.2.4 Location of Elements

In order to extract the markers, the location of the data lines must be known. So far, the images have been analysed from a pixel perspective, but looking at individual pixels is not enough to identify objects. In this step, neighbouring elements are grouped together so they can be interpreted as objects in the image.

The resulting image from the previous step (figure 5.14) has a series of elements drawn by hand against a black background. These elements can be any of the items listed below:

- Top chart (data line only)
- Bottom chart (data lines only)
- Header info (room, lab number, gender...)

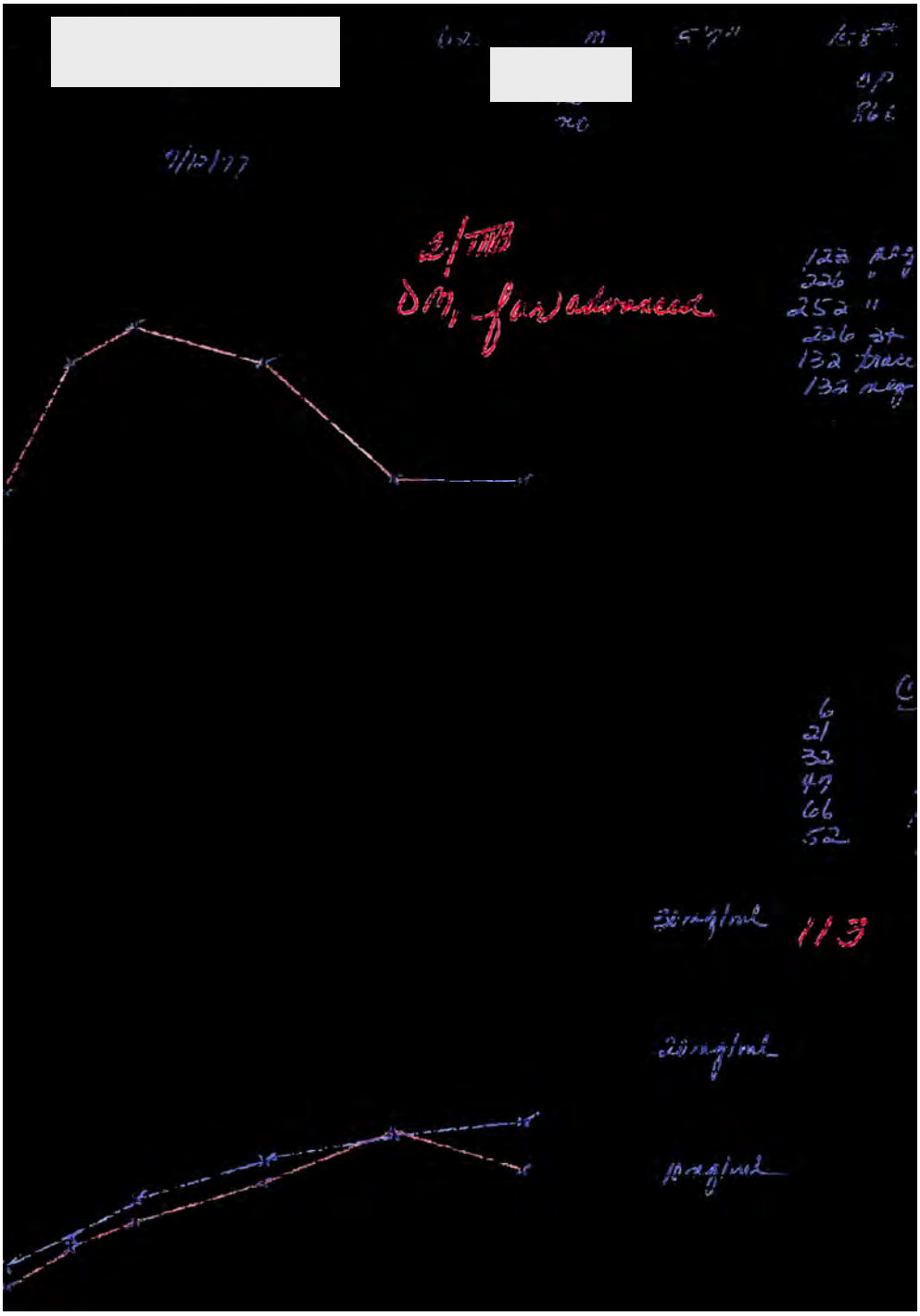


Figure 5.14: Sample form with background removed (names redacted).

- Top chart legend
- Bottom chart legend
- Notes

- Bottom chart hand-drawn scale

Proximity determines which pixels belong to the same object, not colour. Thus, the first step is to make a copy of the image and binarise it. The pixels which are not black are turned white. Figure 5.15 displays the result of this operation.

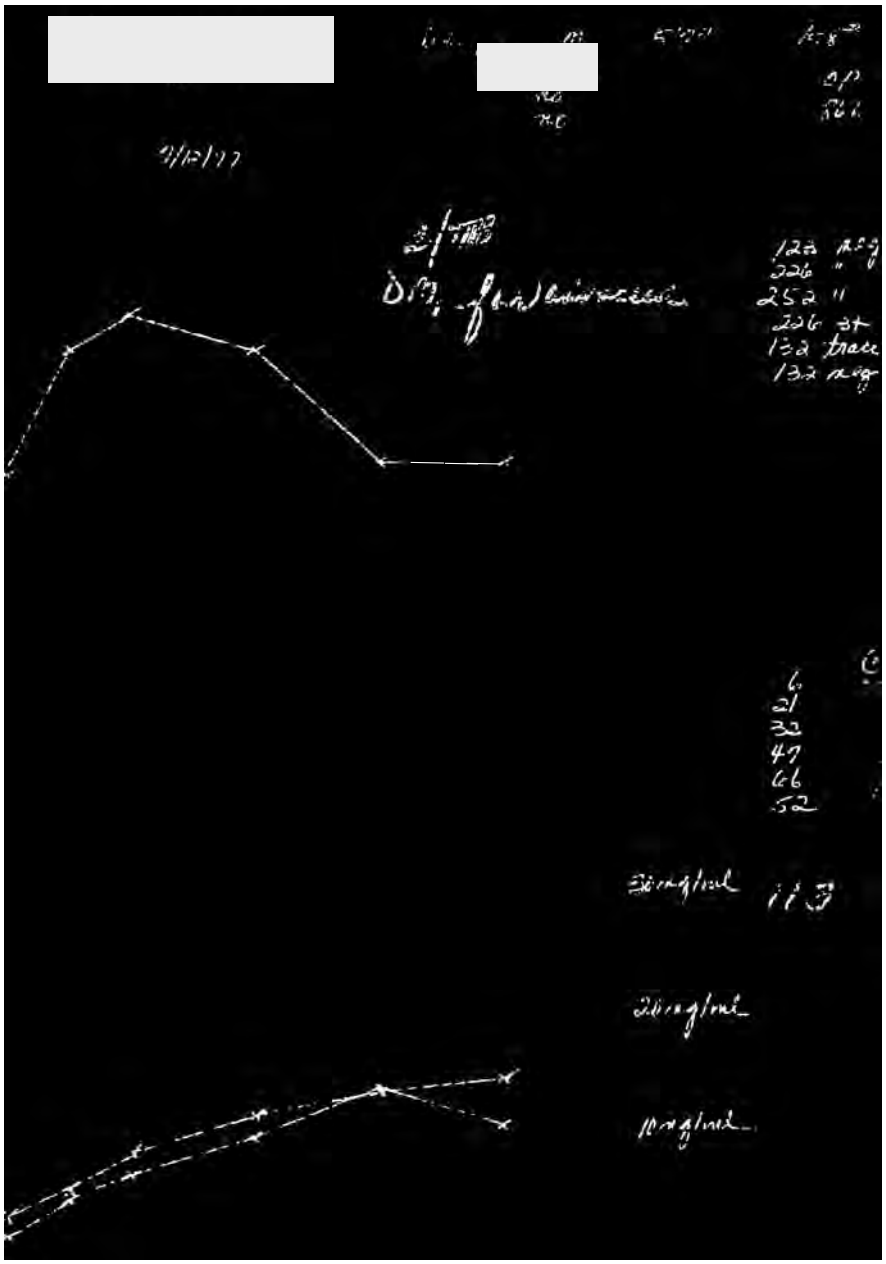


Figure 5.15: Binarized sample (names redacted).

