

University of Southern Queensland
Faculty of Health, Engineering and Sciences

Conversion of Cadastral Survey Information into LandXML Files using Machine Learning

A dissertation submitted by
Óscar Garrido de la Rosa

in fulfilment of the requirements of
ENG4111 and ENG4112 Research Project
towards the degree of
Bachelor of Spatial Science (Honours) (Surveying)
Submitted October 2019

ABSTRACT

Keywords: Cadastral surveying, computer vision, machine learning, LandXML

Although new cadastral surveys can readily be produced in the industry standard LandXML format, there is a vast amount of pre-existing information which is only stored as image files. Automating the back-capture of this information would improve a process which is labour intensive and prone to human error. This project proposes a workflow to automate this process, in relation to Victorian cadastral survey information. Specific algorithms and outcomes are examined using a simplified sample cadastral plan.

The literature review reveals that similar documentation processes have been undertaken in other fields, such as music (Calvo-Zaragoza et al., 2018). In the cadastral context only true to scale cadastral maps have been digitised but not surveyors' sketches or field records (Ignjatić et al., 2018)

A simple plan was created containing a closed parcel and two instrument points for creation and testing of the workflow. An analysis of the tasks required to extract the information needed for the LandXML files was undertaken. A pipeline was designed to perform the data extraction in a machine learning environment, which has been dubbed Double Filter Capture. It consists of two main workflows that handle the graphical information and the text elements separately, by means of Computer Vision and Optical Character Recognition algorithms, respectively. An implementation of the actions in the pipeline was trialled and barriers encountered discussed. Several Machine Learning algorithms were used for the required tasks, such as line detection, corner detection, image rotation, text detection and text extraction.

The project gives some idea of the possibilities and limitations that a larger scale automated back-capture would face, when dealing with records of significantly greater complexity. It also points the way to further research required to refine the extraction process outlined here, for example including elements omitted in this project, such as occupation and other auxiliary information and hand-written records.

This project demonstrates automated accurate data extraction from an image file is possible, however an extensive investment would be required in the programming stage, given the complexity and inconsistencies of existing plans that require back-capture.

DISCLAIMER

University of Southern Queensland
Faculty of Health, Engineering and Sciences
ENG4111/ENG4112 Research Project

Limitations of Use

The Council of the University of Southern Queensland, its Faculty of Health, Engineering & Sciences, and the staff of the University of Southern Queensland, do not accept any responsibility for the truth, accuracy or completeness of material contained within or associated with this dissertation.

Persons using all or any part of this material do so at their own risk, and not at the risk of the Council of the University of Southern Queensland, its Faculty of Health, Engineering & Sciences or the staff of the University of Southern Queensland.

This dissertation reports an educational exercise and has no purpose or validity beyond this exercise. The sole purpose of the course pair entitled “Research Project” is to contribute to the overall education within the student’s chosen degree program. This document, the associated hardware, software, drawings, and other material set out in the associated appendices should not be used for any other purpose: if they are so used, it is entirely at the risk of the user.

CERTIFICATION

University of Southern Queensland
Faculty of Health, Engineering and Sciences
ENG4111/ENG4112 Research Project

Certification of Dissertation

I certify that the ideas, designs and experimental work, results, analyses and conclusions set out in this dissertation are entirely my own effort, except where otherwise indicated and acknowledged.

I further certify that the work is original and has not been previously submitted for assessment in any other course or institution, except where specifically stated.

Óscar Garrido de la Rosa

Student number: XXXXXXXXXX

ACKNOWLEDGEMENTS

Working through this project was truly a team effort and many people contributed to shaping it. To all of them I am deeply grateful. Dr Glenn Campbell encouraged me to dive into a field of unknown and offered guidance along the way. Lachlan Smith offered invaluable help with the technical aspects. Craig Sandy, Surveyor-General of Victoria, and my managers and colleagues, especially Ang Dimas, supported me and granted me the flexibility to pursue this project while in their employment. Jason Heritage, Chris Quinn and the Digital Cadastre Modernisation team made all necessary resources available. Hamed Olfat from the ePlan team offered valuable insights into the field. Lastly and most importantly, my wife Bettina was the engine pulling this train. Our son Arden was the light at the end of the tunnel.

ABBREVIATIONS

AFR: Abstract of Field Records

AI: Artificial Intelligence

CAD: Computer-aided Design

CV: Computer Vision

DCMP: Digital Cadastre Modernisation Project

DFC: Double Filter Capture

DL: Deep Learning

ICSM: Intergovernmental Committee for Surveying and Mapping

LandXML: Land Extensible Markup Language

LASSI: Land and Survey Spatial Information (mapping service provided by Land Use Victoria to search property details)

LASSI – SPEAR: web mapping application that allows the user to search and download survey information from one location. It provides a link between LASSI survey documents available from LANDATA® and provides a feature to download pre-populated ePlans with data from Vicmap and the Survey Marks Enquiry Service (SMES).

MGA: Map Grid of Australia

ML: Machine Learning

OCR: Optical Character Recognition

OMR: Optical Music Recognition

PDF: Portable Document Format

PS: Plan of Subdivision

PSM: Page Segmentation Mode

RE: Record of Having Re-established a Cadastral Boundary

SPEAR: Surveying and Planning through Electronic Applications and Referrals (online platform from Land Use Victoria to enable the electronic lodgement of subdivision applications and planning permits)

TIFF: Tagged Image File Format

TABLE OF CONTENTS

Abstract	i
Disclaimer	ii
Certification	iii
Acknowledgements	iv
Abbreviations	v
Chapter 1: Introduction	1
1.1 Background	1
1.2 The LandXML format	1
1.2.1 Back-capture project in Victoria	2
1.2.2 Back-capture of cadastral survey information	2
1.3 Task statement	4
1.4 Research question	4
1.5 Limitations	5
Chapter 2: Literature review	6
2.1 Machine Learning	6
2.2 Optical Character Recognition	7
2.3 Computer Vision	7
2.4 Similar research in other areas	8
2.5 Contribution to existing research	11
Chapter 3: Methodology	12
3.1 Input Source: Abstracts of Field Records	12
3.1.1 Analysis of a set of Abstracts of Field Records	13
3.1.2 Creation of a custom AFR for input	17
3.2 Output to a LandXML file	21
3.2.1 File format	21
3.2.2 Required data items	22
3.3 Creation of a workflow for the information detection procedure	24
3.4 Development environment	26
Chapter 4: Implementation and results	27
4.1 Workflow: Double Filter Capture	27
4.2 Implementation	28
4.2.1 Phase one: Geometry Filter	29

4.2.2 Phase two: Text Filter	41
4.2.3 Cross-checks and removal of false positives	46
4.3 Summary of results	55
Chapter 5: Further research and conclusion	56
5.1 Further research	56
5.2 Conclusion	58
References	59
Appendix A – Project specification	63
Appendix B – LandXML code	64
Appendix C – Python code	67
1. Removing text	67
2. Simplify lines	68
Thresholding	68
Canny	68
3. Detect lines	69
4. Detect corners	69
5. Image rotation	69
6. Text detection	71
7. Text recognition	72

CHAPTER 1: INTRODUCTION

1.1 Background

Vast amounts of historical survey information are stored in land registry repositories in the form of scanned documents. These provide essential records supporting the re-establishment and creation of cadastral boundaries. The current general move to automation in data management requires that these documents be converted into a machine-readable format, if they are to be preserved.

Many jurisdictions around the world are implementing a land-specific extensible markup language, known as LandXML, as the standard format for recording and storing cadastral survey information (ICSM 2016). Although new surveys can readily be produced in this format, there is a vast amount of pre-existing information which is only stored as an image file. The back-capture of this information is usually done manually and is labour intensive and prone to human error. A compounding difficulty is the level of expertise required from the data entry operators to understand the relevance of the information. This project examines the viability of automating the back-capture process for Victorian cadastral survey information.

1.2 The LandXML format

Since the advent of the generalized use of the Internet and the ability to transfer data quickly and reliably between users, one goal that has been pursued is to agree on a standard format for the transfer of cadastral survey information. LandXML is a sharable data format that allows for manipulation, interrogation, visualisation or rendering of the data in a range of software allowing flexible use of the data. The efforts of the Intergovernmental Committee of Survey and Mapping (ICSM) have resulted in LandXML being accepted as the standard, with the aim of enabling digital lodgement and transfer of intelligent cadastral survey data (ICSMA 2010).

The Victorian state government offers since 2013 the option of lodging plans of subdivision as ePlans. These plans are based on the LandXML format and some survey companies have participated in the development of this feature. The general uptake from the industry has been slow (Olfat et al. 2018) but a sufficient number of submissions exists for a preliminary analysis.

1.2.1 Back-capture project in Victoria

The Victorian cadastre is based on monuments and measurements rather than on co-ordinates (Batchelor 2016). The graphical representation of the cadastre in Vicmap Property™ is co-ordinated to a certain degree, but in the legal determination of cadastral boundaries the monuments take precedence.

Survey information supporting cadastral boundary determinations is recorded in a repository of documents dating as far back as the first colonial surveys. Most of these historical documents have been digitised and are accessible as Portable Document Format (.pdf) files (see figure 1). Since the inception of the Surveying and Planning through Electronic Applications and Referrals (SPEAR) new surveys can be lodged electronically as .pdf files and hard copies are not kept on record. The latest development, ePlan, enables full electronic submission of “intelligent” data in LandXML format, although the uptake of this possibility by the industry has been slow (Olfat et al. 2018).

To maximise the potential of the available information, it would be desirable to develop a state-wide cadastral database which would include both parcel geometry and survey information, being co-ordinates of survey monuments and their connection to parcel corners by bearings and distances (Merz 2011). Ideally, this would include a continuous fabric free of gaps or overlaps that allows for new information to be included as it becomes available. This new information may be the result of new subdivisions or other types of land reconfiguration.

This task is currently a focus of Surveyor-General Victoria, who has created a Digital Cadastre Modernisation Project (DCMP) work team for this purpose. A similar undertaking has been performed in other jurisdictions, notably New Zealand (Rowe 2003) and the Australian Capital Territory (Merz 2011) and overseas (Olfat et al. 2018).

1.2.2 Back-capture of cadastral survey information

The first stage in these projects entails back capturing the survey information that exists in the form of .pdf files. This forms a starting network to which new information can be added. To date this has been done manually by typing the relevant information into databases. In the Victorian DCMP this step is currently underway. The back-capture process involves marking up the plans by an expert who decides which information is relevant by following set rules (Fraser et al. 2018). A process of double entry follows, in which two operators independently type the information into a database. An automated matching routine flags any inconsistencies to ensure accuracy in the entry.

1.3 Task statement

This project aims to attempt application of Machine Learning (ML), or elements thereof, to the back-capture process. This involves “scanning” survey plans in order to automatically collect data and export this data into a LandXML database. To undertake this project, research from other fields has been used for information regarding programming environments and procedures. Victorian Abstracts of Field Records (AFR) have been used as the source material. The project aims to document procedures used by the author, as these are expected to differ from both how data is collected manually and “learned” by humans, and from how traditional computer programming would approach input of this information. This process may have implications as to whether automated or manual processes are selected in future and may impact on accuracy and efficiency of this process.

The ultimate goal, outside of the scope of this project, would be to a tool that would automate the back-capture of cadastral survey information from .pdf plans and convert it into intelligent data in LandXML format. Ideally this would enable faster and more reliable back-capture free from human error. Outside the DCMP context, this tool would be useful for surveyors needing intelligent data from previous surveys when preparing a new survey. Scoping the extent of research necessary to design such a tool is one of the objectives of this project.

1.4 Research question

This project aims to examine the use of ML and its sub-domains of Computer Vision (CV) and Optical Character Recognition (OCR) to determine their viability in completing elements of the back-capture processes. Given the broad scope of this area, this project has been limited to answering the following research question:

Can ML be used to reliably extract survey pertinent data from scanned Victorian AFR?

The above problem was addressed through the following objectives:

1. Define the elements that are required to be detected in a cadastral survey plan.
2. Propose the steps required for automated detection of cadastral information. This will include considering the order in which a machine would need to identify and process these elements.
3. Gather information on methods of ML documented in the research literature from other fields of work (e.g. botanical labels, music scanning) in order to trial these methods in a surveying context. Attempt implementation of these methods to determine if any elements from AFRs can be detected automatically.

The project documents procedures trialled as an outcome of the project. It also documents barriers encountered and possible future approaches that may achieve more accurate, efficient or extensive outcomes.

1.5 Limitations

Computer generated plans were used to facilitate simple evaluation of outcomes. Issues encountered include incorrect and alternate use of symbols, omitted or inaccurate text, data entered further away from the element it is describing due to lack of space on the plan, high plan complexity, poor plan quality and plans written partially or completely by hand. Further issues considered include use of symbols or signs that are made up of lines (e.g. fence markings, symbols). It was therefore attempted to determine most commonly occurring features of plans, and ultimately the creation of a model AFR including only these elements, as described in section 3.1.1.2.

Only one programming language, Python and two repositories of algorithms, Open CV and Tesseract were used. This decision was based on research information from other fields. It is possible that accessing further repositories, customised programming, and/or combining computer programming languages would lead to further positive outcomes. Also to note is the fact that the author is a novice computer programmer and had not previously worked within Python or in the field of ML. This demonstrates the accessibility of this technology and implies that far more efficient outcomes may be able to be achieved if working together with an expert in this area.

The project was time limited by its nature as an academic project. Further time would be required to delve into the issues unearthed.

The project did not aim to evaluate efficiency and accuracy by comparison with manual data entry, as is being used in the Victorian Government back-capture project which sparked the idea for this project. It does however attempt to demonstrate that investigation into this area is warranted particularly if large quantities of existing, or possibly newly created, files require back-capture.

CHAPTER 2: LITERATURE REVIEW

Back-capturing survey plans involves converting visual images and text to a digital database of reference points. The literature shows that the ML environment can address the processing of both images and text.

2.1 Machine Learning

The field of Artificial Intelligence (AI) and ML is as old as the first computers and the work of Alan Turing is recognised as pioneering in the field (Russell & Norvig 2003). The research interest in the field has ebbed and flowed with changes in technology and focus on the different issues involved and is currently experiencing unprecedented attention. This is due to the exponential increase in available data, such as data generated by users of mobile and wearable devices and also to the higher computational power enabled by parallel computing and the use of Graphic Processing Units for computations other than graphics (Alpaydin 2016).

The traditional approach to programming has been to use algorithms to give the computer step by step instructions to solve a task. With ML the focus shifts to problems of which we do not know the steps involved but we have data to provide the required outcome. This way the computer can learn to derive the algorithm to arrive at the required outcome. The more data we can provide, the higher the success rate of the learning, i.e. the higher the probability that the prediction will match the desired outcome (Alpaydin 2016).

A non-exhaustive overview of tasks for which ML has been and is being used can be compiled from several sources in the literature (Alpaydin 2016; Goodfellow et al. 2016; Zhang et al. 2019)

- Classification, with or without missing inputs.
- Regression: prediction of a numerical value given some input (for example, predicting house prices given some variables).
- Transcription, such as OCR: “In this type of task, the machine learning system is asked to observe a relatively unstructured representation of some kind of data and transcribe the information into discrete textual form” (Goodfellow et al. 2016, p. 99).
- Machine translation. As is commonly applied to natural languages.
- Structured output. Broad category including transcription and translation and in general any task that involves labelling data and finding relationships among the data, such as parsing or image captioning (for example labelling roads in an aerial photograph).

- Denoising (for example extracting background from images before OCR).
- A series of other tasks mentioned do not appear to be relevant for this project. They include anomaly detection, synthesis and sampling, imputation of missing values and density estimation.

These tasks have been applied in many different fields, among others medical diagnosis, face recognition, speech recognition and predicting share prices and shopping behaviours.

2.2 Optical Character Recognition

OCR is the ‘translation of handwritten or typewritten text into machine-editable form’ (Dharmana et al 2012, p. 23). Areas of application include automatic text entry, automatic reading of credit cards, post and licence plates, document data compression, and language processing amongst others (Dharmana et al 2012). The aim of this project is to research whether the use of OCR can improve the efficiency of back-capture processes, in terms of speed and accuracy. OCR has been used for over two decades in an array of fields and its development continues, with Deep Learning (DL) (Wei 2018). A limitation of existing OCR engines is their variety, with each engine designed to read a specific type of document. The accuracy of these engines is dependant on fonts, paper quality or deterioration and may not achieve 100% accuracy (Petrescu et al 2019, p. 23).

In the Victorian cadastral surveying context, the use of OCR has been limited to the capture of back-capture of Primary Cadastral Marks and Permanent Marks from tables provided by surveyors. The provision of this table was required by the *Survey co-ordination regulations 2004 (Survey Co-ordination Regulations 2004)* to help improve the integrity of the digital cadastre (Vicmap Property™) but the requirement was revoked in the 2014 regulations.

An attempt was also made to utilise OCR for the extraction of metadata from plans, however this project was abandoned (Olfat 2019, pers. comm. 28 March). Both these uses were limited to “traditional” OCR extracting only text from documents consisting mainly of text. As we will see, this project differs in that it attempts to extract text from AFRs, which is a far more complex undertaking.

2.3 Computer Vision

A subfield of ML with a recent interest spike and which proved useful in this project is CV. A vast amount of research is being done in CV for its use in self-driving vehicles and in scene recognition (Liu et al. 2019). These advances have also lessened the need to rely on the existence of labelled data, as techniques are developed that can worked in an unsupervised fashion, such as Deep Belief Networks, Deep Boltzmann Machines and Stacked Autoencoders

(Voulodimos et al. 2018). Also, techniques such as bootstrapping can minimise reliance on labelled data, as described in (Heidorn & Wei 2008).

This project looks at the potential of CV to detect features in the imaged plan that can guide the interpretation of the text elements extracted with OCR. The line work in the AFR provides a topological diagram representing the spatial relationship between points and using this as a manner of road map has become one of two main pillars for this project, the second pillar being the OCR text recognition.

To quote Ignjatić et al.(2018, p. 3),

“As we are moving towards more complete computer vision, full historic map understanding becomes achievable, as the crucial steps in this process are visual object and pattern recognition, which is becoming more precise and detailed”.

