FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Computer Vision System for Alphanumeric Code and Symbol Recognition in Welding Blueprints

Francisco Tomé Macedo Martins Santos Moreira

Master in Informatics and Computing Engineering

Supervisor: Luís Filipe Teixeira

Second Supervisor: Valter Joaquim Ramos Costa

July 26, 2021

Computer Vision System for Alphanumeric Code and Symbol Recognition in Welding Blueprints

Francisco Tomé Macedo Martins Santos Moreira

Master in Informatics and Computing Engineering

July 26, 2021

Abstract

The construction of large metallic structures in the industrial context requires complex and multi-layered welding blueprints detailing the various manufacturing stages and control of their production. These blueprints, with very large dimensions, are printed to allow welding operators to keep track of progress on the structure and extract information about individual welds and components of the structure.

The field of Computer Vision has been a prime source of innovation and problem solving for industrial automation, both in physical and cognitive automation. The detection and recognition of text and objects have long been a major field of study in Computer Vision, from the inception of this scientific field itself to being at the centre of the Deep Learning revolution.

The goal of this thesis is to help welding operators in their activity, automating the process of reading, searching and interpreting welding blueprints. Also removing the need to print the blueprints, with all the advantages in organization of using them in a digital format, more easily keeping track of progress and making observations. To reach this goal, Computer Vision techniques are used for the automatic detection and recognition of alphanumeric codes and welding symbols.

For Alphanumeric Code detection and recognition, the proposed solution makes use of state-of-the-art text detection and text recognition pre-trained models with CRAFT and Tesseract. Welding Symbol detection is approached using a Robust Feature Matcher, applying several enhancements to obtain finer matches in the generally feature-lacking shape of welding symbols.

A dataset of 2893 alphanumeric codes and 1173 welding symbols was prepared and its characteristics, difficulties and limitations are thoroughly analysed. On this dataset, the proposed solutions obtained an accuracy of 94.50% in alphanumeric code detection and recognition and 78.09% in symbol detection.

Additionally, a proof of concept application was developed, making use of the alphanumeric code detection and recognition solution, to help the welding operator navigate between blueprints and highlight the codes that appear in multiple blueprints.

Keywords: Computer Vision, Text Detection, Text Recognition, Object Detection, Object Recognition, Alphanumeric Code, Symbol, Welding

Resumo

A construção de grandes estruturas metálicas no contexto industrial requer plantas de soldadura complexas e multicamadas, detalhando as várias fases de fabrico e controlo da sua produção. Estas plantas, com dimensões muito grandes, são impressas para permitir aos operadores de soldadura acompanhar o progresso da estrutura e extrair informações sobre soldaduras individuais e os componentes da estrutura.

O campo de *Computer Vision* tem sido uma fonte de inovação e de resolução de problemas para a automação industrial, tanto na automação física como cognitiva. A deteção e reconhecimento de texto e objetos tem sido, desde o início deste campo científico até estar no centro da revolução *Deep Learning*, um dos principais campos de estudo de *Computer Vision*.

O objetivo desta tese é ajudar os operadores de soldadura na sua atividade, automatizando o processo de leitura, pesquisa e interpretação de folhas de soldadura. Eliminando também a necessidade de imprimir as folhas, com todas as vantagens na organização da sua utilização em formato digital, mais facilmente acompanhando o seu progresso e observações. Para atingir este objetivo, são utilizadas técnicas da área de *Computer Vision* para a deteção e reconhecimento automático de códigos alfanuméricos e símbolos de soldadura.

Para a deteção e reconhecimento de códigos alfanuméricos, a solução proposta usa modelos *Deep Learning* pré-treinados de deteção e reconhecimento de texto com *CRAFT* e *Tesseract*. A deteção de símbolos é abordada através de um *Robust Feature Matcher*, aplicando vários aperfeiçoamentos para obter correspondências mais precisas nos desenhos, geralmente com poucas caraterísticas, dos símbolos de soldadura.

Foi preparado um *dataset* de 2893 códigos alfanuméricos e 1173 símbolos de soldadura e as suas caraterísticas, dificuldades e limitações são cuidadosamente analisadas. Neste conjunto de dados, as soluções propostas obtiveram uma precisão de 94,50% na deteção e reconhecimento de códigos alfanuméricos e de 78,09% na deteção de símbolos.