Then, the binarised copy is dilated with a 45x45 kernel so that the discontinuities

are removed and neighbouring elements are merged together. Figure 5.16 displays the result of this operation.

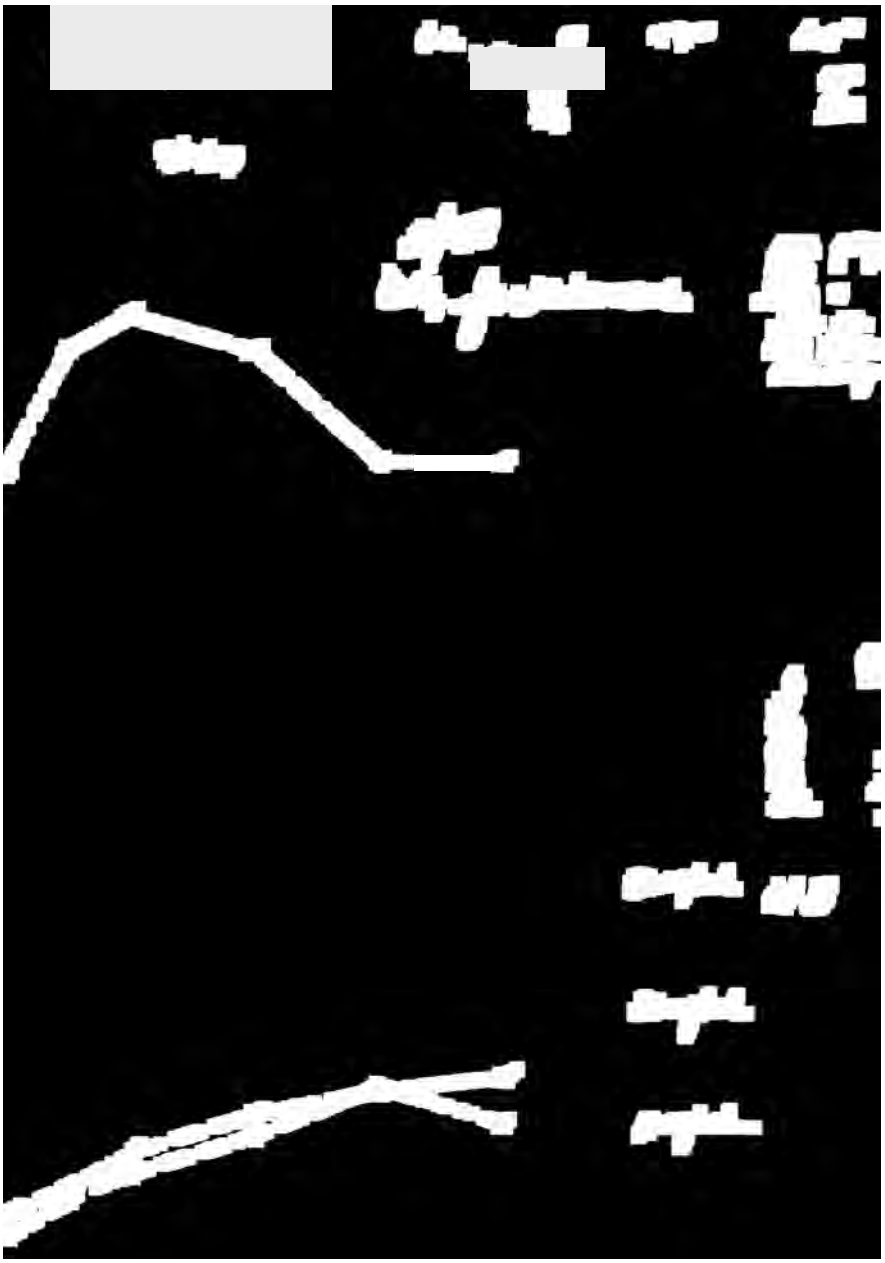


Figure 5.16: Form dilated after binarisation (names redacted).

5.2.5 Classification of Elements

The items that are relevant to the marker extraction are the data lines contained in the top and bottom charts. Figure 5.17 displays their locations in the image generated in the previous step.