2.4 Similar research in other areas

Two areas found in the literature with many commonalities to this project are Optical Music Recognition (OMR) (Bainbridge & Bell 2001; Calvo-Zaragoza et al. 2018) (see figure 2) and botanic and museum specimen labelling (Heidorn & Wei 2008; Kirchhoff et al. 2018) (see figure 3).

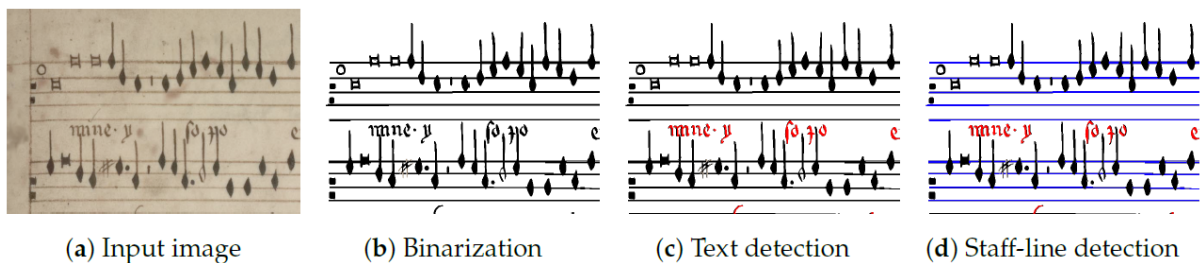


Figure 2: Typical sub-tasks for processing musical documents, from (Calvo-Zaragoza et al. 2018)

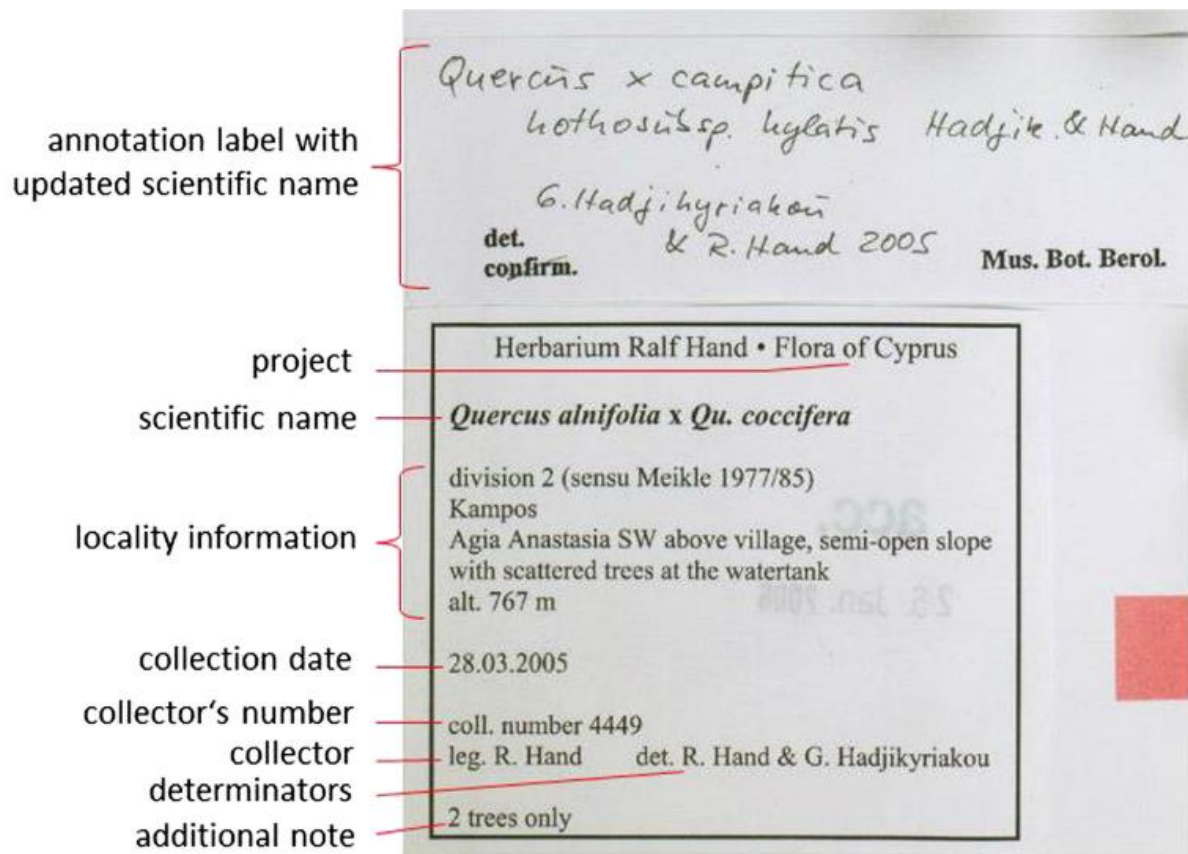


Figure 3: Information on botanic specimen labels, from (Kirchhoff et al. 2018)

In both cases the problem consists of extracting information from hand written or typeset documents with a variety of styles, formats, codes and layouts. In the case of OMR, the music notation can have many levels of complexity, with symbols connected and overlaid with other notations. This is also the case in this project, where handwritten notations from the plan examiner may cross Computer-aided Design (CAD) produced lines or text with relevant information. Although the Surveying Practice Handbook contains guidelines for the drafting of plans and abstracts of field records, these are not statutory legislation and therefore are only loosely followed by surveyors and transmitted in an expert and apprentice fashion (Victoria 1997). The variability of drafting styles and conventions poses difficulties for automating the recognition and understanding of the information.

More specifically, there are two main similarities between OMR and the extraction of cadastral information:

- Symbols do not convey meaning by themselves but gain meaning from their position relative to lines (staff in music). It is not sufficient to capture the symbols in the order that they appear, as it would be with ordinary text-based OCR. The human reader interprets the output of OCR in the form a text document (Bainbridge & Bell 2001). In

OMR and cadastral information, this interpretation needs to happen before the output, and this represents the single major challenge of the process.

- In music, a measure contains a stipulated number of notes that are not necessarily distributed proportionally in the visual space. OMR software is generally programmed so that it checks the contents of each measure against the position of bar lines. (Padilla et al. 2014). This is the equivalent of closed shapes in AFRs, where the lines are not drawn to scale. The dimensions shown as text on the lines are the actual information of interest and the lines can be used for validation.

The steps identified by the authors of (Bainbridge & Bell 2001) for the OMR workflow can therefore be used as a guide for the retrieval of cadastral information:

- Staff line identification (equivalent to detection of lines and line styles in the AFR).
- Musical object location (equivalent to detection of symbols and line intersections).
- Musical feature classification (equivalent detection of text and allocation of this text to lines).
- Musical semantics (equivalent to validation of dimensions against line work and allocation of co-ordinates).

As in OMR, the last two steps require specialised knowledge and may not be fully automatable.

Heidorn & Wei (2008, p. 58) describe similar challenges faced when extracting metadata from museum specimen labels, mainly “the volume and heterogeneity of the data” which makes it “expensive for humans to type in and extract critical information by hand.”

Once extracted, this information needs to be transformed into smart data than can be searched and manipulated. Issues around deciding which information to capture, how to eliminate not relevant information safely and how to structure the output are common with this project. The rationale for the interest in a ML approach in all these fields stems from the expense of utilising highly qualified personnel for the menial tasks of labelling, be it specimens, music notation or in this case dimensions in an Abstract of Field Records (AFR).

The use of ML in cartography in general and in cadastral maps in particular has been limited (Ignjatić et al. 2018). Although many countries have digitised their map bases since the 1980s, this has been done by vectorisation of image files by traditional algorithm-based programming, and DL techniques such as neural networks have only been used in the text recognition stage (Katona & Hudra 1999).

2.5 Contribution to existing research

The existing research documents use of OCR and ML techniques in other fields attempting to import graphical data and convert it into digital data, however no research documentation has been found by the author to date specifically examining the extraction of cadastral information from scanned survey plans.

This project attempts to contribute to the existing knowledge by applying ML Techniques to cadastral survey information in the Victorian context. This may in future lead to further research attempting to fully-automate Victorian back-capture, attempts in other locations with surveys following different presentation guidelines, and possibly inform further ML or DL research in other fields.

CHAPTER 3: METHODOLOGY

In order to undertake the process of automatically extracting cadastral survey information, the following single case study research design was devised:

1. Determine the required Inputs from a set of AFR plans. The set provided by the DCMP Team was analysed to identify the most common features that could be included and less common features to be excluded from a model AFR. From this information, Research Outcome 1 will be devised, namely, an Information Extraction Model (Model AFR). This is a simplified AFR containing only key elements to test programming modules.
2. Determine required outputs based on available LandXML examples. Create a proposed workflow of anticipated steps and their corresponding CV algorithms to detect items from the plan. This completes Research Outcome 2.
3. Implementation. Existing algorithms will be tested within the proposed workflow and their outcomes evaluated. The evaluation of this pilot project into automated data capture of AFR plans completes Research Outcome 3.

The development of each of these processes will be outlined in detail below.

3.1 Input Source: Abstracts of Field Records

In Victoria, any land transaction that involves creating new cadastral boundaries or modifying existing cadastral boundaries is required by law to be supported by a survey performed by a Licensed Surveyor. This includes transactions such as subdivisions and consolidations of land, and boundary realignments to agree with occupation. The supporting survey is to be documented in the form of an AFR and a Licensed Surveyor's report, outlining among other particulars the "relationship with other relevant cadastral surveys and the manner in which the boundaries of the subject land have been determined" (*Surveying (Cadastral Surveys) Regulations 2015 – Regulation 15*). In the case of subdivisions and consolidations, a new plan is generated and the Land Registrar issues new titles for the new parcels. This plan is drawn to scale and depicts the boundaries and their dimensions and a connection to a road intersection or a Crown boundary. Additionally, easements and other secondary interests are recorded in the plan.

The AFR is a schematic representation of the fieldwork undertaken by a surveyor. It contains information about the parcel boundaries, their connection to existing and newly established reference marks and any relevant evidence of occupation along the boundaries. Importantly for this project, this representation is not drawn to scale, and the information required is contained in the text elements rather than the drawing elements. However, the line work can still be used as a skeleton to indicate how the points are connected. The AFR provides the link

between cadastral boundaries and the network of co-ordinated reference marks and as such contains vital information for a robust cadastre.

3.1.1 Analysis of a set of Abstracts of Field Records

A batch of Plans of Subdivision provided by the DCMP team was analysed in order to gain an understanding of the type of information typically recorded and of which information should be captured or could be excluded.

The documents reviewed include plans from the Whittlesea local government area that have a Plan of Subdivision (PS) number greater than PS600000. This ensured that most plans were computer drafted, as these are the most recent plans lodged with Land Registry. The Whittlesea City Council covers a peri-urban area to the north of Melbourne which has recently experienced a notable growth. As such, this batch includes some 2500 plans, with the oldest plan being only 10 years old.

The data provided consists of a complex array of documents. For each output of one LandXML file there are as a minimum two .pdf documents from which information is extracted. One document typically contains the boundary dimensions of the parent parcel and the newly created internal boundaries plus all the relevant administrative information. This is the actual PS. The other document is a bundle collated from different sources and can include reports from the surveyor, the examiner and the AFR documenting the field measurements and the connections from the boundaries to the survey and to previous surveys. The information that this project aims to capture is contained in the AFR.

3.1.1.1 AFR file format

The files were provided as raster .pdf files. While most of the submissions are vector .pdf files generated from CAD software, some are scans of print outs or hand drafted plans. Additionally, these documents undergo an examination by a registration officer who adds hand written notations and subsequently scans them converting them to Tagged Image File Format (.tiff). SPEAR in turn prints these files to .pdf. For consistency, all documents, including accompanying reports and other documents that mostly contain text, are kept as raster .pdf files. The file size can vary considerably, and the files provided by the DCMP team ranged from 29KB to 359KB (215KB and 1.03MB respectively when converted to .tiff at 96dpi, the image size being 1654 x 2339pixels).

3.1.1.2 Project scope limitation

As a starting point, the most recent 183 plans were selected, with PS numbers higher than PS800000. A further sample reduction was undertaken, with the following criteria, developed on inspection of the plans:

- 40 plans did not contain an AFR. This could be due to a few reasons. Some are staged subdivisions that only contain a plan linking diagram to link the plan to the parent subdivision. Some subdivisions were not based on survey. These plans were discarded.
- 26 plans contained Abstracts of Field Records over multiple sheets. These were also discarded to simplify the one to one relationship between input and output.
- 41 plans contained arcs as boundary lines. The information shown for the dimensions of arcs (chord length, arc length, radius, chord bearing) would add much complexity to this task and these plans were also discarded.
- 2 plans were handwritten, and it was estimated that OCR would not be accurate enough to recognise the text. Hand-written text is notoriously more difficult to process for OCR engines (Ouwayed & Belaïd 2012).

In total, 100 plans were discarded (note that some plans contained more than one of the issues described above).

The remaining 83 plans were considered suitable in first instance. From those:

- 52 include splay corners. In most cases the intersection point is shown as well as the approaching distance to it from each line. The truncation distance is generally short, very commonly in the order of 3.05m (10ft) and can be represented in a variety of ways in different plans, namely as a point to point distance or in a chainage/offset style, or a combination of both. Figure 4 shows examples of these cases.

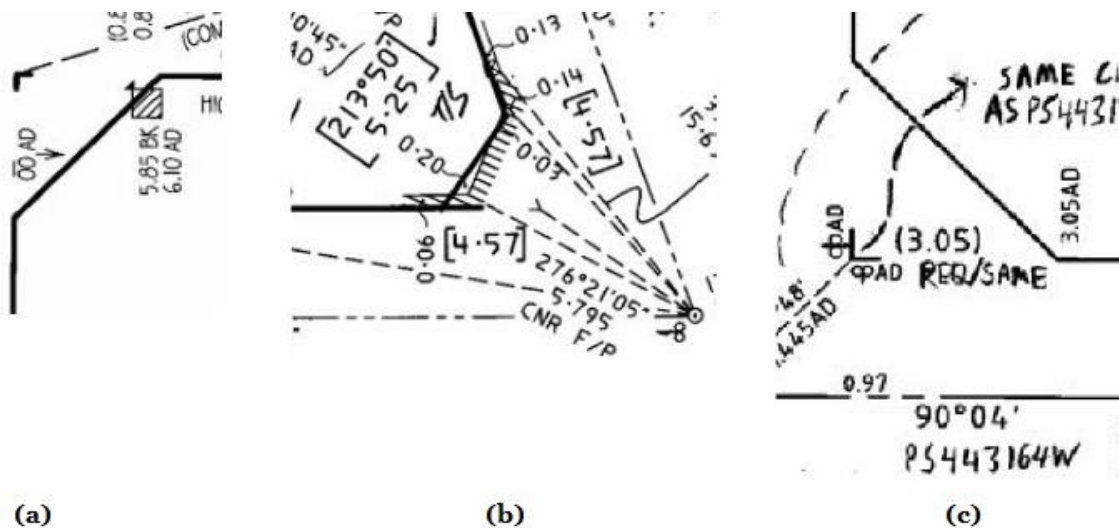


Figure 4: Splay distance shown as

- running chainage in (PS813845K)
- point to point distance, in brackets as a boundary dimension (PS814185W)
- both running chainage and point to point distance (PS815019K)

- 69 include running chainages additionally to the point to point distances displayed as dimensions on boundary lines. Running chainages are often accompanied by offsets shown in the same text orientation. Other times the offsets are shown as running chainages perpendicular to the traverse or boundary lines. This is a prime example of human understanding required to discern to which category these figures belong.

It is noteworthy that the chainage/offset situation applies mostly to the way occupation is shown in field records, as it conveys in a much clearer way the relevant information, being the offset from the occupation to the boundary line. The chainage/offset style depicts the traditional method of measuring occupation, that is laying a chain along the traverse line and measuring offsets with the offset tape. Modern surveyors prefer to radiate occupation from an instrument point and independently check the measurement by way of sighting to an offset tape along the traverse line. Showing these radiations in the field records would be a truer depiction of the field work but in most cases would result in a plan that is visually very busy and hard to read. Chainage and offset is therefore the preferred method to show the relationship between occupation and boundary in field records. The DCMP in its current phase is not capturing any occupation and focussing on survey monuments and boundary dimensions. “Traversing that does not connect parcel corners to control marks and/or reference marks, but fences only, should not be captured” (Fraser et al. 2018).

Figure 5 shows examples of offsets oriented parallel and perpendicular to the boundary line.

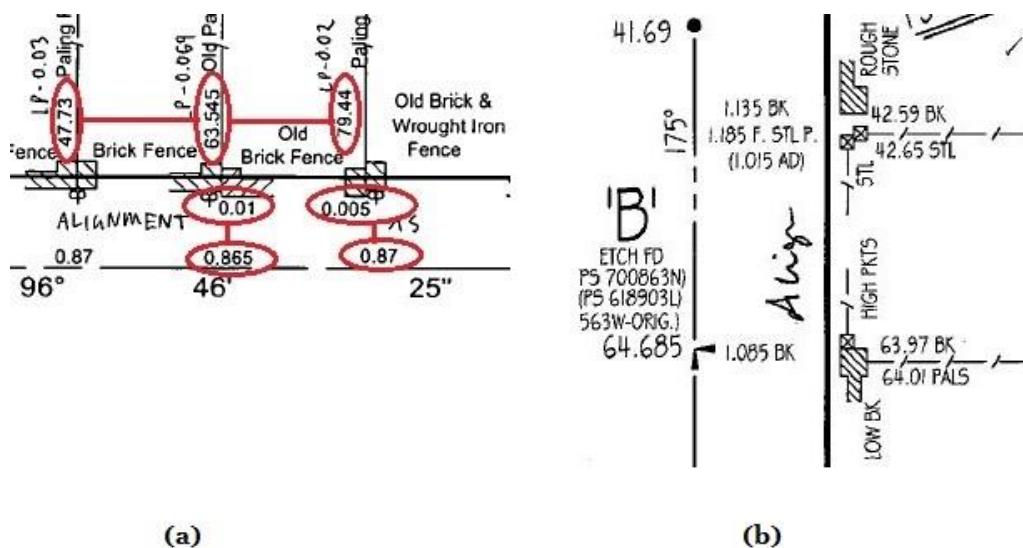


Figure 5: Offsets shown as

- a) running chainages perpendicular to traverse line (PS825585S)
- b) in the same orientation as chainages (PS802358J)

Consequently, the abstract was examined twice by two different examiners at Land Registry and the registered versions show two different sets of hand-written annotations.