Adicionalmente, foi desenvolvida uma aplicação prova de conceito, fazendo uso da solução desenvolvida para deteção e reconhecimento de códigos alfanuméricos, para navegar entre plantas e destacar os códigos que aparecem em múltiplas plantas.

Palavras-chave: Visão por Computador, Deteção de Texto, Reconhecimento de Texto, Deteção de Objetos, Reconhecimento de Objetos, Código Alfanumérico, Símbolo, Soldadura

Acknowledgements

My most profound thank you to everyone that accompanied me in this stage of my life: my parents, my brother and the rest of my family, for all the support they gave me.

To my supervisor Doctor Luís Filipe Teixeira and co-supervisor Doctor Valter Costa for their guidance, suggestions and corrections that greatly contributed to this dissertation.

A thank you to all my friends.

Francisco Tomé Moreira

This dissertation was developed under the project with the reference POCI-01-0247-FEDER-068492– AARM 4.0: Aços de Alta Resistência na Metalomecânica 4.0, cofinanced by Programa Operacional Competitividade e Internacionalização (COMPETE 2020), through Fundo Europeu de Desenvolvimento Regional (FEDER).

Cofinanciado por:

COMPETE 2020

PORTUGAL 2020

UNIÃO EUROPEIA
Fundo Europeu
de Desenvolvimento Regional

"In preparing for battle, I have always found that plans are useless, but planning is indispensable."

Dwight D. Eisenhower

Contents

List of Figures

List of Tables

Abbreviations

AGAST	Adaptive and Generic Accelerated Segment Test
AKAZE	Accelerated KAZE
BBS	Best-Buddies Similarity
BRIEF	Binary Robust Independent Elementary Features
CCA	Connected Components Analysis
CCL	Connected Components Labelling
CNN	Convolutional Neural Network
CRAFT	Character Region Awareness for Text Detection
CRNN	Convolutional Recurrent Neural Network
CTC	Connectionist Temporal Classification
CTPN	Connectionist Text Proposal Network
DDIS	Deformable Diversity Similarity
EAST	Efficient and Accurate Scene Text
FAST	Features from Accelerated Segment Test
FCN	Fully Convolutional Network
FPN	Feature Pyramid Network
HOG	Histogram of Oriented Gradients
LSTM	Long Short Term Memory
OCR	Optical Character Recognition
ORB	Oriented FAST and Rotated BRIEF
PDF	Portable Document Format
PSM	Page Segmentation Methods
PNG	Portable Network Graphics
R-CNN	Region-Based Convolutional Neural Network
RAM	Random Access Memory
RANSAC	RANdom SAmple Consensus
RNN	Recurrent Neural Network
SIFT	Scale-Invariant Feature Transform
SSD	Single Shot MultiBox Detector
SURF	Speeded-Up Robust Features
SVM	Support-Vector Machines
SWT	Stroke Width Transform
YOLO	You Only Look Once

Chapter 1

Introduction

Every industry continuously aims to improve its efficiency and productivity. For many decades, increasing automation has been the leading path to this objective. Automation makes human tasks easier and more efficient and reduces their potential dangers. The manufacturing industry makes extensive use of automation and has done so since its inception.

Welding is a centuries-old fabrication process that closely accompanied the second Industrial Revolution. While welding has suffered many technological improvements over the decades, including an ever-increasing degree of automation, it still requires operators to be highly trained and have a complex understanding of its techniques and the symbols used in welding blueprints.

While automation has classically been focused on physical tasks, there is an ever-increasing use of automation to perform cognitive tasks. The field of Computer Vision has been a great contributor to this aspect, allowing computers to help humans with visual cognitive tasks.

1.1 Motivation

During the welding process of large structures, detailed blueprints of the procedure are used to control and accompany the production status. These blueprints contain various Welding Symbols and Alphanumeric References. The process of interpreting the symbols and codes used and structuring the blueprint is slow and error-prone. Figure 1.1 shows an example of a region in a blueprint with a mix of Welding Symbols and Alphanumeric References.