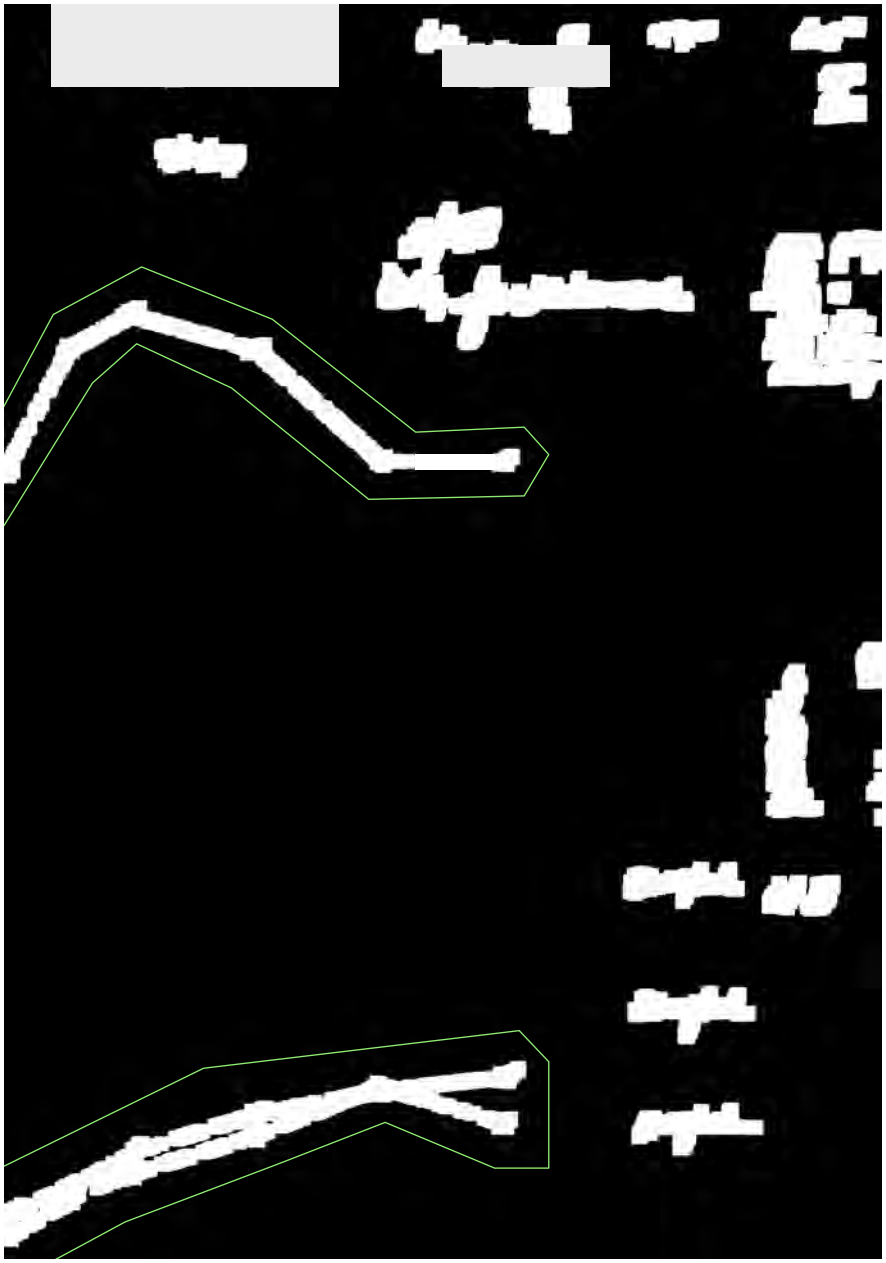


Figure 5.17: Regions relevant for processing highlighted in green (names redacted).

While it is easy for the human eye to identify them, it is not clear for the machine

which of the white blobs correspond to the regions of the data lines. Thus, a strategy is devised to make them delineate data lines from other types of objects.

First, the data set is manually analysed to understand in what regions the lines usually are. This is no simple task as there is considerable variation across the samples. Moreover, the regions to be defined cannot have any other elements other than the data lines. By following this reasoning, a region is defined for each chart as displayed by figure 5.18.

The contours that intersect with the first region are classified as "chart 1", whereas the contours that intersect with the second region are classified as "chart 2". The output of this step are the contours that intersect with the regions and their respective labels.

5.2.6 Extraction of Markers

The goal is to extract the y value of each marker in relation to the pixel grid. In the previous case study there was only one data line, but in this one there are two data lines for the bottom chart. Thus, the charts have to be processed separately and with different strategies.

The labeled contours generated in the previous step are used to create two distinct images. First, the cropped image without background (figure 5.14) is copied twice. The contours labeled "chart one" are used to define what pixels remain in the first copy, while the contours labeled "chart two" are used to define what pixels remain in the second copy. Figure 5.19 displays the resulting images.

Top Chart

The process used to extract the markers for the top chart is inspired by the first case study. In both cases, there is a single data line, and both present similar spatial characteristics. Thus, a similar method is applied.

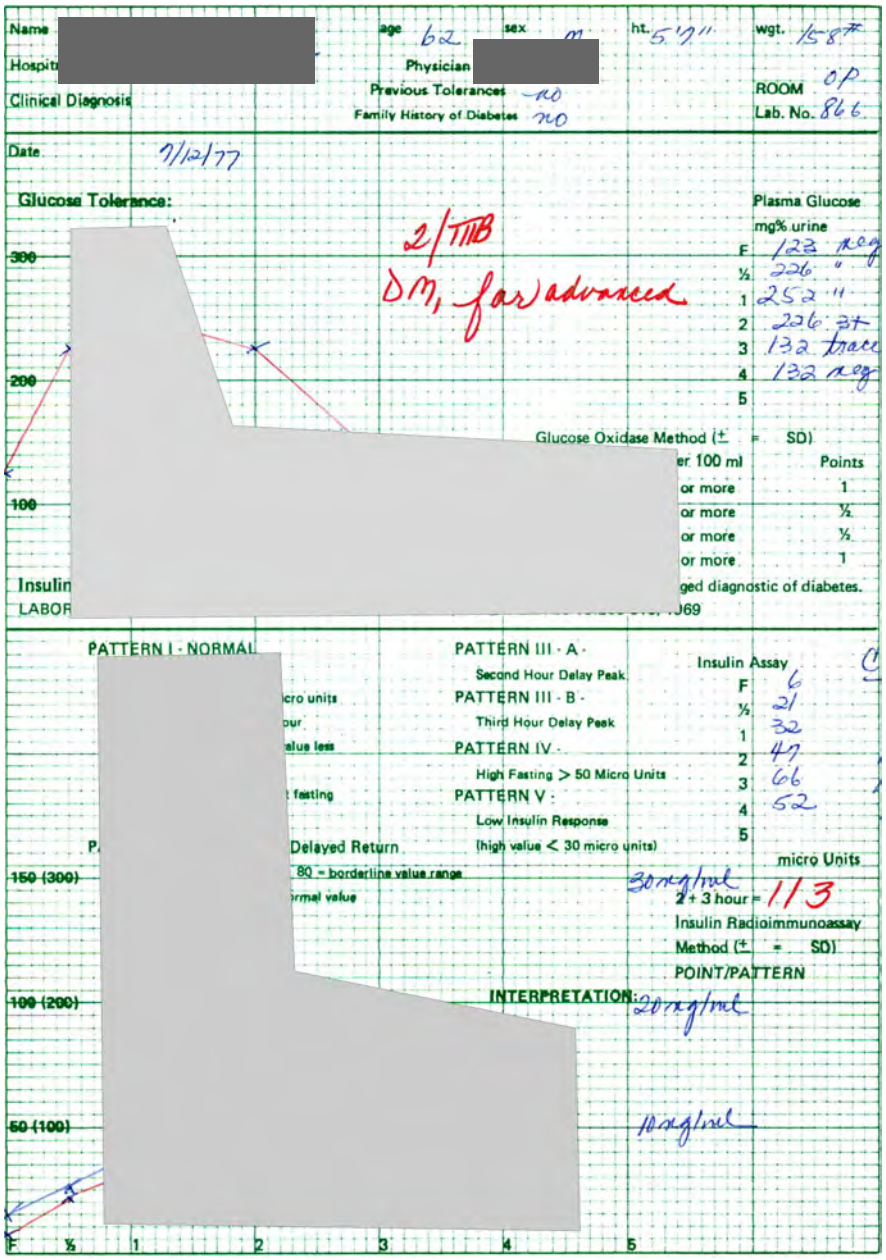
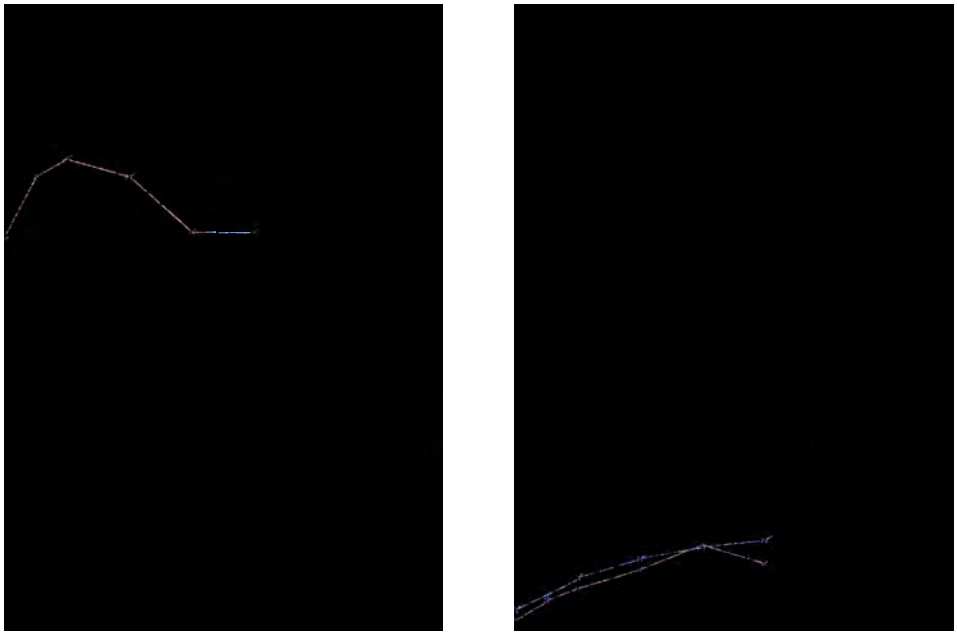


Figure 5.18: The light-grey polygons represent the regions in which the charts are usually drawn.