3.1.2 Creation of a custom AFR for input

From this discussion it can be seen that the diversity of formats, information and styles contained in AFRs is far from uniform. The idea of using a set of AFRs with their corresponding LandXML files in a ML or DL environment to progressively train and predict a LandXML output may seem attractive. However, given this complexity and variety, great amount of manipulation would be required to obtain a workable data set from these sources. The preferred approach was to create a custom made, clean and simplified model AFR with the minimum required information. This model AFR was specifically designed to test different algorithms for the necessary steps until a satisfactory result has been achieved.

To illustrate the information required for this model, a simple AFR can be reviewed. Figure 8 shows a typical AFR for a PS. The relevant text information that needs to be captured has been highlighted in yellow. The points circled in red are the survey marks connected to. The arrowed nodes represent boundary corners and connections to road intersections.

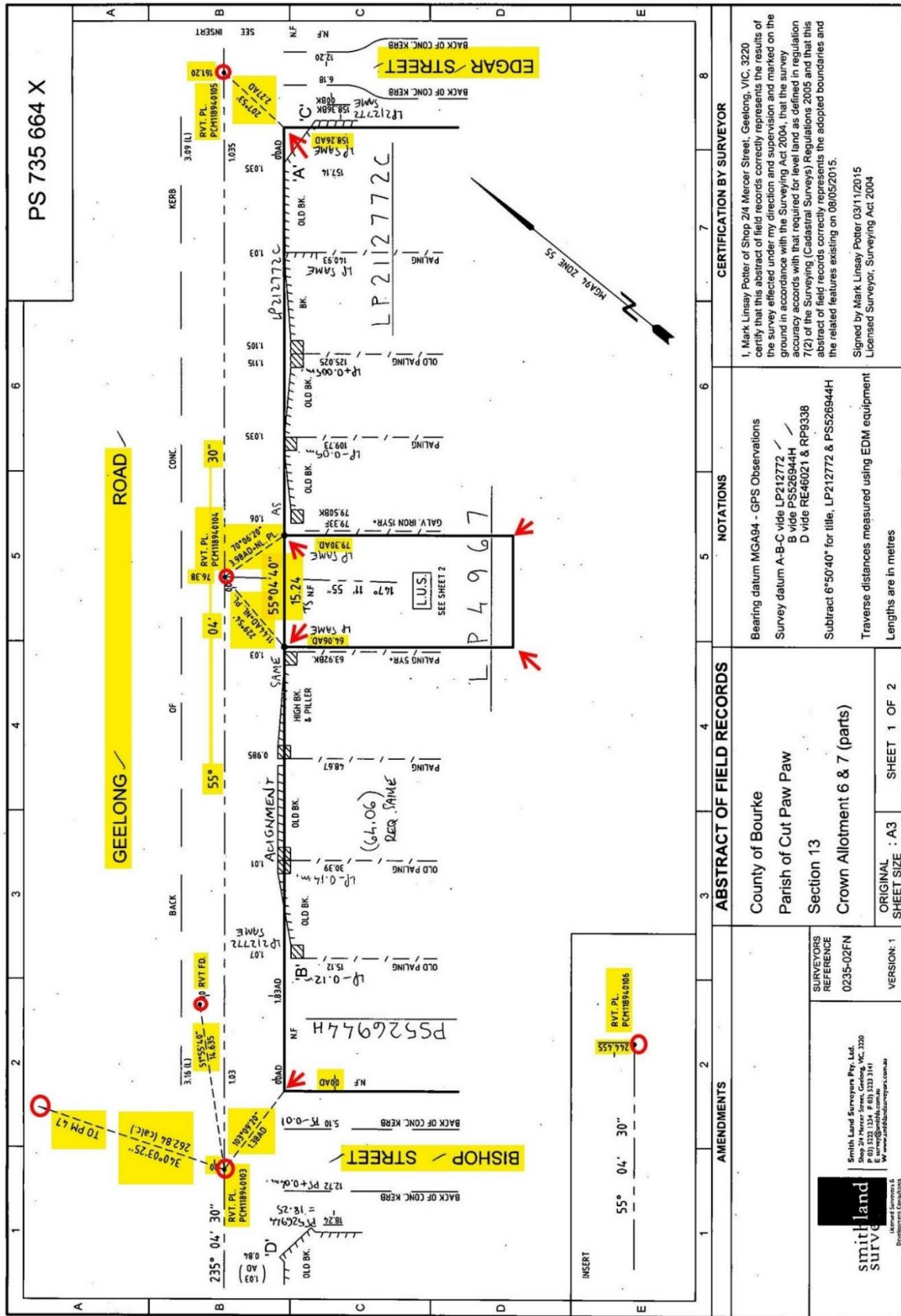


Figure 8: Abstract of Field Records for a Plan of Subdivision. The information to be captured has been highlighted: text in yellow, nodes with arrows and symbols circled in red.

The relevant required information is in the form of written text in the original document. This text can be one of the following:

- Bearings (directions) in the form (DD)DMM(SS), such as 96°45'
- Distances in the form xx.xx, such as 20.19, be it as point to point or as running distance.
- Other text elements such as road names, descriptions and identifiers of survey marks, and descriptions of occupation.

This text is associated to a network of lines that represent a visual skeleton to connect the points linked by the dimensions. Importantly, this line work is not drawn to scale and cannot be relied on for the position of the different elements in the plan. Only the position relative to each other can be used as an indicative guide.

Three different line styles are traditionally used, as depicted in figure 9 (Victoria 1997):

- Solid boundary lines, thick for the subject lot and thin for abutting boundaries.
- Dashed radiations.
- “Chain” style for traverse lines.

LINES — TYPES AND DIMENSIONS






1 Designating letter	2 Type of Line	3 Example of Line	4 Minimum thickness according to Sheet Size, mm		
			A0	A1	A2, A3 A4
A	Continuous—thick		0.7	0.5	0.35
B	Continuous—thin		0.35	0.25	0.18
E	Dashed—thin		0.35	0.25	0.18
F	Dashed—thick	 S = 1 mm minimum q = 2S to 4S	0.7	0.5	0.35
K	Chain—thin, double dashed		0.35	0.25	0.18

Figure 9: Recommended line styles for survey plans in Victoria. Extract from “Survey Practice Handbook Victoria”

3.1.2.1 Elements included and excluded from the Model AFR

Given all the challenges and barriers discussed above, a model AFR was created that includes:

- A simple quadrilateral parcel. Three boundaries are at right angles while the eastern boundary intersects the northern and southern boundaries at oblique angles.
- The connection from the parcel to the nearest road intersection.
- The three different line styles for boundary (solid line style, thick for the subject land, thin for the abuttals), traverse (“chain” style) and radiation (dashed style).
- Conventional triangular symbol for two instrument stations.
- Two road names.
- Bearings and distances for all relevant line segments. One segment only shows a distance and two segments do not show bearing or distance. Some interpretation is required here as will be seen in the discussion of results (4.2.3)

The following elements are commonly found in AFRs and have been omitted deliberately to obtain results as unequivocal as possible:

- Offsets and running distances. All distances are shown as point to point.
- Occupation along boundaries shown diagrammatically.
- North arrow, containing both lines and text, usually the denomination of the bearing datum used.
- Descriptions of survey marks.
- Legal description of subject parcel and abuttals and surface area.
- Additional instrument stations and other survey marks connected to by radiation.
- Other miscellaneous text elements, for example computations and annotations from the examiner are frequently found in registered AFRs. These are almost always handwritten and would also need to be removed in pre-processing.

3.1.2.2 Model AFR

The plan depicted in figure 10 was created as a Model AFR following the criteria discussed above. A matching LandXML file will also be created as a guide for the required outcome and is contained in Appendix B. The Model AFR is the first research outcome.

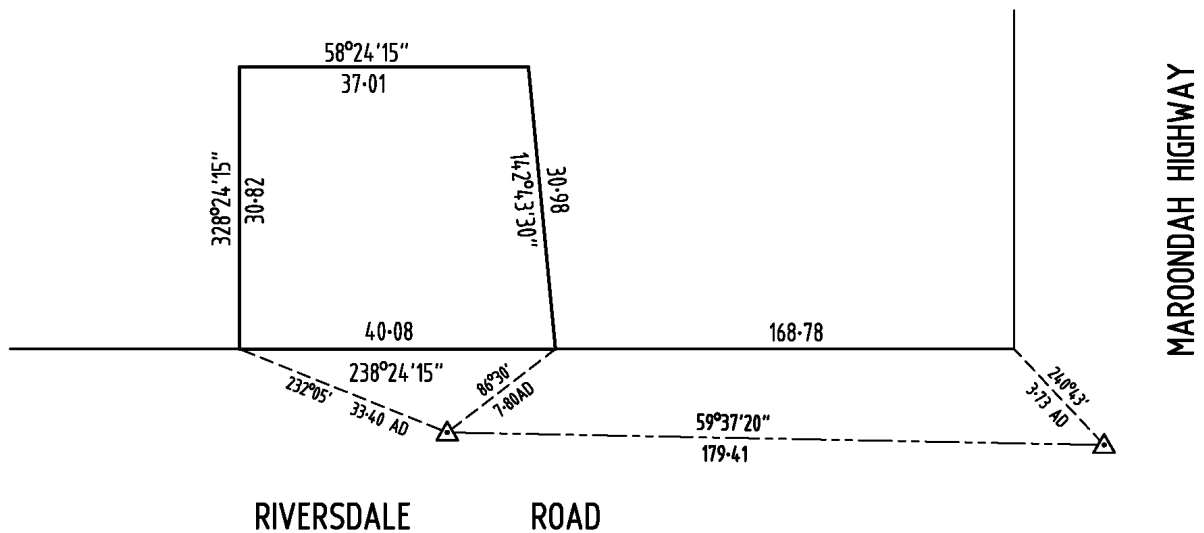


Figure 10: Model AFR

3.2 Output to a LandXML file

Some plans of subdivision and their accompanying survey information have already been submitted as ePlans in LandXML format. The accompanying AFR for these plans has been recorded in the traditional .pdf format. At this stage, the LandXML data provided by the surveyor who created the plan is not being made available and the LandXML data provided by LASSI – SPEAR only contains approximate co-ordinates of property corners as plotted in Vicmap Property™. These co-ordinates are not necessarily survey accurate and are not based on the information provided by the surveyor (Olfat 2019, pers. comm. 28 March). This data has been used for learning purposes.

3.2.1 File format

A typical LandXML file for an ePlan submission has the following structure (See figure 11). The example is taken from an existing registered ePlan in Victoria:

```

1  <?xml version="1.0" encoding="UTF-8"?>
2  <LandXML xmlns="http://www.landxml.org/schema/LandXML-1.2"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.landxml.org/schema/LandXML-1.2
   http://www.landxml.org/schema/LandXML-1.2/LandXML-1.2.xsd"
   version="1.0" date="2017-04-12" time="00:00:00">
3  <Units>
6  <CoordinateSystem datum="Local" horizontalDatum="Local"/>
7  <Application name="Software name" version="1.0"/>
8  <FeatureDictionary name="
   "xml-gov-au-vic-icsm-eplan-cif-protocol" version="1.8"/>
9  <Parcels>
229 <CgPoints>
271 <Survey>
424 <Monuments>
432 </LandXML>
433

```

eXtensible Markup Language file length : 22,640 lines : 433

Figure 11: Structure of a LandXML file for an ePlan

3.2.2 Required data items

Of interest for the back-capture of relevant information are the following headers:

- <Parcels> includes the geometry of all the parcels represented in the plan. As a minimum, this would typically mean one closed polygon for the parcel being re-established and two road “parcels” to which intersection the main parcel is connected. In the case of plans of subdivision, the number of parcels would increase to include newly created lots and common properties inside of the parent lot. This information should be readily available in the plan being back-captured in the form of text elements showing the boundary dimensions.
- <CgPoints> includes the co-ordinates of all the nodes within the plan. These are both monuments such as survey marks, be it traverse stations or side shots, and boundary corners. Generally, the co-ordinates must be calculated from the information provided in the plan, so this is a step that happens after the back-capture has been completed.
- <Survey> includes three elements:
 - The <SurveyHeader> contains administrative information such as the name of the plan, jurisdiction, local government area details, etc. This information is

usually given in the plan under the “location of land” field and could be readily captured by means of traditional text-based OCR. This information is also available directly from LASSI – SPEAR.

- `<InstrumentSetup>` follows and correlates each instrument station to a point identifier. This requires interpreting the plan to identify the instrument stations, which can be represented as symbols or merely as the intersection of multiple lines or both. One of two main pillars of this project is attempting to automate the identification of these points using CV techniques.
- `<ObservationGroup>` lists reduced observations, that is bearings and distances, from each instrument setup. This information should be readily available in the plan being captured in the form of text elements showing bearings and distances between instrument points.
- `<Monuments>` gives descriptions of the survey marks visited and correlates them with point identifiers. These descriptions could also be an element to be back-captured from the plan.

A LandXML file following this structure and requirements was created to match the Model AFR as a guide for the be the desired end result of the capture and conversion and is contained in Appendix B. Co-ordinates were computed starting on an arbitrary point at 5000, 1000 on CGPNT-1. The remaining sections of the code were completed as per the above description, including correlating station setups with points, observations between stations and monument descriptions.

In Figure 12, the Model AFR has been annotated to show the labels allocated to the different elements in the code contained in Appendix B. It contains point IDs allocated to parcel corners and survey marks. Point IDs have also been given to points at the centre of the road names, for visualisation purposes.

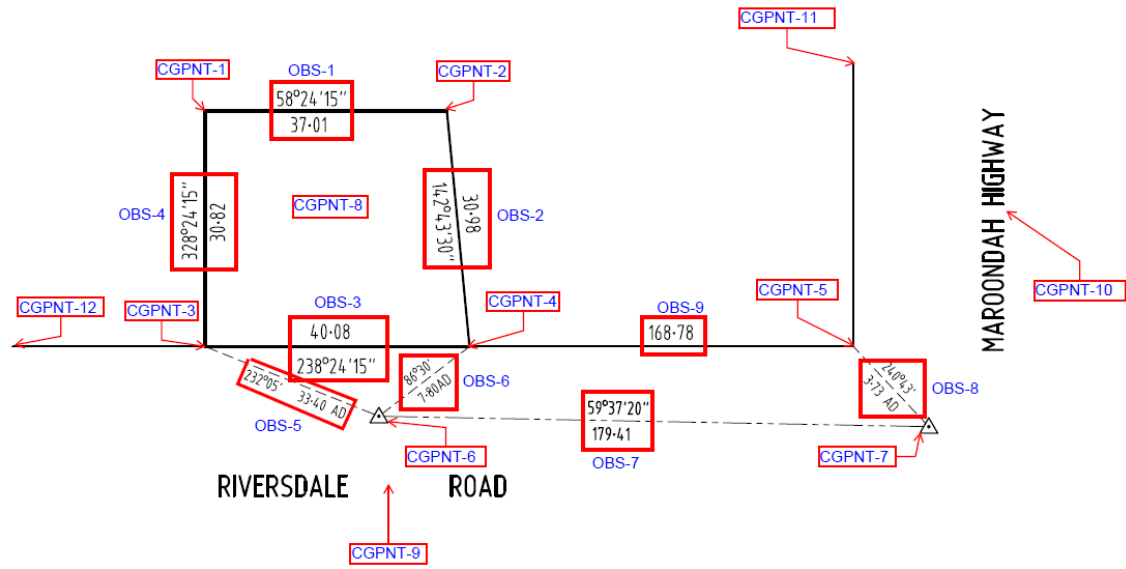


Figure 12: Model AFR annotated

3.3 Creation of a workflow for the information detection procedure

As described above, some of the elements required for the LandXML file are readily available as text elements in the input .pdf:

- Bearings/distances, of three types: boundary, traverse and side shot.
- Monument descriptions.
- Road names.

The following elements need to be derived before been incorporated into the LandXML file. They require the interpretation of the graphical content in the plan:

- Point IDs: First, the points of interest need to be identified. All nodes in the image are points of interest. Nodes represent a point with co-ordinates and refer to edges starting at that point. Edges are areas of high contrast, which, in the case of this project are coincident with lines. Following the terminology used in (Katona & Hudra 1999), nodes can be endpoints (one edge, such as survey marks measured by radiation), breakpoints (two edges, such as bends in an alignment or marks radiated from two different instrument points) or junction points (multiple edges, such as most boundary corners, where at least two boundary lines and one radiation line will meet). An algorithm could read the image from left to right and generate successive point IDs to each node encountered.
- Co-ordinates: generally, there will be at least one Permanent Mark in the plan with known Map Grid of Australia (MGA) co-ordinates, available through SMES. This could

be used as the origin of the co-ordinates of the remaining points. For this purpose, though, to maintain the simplicity of this initial stage, arbitrary co-ordinates were allocated to an initial point and local co-ordinates calculated. This removes the need to reach for one additional external source of information.

- The purpose of each dimension (boundary, traverse or side shot) can be identified through the different line styles.

The project proceeded under the assumption that text elements would be captured by means of OCR whereas graphic elements would be captured with CV feature detection algorithms. This dual approach yields a series of outputs that are at times redundant. Redundancy provides validation and helps eliminating false positives, as will be discussed in the results section (4.2.3)

There is a series of steps involved in the capture and conversion to LandXML using CV which have been identified in other areas in the literature. Some of these steps are:

1. Background/foreground separation (Roy et al. 2012; Westphal et al. 2018) This will not be necessary for the first attempts with a clean plan specifically generated for the purpose of creating a model, but it will become relevant once real-world examples are being captured.
2. Text/non text separation (Bhowmik et al. 2018). AFR documents fall clearly within Class-2 of Bhowmik's taxonomy of "general offline document images". They contain text scattered over the page in multiple sizes and orientations. They also contain many non-text elements such as lines and symbols. Finally, they may contain "a significant amount of noise similar to the content", such as added annotations.
3. Line/symbol separation. Once the background and the text have been removed, all that remains are often overlapping lines and symbols. Separating these classes is one of the main challenges in OMR (Calvo-Zaragoza et al. 2018).
4. The CV field has brought improvements in the detection and recognition of skewed and multi-oriented text (Ouwayed & Belaïd 2012; Liu et al. 2019). In the case of this project, relevant text is always aligned or perpendicular to a line. This may simplify the task of detecting relevant text as it can be restricted to a set of given orientations (see steps 6- 7).