Welders are required to print the welding blueprints in the largest paper format possible to manually keep track of completed welding sections and search for each section Alphanumeric References.

Computer Vision techniques can be used to automatically detect and recognize the Alphanumeric References and Welding Symbols used. This will allow welders to no longer require printing the blueprints, instead keeping track of their progress digitally, more easily search for Alphanumeric References between different blueprints and have an auxiliary to the Welding Symbol interpretation.

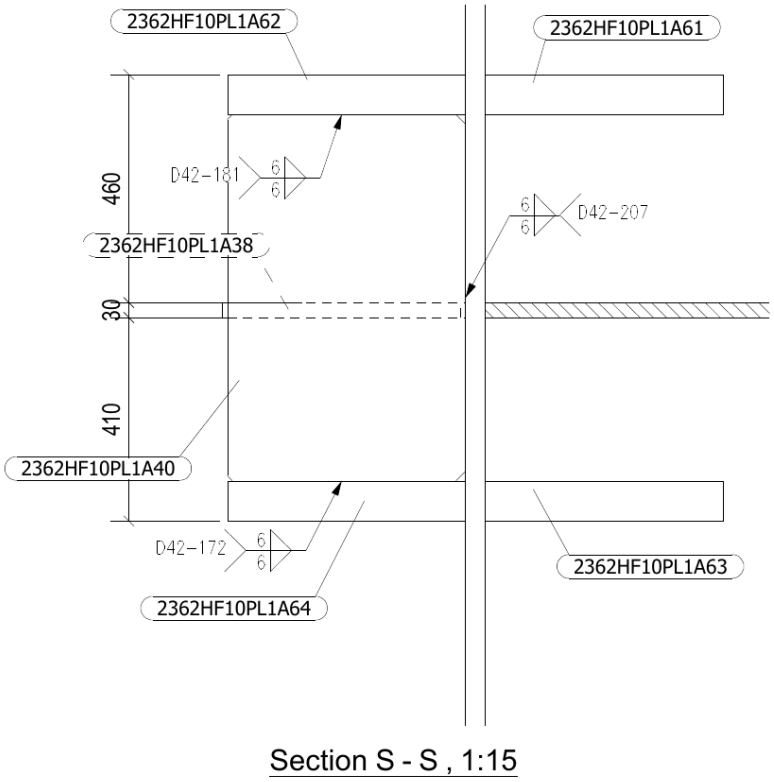


Figure 1.1: Example of a welding blueprint region with 3 welding symbols and 5 alphanumeric references.

1.1.1 Alphanumeric References

Alphanumeric References contain information about the material and structure used in a section. It is important that operators correctly identify the material and structure utilized, or the referred section, at the risk of defects being in the final product. The problem to tackle is then the automatic detection of Alphanumeric References and their textual recognition. This consists of localizing an Alphanumeric Reference in the blueprint - text detection - and recognising the alphanumeric code contained in it - text recognition. Figure 1.2 shows an example of an Alphanumeric Reference pointing to a segment of a blueprint.

Figure 1.2: Example of a welding blueprint Alphanumeric Reference.

1.1.2 Welding Symbols

Welding Symbols contain information about the weld to be applied at a specific location, typically a junction of elements. These Welding Symbols provide a large array of information including the type of the weld, the relative position of the weld, the length of the weld, the continuity of the weld, etc. It is very important that operators properly extract the information about the weld to perform it correctly. Different types of welds also require a different level of expertise from operators, meaning that a welding operator may not be allowed to perform all kinds of welds present in a blueprint. The problem to tackle is then the automatic detection and recognition of these Welding Symbols. Figure 1.3 shows an example of a Welding Symbol pointing to a segment of a blueprint.

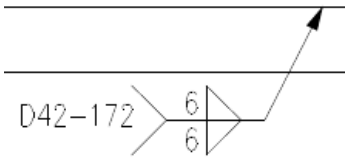


Figure 1.3: Example of a Welding Symbol.

1.2 Objectives

The objective of this thesis is to help welding operators in the reading, searching and interpretation of welding blueprints, while simultaneously removing the need to print the welding blueprints, bringing the organizational advantages of keeping the blueprints digital, more easily keeping track of the state of the production and recording observations.