First, the image is binarised by painting in white the pixels which are not black. Then, the opening operation – erosion followed by dilation – is applied with a 9x9 kernel. This operation is used to remove the line and keep the markers.

Finally, the contour detection operation is applied. At this stage only the markers



(a) Top chart (b) Bottom chart

Figure 5.19: Images generated for the processing of each chart.

are present, so this operation will find their contours. Once they are found, the centroid of each contour is detected. The output of this step is the centroid values, which are the marker positions in relation to the pixel grid. Figure 5.20 illustrates this process.

Bottom Chart

The bottom chart has two lines. One line is always blue with blue markers, while the other line is either blue or red with red markers. By using the algorithm applied to the top chart, it is not possible to identify to which line a marker belongs. Thus, another approach has to be devised for this scenario. The first step is to narrow down to the chart area. This is accomplished by copying the chart region to another rectangular image. Figure 5.21 illustrates this process.

The data points have been collected at regular intervals. The first measurement was taken thirty minutes after the exam. The second, after one hour. The third, after two hours, and so on. The charts may have five to eight measurements, which may result in

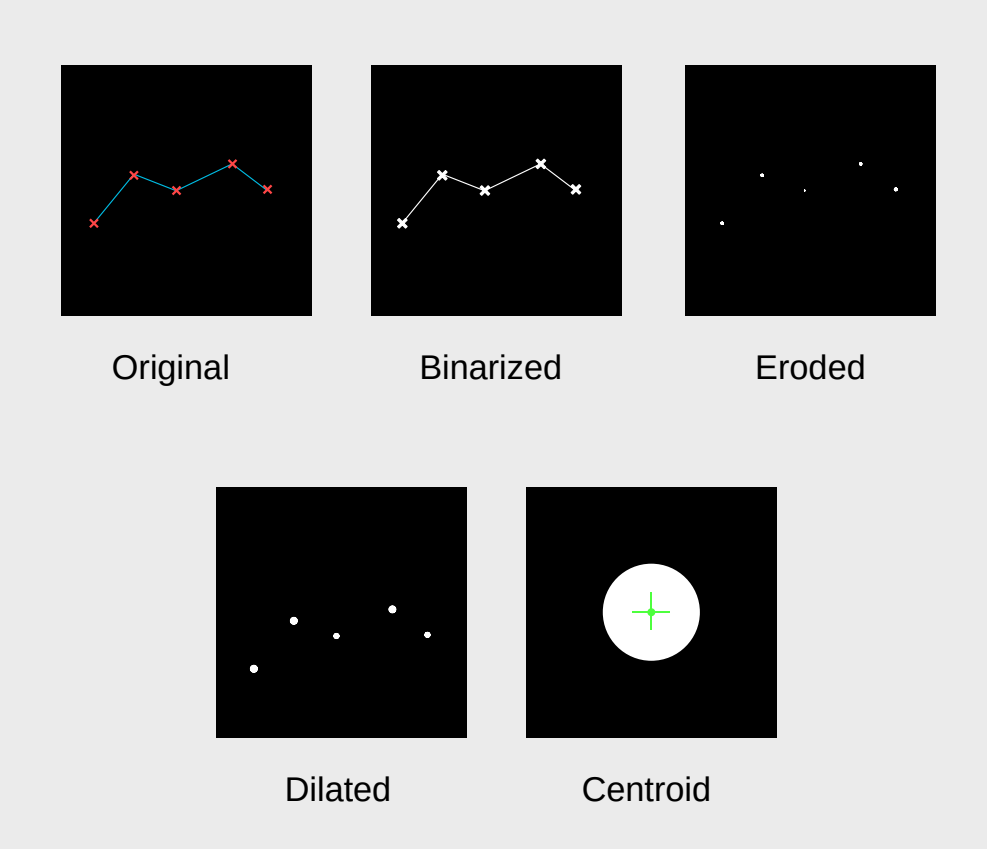


Figure 5.20: From top-left to bottom-right: process used to extract the marker values in relation to the pixel grid. The goal is to find the centroid of each marker.

up to 16 markers in total for the bottom chart.

The markers always appear in roughly the same vertical strips. The first step is to cut these strips off the smaller image, which may be wider or narrower depending on the number of measurements.

The strips are set to be 50 pixels wide. This value was based on a manual inspection of the data set. They start at columns 25, 210, 407, 805, 1208, and optionally 1605, 2005 and 2405 if markers are present. The height of all strips is equal to the height of the small image. The strip can be in one out of two states: one marker (markers intersect) or two markers (markers do not intersect). Figure 5.22 displays the location of the strips.

Each strip is processed separately. The strip is binarised, which turns foreground

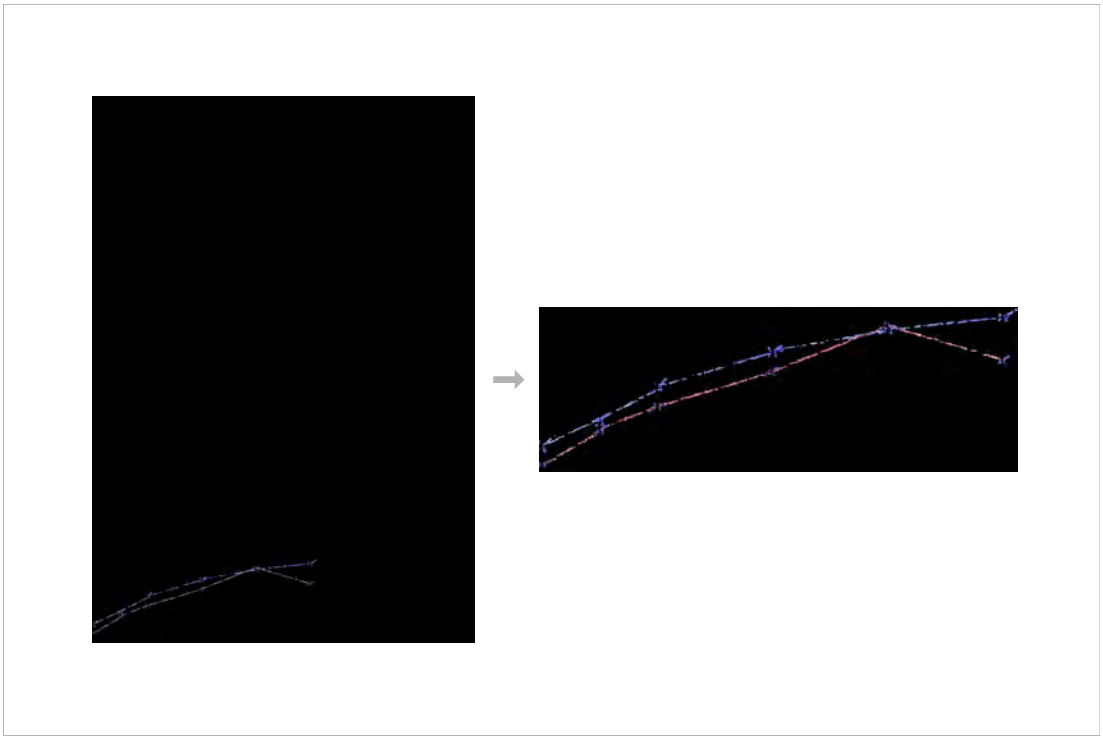


Figure 5.21: Chart is copied to a new rectangular image that rightly fits the chart region.