Open source repositories of CV algorithms are publicly available for implementation of CV projects. For each step of the proposed workflow, a pre-programmed algorithm was sought and tested to determine whether any usable results could be acquired for export to the LandXML file.

Each step will be discussed individually in Chapter 4. Some algorithms allow for manipulation of variables to improve output. Chapter 4 includes detailed discussions of the algorithms used and their adequacy as well as barriers encountered during implementation.

3.4 Development environment

The experiments were conducted on a general-purpose computer with an Intel® Core™ i5-4210U CPU @ 1.70GHz 2.40 GHz processor, 4.00GB RAM and 64-bit Windows 7 Professional operating system.

The Model AFR was drafted using AutoCAD, printed as a vector .pdf and converted to .tiff with Adobe Acrobat XI Pro.

The programming language used was Python 3.0 through its Anaconda distribution. A Jupyter lab was setup to run the code.

OpenCV was the library of choice for the feature extraction and image processing functions required.

OCR was performed with Tesseract, through the Tesseract wrapper.

Adobe Acrobat DC was used to trial text detection within a commercial application.

A note about pixel geometry: most computations performed by the program are nothing other than co-ordinate geometry on the image pixels. Due to the nature of the computer processor reading files from left to right and top to bottom, the conventional origin of co-ordinates is the top left corner of the image, with the y co-ordinate increasing downward.

CHAPTER 4: IMPLEMENTATION AND RESULTS

4.1 Workflow: Double Filter Capture

As discussed, a characteristic of AFRs is that they contain information coded in two parallel ways: text on the one hand and graphics, being lines or symbols, on the other. The drawing is not to scale and as such, the lines provide a guide for the topology of the points of interest but do not convey any dimensional information. This project takes advantage of this characteristic and captures the information in two parallel streams. The redundancies obtained by this dual approach provide validation and help eliminate false positives.

The workflow depicted in figure 13 was progressively created as a guide for a program that would:

- a. extract the relevant information from an imaged AFR of a cadastral survey (labelled “Image”); and
- b. output point IDs and the bearings and distances connecting them. These would be presented in a suitable format to be incorporated in a LandXML file. These outputs are labelled “LandXML” in the workflow.

The multiple steps contained in this workflow are connected and interdependent, however a visual representation of the required sequence was deemed useful to assist both the reader in understanding the processes undertaken and the author in testing algorithms on plans. The process was separated into two phases. In the first phase, steps relating to the capture of graphic related items, labelled “Geometry Filter”. In the second phase, steps relating to text, using OCR, labelled as “Text Filter” are completed.

These two filters can occur simultaneously and independently of each other. A third phase follows, in which the results from the Geometry and Text filters are validated through the redundancies obtained. Due to this dual filtering of information, this process has been labelled “Double Filter Capture” (DFC).

Reading the first left column downwards, within the “Geometry Filter”, the first sequence of transformations to be performed on the image can be found, numbered from 1 to 4. These processes yield three sets of data that are to be stored in arrays, shown in the second column. These are:

- text boxes with their location, in pixel co-ordinates,
- lines and their orientation, in radians, and
- points with their position, in pixel co-ordinates.

After this first block, steps 5 to 7 (the “Text Filter”) follow. Steps 5 and 6 occur multiple times, as they require that the original image (where the text has not been removed) is rotated to each of the orientations stored as a result of step 3 and OCR performed on each of those rotated images. Text data is obtained in step 7 which is stored in an array including their location and a flag about their content: bearing or distance.

Steps 8 and 9 cross reference the results from the phases one and two to identify and eliminate false positives and to ensure that every pair of detected points can be reliably allocated two text elements representing the bearing and distance between the points (steps 10 and 11).

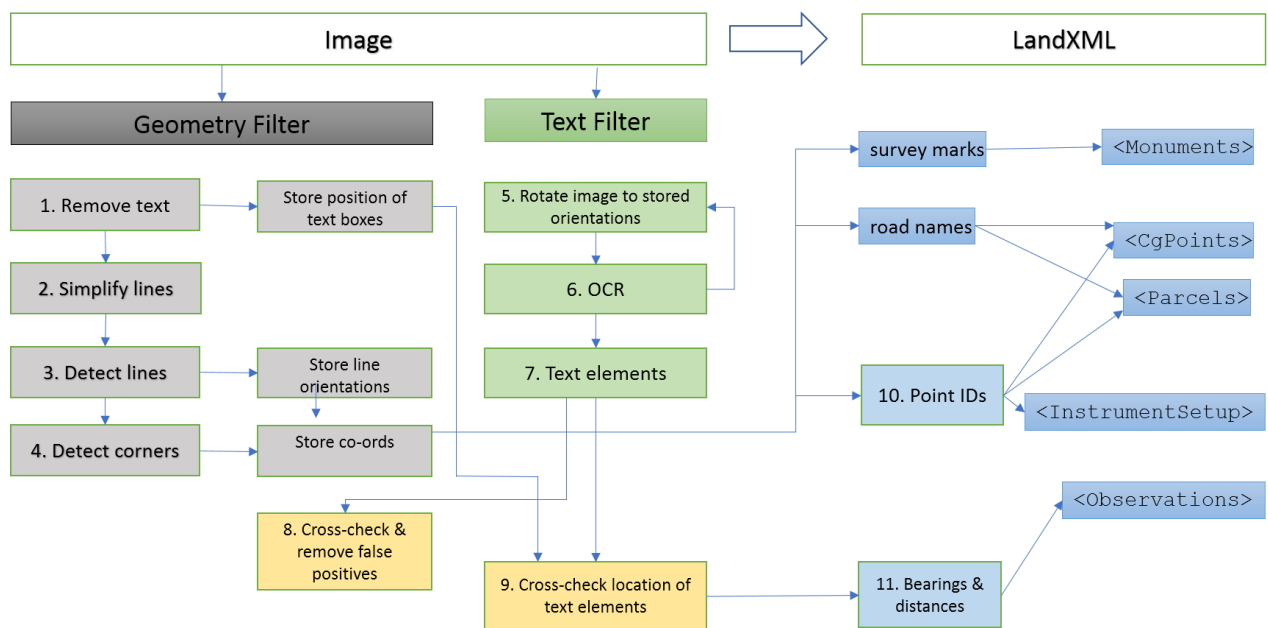


Figure 13: Double Filter Capture

4.2 Implementation

In order to implement the DFC process, algorithms were selected from the OpenCV repository. For this initial case study, the aim was to determine at least one algorithm that could at least partially complete each step of the DFC. It is acknowledged that further algorithms may exist which complete the tasks more accurately, however the aim of this study was not to complete a full review of the available repositories, but rather to evaluate whether further refinement and expansion of this model using CV was warranted.

Each step in the phase is outlined below, with results indicated for each individual step.

4.2.1 Phase one: Geometry Filter

4.2.1.1 *Remove text*

Aim

The aim of the first series of transformation is to extract the topology of the required elements. It is necessary to identify which points and which lines connecting them are of interest. To facilitate this task and to remove the risk of an unmanageable number of false positives, the image needs to be “cleaned up” and reduced to a skeleton of lines.

A text detection algorithm could be used to remove all text fields and at the same time extract the location and orientation of the bounding boxes of those text elements. The required output is a 4-column array with as many rows as text elements found. The fields for each column would be:

- x: horizontal distance from the left edge of the box to the co-ordinate origin, being the left edge of the image.
- y: vertical distance from the top edge of the box to the co-ordinate origin, the top of the image.
- w: width of the bounding box in pixels.
- h: height of the bounding box in pixels.

Result

Text removal can be partially achieved by the method `GetComponentImages` of the Tesseract library. This method also extracts the text and provides a confidence level of the text recognition. The orientation of the box is not recorded, and all boxes are rectangular and oriented parallel to the co-ordinate grid.

Text was removed with `GetComponentImages`. The five available page iteration levels (RIL) were tested:

- `RIL.SYMBOL` returned 123 text elements (see figure 14)
- `RIL.WORD` returned 18 text elements (see figure 15)
- `RIL.BLOCK` returned 14 text elements (see figure 16)
- `RIL.TEXTLINE` returned 15 text elements (see figure 16)
- `RIL.PARA` returned 14 text elements (see figure 16)

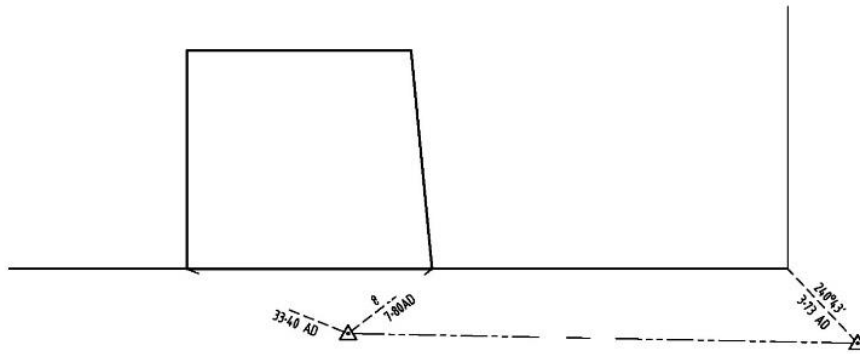


Figure 14: Text removed with RIL.SYMBOL

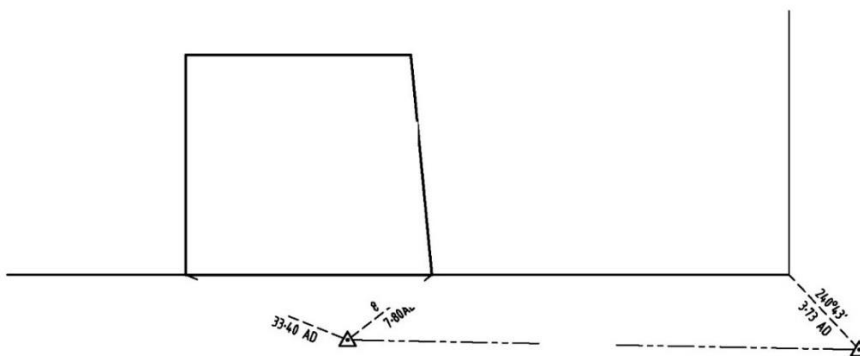


Figure 15: Text removed with RIL.WORD

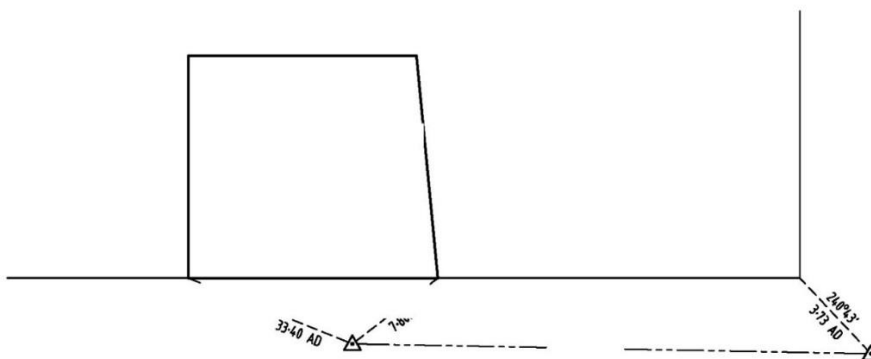


Figure 16: Text removed with RIL.BLOCK, RIL.PARA and RIL.TEXTLINE

Two main difficulties were encountered when using this approach:

- Only text that is oriented horizontally or vertically or near horizontally or vertically is detected. The remaining text elements pose obstacles to the following line detection tasks.

- Removing the detected text also removes lines that fall within the bounding box. This creates additional line ends that would be false positives in the line detection process.

Given the limitations detected in this step, an existing commercial software option was trialled. Adobe Acrobat Pro DC (Adobe, 2019) was used to automatically detect text and convert to editable format. Detection of presence of text, including on non-horizontal and non-vertical orientations was highly accurate. At this point the same barriers were encountered as using tesseract algorithms, namely, partial or full deletion of lines (see figure 17). Text located further away from lines was able to be rapidly removed with a “click and delete” approach, however further text required manual deletion in a graphic program or order to allow further stages of this project to be tested and implemented.

This step would require testing or programming from a more skilled computer programmer or for creation of a new algorithm for this purpose and trained using a true ML approach which is beyond the scope of this project.

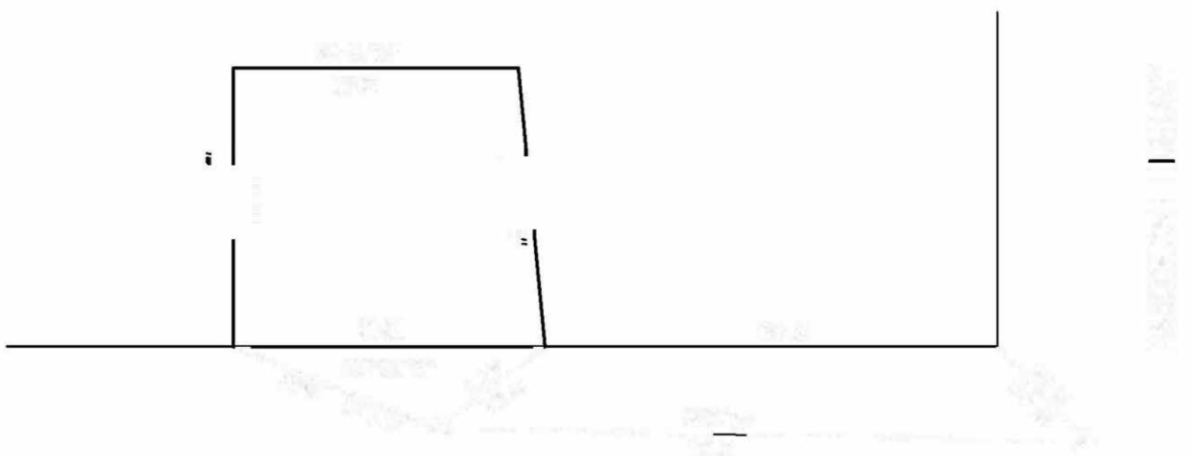


Figure 17: Text removed automatically using Adobe Acrobat Pro DC.

An alternative approach, namely to detect lines and keep them rather than identifying text for removal has been considered, as will be used later to work with lines of the file (see section 4.2.1.3). This approach was trialled using the HoughLinesP (probabilistic Hough transform) algorithm. While lines are successfully deleted, additional false positive lines are created from text which is falsely identified due to multiple ‘o’ value pixels in the same orientation. See figure 18.

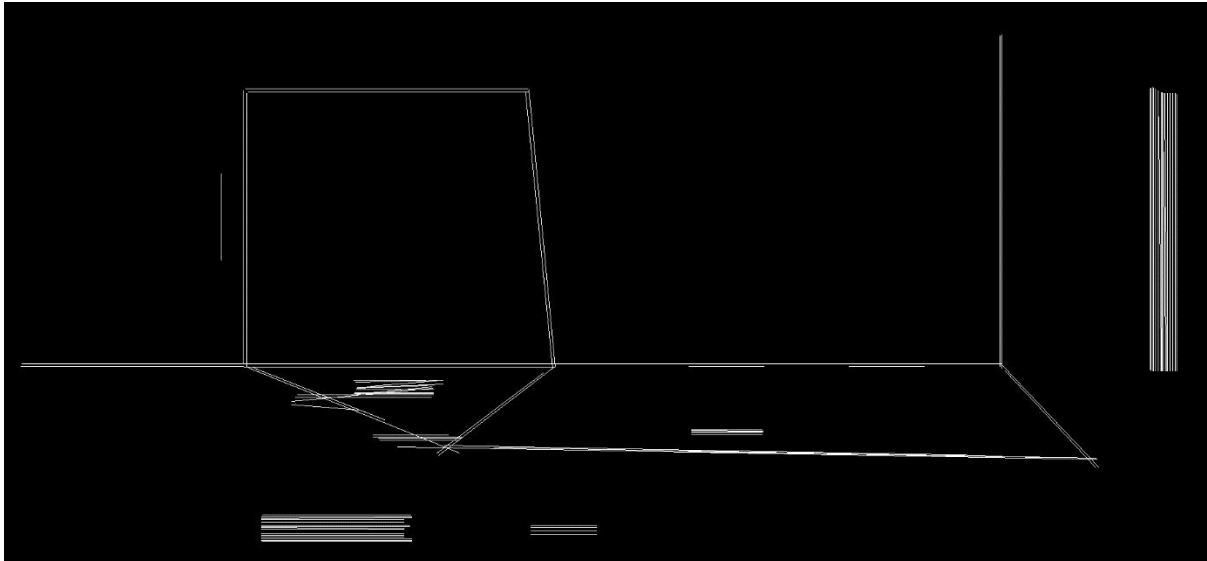


Figure 18: Detection of lines using HoughLinesP as method of text removal.

For the purpose of further stages of this project, the text elements were removed manually (see Figure 19).