Techniques in the field of Computer Vision for the automatic detection and recognition of text and objects are studied and assembled in a proposed solution that performs these objectives on digital blueprints.

Additionally, a proof of concept application is presented, making use of the Alphanumeric Reference detection and recognition solution, to navigate between blueprints and highlight the Alphanumeric References that appear in multiple blueprints.

1.3 Thesis Outline

This document is comprised of six chapters, including the introduction. A literature review is presented in Chapter two, analysing various techniques in traditional computer vision and deep learning that tackle the identified problems. The dataset is thoroughly analysed and quantified in Chapter three in preparation for the development of the proposed solutions which are enunciated in Chapter four with a detailed analysis of their steps and iterative development. The results obtained by the proposed solutions on the dataset are examined in Chapter five, with Chapter six doing a final discussion of the solutions and results, and making suggestions for future work in both problems.

Chapter 2

State of the Art

This chapter explores techniques used for Text Detection, Text Recognition and Object Detection as well as some existing applications of Computer Vision in industrial processes, especially in welding. Image pre-processing techniques will also be explored which will facilitate the rest of the process. Traditional and Deep Learning techniques for these problems were explored and are presented separately. While traditional approaches rely on the manual crafting of a features model able to perform in a multitude of scenarios, Deep Learning approaches make use of automatic learning of the features model.

2.1 Computer Vision in Welding

Welding operations already count with the assistance of Computer Vision for many tasks. Computer Vision helps operators more easily program welding robots by automatically identifying weld seams [1]. Systems have been created to automatically analyse weld bead geometry to help operators identify discontinuities in the weld [2]. Vision systems help welding robots be more adapted to adverse situations in the middle of a weld [3]. Image processing techniques can be used to detect and locate defects in the weld [4].

2.2 Image Pre-Processing

Before applying Text Detection, Text Recognition and Object Detection techniques in images, some pre-processing steps may be required to facilitate the core functioning of the system. These steps aim to enhance the features of the image and facilitate the separation of different parts of the image. When dealing with real scene images this step must also deal with problems that arise from image capture such as brightness, contrast, distortion and blur.

2.2.1 Binarization

Binarization involves transforming an image into a pure black-and-white image. It is used to better separate and distinguish objects from the background in real scene images and properly

separate the white background from the black text and symbols in digital documents which often use colour blending. The process works by finding a threshold value in the image histogram that properly divides the image into two parts. The decision for this threshold value varies between techniques, some utilize statistical information such as the mean, median or entropy while others analyse the shape of the histogram. The choice of the algorithm ultimately depends on the image characteristics, one that works for a type of image may not work for others [5]. Figure 2.1 shows an image binarization using Otsu's method. This method automatically calculates a threshold value by minimizing intra-class intensity variance [6].

Figure 2.1: Image Binarization [7].

2.2.2 Noise Elements

Welding blueprints, like most engineering drawings, may contain shapes, lines and whole regions that can conflict with the Text Detection, Text Recognition and Object Detection techniques. It is desirable to identify these objects and remove them from the blueprint. The techniques to achieve this depends on the specific requirements of the image. In the case of technical drawings, it may be desirable to remove borders, regions or certain shapes known to be irrelevant to the information desired to extract and that can potentially disturb extraction techniques.

2.2.2.1 Hough Transform

The packed space of blueprints often contains concurrent lines representing different information. It can be desirable to identify these lines with the purpose of removing or processing them. The Hough Transform is a feature extraction technique to generically detect imperfect instances of objects using a selection procedure. The classical technique centred on the identification of lines in images, referred to as Hough Line Transform, later advancements extended the Hough Transform for arbitrary shapes [8]. Figure 2.2 shows the Hough Transform for an input image with two lines. The two brightest spots correspond to the two input lines, with them it is possible to determine those lines angle and distance from the centre in the input image. Knowing the length and angle of these lines it is possible to discard the ones which are deemed irrelevant, either by their length or angle.