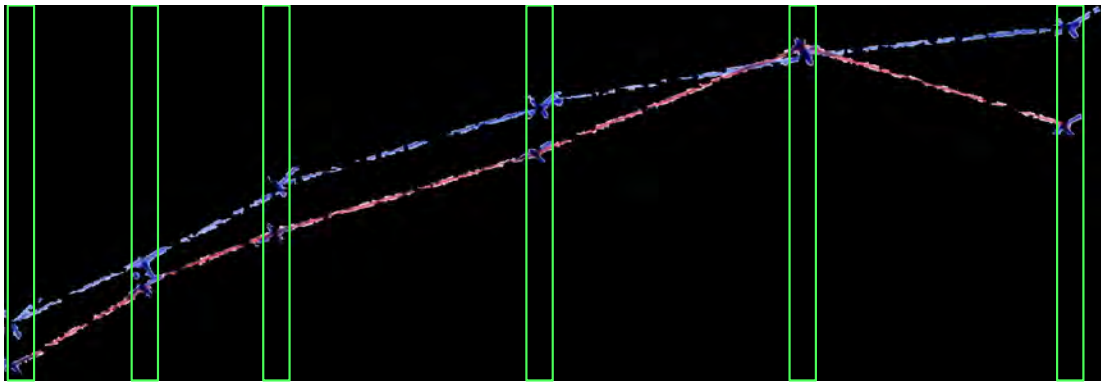


Figure 5.22: Strips used for processing highlighted in green.

pixels white and background pixels black. Next, its contours are detected. If only one contour is found, then the data lines are intersecting and the centroid of the contour is attributed to both markers. If two contours are found, two distinct markers are present. Figure 5.23 displays the two possible states of a strip.

The second case, though, is a little more complex. To determine to which data line each of them belong, the region inside one of the contours is checked against the

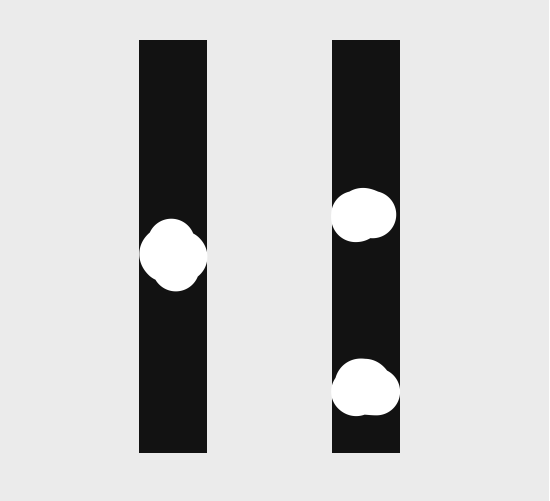


Figure 5.23: Sections of two strips displaying the possible states. The strip on the left is the result of an intersection while the one on the right presents two data points.

original strip. If at least 5% of the pixels are within the red colour range, then this contour represents a marker for the data line that has red in it. If it doesn't, then it belongs to the blue data line. If one marker is classified as belonging to the red line, the other belongs to the blue line. If it is classified as belonging to the blue line, then the other belongs to the red line. This logic is detailed in the algorithm outlined in Figure 5.24.

The procedure above finds the marker locations relative to the smaller image grid. In order to find their absolute positions, an extra step is necessary: the x and y coordinates of the origin of the smaller image in relation to the original image must be added to the x and y coordinates of the markers, respectively.

The output of this step is a list of markers for the top chart and two lists for the bottom chart. The lists for the bottom chart are separated based on the lines the markers belong to. The coordinates are in relation to the pixel grid.

```
forEach strip in strips:
   marker1 <- empty // red data line
   marker2 <- empty // blue data line
   binarised <- copy and binarise strip
   contours <- find contours of binarised copy
   if contours has 1 item:
      //(markers are intersecting)
      marker1 <- centroid of the only item
      marker2 <- centroid of the only item
   else if contours has 2 items:
      //(strip has two markers)
      contour <- any item from contours
      if red pixels > 5% of total area of the contour:
        marker1 <- centroid of contour
        marker2 <- centroid of other contour
      else
        marker1 <- centroid of other contour
        marker2 <- centroid of contour
    else
      //(error)
```

Figure 5.24: Algorithm that scans each strip for markers to determine to which data line each marker belongs.

5.2.7 Scale Conversion

As it was explained in the first case study, an image is represented by a grid of pixels with the origin at the top-left corner. The x dimension grows to the left and the y dimension grows to the bottom.

In the first case study, there were only two axes, one horizontal and one vertical for the x and y values, respectively. This resulted in two scale converters. In this dataset, each form has three scales: One is used for the top chart and the the other two are used in the bottom chart. The bottom chart may or may not use the same scale for both lines. Another difference is that, in the previous scenario, the x and y values were extracted for each marker, while in this scenario only the y value is relevant. Figure 5.25 displays where the scales are located in a sample.