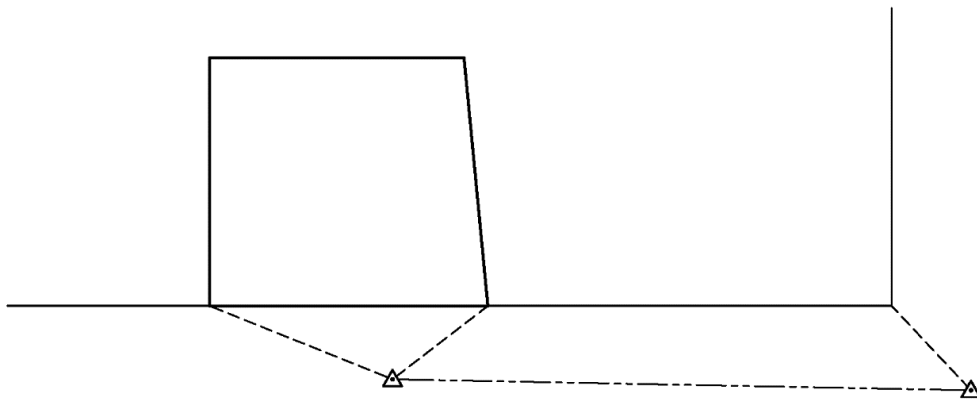


Figure 19: Text removed manually using graphics software.

4.2.1.2 Simplify lines

Aim

In the mentioned examples from the literature, initial steps involve pre-processing the images to reduce noise (such as dirt speckles in scans, through bleed from the reverse etc) and to adjust contrast levels and simplify colours (Calvo-Zaragoza et al. 2018; Ignjatić et al. 2018).

As the Model AFR was specifically prepared for this purpose, no issues arising from scanning quality are present, however, some pre-processing is still required. The corner detection algorithm that will be used later to detect the line intersections requires that the input image is binary rather than greyscale. The Model AFR image is black and white, and most pixels have the values of either 0 (black) or 255 (white), but there are grey values along the edges of lines and symbols. In “real world” AFRs, these greyscale values can be a by-product of the scanning process or of the .pdf to .tiff conversion. To ensure that all the relevant shapes are recognised by the line detection algorithm, these values need to be simplified. This pre-processing step also has the added benefit of a reduction of the file size, which in turn increases computation efficiency.

Algorithm: Thresholding

An easy way to achieve this is to apply a threshold that converts all pixel values higher than a given threshold to 255 (white). This in turn reduces the size of the image, as most pixels are “turned off” into a 0 value (black). In figure 20 the median value 128 has been applied as the threshold. The `thresh_binary_inv` function of the OpenCV library was used for this step. Note that in this case the inverse function was used as the vast majority of pixels are “background”.

Result

The threshold gives a clean image free of grey shades, yet the results in the steps that follow yield numerous false positives. These can be traced back to the line thicknesses and can be avoided in this step by instead using an algorithm that detects edges, the Canny algorithm.

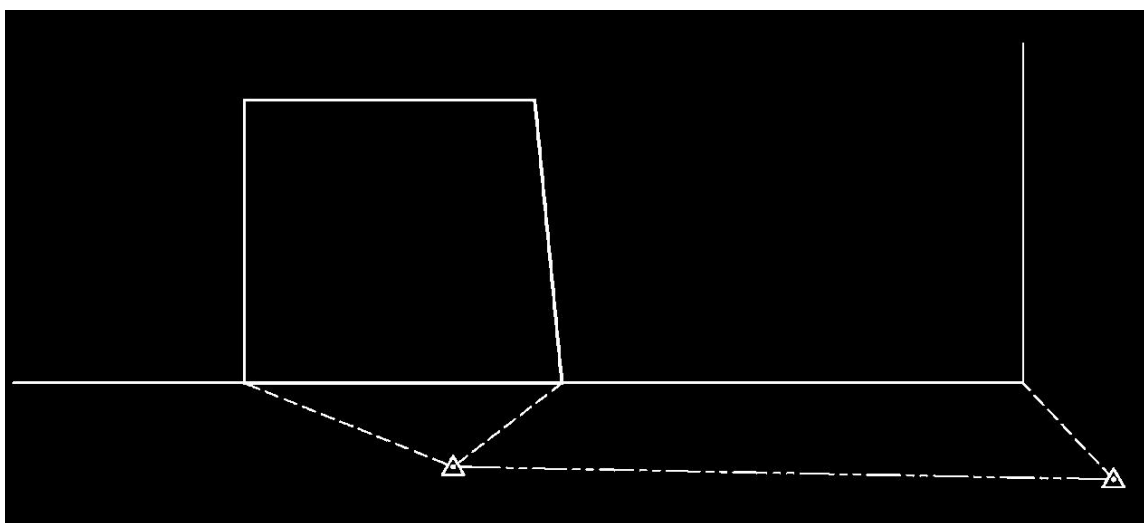


Figure 20: Threshold 128 applied

Algorithm: Canny

The Canny86 algorithm is applied to detect edges in the image. The multi-step algorithm is described in Canny (1986) and detects regions of sharp contrast in brightness. In summary, the image is blurred at first, then the intensity gradient in each direction is calculated for each pixel and non-maximum values suppressed, thus eliminating weak edges. Then a double threshold is applied to eliminate false positives caused by noise or colour variation. The algorithm was designed to detect the edge of solid objects in front of a background, however it is effective for lines in this case as no other solid objects are present.

Result

As a result of this process, all lines are seemingly duplicated (figure 21), as either side of each line is detected as an edge. The symbols also seem to have been made more complex with additional lines. This is in fact an advantage as will be seen in the next step. It could appear that accuracy might be lost as the actual point of intersection between lines has disappeared. Here it is worthwhile to remember that the plan being processed is a sketch of the surveyor's field work and is not drawn to scale. An approximate location indicative of the relationship between point is all that is required, and a displacement of a few pixels will still give a good result.

The Canny function allows for manipulation of three parameters:

- Minimum threshold.
- Maximum threshold. Canny recommends this to be triple the minimum threshold.
- Aperture size of the Sobel operator (the edge detection operator).

The Model AFR is a greyscale image of a black and white document (note that Canny is used as a replacement for thresholding). As such the amount of filtering required is minimal and the output image only contains the values 0 (black) and 255 (white). Different thresholds were tested and it was found that a maximum threshold higher than 1300 resulted in information loss. A minimum threshold higher than 800 also led to information loss. The final thresholds used were 50 and 150.

The aperture of the Sobel operator must be 3, 5 or 7. This is the size of a pixel matrix used to detect edges. The number of white pixels in the output indicates the range of sensitivity of this parameter (see table 1):

Value \ Aperture	3	5	7
0 (black)	714435	714321	714299
255(white)	7309	7423	7445

Table 1: Values obtained with different apertures of the Sobel operator

Although it does not seem significant, the number of false corners detected in step 4 increases if aperture 7 is applied here. Corner detection results did not change between values 3 and 5.

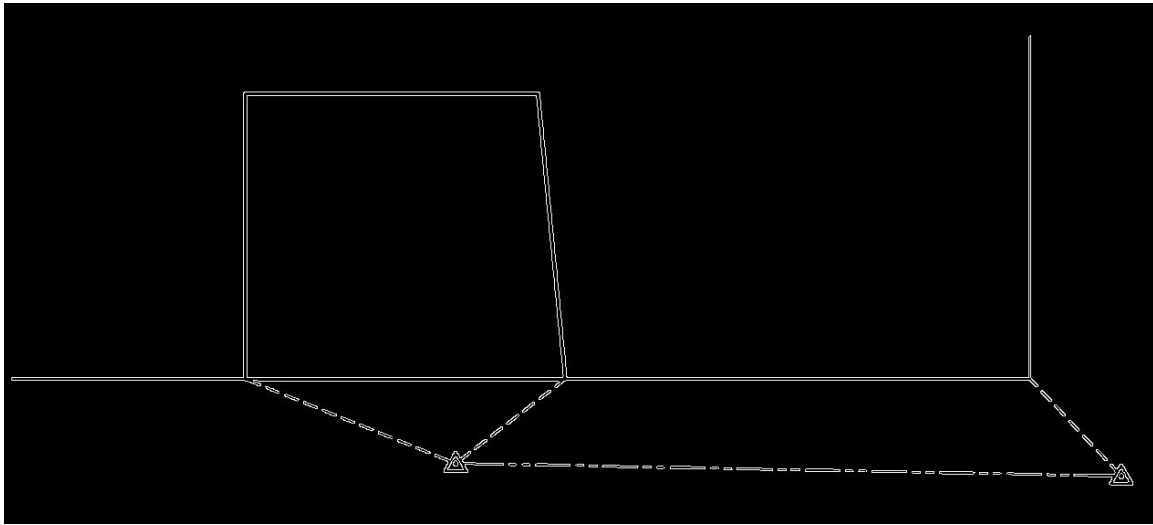


Figure 21: Canny algorithm applied

4.2.1.3 Detect lines

Aim

It is necessary to identify nodes (line intersections) as a guide for the text objects. As discussed above, these lines are usually drawn in one of at least three different types of line styles. Dashed and “chain” style lines are made up of line segments that could be detected as a line end by an algorithm detecting corners. To avoid this and ensure that only the relevant nodes are detected an intermediate step is required that converts broken lines into solid lines.

Algorithm: HoughLinesP (probabilistic Hough transform)

Hough transform is used to detect simple shapes that can be expressed by only a few parameters, such as a line, which can be represented by two parameters: slope and intercept (Duda & Hart 1972). This algorithm is also useful when detecting incomplete shapes, as is the case in the broken line styles. The output of a regular Hough transformation is a two-element vector (ρ, θ) (see figure 22) whereby ρ is the normal distance from the co-ordinate origin (top left corner of the image) and θ is the angle (orientation) of the line in radians.

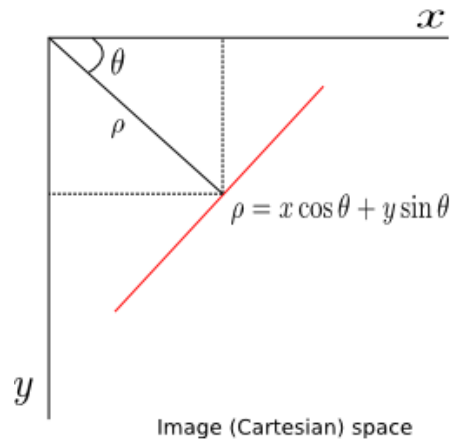


Figure 22: Hough transformation output. Image taken from <https://nabinsharma.wordpress.com/2012/12/26/linear-hough-transform-using-python/>

The location of the nodes (line intersections) is the required information, therefore the probabilistic Hough Transformation was used (see figure 23). This algorithm outputs a four-element vector with x and y co-ordinates of both ends of each line.

Several parameters can be manipulated for this task:

- resolution of the parameter ρ in pixels
- resolution of the parameter θ in radians
- threshold
- minimum line length, ensuring that short line segments such as in the symbols are discarded.
- minimum line gap, ensuring that only genuine broken line styles are detected.

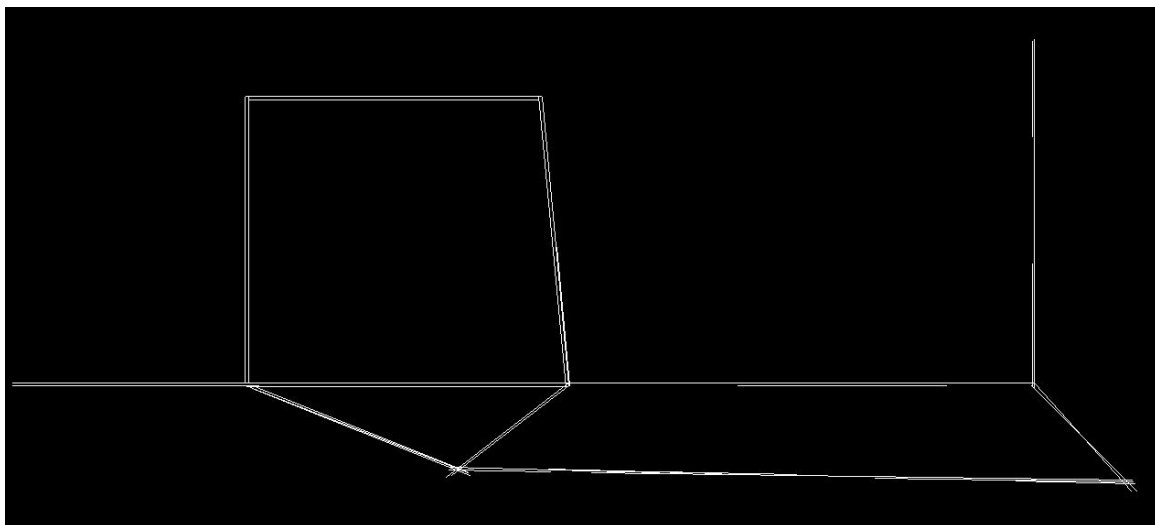


Figure 23: Probabilistic Hough Transform applied

Result

The expected number of lines was 22, considering the doubling caused by the Canny algorithm. The closest number of lines detected higher than 22 is 24, with the values $\rho = 2$ and $\theta = 30$. Given variables are set based on line thicknesses rather than details, it can be assumed that these variables would be adequate for other AFRs with a similar level of complexity. This is an area for further investigation and is beyond the scope of this project. A number lower than 22 would be missing information, higher would mean redundancy, which is preferred. It should be noted, values do not need to be identical but rather be so close as to indicate that they pertain to the same line, or segments of the same line, which has been duplicated in the previous step. These values also yield the correct number of corners (7) as will be seen in the next step (see table 2).

ρ	1	2	3	4	5	6
θ	15'	15'	15'	15'	15'	15'
corners	6	7	8	8	7	9
lines	20	29	25	43	38	39

ρ	1	2	3	4	5	6
θ	30'	30'	30'	30'	30'	30'
corners	7	7	7	8	8	9
lines	17	24	34	33	36	38

ρ	1	2	3	4	5	6
θ	1°	1°	1°	1°	1°	1°
corners	10	7	9	8	9	10
lines	18	21	29	31	38	35

ρ	1	2	3	4	5	6
θ	2°	2°	2°	2°	2°	2°
corners	9	9	8	8	10	10
lines	21	25	32	31	35	38

ρ	1	2	3	4	5	6
θ	3°	3°	3°	3°	3°	3°
corners	11	8	9	8	8	6
lines	17	26	29	27.5	32	37

Table 2: Number of corners and lines detected by the probabilistic Hough transform with different ρ and θ values.

The lines detected by the probabilistic Hough transform algorithm with the values $\rho = 2$ and $\theta = 30$ are listed in table 3. Line IDs have been allocated for convenience. Inspection of the coordinates on the table reveals:

- There are seven pairs of lines with neighbouring start and end points, consistent with the duplication in step 2. These pairs of lines are 3 – 5, 6 - 10, 7 – 11, 8 – 9, 13 – 14, 16 – 19, 18 - 24.
- There are eight possible false positives, which appear to be segments of other lines.

These multiple lines will be valuable in the validation process as discussed in section 4.2.3.2.

Index	x	y	x	y	duplicate of	segment of
0	27	548	1514	548		
1	25	552	363	552		0
2	910	680	1658	693		4
3	1511	552	1511	51		
4	742	674	1656	690		
5	1514	555	1514	49	3	
6	368	553	368	131	10	
7	366	131	794	131	11	
8	795	132	836	553		
9	791	137	831	555	8	
10	363	547	363	134		
11	370	136	790	136		
12	1218	552	1332	552		0
13	370	554	690	683		
14	378	554	687	676	13	
15	661	675	751	675		4
16	1520	560	1656	705		
17	1333	552	1517	552		0
18	659	687	834	553		
19	1515	550	1662	705	16	
20	668	672	1272	683		4
21	737	676	927	680		3
22	364	553	835	553		
23	1048	552	1162	552		0
24	656	685	812	568	18	

Table 3: Multiple lines detected by probabilistic Hough transform

4.2.1.4 Detect corners

Aim

In the Model AFR, all points of interest are intersections of lines. This will not always be the case in a ground truth scenario, as some line intersections do not represent points of interest. This would ideally be detected and rectified in the cross-checking phase (steps 8 – 9).

Algorithm: Harris Corner Detector

The corner detection algorithm described in Harris & Stephens (1988) returns points where two edges intersect. As the direction of those edges changes, the gradient of the image varies in both directions. This fact is used to detect the edge. A neighbourhood window size is required, and the image is broken down in squares of that size that are evaluated as containing a corner or not.

Result

The Harris corner detector was found to be too sensitive and it gave too many false positives in lines other than at 0° or 90° (see figure 24).

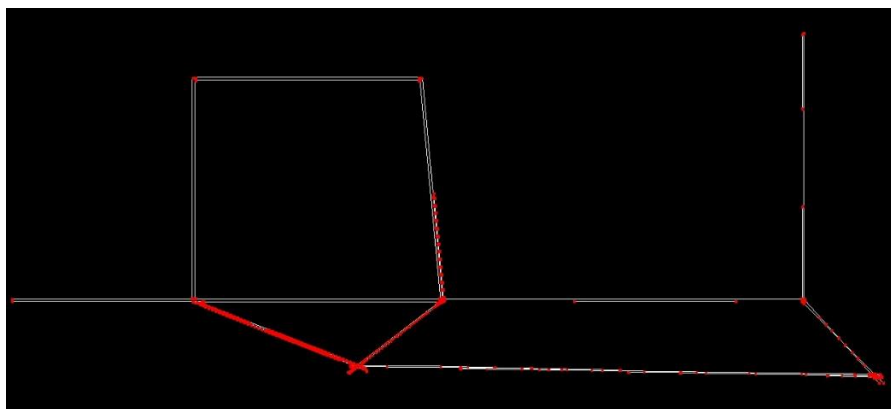


Figure 24: Corners detected by the Harris algorithm in red

Algorithm: Shi/Tomasi Corner Detector

Shi-Tomasi corner detector (Shi & Tomasi 1994) refines the Harris algorithm by grading the results according to the quality of the corners detected. The parameters that can be manipulated are:

- A maximum number of corners can be specified. If the number of detected corners exceeds the maximum, only the strongest are returned.
- Quality threshold, which is a factor of the quality of the best corner detected.

- minDistance is the minimum possible Euclidean distance in pixels between returned corners. It is important to note that high accuracy is not necessarily required for this step. As the plan is not drawn to scale and is only an indicative diagram, the positions of the points are approximate. It is sufficient to detect the general area of a node. Limiting the minimum distance between nodes ensures that duplications are avoided.