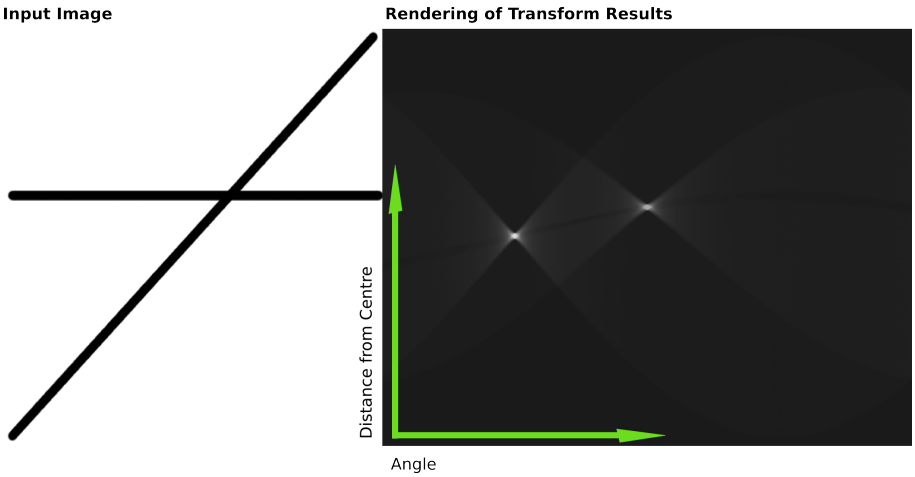


Figure 2.2: Hough Transform [9].

2.2.2.2 Connected Components Analysis

The Hough Transform can be used to detect and remove problematic lines in welding blueprints, however, it is not suitable to remove compact elements such as characters or symbols. For this, Connected Components Analysis (CCA) can be used [10]. This technique groups together pixels, typically in white over a binarized image, which are connected. This connectivity can be considered in four directions, considering only the pixel edge neighbours, or in all eight directions depending on the application. After the connected components groups are formed, several metrics can be calculated for them, such as its area or the bounding box produced by it. This can be used to remove small elements, by their area, or look for specific shapes, analysing the distribution of the group. Figure 2.3 illustrates the two groups, in red and in blue, formed by CCA using eight direction connectivity on a sample binary image.

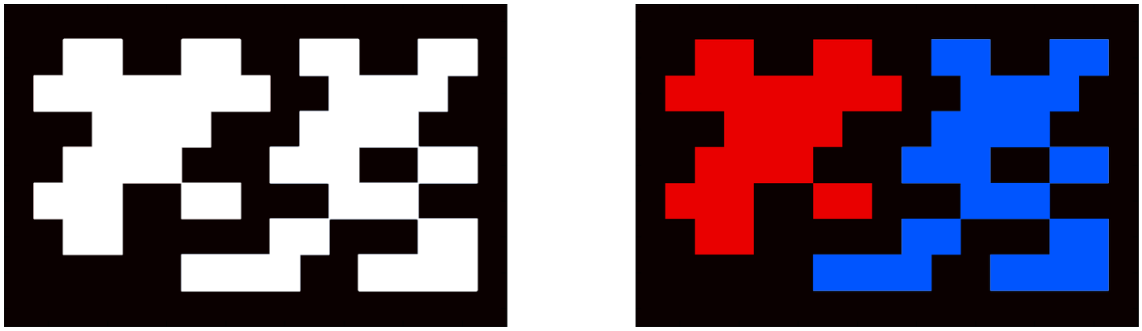


Figure 2.3: Connected Components Analysis [11].

2.2.3 Enhancing Features

After the image Binarization, some of the desired features of the image may not be sufficiently pronounced or have become noisy due to an imperfect Binarization. Morphology transformations are simple operations that can be applied to binary images based on the shapes found. The two

basic morphological operators are Erosion and Dilation [12]. These operate using a kernel of a defined size which goes around shapes found in the image and either removes the shape pixels inside the kernel, in the case of Erosion, or adds them to the shape, in case of Dilation. Using combinations of these two basic operators, more complex operations can be achieved. For example, the Closing transformation consists of an Erosion after a Dilation and is useful to deal with noise left inside shapes after the Binarization, leaving the overall shape untouched but filling the holes found inside. Figures 2.4 and 2.5 show an example of an Erosion and a Dilation applied to a white square using a square kernel.