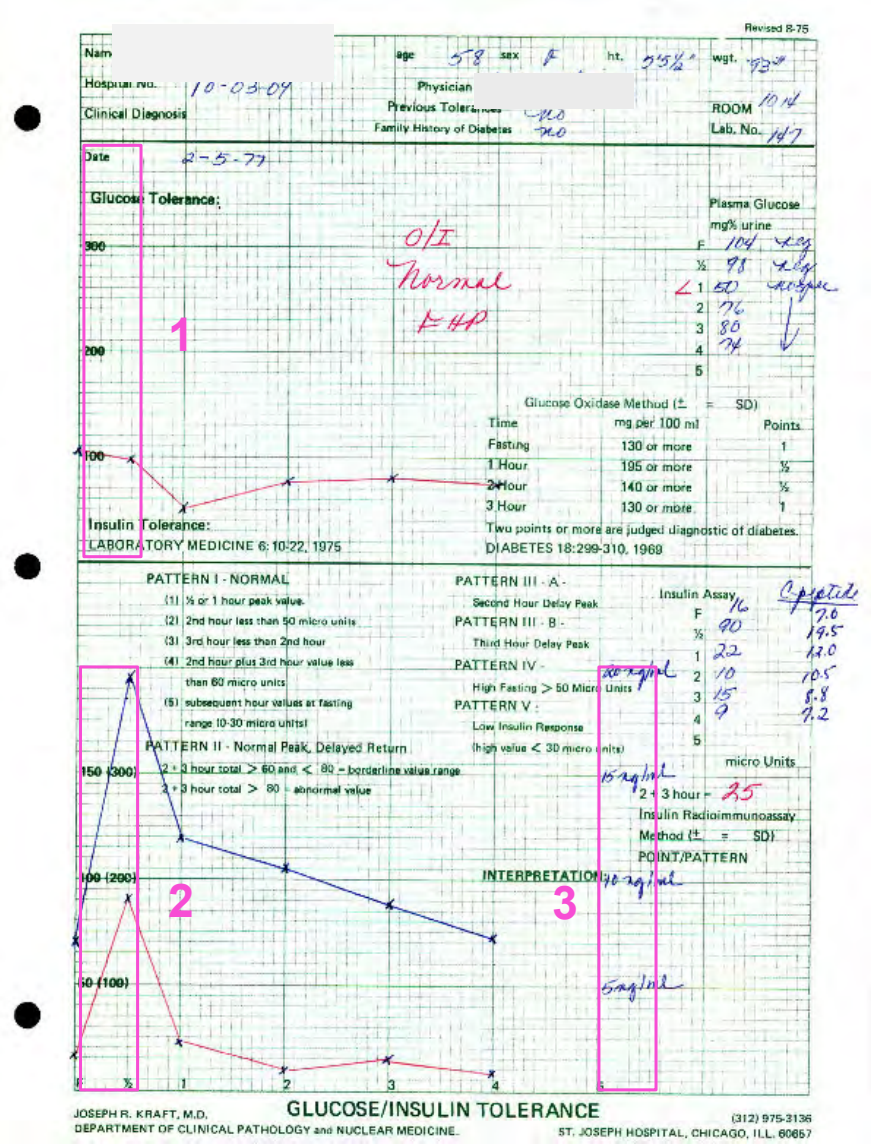


Figure 5.25: Location of the scales highlighted in pink (names redacted). 1. Top Scale; 2. Bottom Scale 1; 3. Bottom Scale 2.

The markers positions are known in relation to the pixel grid. Thus, it is necessary to convert between the pixel scale and the chart scale to extract the final values. In the previous case study, the converter dynamically extracted the scale for the axes by inspecting and recognizing the numerical labels. In this case study, the scales are passed as parameters before the processing takes place.

Three elements are used to build each scale converter and extract the final value:

two points and the marker. Each point provides two pieces of information: the y value in relation to the pixel grid and in relation to the chart scale. The marker provides only the y value in relation to the pixel grid. The elements used in the converter formula are as follows:

- PixelRange: the absolute difference between the highest and lowest values in the y-axis in relation to the pixel grid ;
- ActualRange: the absolute difference between the highest and lowest values in the y-axis in relation to the chart scale;
- PixelMax.Y: the lowest values in the y-axis in relation to the pixel grid;
- Marker.Y: the y value of the marker in relation to the pixel grid;
- Scaler: The PixelRange divided by the ActualRange.

These values are used in the formula below. The final value is the marker value in relation to the chart scale.

$$finalValue = PixelMax.Y - (Marker.Y * Scaler) \tag{5.1}$$

Six different scales are used throughout the data set, two in the top charts and four in the bottom charts. These scales are listed in Table 5.2. A converter is created for each scale.

Table 5.2: Original scales contained in the forms.

Chart	**Scale Range**
Top	0-400
Top	0-800
Bottom	0-200
Bottom	0-400
Bottom	0-20
Bottom	0-40

Before any processing takes place, the data set is manually grouped according the

the combination of scales used in each document. As a result, eight groups are created. These groups are listed in Table 5.3.

Table 5.3: Groups created based on scales contained in the documents.

Group	Top Scale	Bottom Scale 1	Bottom Scale 2
Group 1	0-400	0-200	0-20
Group 2	0-400	0-200	0-40
Group 3	0-400	0-400	0-40
Group 4	0-800	0-200	0-20
Group 5	0-800	0-200	0-40
Group 6	0-400	0-400	0-250
Group 7	0-400	0-500	0-500
Group 8	0-800	0-200	0-250

The scale converters are passed as parameters for each group. The final values are obtained once this step is finalized.

5.3 Attempt at Extracting Values from the Legend

Each chart in the form has a corresponding legend. These legends hold the markers values. An attempt was made to use OCR – optical character recognition - to access these values. This approach was not further pursued because:

- The legend presents more variation and is more inconsistent than the data lines;
- The preliminary results were unsatisfactory, with around 40% of the values being misidentified.

The OCR tools used for this part of the experiment were the Google Vision API, Azure Computer Vision and Amazon Textract. These are cloud services provided by Google, Microsoft and Amazon, respectively. The best-performing tool was the Google Vision API, with around 60% of the values being correctly identified.

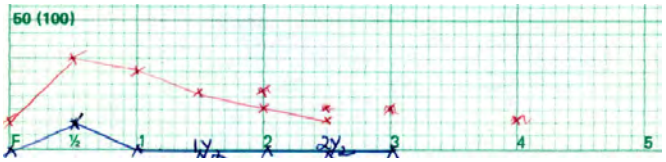
5.4 Results

The actual values of the markers were manually extracted to a spreadsheet prior to the experiment for 419 out of the 478 forms. These values were extracted for a different research project and by a different researcher. The spreadsheet was provided so that the results of this case study could be compared to the ground truth. Only 419 charts were provided because the aforementioned research project only required a subset of the dataset. In order to measure the performance of the algorithm, the actual value must be known so that it can be compared to the predicted value. Consequently, only these 419 samples were utilised. Each form contains two charts, which totals 838 charts. Out of the 838 charts, 506 had their markers successfully extracted. Figure 5.26 displays three sample scenarios in which the markers could not be obtained. When the extraction was not possible, it was due to one the following reasons:

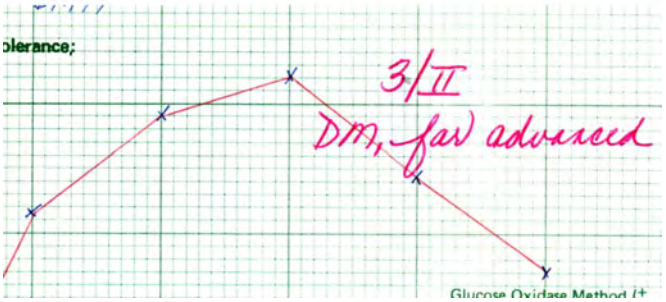
- Presence of unexpected elements (cross-outs and smudges) close to the data line;
- Discontinuities in the data line caused in the background detection step.

The metric used quantify the errors was described in section 3.2.1. The absolute difference between the actual and extracted values of a marker was calculated for all the extracted data points. Then, the result was divided by the range of their respective scales. These figures were then grouped by the scale they were based on and the arithmetic mean was calculated for each group. Only the charts from which it was possible to extract the markers were considered in the calculations. The metric was broken down according to the six scales present in the data set as presented in Table 5.4.

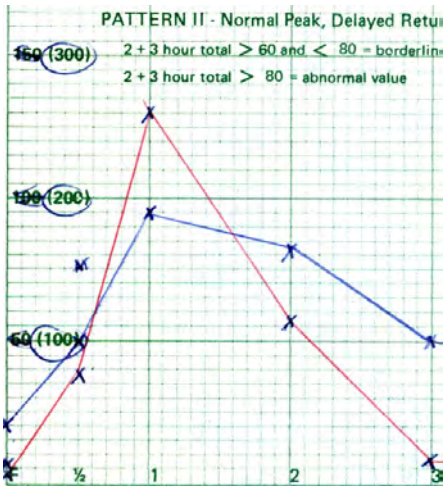
The error for the 0-400 bottom scale range was an outlier. While the error rate remained below 4% for the other chart ranges, the 0-400 range had an error of 9.20%. The explanation for this is the interference between the two lines at the bottom chart. The method attributed the same value for the markers in both lines if the markers were



(a) Human error when drawing the markers.



(b) Note intersecting with data line.