Result

The corner detection parameters were set as follows:

- maxCorners has been set to 25. This is only an empirical value obtained by observation. In this case we know that 7 corners are required to be returned.
- qualityLevel. If set too low, it was found that corners were returned along lines other than lines in cardinal directions, due to the low resolution of the picture. If too high, some relevant corners were not detected (see table 4).
- minDistance between returned corners. Setting this value too low would return double corners at the symbol points. A minimum of 20 was required to obtain good results. Higher than 150 would cause loss of corners. The distance between the two nearest corners in this plan is approximately 150 pixels. An intermediate value of 85 was used.

q level	0.9	0.85	0.8	0.75	0.7	0.65	0.6	0.55	0.5	0.45	0.4	0.35	0.3	0.25	0.2	0.15	0.1	0.05
corners	2	2	5	5	5	5	6	7	7	7	7	7	7	11	16	16	17	17

Table 4: Corners detected with different quality thresholds.

Applying this algorithm to the previously obtained image successfully yields 7 corners (see figure 25). The co-ordinates of the identified nodes are listed below. These nodes can be stored in an array that will be validated in step 8 (see 4.2.3.3) against the detected text boxes. After validation, these nodes can be allocated ID numbers as required for the LandXML file.

Index	x	y
0	1512	548
1	368	548
2	830	548
3	370	133
4	789	138
5	670	676
6	1648	695

Table 5: Table of co-ordinates for the detected corners

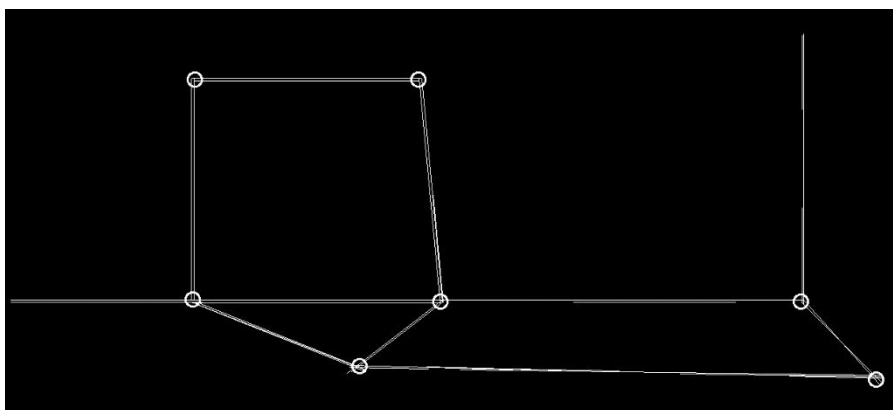


Figure 25: Shi/Tomasi algorithm applied

4.2.2 Phase two: Text Filter

For the following steps the original image with text is used again. This image will be subject to a series of rotations and text extractions and these outputs will then be validated against the data obtained in the first phase.

4.2.2.1 Image rotation

Aim

In order to facilitate the function of the OCR engine it must be ensured that the text to be captured appears horizontal on the screen. As seen previously, the relevant text contains the bearing and distance of each line, and these text elements are written along the corresponding line. The original image will be rotated to the orientation of each of lines detected previously. The transformations undertaken here are affine transformations which preserve the distance ratios and the collinearity of points. OpenCV includes a function called `warpAffine` for this purpose.

As the text elements are aligned to the corresponding lines, the co-ordinates obtained in section 4.2.1.3 can be used to compute the orientation of those lines, and then rotate the image to those orientations to obtain the text. This can be achieved by means of co-ordinate geometry. The rotation angle α must be given as a positive number for counter clockwise rotation and negative for clockwise rotation. The formula to apply is:

$$\alpha = -\tan^{-1}(\Delta y / \Delta x)$$

This formula takes the fact into account that the origin of the y axis is the top edge of the image. As such, a line sloping “south” will have a positive Δy .

As the rotation occurs about the centre point, the image must be translated to the centre point of the destination window. The magnitude of this shift is the coordinate difference between both centres:

$$Xshift = dst_rows/2 - rows/2$$

$$Yshift = dst_cols/2 - cols/2$$

Now the rotation can be performed. A rotation matrix is calculated using the function `getRotationMatrix2D` and applied to the translated image with `warpAffine`.

Result

In Figures 26 and 27 the image has undergone the described transformations and been rotated by 22° so the text reading “232°05’ 33.40AD” appears horizontal.

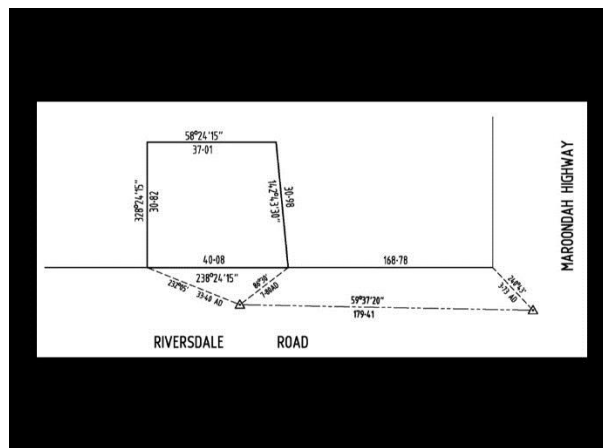


Figure 26: The destination window has been resized and the imaged shifted to the window's centre

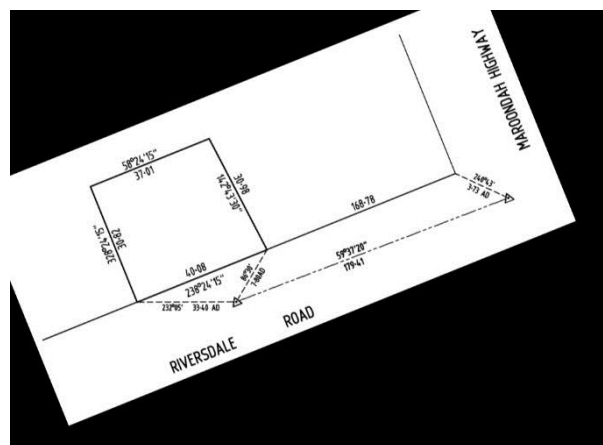


Figure 27: The image has been rotated by 22°

4.2.2.2 Text detection

Aim

Initially, the position of a box wrapping each text element is required, rather than each individual character. This can be achieved by the function `GetComponentImages`.

Result

Position of boxes wrapping each individual character was obtained (see figure 28). The test detected 15 text boxes. 8 boxes were correctly detected, 1 box extends further than the desired width and 3 text elements with skewed text are not detected.

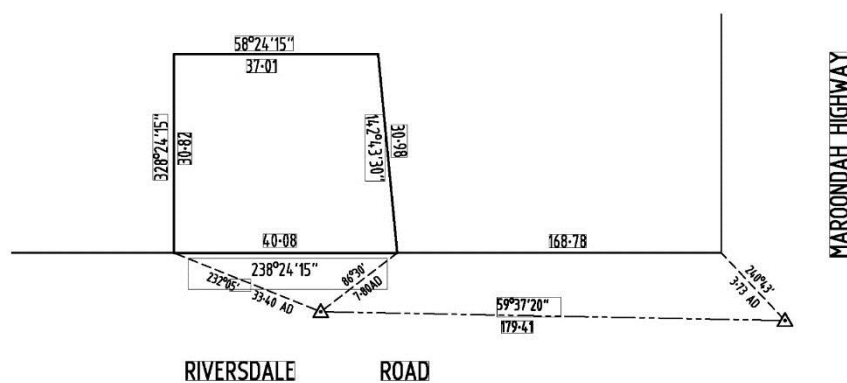


Figure 28: Text boxes detected by Tesseract

4.2.2.3 Text recognition

Aim

The text elements detected must be classified into bearings, distances and other.

A bearing will have the format $(nn)n^{\circ}nn'$ (nn'') where all characters are numeric and those in brackets may or may not be present.

A distance will have the format $(nnn)n.nn^*$ where all characters are numeric, those in brackets may or may not be present and there may be additional characters to the right than can be safely ignored, such as “AD” in the Model AFR.

A text element may contain both bearing and distance. The format will be $(nn)n^{\circ}nn'$ (nn'') $(nnn)n.nn^*$ where all characters are numeric, those in brackets may or may not be present and there may be additional characters to the right than can be safely ignored, such as “AD” in the Model AFR.

The array of text elements will have the format `[x, y, w, h, text, type]` where

- `x`: horizontal distance from the left edge of the box to the co-ordinate origin, being the left edge of the image.
- `y`: vertical distance from the top edge of the box to the co-ordinate origin, the top of the image.
- `w`: width of the bounding box in pixels.
- `h`: height of the bounding box in pixels.
- `text`: the detected text.
- `type`: bearing, distance or other (in this case, road names)

Result

Text detection was performed by means of Tesseract OCR, an open-source project started by Hewlett-Packard and further developed by Google. The `tesseract` Python wrapper was used.

13 different page segmentation modes (PSM) are available in Tesseract. This is from the Tesseract documentation (Smith 2019)

“Page segmentation modes:

- 0 Orientation and script detection (OSD) only.
- 1 Automatic page segmentation with OSD.
- 2 Automatic page segmentation, but no OSD, or OCR.
- 3 Fully automatic page segmentation, but no OSD. (Default)
- 4 Assume a single column of text of variable sizes.
- 5 Assume a single uniform block of vertically aligned text.
- 6 Assume a single uniform block of text.
- 7 Treat the image as a single text line.
- 8 Treat the image as a single word.
- 9 Treat the image as a single word in a circle.
- 10 Treat the image as a single character.
- 11 Sparse text. Find as much text as possible in no particular order.
- 12 Sparse text with OSD.

13 Raw line. Treat the image as a single text line, bypassing hacks that are Tesseract-specific.”

The image was processed through each of these PSMs and the best result was obtained with the default setting.

The following table shows the text recognised in the first instance by the text location algorithm (see table 6). The second column shows that text recognition improves considerably after rotation of the relevant text box.

Text box	OCR original	OCR rotated	rotation
0	328°24'15"	328°24'15"	-90
1	58°24'15"	58°24'15"	0
2	37:01:00	37-01	0
3	30-82	30-82	-90
4	40-08	40-08	0
5	WDEELT	14204330	90
6	86-0t	30-98	90
7	168-78	168-78	0
8	\%\ BEWE o8	238°24'15"	0
9)5~		0
10	RIVERSDALE	RIVERSDALE	0
11	ROAD	ROAD	0
12	s	59°37'20"	0
13	179-41	179-41	0
14	MAROONDAH HIGHWAY	MAROONDAH HIGHWAY	-90

Table 6: Comparison of text detection before and after rotation

However, rotations other than 90° do not provide good results. This is due to distortions introduced by rotations that do not line up with the pixel grid.

This example has been rotated by 84° to bring the text to horizontal (see figure 29). The OCR result is ' 172* ' : . 3 ' 30 "

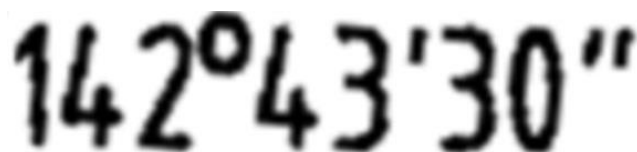


Figure 29: Text rotated to be horizontal. OCR result is ' 172* ' : . 3 ' 30 "

The same text rotated by 90° is not horizontal however yields a better result (14204330) yet still does not recognise the symbols for degree, minutes and seconds (see figure 30):

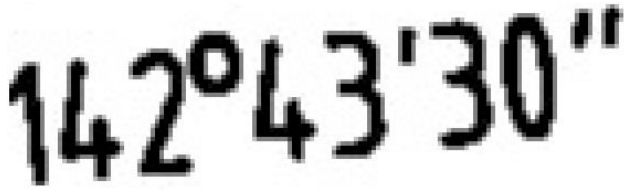


Figure 30: Text rotated by 90°. OCR result is 14204330

4.2.3 Cross-checks and removal of false positives

As has been discussed, several of the steps undertaken are likely to produce false positives. To ensure that false positives are eliminated the obtained data must undergo a series of validations:

- The number of lines from step 3 (4.2.1.3) should be greater than number of rotation angles for step 5 (4.2.2.1).
- The number of points from step 4 (4.2.1.4) should be lesser than the number of line ends in step 3 (4.2.1.3).
- Maximum number of text boxes from step 6 (4.2.2.2) must be double the number of lines from step 3 (4.2.1.3), as each line will have as a maximum one text element for its bearing and one for its distance. Some lines may have bearing and distance within one same text element. Some lines might only have a bearing or a distance. Different rules must be applied for these cases. If only a distance is given (such as 168.78 in the Model AFR example), the bearing of an adjacent text element in the same orientation that is assigned a line in the same orientation can be given. If only a bearing is given (such as 328°24'15" in the Model AFR), a nominal distance to be decided can be given.

4.2.3.1 Output summary

Throughout the process of Double Feature Capture, the following data has been stored in variables:

- An array of lines, with four elements each, being the x and y co-ordinates of the start and end of each line. This was the outcome of section 4.2.1.3 (see table 7).

Index	Start		End	
	x	y	x	y
0	27	548	1514	548
1	25	552	363	552
2	910	680	1658	693
3	1511	552	1511	51
4	742	674	1656	690
5	1514	555	1514	49
6	368	553	368	131
7	366	131	794	131
8	795	132	836	553
9	791	137	831	555
10	363	547	363	134
11	370	136	790	136
12	1218	552	1332	552
13	370	554	690	683
14	378	554	687	676
15	661	675	751	675
16	1520	560	1656	705
17	1333	552	1517	552
18	659	687	834	553
19	1515	550	1662	705
20	668	672	1272	683
21	737	676	927	680
22	364	553	835	553
23	1048	552	1162	552
24	656	685	812	568

Table 7: Lines identified in section 4.2.1.3

- An array of points. These were the corners detected in section 4.2.1.4 and are stored in a 2-column array with their co-ordinates (see table 8):

Index	x	y
0	1512	548
1	368	548
2	830	548
3	370	133
4	789	138
5	670	676
6	1648	695

Table 8: Corners identified in section 4.2.1.4

- Text boxes, as output of either section 4.2.1.1 or section 4.2.2.2: a 5-column array with x and y back-capture of the top left corner, width and height in pixels and the detected text. For this test, as the OCR results were incomplete, this table has been completed manually. The position of text boxes 15 – 20 was measured on screen with the help of the “Online pixel ruler” at www.rapidtables.com. See table 9.

Index	x	y	w	h	Output
0	322	260	31	138	328°24'15"
1	494	96	122	31	58°24'15"
2	518	145	63	28	37.01
3	374	299	29	67	30.82
4	553	511	70	28	40.08
5	767	261	41	139	142°43'30"
6	821	282	34	68	30.98
7	1152	512	78	28	168.78
8	396	562	417	65	238°24'15"
10	391	778	228	40	RIVERSDALE
11	800	778	100	40	ROAD
12	1043	644	134	41	59°37'20"
13	1052	693	66	26	179.41
14	1740	130	40	430	MAROONDAH HIGHWAY
15	430	590	80	60	232°05'
16	520	640	100	70	33.40 AD
17	710	600	70	50	86°30'
18	740	600	75	65	7.80 AD
19	1560	570	60	60	240°43'
20	1530	590	65	65	3.73 AD

Table 9: Text boxes

4.2.3.2 Line validation

As seen in 4.2.1.3, there were some false positives in the output of the line detection algorithm, in the form of duplications and line segments. These can be eliminated to simplify the array of lines by means of co-ordinated geometry.

Calculating the slope and y -intercept of each line helped discern nine distinct lines. Of those, seven were duplicate lines, consistent with the expected output of the Canny algorithm in 4.2.1.2. Comparing the back-capture of the start and end points of each of these line pairs shows that they all lie within a safe buffer, with the largest distance being 26.63 pixels.

The remaining two were the longest lines in the plan and they have been segmented several times through text elements and other lines intersecting them. These lines appear as clusters of line segments in table 10 below, which has been sorted by slope and y -intercept to help

visualise similar lines. These include overlapping lines, gaps and lines of different distance but with one coincident end point and same orientation. An indicative example is shown in figure 31. These two clusters will be referred to as A and B in the forthcoming sections. The shading in table 10 clarifies line pairs and clusters.

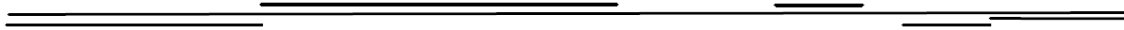


Figure 31: Segment cluster, *y* axis exaggerated for clarity

Index	slope	y-intercept	same as	segment of	Δ start	Δ end
24	-0.75	1177.00	18		3.61	26.63
18	-0.77	1191.61				
7	0.00	131.00	11		6.40	6.40
11	0.00	136.00				
0	0.00	548.00		A		
1	0.00	552.00		A		
12	0.00	552.00		A		
17	0.00	552.00		A		
23	0.00	552.00		A		
22	0.00	553.00		A		
15	0.00	675.00		B		
2	0.02	664.18		B		
4	0.02	661.01		B		
20	0.02	659.83		B		
21	0.02	660.48		B		
14	0.39	404.76	13		8.00	7.62
13	0.40	404.84				
19	1.05	-1047.45	16		11.18	6.00
16	1.07	-1060.59				
8	10.27	-8031.29				
9	10.45	-8128.95	8		6.40	5.39
10	363.00	-131222.00				
6	368.00	-134871.00	10		7.81	5.83
3	1511.00	-2282569.00				
5	1514.00	-2291641.00	3		4.24	3.61

Table 10: Line validation

4.2.3.3 Corner validation

The next check consists of comparing the corners identified in 4.2.1.4 and matching them to beginnings and ends of lines in 4.2.1.3. To achieve this, the distance between each corner and each line start or end was calculated. This step effectively validates the identified corners in 4.2.1.4 as valid line intersections that need to be captured.

Most points fall within the 30-pixel buffer of a line start or end. The exceptions are in red in table 11 below and explained as follows:

- For each of the two clusters of line segments, there are two identified corners contained in the line. This is a further check on the validation of these points. As this computation compares each line end to the nearest identified corner, most distances obtained exceed the reasonable buffer. These are the segmentation points that are not validated by other lines or the corner detection and are not points of interest.
- Line 3 has been duplicated as 5 as a result of the Canny algorithm in step 2 and as such it has successfully identified as a line. However, only one end can be matched to an identified corner. This issue will need to be handled at a later stage, as an assessment will be made about the need of unidentified corners to allocate text elements.
- Line 22 has both extremes clearly matched with identified points, however has not been duplicated. This is due to it being a segment of a longer line.