(c) Element intersecting with data line and extra marker in the wrong place.

Figure 5.26: Scenarios in which the marker extraction was not successful.

Table 5.4: Error in relation to the scale range grouped by scale type.

Chart	Scale	Global Error
Top	0-400	2.27 %
Top	0-800	3.22 %
Bottom	0-200	3.57 %
Bottom	0-400	9.20 %
Bottom	0-20	0.25 %
Bottom	0-40	0.72 %

close enough. The 0-400 range was the largest range for the bottom chart, and this fact combined with the merging of different markers resulted in a higher error rate.

5.4.1 Straightening

The success of the straightening step depends on the footer region selected to be used as a reference for the transformation. The angle used to skew the image is based on the bounding box of the textual content within it.

Figure 5.27 illustrates a scenario in which cut-off text is present in the region. In this case, the resulting rectangle will not be tilted in any direction, the calculated angle will be zero and, thus, the sample will not be straightened. This interference occurred in approximately 30% of the samples. No rotation is applied to the input when interference occurs, and the process continuous normally. The consequence is that, for the samples that are not straightened, the final error is slightly higher.

Figure 5.27: Footer region used for straightening with cut-off text.

5.4.2 Background Removal

The background detection procedure performed well on the identification of background pixels. In some instances, though, it misidentified data line pixels as background pixels, which created discontinuities. Figure 5.28 illustrates this scenario. The discontinuities occurred due to two distinct reasons: the HSV ranges utilised to isolate the background performed sub-optimally for the cases when the pen ink was faded due to the aging of the dataset and the scanner light brightened up spots in the data line.

The consequence of these discontinuities is that parts of the line are not inspected when the markers are being located, which may result in false negatives. Nonetheless,

Figure 5.28: Data line with discontinuities due to background removal.

this scenario was unusual, occurring in less than 3% of the samples.

5.4.3 Data Line Detection

If an element is too close to the data line, it may be interpreted as being part of it. This accidental merging was the main reason for the failures in extracting markers. This scenario was present in almost all the documents from which the values could not be properly extracted. Figure 5.29 shows a case in which handwritten text is too close to the data line.

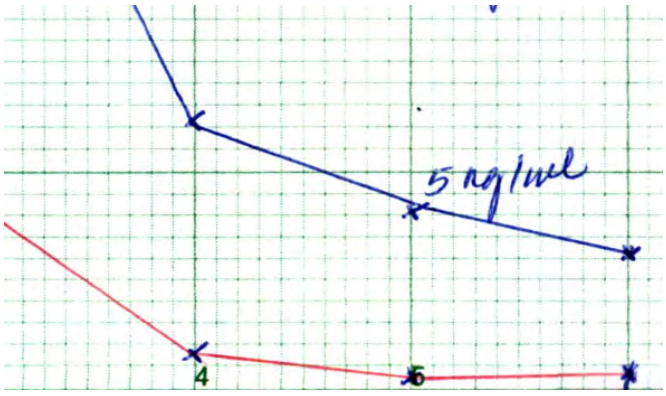


Figure 5.29: Textual content in close proximity to the data line.

5.4.4 Marker Detection

The regions where the chart lines lie are isolated before the marker detection step occurs. Once the detection is performed, two possibilities exist: false negatives and false positives. False positives occurred mainly when there was an intersection between an external element and the data line, and when smudges and cross-outs were present.

In this case, the intersecting elements would be seen as part of the data line and would be processed accordingly.

False negatives occurred when parts of the lines were not inspected due to discontinuities and when the markers were too small to be detected after the morphological transformations. False negatives were more infrequent than false positives as the causes for false negatives were far less common than the main cause for false positives – elements intersecting with the data line.

5.5 Discussion

The approach utilised in this use case was able to extract the markers for 61% of the charts. The main reason why this procedure failed in 39% of the charts was the recurring presence of elements intersecting with the data lines. Thus, improving the line detection step is the action most likely to yield a significant improvement to the success rate.

The metric used to measure the accuracy of the values extracted revealed one relevant trend. The bigger the step in the scale, the larger the error. This happens because all the scales cover a similar amount of pixels. As a result, the bigger the step, the denser the pixel ratio is. As an example, one scale using the step of 50 units and another using the step of 5 units will cover the same amount of pixels. However, the latter captures the actual values with higher precision as it is sparser than the former. Error in the extraction of values is expected as it can be hard even for a human to determine the value a marker holds just by looking at it. Furthermore, the interference between the markers in the bottom chart increased the error for the groups measuring the bottom scales.

The forms that were not straightened due to the issue with the cut-off content may have potentially influenced the error metric as the markers ended up slightly tilted. Nonetheless, the items in the data set were fairly straight before processing, so the

impact was low.

The background removal step caused some discontinuities in the data lines. This had a low impact in the error, but some influence in the false negatives as some discontinued sections of the line may be ignored in the marker location step. The data line detection step was the stage that affected the success rate of extraction the most due to the elements intersecting with the data lines.

The algorithm described in the first case study is more generic as it does not need to be provided with colours as parameters. While the algorithm from the first case study is carried out independently of the colours used in the charts, this one depends on knowing this information beforehand. In contrast, the previous algorithm is sensitive to noise introduced by digitization processes as it depends on colours being fairly constant, whereas this algorithm is tolerant to it as it works with colour ranges.

Finally, the first case study performed a full scan of the image to locate the markers. This was shown to take a long processing time. In the second case study, the performance was improved by using a vectorised approach to locate the centroid of each marker.

5.5.1 Limitations

The approach used in this case study presents some limitations. The scales had to be provided prior to the processing as they were highly heterogeneous across the documents. The inherent human unpredictability and subjectivity makes it harder for computers to process freehand content. Even though the form elements are relatively constant across the data set, notes in random places and cross-outs can easily interfere with the results, which can happen especially if elements intersect with the data line.

Opportunities for improvements and future work are listed below:

- Automate the scale extraction;
- Improve the straightening procedure;

- Devise a method to segment the image in a way that neighbouring elements too close in proximity are not seen as one.

Chapter 6

Conclusion

The goals of this work were to extract the values from line charts contained in scanned medical documents and to contribute to the body of knowledge on chart recognition. Two case studies were devised to achieve these objectives. The first case study covered line charts with a single line for scenarios with low levels of noise. The second case study stemmed from the first one and is best suited for digitised documents with handwritten content.

It was possible to perform the extraction on 61% of the charts present in the scanned documents. The error rate for the markers was below 10% in relation to their corresponding chart scales. In general, the larger the range, the bigger the error. A scale with a range of 600 units will produce more significant errors than a scale with 60 units if both take up the same amount of pixels. This is expected as it is infeasible to manually draw markers small enough to account for denser scales.

The experiments with the machine-generated dataset presented a low level of false negatives, with only six out of the 2000 data points not being detected. The error between the extracted and actual values was low. The charts had two scales with the range of 100 units. The error in the extraction was below one unit for all marker types.

6.1 Research Achievements

This research has made two contributions. Firstly, a method to extract markers from machine generated charts with one single data line has been proposed. Experimentation showed that this approach was able to achieve significant accuracy. The experiments also demonstrated that the method is robust to the use of different colours and marker styles. Secondly, a method to extract markers from digitised form sheets with handwritten content has been proposed. Experimentation showed that this approach was able to process around 60% of the charts in the provided dataset. It has been demonstrated that the error rate for this method is low in relation to the chart scale.