Cluster	Line Index	from		To	
		Nearest point index	Distance	Nearest point index	
	24	5	16.64	2	26.91
	18	5	15.56	2	6.40
	7	3	4.47	4	8.60
	11	3	3.00	4	2.24
A	0	1	341.00	0	2.00
A	1	1	343.02	1	6.40
A	12	0	294.03	0	180.04
A	17	0	179.04	0	6.40
A	23	0	464.02	0	350.02
A	22	1	6.40	2	7.07
B	15	5	9.06	5	81.01
B	2	5	240.03	6	10.20
B	4	5	72.03	6	9.43
B	20	5	4.47	6	376.19
B	21	5	67.00	5	257.03
	14	1	11.66	5	17.00
	13	1	6.32	5	21.19
	19	0	3.61	6	17.20
	16	0	14.42	6	12.81
	8	4	8.49	2	7.81
	9	4	2.24	2	7.07
	10	1	5.10	3	7.07
	6	1	5.00	3	2.83
	3	0	4.12	0	497.00
	5	0	7.28	0	499.00

Table 11: Corner validation. Shading clarifies duplicate lines and segment clusters.

4.2.3.4 Text boxes

The position of the text boxes can now be compared with the validated lines. As the text is oriented along the corresponding line and approximately centred, the distance between the centre point of each box and the centre point of each line can be calculated and the text can be allocated to the closest line. Details can be seen in table 12 below.

Most text boxes are allocated correctly, with four exceptions:

- Box 7 is too far (156 pixels) from the closest line, number 4, to be allocated with confidence. In reality, this text element relates to a line connecting points 0 and 2 which has not been detected. This issue will need to be handled separately and could be an opportunity for a “human intervention required” flag.

- Boxes 10, 11 and 14 are too far from any line to be reliably allocated, the shortest distance being 253 pixels. These text elements are road names and will have to be handled separately, as discussed in the following section on LandXML elements (4.2.3.5).

Each line has been allocated two text boxes, consistent with the two pieces of information required for each line: bearing and distance. This confirms the success of the process of validation between the outputs of the two pillars within the DFC.

Text Box Index	Nearest line Index	dist	from	to	text	type
0	10	27.97	1	3	328°24'15"	bearing
1	11	35.00	3	4	58°24'15"	bearing
2	11	38.20	3	4	37.01	dist
3	10	26.73	1	3	30.82	dist
4	22	30.27	1	2	40.08	dist
5	8	30.46	4	2	142°43'30"	bearing
6	8	34.76	4	2	30.98	dist
7	4	156.20	5	6	168.78	dist
8	22	41.80	1	2	238°24'15"	bearing
10	22	262.59	1	2	RIVERSDALE	road
11	18	350.39	5	2	ROAD	road
12	4	90.70	5	6	59°37'20"	bearing
13	4	116.50	5	6	179.41	dist
14	3	252.77	0		MAROONDAH HIGHWAY	road
15	13	60.02	1	5	232°05'	bearing
16	13	69.23	1	5	33.40 AD	dist
17	18	5.22	5	2	86°30'	bearing
18	18	33.43	5	2	7.80 AD	dist
19	16	32.56	0	6	240°43'	bearing
20	16	27.39	0	6	3.73 AD	dist

Table 12: Allocation of text boxes to lines

4.2.3.5 LandXML elements

The elements required for the LandXML file have been discussed in section 3.2. With the output validated as per the previous section, it is possible to allocate each object to the different elements required. As all the identified corners have been validated, they can be readily allocated <CgPoint> and <InstrumentSetup> numbers (see table 13). This constitutes step 10 of the DFC.

Pt Index		
0	CGPNT-1	IS-1
1	CGPNT-2	IS-2
2	CGPNT-3	IS-3
3	CGPNT-4	IS-4
4	CGPNT-5	IS-5
5	CGPNT-6	IS-6
6	CGPNT-7	IS-7

Table 13: Allocation of point identifiers

These points can also be joined via the validated lines to form the elements of the <ObservationGroup>. Table 14 below combines data from tables 12 and 13, with the corner indices translated into instrument setups. This constitutes step 11 of the DFC.

Line index	from	to		bearing	distance
8	IS-5	IS-3	OBS-1	142°43'30"	30.98
10	IS-2	IS-4	OBS-2	328°24'15"	30.82
11	IS-4	IS-5	OBS-3	58°24'15"	37.01
13	IS-2	IS-6	OBS-4	232°05'	33.40 AD
16	IS-1	IS-7	OBS-5	240°43'	3.73 AD
18	IS-6	IS-3	OBS-6	86°30'	7.80 AD
22	IS-2	IS-3	OBS-7	238°24'15"	40.08

Table 14: Initial allocation of observations

The remaining text elements, yet to be allocated, are shown in table 15 below and explained following:

Text Box ID	Line ID	dist	from	to	text	type
7	4	156.20	5	6	168.78	dist
10	22	262.59	1	2	RIVERSDALE	road
11	18	350.39	5	2	ROAD	road
12	4	90.70	5	6	59°37'20"	bearing
13	4	116.50	5	6	179.41	dist
14	3	252.77	0		MAROONDAH HIGHWAY	road

Table 15: Unallocated text elements

Text boxes 12 and 13 are related to line 4, which is part the clusters of line segments referred to as cluster B. All line segments within this cluster have point 5 or point 6 as either their beginning or end. As such, it can be safely concluded that all the line segments in cluster B belong to an undetected line that joins points 5 and 6. This line will be called OBS-8 and be allocated the bearing and distance contained in boxes 12 and 13.

The case of text box 7 is similar, however the line allocation is less simple. The calculation has erroneously detected line 4 as the closest, but the distance to its centre is 156 pixels. This misallocation is a result of the line connecting points 0 and 2 not being detected. This line is again part of a cluster of segments, cluster A. Line 22, connecting points 1 and 2, was already detected as part of this cluster. The corner most successfully detected within cluster A was point 0, identified 3 times within 6.4 pixels of a segment end. If points 2 and 0 are joined as a new line and its midpoint calculated, it can be seen that its distance to the centre of box 7 is only 29.7 pixels. This line can safely be called OBS-9 and allocated the text in box 7.

A bearing is still needed for this line. As discussed, previously identified line 22 was also part of cluster A and hence has the same orientation. As such, the same bearing can safely be allocated to OBS-9.

The remaining unallocated text elements are road names. Road names require a co-ordinated location in the plan for plotting purposes and must be associated with a line for orientation. In this case, and commonly in AFRs, the ends of the lines along which road names are oriented have not been measured and are shown in the diagram for reference only without a dimension. For this reason, the corner detection algorithm has not identified them as points of interest, as there are no other intersecting lines at those points. Nevertheless, their co-ordinates have been captured twice in 4.2.1.3, in both cases as extremes of duplicate lines 0/1 and 3/5. Therefore, these points can be given the names CGPOINT-8 and CGPOINT-9. The line joining CGPOINT-8 and CGPOINT-1 is the closest to text boxes 10 and 11. Note that although lines 22 and 18 were returned as the closest to text boxes 10 and 11, these lines are in a different orientation, so this information was discarded.

Similarly, the line joining CGPOINT-1 and CGPOINT-9 is the closest to text box 14, in this case correctly returned as line 3.

At this point and with only two exceptions, all text elements have been allocated to lines and every line has been allocated a bearing and a distance. There are two exceptions, being the new lines created in the immediately previous step. The line joining CGPOINT-8 and CGPOINT-1 can be safely allocated the same bearing as OBS-7 as it is part of the same segment cluster. The required distance can be calculated from the pixel co-ordinates. The line joining

CGPOINT-1 and CGPOINT-9 requires both bearing and distances, which can also be calculated from the pixel co-ordinates. Note that these are pixel distances and are only needed for visualisation purposes.

One element that has not been implemented due to time constraints is the allocation of observation purposes in the LandXML file. The different line types in the original AFR carry the information of the purpose of each observation. This is required information in the LandXML file, and the values can be “normal” (boundary line, solid line type), “sideshot” (radiation, dashed line type) or “traverse” (measurement between instrument stations, “chain” line types). As the different line types were lost in the Hough Transform, this information has not been carried over to the LandXML file. An identification of line types needs to be undertaken prior to that step, however this has not been undertaken due to time constraints. This omission has a flow on effect on the computation of the parcels. The parcel subject to survey should be the only closed parcel in the plan bounded by solid lines. Again, as all lines have been converted to solid with the Hough Transform algorithm, there are two additional possible parcels that need to be discarded.

4.3 Summary of results

This project has used ML to extract some cadastral survey information from a model AFR.

The Model AFR was created upon analysis of existing AFRs lodged in Victoria to include only the minimal information required as a starting point for the design of this method of back-capture. The Model AFR fulfils objective 1 of this project, as stated in 1.4.

The DFC method described provides a roadmap for an automation process for the retrieval of information from AFRs. The relevant information is contained in text form and is captured by means of OCR. The graphical information that supports the text can be captured by means of CV and used to check and validate the information in the text elements and to aid with allocation of bearings and distances to points. The DFC is the outcome of objective 2 (see 1.4).

A selection of algorithms informed by the literature review was trialled to undertake each of the tasks required for the back-capture. Successes and challenges were discussed in chapter 4, fulfilling objective 3 (see 1.4).

The method described shows that a degree of automation is possible, and the double filter provides checks and balances to ensure accurate capture of relevant information and to flag instances where human intervention may be required. This project has documented an initial case study providing evidence that further investigation and development in this area is warranted.

CHAPTER 5: FURTHER RESEARCH AND CONCLUSION

5.1 Further research

This study has mapped out the steps that would be required to extract point identification and dimension information from an image file for storage in a searchable LandXML file. It has also outlined one possible implementation of those steps, the Double Filter Capture, from a simple AFR created for this purpose.

This project has documented an initial case study demonstrating that some degree of automation is possible. The significant constraints and limitations to the scope of this first trial that have been outlined throughout this document could provide the first steps for further research in this area.

In the present study the DFC was only tested using a model AFR developed after analysis of a set of AFRs of great similarity. The study should be independently replicated and the DFC should undergo further validation by testing other model AFRs of similar complexity. Then research could be furthered by introducing successive levels of complexity. These could include:

- Splay corners.
- Arcs.
- Multiple pages.
- Hand writing.
- Capturing occupation shown in the AFR. The DCMP is not capturing occupation as it is too resource expensive, although the information would be very valuable if captured. Automating this part of the capture would be of a great benefit and this is a field worth exploring further.
- Chainage / offset measurements, with text perpendicular to the corresponding line. This would require investigating how to interpret different pieces of text in the same orientation but with different meaning.
- Allocation of observation purposes in the LandXML file. This is the information carried the different line styles and should be captured before the Hough Transform converts all lines to a solid line style.

The validations that link the outcomes of the DFC's two phases were undertaken manually, however the processes outlined can be a guideline for an automated checking mechanism.

A further validation that could be built in would be the detection of closed figures by means of CV and then computation of the closures with the corresponding dimensions extracted by OCR. This would validate not only the capture but also the information itself.

While investigating the links with OMR can prove fruitful, one essential difference between sheet music and the capture of cadastral information needs to be considered. Whereas in sheet music the information is laid out in a consistent pattern from left to right and with added “depth” on the y axis, each cadastral plan represents a potentially brand new layout and myriad configurations of patterns are possible. This increases the difficulty in automating pattern detection.

Further research could include expansion of this model by a surveyor to include all steps required to identify and extract all relevant survey information.

Given this project has demonstrated that a level automatic extraction is indeed possible, a further step would be refining the documented algorithms to include more accurate use of inbuilt variables, customisation of the algorithms used and programming so that the data output is automatically entered into the LandXML database. One example of this could be investigating the use of the “stable paths” method (Dos Santos Cardoso et al. 2009) for the removal of lines.

An additional direction of research may be comparing accuracy of scanning vector .pdf files compared with scanned raster images of CAD generated plans. The latter form the majority of plans existing in the cadastre. This is so despite the fact that most surveyors submit vector .pdf files which are imaged into the system. Vector .pdf files should yield a better result when processed through OCR. While comparing these processes was outside of the scope of this project, it could be the object of further research.

For implementation of this research into practical usage or a commercial product, this project has shown that these tasks may be better approached in a multidisciplinary manner by computer scientists together with surveyors.

This research demonstrates that there is value in attempting automated data extraction, however it remains unclear as to what volume of data is required for this method to become more time and resource efficient when compared with manual entry. It would be of interest for the abovementioned expansions of this project to be developed in a formal research context, also measuring resource cost (time, salaries and so on) to determine efficiency and productivity measures of developing such software compared with the current procedure in place using double entry, completed by data entry experts rather than surveyors. This may inform how similar projects are undertaken in other fields.

In the case of the Victorian cadastre, a large amount of survey information has been captured manually by the DCMP. These data could be used to train DL neural networks to learn to recognise each of the elements discussed in this project. Ultimately, this could lead to a systemic approach where newly captured plans are incorporated into training sets and the capture becomes progressively more automated.

5.2 Conclusion

This project provides a single case study, demonstrating that automated data extraction from an image file of a simple cadastral file is possible. The steps required to extract point identifications have been mapped in a pipeline workflow and successfully implemented within the limitations of using pre-existing ML algorithms by a novice computer programmer. The steps required to extract dimensions from plans have also been outlined and initial trials undertaken, which require further development.

It has also shown that the expertise of a surveyor enables the steps required for data extraction to be conceptualised, modified, and implemented. It also clarifies that greater expertise or a multidisciplinary approach involving also experts from the field of ML in computer would most likely make expansion and implementation of the project more efficient and potentially more accurate and extensive. Given the complexity and inconsistencies of existing plans requiring back-capture it also highlights that an extensive investment will be required in programming, refining and testing authentic survey plans.

It is hoped that this research project will facilitate development of tools for the automatic data extraction of survey plans and further afield by informing application of rapidly developing ML technology to other areas.

To quote Broussard (2018, pp. 176-7), “Automation will handle a lot of the mundane work; it won’t handle the edge cases (...) You need to build in human effort for the edge cases”. The capture of cadastral survey information is one such “edge case”. A “human-in-the-loop” system could be a solution where expert input is required to overcome the limitations of ML algorithms. Complete automation remains an aspirational goal, however interested organisations may be encouraged by this project to undertake further research.

REFERENCES

Alpaydin, E 2016, *Machine Learning: The New AI*, The MIT Press.

Bainbridge, D & Bell, T 2001, 'The challenge of optical music recognition', *Computers and the Humanities*, vol. 35, no. 2, pp. 95-121.

Batchelor, J 2016, 'Towards a Survey-Accurate Digital Cadastre for Victoria', RMIT University, Melbourne.

Bhowmik, S, Sarkar, R, Nasipuri, M & Doermann, D 2018, 'Text and non-text separation in offline document images: a survey', *International Journal on Document Analysis and Recognition (IJDAR)*, vol. 21, no. 1, pp. 1-20.

Broussard, M., 2018. *Artificial unintelligence: how computers misunderstand the world*. MIT Press.

Calvo-Zaragoza, J, Castellanos, FJ, Vigliensoni, G & Fujinaga, I 2018, 'Deep neural networks for document processing of music score images', *Applied Sciences*, vol. 8, no. 5, p. 654.

Canny, J 1986, 'A computational approach to edge detection', in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-8, no. 6 pp. 679-98.

Dharmana, J, Cherku, R, Prasad, K & Das, S 2012. *An Overview of Optical Character Recognition Systems Research on Telugu Language*, vol. 2, no. 9, pp. 23-9.

Dos Santos Cardoso, J, Capela, A, Rebelo, A, Guedes, C & Pinto Da Costa, J 2009, 'Staff detection with stable paths', *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, no. 6, pp. 1134-9.

Duda, RO & Hart, PE 1972, 'Use of the Hough transformation to detect lines and curves in pictures', *Communications of the ACM*, vol. 15, no. 1, pp. 11-5.

Fraser, R, Boyle, D, Heritage, J & Olfat, H, *Cadastral back capture instruction manual. Creation of LandXML files from the back-capture of registered title plans and cadastral surveys*, 2018, L Department of Environment, Water and Planning, Melbourne.

Goodfellow, I, Bengio, Y & Courville, A 2016, *Deep Learning*, The MIT Press, viewed 10 July 2019, <<http://www.deeplearningbook.org>>.

Harris, CG & Stephens, M 1988, *A combined corner and edge detector*, Plessey Research Roke Manor, UK, vol. 15, no. 50, pp. 147-51.

Heidorn, PB & Wei, Q 2008, 'Automatic metadata extraction from museum specimen labels', in *International Conference on Dublin Core and Metadata Applications*, pp. 57-68.

Ignjatić, J., Nikolić, B. and Rikalović, A., 2018. 'Deep learning for historical cadastral maps digitization: overview, challenges and potential', *Proceedings of the 26 International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision WSCG 2018, Posters Proceedings*, Plzen, Czech Republic, pp. 42-7

Intergovernmental Committee on Surveying and Mapping (ICSM) 2010, *ePlan Model v1.0*, viewed 21 March 2019 <<https://www.icsm.gov.au/sites/default/files/ICSM-ePlan-Model-v1.0%20%281%29.pdf>>.

Intergovernmental Committee on Surveying and Mapping (ICSM) 2016, *ePlan Protocol LandXML Mapping*, viewed 21 March 2019 <https://icsm.gov.au/sites/default/files/ePlan-Protocol-LandXML-Mapping-v2.1.4.pdf>.

Katona, E & Hudra, G 1999, 'An interpretation system for cadastral maps', *Proceedings of the 10th International Conference on Image Analysis and Processing*, pp. 792-7.

Kirchhoff, A, Bügel, U, Santamaria, E, Reimeier, F, Röpert, D, Tebbje, A, Güntsch, A, Chaves, F, Steinke, K-H & Berendsohn, W 2018, 'Toward a service-based workflow for automated information extraction from herbarium specimens', *Database*, vol. 2018, pp. 1-11.

Liu, X, Meng, G & Pan, C 2019, 'Scene text detection and recognition with advances in deep learning: a survey', *International Journal on Document Analysis and Recognition (IJ DAR)*, vol. 22, no. 2, pp. 143-62.

Merz, SK, *Business Case For A Spatially Accurate Map Base*, 2011, DoSa Environment, Melbourne.

Olfat, H, Shojaei, D, Briffa, M, Maley, S & Rajabifard, A 2018, 'Strategic actions for increasing the submission of digital cadastral data by the surveying industry based on lessons learned from Victoria, Australia', *ISPRS International Journal of Geo-Information*, vol. 7, no. 2, p. 47.

Ouwayed, N & Belaïd, A 2012, 'A general approach for multi-oriented text line extraction of handwritten documents', *International Journal on Document Analysis and Recognition (IJDAR)*, vol. 15, no. 4, pp. 297-314.

Padilla, V, Marsden, A, McLean, A & Ng, K 2014, 'Improving OMR for digital music libraries with multiple recognisers and multiple sources', in *Proceedings of the 1st International Workshop on Digital Libraries for Musicology*, ACM, London, United Kingdom, pp. 1-8.

Petrescu, R, Manolache, S, Boiangiu, CA, Vlăsceanu, GV, Avatavului, C, Prodan, M & Bucur, I 2019, 'Combining tesseract and aprise results to improve OCR text detection accuracy', *Journal of Information Systems & Operations Management*, vol. 13, viewed 7 October 2019 <https://www.researchgate.net/profile/Costin-Anton_Boiangiu/publication/333968802_COMBINING_TESSERACT_AND_ASPRISE_RESULTS_TO_IMPROVE_OCR_TEXT_DETECTION_ACCURACY/links/5d0fe230a6fdcc2462a02724/COMBINING-TESSERACT-AND-ASPRISE-RESULTS-TO-IMPROVE-OCR-TEXT-DETECTION-ACCURACY.pdf>.

Roy, PP, Pal, U & Lladós, J 2012, 'Text line extraction in graphical documents using background and foreground information', *International Journal on Document Analysis and Recognition (IJDAR)*, vol. 15, no. 3, pp. 227-41.

Rowe, G 2003, 'The survey conversion project—making a survey-accurate digital cadastre for New Zealand a reality', *New Zealand Surveyor*, vol. 293, pp. 31-8.

Russell, SJ & Norvig, P 2003, *Artificial Intelligence: A Modern Approach*, Pearson Education.

Shi, J & Tomasi, C 1994, 'Good features to track', *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pp. 593-600.

Smith, R 2019, *Tesseract Manual*, viewed 31 August 2019, <<https://github.com/tesseract-ocr/tesseract/blob/master/doc/tesseract.1.asc>>.

Victoria, SBo 1997, *Survey Practice Handbook Victoria*, Melbourne.

Voulodimos, A, Doulamis, N, Doulamis, A & Protopapadakis, E 2018, 'Deep Learning for Computer Vision: A Brief Review', *Computational Intelligence and Neuroscience*, vol. 2018.

Wei, TC 2018, 'Improved Optical Character Recognition With Deep Learning'.

Westphal, F, Grahn, H & Lavesson, N 2018, 'Efficient document image binarization using heterogeneous computing and parameter tuning', *International Journal on Document Analysis and Recognition (IJDAR)*, vol. 21, no. 1, pp. 41-58.

Zhang, A, Lipton, ZC, Li, M & Smola, AJ 2019, *Dive into Deep Learning*.

APPENDIX A – PROJECT SPECIFICATION

University of Southern Queensland
FACULTY OF ENGINEERING AND SURVEYING

ENG4111/4112 Research Project **PROJECT SPECIFICATION**

FOR: **Oscar GARRIDO DE LA ROSA**

TOPIC: Conversion of cadastral survey information into LandXML files using Machine Learning

SUPERVISOR: Dr Glenn Campbell

PROJECT AIM: To investigate and assess the suitability of Machine Learning in the automation process of the back capture of cadastral survey plans for their conversion into LandXML files.

PROGRAMME: Revision B, 7 October 2019

1. Perform literature review covering the following areas:
 - a. Benefits of digitising survey information
 - b. LandXML and ePlan
 - c. OCR, Machine Learning and Neural Networks
2. Conduct research to determine available open source Machine Learning repositories.
3. Review survey information from the Victorian cadastre, identify relevant elements suitable to automation and draft model plan for testing.
4. Determine required steps to retrieve survey information through Machine Learning algorithms.
5. Test algorithms, and document results and barriers encountered.
6. Discuss validation methods.
7. Write dissertation.

APPENDIX B – LANDXML CODE

```
<?xml version="1.0" encoding="UTF-8"?>
<LandXML xmlns="http://www.landxml.org/schema/LandXML-1.2"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.landxml.org/schema/LandXML-1.2
http://www.landxml.org/schema/LandXML-1.2/LandXML-1.2.xsd" version="1.0"
date="2019-03-23" time="00:00:00">
  <Units>
    <Metric linearUnit="meter" temperatureUnit="celsius"
volumeUnit="cubicMeter" areaUnit="squareMeter" pressureUnit="milliBars"
angularUnit="decimal dd.mm.ss" directionUnit="decimal dd.mm.ss"/>
  </Units>
  <CoordinateSystem datum="Local" horizontalDatum="Local"/>
  <Application name="Software name" version="1.0"/>
  <FeatureDictionary name="xml-gov-au-vic-icsm-eplan-cif-protocol"
version="1.8"/>
  <Parcels>
    <Parcel name="1\LP999999" class="Lot" state="existing"
parcelType="Single" parcelFormat="Standard" area="1136">
      <Center pntRef="CGPNT-8"/>
      <CoordGeom name="CG-8">
        <Line>
          <Start pntRef="CGPNT-1"/>
          <End pntRef="CGPNT-2"/>
        </Line>
        <Line>
          <Start pntRef="CGPNT-2"/>
          <End pntRef="CGPNT-4"/>
        </Line>
        <Line>
          <Start pntRef="CGPNT-4"/>
          <End pntRef="CGPNT-3"/>
        </Line>
        <Line>
          <Start pntRef="CGPNT-3"/>
          <End pntRef="CGPNT-1"/>
        </Line>
      </CoordGeom>
    </Parcel>
    <Parcel name="ROAD-1" class="Road" state="existing"
parcelType="Single" parcelFormat="Standard" desc="RIVERSDALE ROAD">
      <Center pntRef="CGPNT-9"/>
      <CoordGeom name="CG-9">
        <Line>
          <Start pntRef="CGPNT-3"/>
          <End pntRef="CGPNT-5"/>
        </Line>
      </CoordGeom>
    </Parcel>
    <Parcel name="ROAD-2" class="Road" state="existing"
parcelType="Single" parcelFormat="Standard" desc="MAROONDAH HIGHWAY">
      <Center pntRef="CGPNT-10"/>
      <CoordGeom name="CG-10">
        <Line>
          <Start pntRef="CGPNT-5"/>
          <End pntRef="CGPNT-11"/>
        </Line>
      </CoordGeom>
    </Parcel>
  </Parcels>
</LandXML>
```

```

        </CoordGeom>
    </Parcel>
</Parcels>
<CgPoints>
    <CgPoint state="existing" pntSurv="boundary" name="CGPNT-
1">5000.000 1000.000</CgPoint>
    <CgPoint state="existing" pntSurv="boundary" name="CGPNT-
2">5019.390 1031.524</CgPoint>
    <CgPoint state="existing" pntSurv="boundary" name="CGPNT-
3">4973.740 1016.148</CgPoint>
    <CgPoint state="existing" pntSurv="boundary" name="CGPNT-
4">5019.390 1031.524</CgPoint>
    <CgPoint state="existing" pntSurv="boundary" name="CGPNT-
5">5083.166 1194.047</CgPoint>
    <CgPoint state="existing" pntSurv="traverse" name="CGPNT-
6">4994.263 1042.522</CgPoint>
    <CgPoint state="existing" pntSurv="traverse" name="CGPNT-
7">5084.991 1197.301</CgPoint>
    <CgPoint state="existing" pntSurv="reference" name="CGPNT-
8">4996.965 1024.490</CgPoint>
    <CgPoint state="existing" pntSurv="reference" name="CGPNT-
9">4967.170 1043.717</CgPoint>
    <CgPoint state="existing" pntSurv="reference" name="CGPNT-
10">5104.366 1177.900</CgPoint>
</CgPoints>
<Survey>
    <InstrumentSetup id="IS-1" stationName="IS-1"
instrumentHeight="0">
        <InstrumentPoint pntRef="CGPNT-1"/>
    </InstrumentSetup>
    <InstrumentSetup id="IS-2" stationName="IS-2"
instrumentHeight="0">
        <InstrumentPoint pntRef="CGPNT-2"/>
    </InstrumentSetup>

    <InstrumentSetup id="IS-3" stationName="IS-4"
instrumentHeight="0">
        <InstrumentPoint pntRef="CGPNT-4"/>
    </InstrumentSetup>
    <InstrumentSetup id="IS-4" stationName="IS-3"
instrumentHeight="0">
        <InstrumentPoint pntRef="CGPNT-3"/>
    </InstrumentSetup>
    <InstrumentSetup id="IS-5" stationName="IS-5"
instrumentHeight="0">
        <InstrumentPoint pntRef="CGPNT-5"/>
    </InstrumentSetup>
    <InstrumentSetup id="IS-6" stationName="IS-7"
instrumentHeight="0">
        <InstrumentPoint pntRef="CGPNT-7"/>
    </InstrumentSetup>

    <InstrumentSetup id="IS-7" stationName="IS-6"
instrumentHeight="0">
        <InstrumentPoint pntRef="CGPNT-6"/>
    </InstrumentSetup>
    <ObservationGroup id="OG-1">
        <ReducedObservation name="OBS-1" purpose="normal"
setupID="IS-1" targetSetupID="IS-2" azimuth="58.2415"
horizDistance="37.01"/>
        <ReducedObservation name="OBS-2" purpose="normal"

```

```

setupID="IS-2"          targetSetupID="IS-4"          azimuth="142.4330"
horizDistance="30.98"/>
    <ReducedObservation    name="OBS-3"    purpose="normal"
setupID="IS-4"          targetSetupID="IS-3"          azimuth="238.2415"
horizDistance="40.08"/>
    <ReducedObservation    name="OBS-4"    purpose="normal"
setupID="IS-3"          targetSetupID="IS-1"          azimuth="328.2415"
horizDistance="30.82"/>
    <ReducedObservation    name="OBS-5"    purpose="sideshot"
setupID="IS-6" targetSetupID="IS-3" azimuth="232.05" horizDistance="33.4"/>
    <ReducedObservation    name="OBS-6"    purpose="sideshot"
setupID="IS-6" targetSetupID="IS-4" azimuth="86.30" horizDistance="7.8"/>
    <ReducedObservation    name="OBS-7"    purpose="traverse"
setupID="IS-6"          targetSetupID="IS-7"          azimuth="59.3720"
horizDistance="179.41"/>
    <ReducedObservation    name="OBS-8"    purpose="sideshot"
setupID="IS-7" targetSetupID="IS-5" azimuth="240.43" horizDistance="3.73"/>
    <ReducedObservation    name="OBS-9"    purpose="normal"
setupID="IS-5" targetSetupID="IS-4" azimuth="238.2415" horizDistance=
    "168.78"/>
    </ObservationGroup>
</Survey>
<Monuments>
    <Monument name="MON-1" pntRef="CGPNT-6" type="Rivet" state="New"
condition="Placed"/>
    <Monument name="MON-2" pntRef="CGPNT-7" type="Permanent Mark"
state="Existing" condition="OK"/>
</Monuments>
</LandXML>

```

APPENDIX C – PYTHON CODE

1. Removing text

```
import numpy as np
import cv2
from matplotlib import pyplot as plt
from tesseract import PyTessBaseAPI, RIL

cv_img = cv2.imread('AFR_model_1.bmp')

with PyTessBaseAPI() as api:
    api.SetImageFile('AFR_model_1.bmp')
    boxes = api.GetComponentImages(RIL.PARA, True)
    print('Found {} textline image components.'.format(len(boxes)))
    for i, (im, box, _, _) in enumerate(boxes):
        # im is a PIL image object
        # box is a dict with x, y, w and h keys
        api.SetRectangle(box['x'], box['y'], box['w'], box['h'])
        ocrResult = api.GetUTF8Text()
        conf = api.MeanTextConf()
        print(u"Box[{0}]: x={x}, y={y}, w={w}, h={h}, "
              "confidence: {1}, text: {2}".format(i, conf, ocrResult,
**box))

        ## delete boxes
    for (im,box,_,_) in boxes:
        x,y,w,h = box['x'],box['y'],box['w'],box['h']
        cv_img[y:y+h,x:x+w] = 255
        print(x,y,w,h)
```

2. Simplify lines

Thresholding

```
img = cv.imread(image,0)
th, dst = cv.threshold( img, 128, 255, cv.THRESH_BINARY_INV)
print(dst)
print(dst.shape)
#print(dst[500,497])
#pic = img
pic = np.zeros(dst.shape) #creates blank window
of same shape
#edges = cv.Canny(dst,50,150,apertureSize = 3) #applies Canny to find
edges
```

Canny

```
img = cv.imread(image,0)
print(img)
print(img.shape)
print(img[500,495])

pic = np.zeros(img.shape) #creates blank window
of same shape
edges = cv.Canny(img,50,150,apertureSize = 3) #applies Canny to find
edges
cv.imwrite('Canny.jpg',edges)
print('255 count = '+str(((edges == 255).sum()))))
print('0 count = '+str(((edges == 0).sum()))))
```

3. Detect lines

```
lines =  
cv.HoughLinesP(edges,2,0.5*np.pi/180,100,minLineLength=50,maxLineGap  
=50) #HoughLinesP to convert broken lines to solid  
  
#draw hough lines on blank window  
for line in lines:  
    x1,y1,x2,y2 = line[0]  
    cv.line(pic, (x1,y1), (x2,y2), (255,255,255),1)  
cv.imwrite('houghlines.jpg',pic)
```

4. Detect corners

```
corners = cv.goodFeaturesToTrack(img,25,0.4,85)  
corners = np.int0(corners)  
pic = np.zeros(img.shape)  
  
for i in corners:  
    x,y = i.ravel()  
    cv.circle(img, (x,y),10,255,2)  
cv.imwrite('corners.jpg',img)
```

5. Image rotation

```
img = cv2.imread('AFR_model_1.bmp',0)  
th, img = cv2.threshold( img, 128, 255, cv2.THRESH_BINARY)  
  
##scale picture for viewing  
scale_percent = 50 # percent of original size  
width = int(img.shape[1] * scale_percent / 100)  
height = int(img.shape[0] * scale_percent / 100)  
dim = (width, height)
```

```

## resize image
img = cv2.resize(img, dim, interpolation = cv2.INTER_AREA)

rows,cols = img.shape
print('Original image :' , img.shape)
centre = (rows/2 , cols/2)
angle = 45
scale = 1

#cv2.namedWindow('image',cv2.WINDOW_NORMAL)
cv2.imwrite('image.bmp',img)
cv2.imshow('image',img)
cv2.waitKey(0)
cv2.destroyAllWindows()

rot_rows = np.int(np.around(np.sqrt(rows**2 + cols**2))+1)
rot_cols = rot_rows
rot_centre = (rot_rows/2 , rot_cols/2)

#translation (resize window to target, shift centre to centre
of target)
Xshift = rot_centre[1] - centre[1]
Yshift = rot_centre[0] - centre[0]
M = np.float32([[1,0,Xshift],[0,1,Yshift]])
dst = cv2.warpAffine(img, M , (rot_cols,rot_rows))

#removing black frame around img
x_lim = 1 + rot_centre[0] - cols/2
y_lim = 1 + rot_centre[1] - rows/2

dst[:int(y_lim)] = [255]
dst[-int(y_lim):] = [255]
dst[:, :int(x_lim)] = [255]

```

```

dst[:, -int(x_lim):] = [255]

#cv2.namedWindow('image', cv2.WINDOW_NORMAL)
cv2.imwrite('translated.bmp', dst)
cv2.imshow('translated', dst)
cv2.waitKey(0)
cv2.destroyAllWindows()

#rotation
M = cv2.getRotationMatrix2D(rot_centre, angle, scale)
dst = cv2.warpAffine(dst, M, (rot_cols, rot_rows))
print('Rotated : ', dst.shape)

#cv2.namedWindow('image', cv2.WINDOW_NORMAL)
cv2.imwrite('rotated.bmp', dst)
cv2.imshow('rotated', dst)
cv2.waitKey(0)
cv2.destroyAllWindows()

```

6. Text detection

```

import numpy as np
import cv2
from matplotlib import pyplot as plt
from tesseract import PyTessBaseAPI, RIL

cv_img = cv2.imread('AFR_model_1.bmp')

with PyTessBaseAPI() as api:
    api.SetImageFile('AFR_model_1.bmp')
    boxes = api.GetComponentImages(RIL.PARA, True)
    print('Found {} textline image components.'.format(len(boxes)))
    for i, (im, box, _, _) in enumerate(boxes):

```



```

# im is a PIL image object
# box is a dict with x, y, w and h keys
api.SetRectangle(box['x'], box['y'], box['w'], box['h'])
ocrResult = api.GetUTF8Text()
conf = api.MeanTextConf()
print(u"Box[{0}]: x={x}, y={y}, w={w}, h={h}, "
      "confidence: {1}, text: {2}".format(i, conf, ocrResult,
**box))

## draw boxes
for (im,box,_,_) in boxes:
    x,y,w,h = box['x'],box['y'],box['w'],box['h']
    cv2.rectangle(cv_img, (x,y), (x+w,y+h), color=(0,0,255))
    print(x,y,w,h)

```

7. Text recognition

```

import numpy as np
import cv2
import tesseract
from matplotlib import pyplot as plt
from tesseract import PyTessBaseAPI, RIL, PSM

with PyTessBaseAPI() as api:

    api.SetImageFile('AFR_model_1.bmp')
    print(api.GetUTF8Text())

```