6.2 Future Work

This work also presents opportunities for future research. The first case study expected the charts to have a series predetermined elements – a grid, a data line with markers and labels for the x and y axes. The method could be improved by being made more tolerant to the presence or absence of unexpected items. Also, it expects the chart to have only one data line, which could be expanded to work with any number of lines.

The second case study demands that a series of parameters are passed in so that the form sheets can be processed. Three areas of improvement have been identified for this scenario. Firstly, the scale range is received through a parameter, unlike the first case study. Devising a method to automatically extract the scale from the document itself would make the method more flexible. Secondly, the straightening procedure has a high error rate. Finding a new strategy to straighten the images before processing have the potential to slightly improve the error in the marker extraction. Finally, the intersection between notes, smudges and cross-outs was the single main cause for the failure in the processing of 39% of the documents. As a result, devising a method to detect the data

lines even when intersecting elements are present is the action most likely to yield the biggest gains.

References

Al-Zaidy, R. A. & Giles, C. L. (2015). Automatic extraction of data from bar charts. In *Proceedings of the 8th international conference on knowledge capture* (p. 30).

Amara, J., Kaur, P., Owonibi, M. & Bouaziz, B. (2017). Convolutional neural network based chart image classification.

Arica, N. & Yarman-Vural, F. T. (2001). An overview of character recognition focused on off-line handwriting. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, *31*(2), 216–233.

Bartz, C., Yang, H. & Meinel, C. (2018). See: towards semi-supervised end-to-end scene text recognition. In *Thirty-second aaai conference on artificial intelligence.*

Bradski, G. (2000). The OpenCV Library. *Dr. Dobb's Journal of Software Tools*.

Cireşan, D., Meier, U. & Schmidhuber, J. (2012). Multi-column deep neural networks for image classification. *arXiv preprint arXiv:1202.2745*.

Crofts, C., Schofield, G., Zinn, C., Wheldon, M. & Kraft, J. (2016). Identifying hyperinsulinaemia in the absence of impaired glucose tolerance: An examination of the kraft database. *Diabetes research and clinical practice*, *118*, 50–57.

Heijmans, H. J. (1994). *Morphological image operators* (Vol. 4). Academic Press Boston.

Huang, W., Tan, C. L. & Leow, W. K. (2003). Model-based chart image recognition. In *International workshop on graphics recognition* (pp. 87–99).

Jung, D., Kim, W., Song, H., Hwang, J.-i., Lee, B., Kim, B. & Seo, J. (2017). Chartsense: Interactive data extraction from chart images. In *Proceedings of the 2017 chi conference on human factors in computing systems* (pp. 6706–6717).

Major causes of death. (2018, Aug). Ministry of Health NZ. Retrieved from `https://www.health.govt.nz/our-work/populations/maori-health/tatau-kahukura-maori-health-statistics/nga-mana-hauora-tutohu-health-status-indicators/major-causes-death`

Mohammad, F., Anarase, J., Shingote, M. & Ghanwat, P. (2014). Optical character recognition implementation using pattern matching. *International Journal of Computer Science and Information Technologies*, *5*(2), 2088–2090.

Nair, R. R., Sankaran, N., Nwogu, I. & Govindaraju, V. (2015). Automated analysis of line plots in documents. In *2015 13th international conference on document analysis and recognition (icdar)* (pp. 796–800).

Otsu, N. (1979). A threshold selection method from gray-level histograms. *IEEE*

transactions on systems, man, and cybernetics, *9*(1), 62–66.

Poco, J. & Heer, J. (2017). Reverse-engineering visualizations: Recovering visual encodings from chart images. In *Computer graphics forum* (Vol. 36, pp. 353–363).

Poco, J., Mayhua, A. & Heer, J. (2017). Extracting and retargeting color mappings from bitmap images of visualizations. *IEEE transactions on visualization and computer graphics*, *24*(1), 637–646.

Rakap, S., Rakap, S., Evran, D. & Cig, O. (2016). Comparative evaluation of the reliability and validity of three data extraction programs: Ungraph, graphclick, and digitizeit. *Computers in Human Behavior*, *55*, 159–166.

Redmon, J., Divvala, S., Girshick, R. & Farhadi, A. (2016). You only look once: Unified, real-time object detection. In *Proceedings of the ieee conference on computer vision and pattern recognition* (pp. 779–788).

Savva, M., Kong, N., Chhajta, A., Fei-Fei, L., Agrawala, M. & Heer, J. (2011). Revision: Automated classification, analysis and redesign of chart images. In *Proceedings of the 24th annual acm symposium on user interface software and technology* (pp. 393–402).

Smith, R. (2007). An overview of the tesseract ocr engine. In *Ninth international conference on document analysis and recognition (icdar 2007)* (Vol. 2, pp. 629–633).

Zhou, X., Yao, C., Wen, H., Wang, Y., Zhou, S., He, W. & Liang, J. (2017). East: an efficient and accurate scene text detector. In *Proceedings of the ieee conference on computer vision and pattern recognition* (pp. 5551–5560).

Zimmermann, O., Stocker, M., Lübke, D. & Zdun, U. (2017). Interface representation patterns: crafting and consuming message-based remote apis. In *Proceedings of the 22nd european conference on pattern languages of programs* (p. 27).

Zuo, Z., Shuai, B., Wang, G., Liu, X., Wang, X., Wang, B. & Chen, Y. (2015, June). Convolutional recurrent neural networks: Learning spatial dependencies for image representation. In *The ieee conference on computer vision and pattern recognition (cvpr) workshops.*

Appendix A

Ethics Approval

A copy of the Ethics Approval letter is presented in the following page.

Auckland University of Technology Ethics Committee (AUTEC)

Auckland University of Technology
D-88, Private Bag 92006, Auckland 1142, NZ
T: +64 9 921 9999 ext. 8316
E: ethics@aut.ac.nz
www.aut.ac.nz/researchethics

1 June 2018

Russel Pears
Faculty of Design and Creative Technologies

Dear Russel

Re Ethics Application: **18/171 Applying artificial neural networks to chart recognition**

Thank you for providing evidence as requested, which satisfies the points raised by the Auckland University of Technology Ethics Committee (AUTEC).

Your ethics application has been approved in stages for three years until 1 June 2021.

This approval is for the Honours project only. The Executive Secretary is engaging with Dr Crofts about the creation of a databank using the 'Kraft data'.

Standard Conditions of Approval

1. A progress report is due annually on the anniversary of the approval date, using form EA2, which is available online through http://www.aut.ac.nz/researchethics.
2. A final report is due at the expiration of the approval period, or, upon completion of project, using form EA3, which is available online through http://www.aut.ac.nz/researchethics.
3. Any amendments to the project must be approved by AUTEC prior to being implemented. Amendments can be requested using the EA2 form: http://www.aut.ac.nz/researchethics.
4. Any serious or unexpected adverse events must be reported to AUTEC Secretariat as a matter of priority.
5. Any unforeseen events that might affect continued ethical acceptability of the project should also be reported to the AUTEC Secretariat as a matter of priority.

Please quote the application number and title on all future correspondence related to this project.

AUTEC grants ethical approval only. If you require management approval for access for your research from another institution or organisation then you are responsible for obtaining it. You are reminded that it is your responsibility to ensure that the spelling and grammar of documents being provided to participants or external organisations is of a high standard.

For any enquiries, please contact ethics@aut.ac.nz

Yours sincerely,

Kate O'Connor
Executive Manager
Auckland University of Technology Ethics Committee

Cc: , rgr4559@autuni.ac.nz; Muhammed Naeem