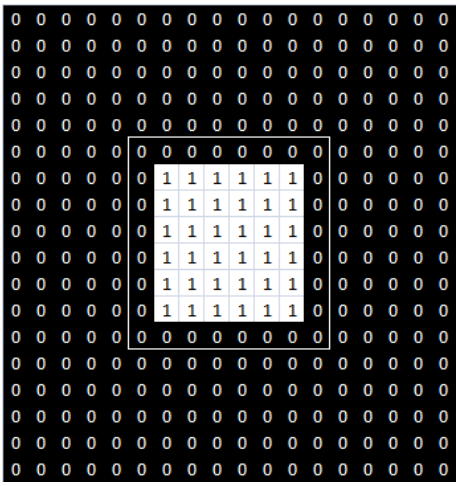


Figure 2.4: Erosion [13].

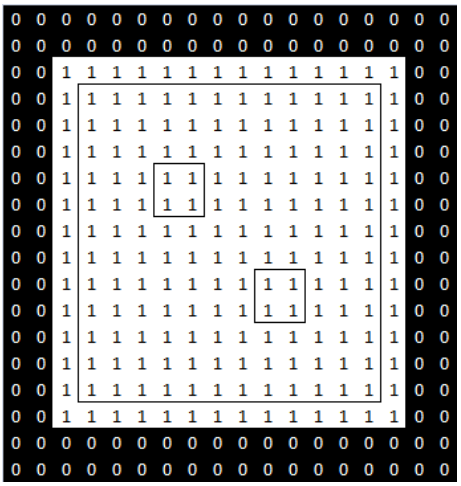


Figure 2.5: Dilation [14].

2.3 Text Detection and Recognition

Being a complex problem with many real-world applications, text detection and recognition has been the target of continuous research over many years. Many of these studies deal with natural scene text detection which brings difficult problems. These involve dealing with the diversity of the text present in a scene which can have varying size, colour, font and orientation, dealing with background noise in the scene which can resemble text and dealing with the natural problems coming from real scene capture such as blur, distortion, varying illumination and partial occlusion. Text detection and recognition in digital documents and images does not have to deal with the problems involved in image capture, however, many of the previously mentioned text-related problems are still relevant. While Text Detection and Text Recognition are more often treated as individual problems, some approaches analyse both problems at the same time as a complete solution [15]. Figure 2.6 illustrates the difference between end-to-end text detection and recognition and solutions that separate detection and recognition.

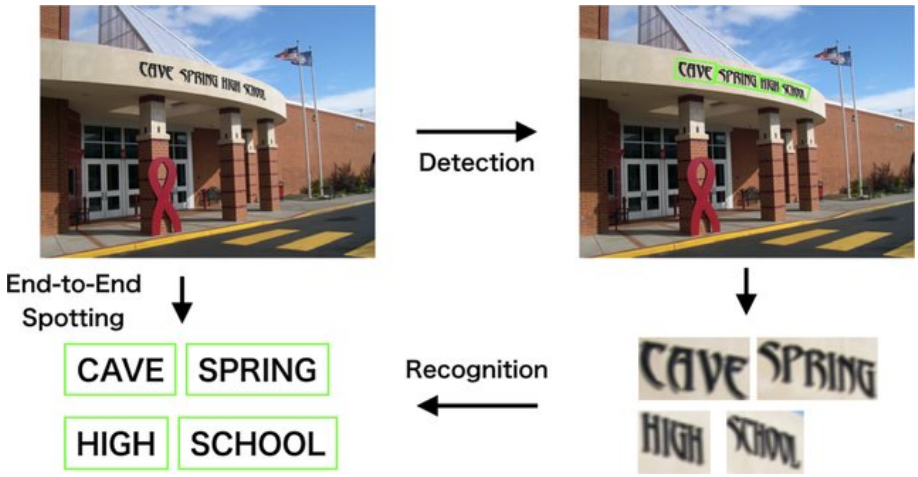


Figure 2.6: Text Detection and Recognition [16].

2.3.1 Text Detection

Text detection is a prerequisite for text recognition and plays a key role in the process of textual information extraction. While text detection can be considered a specific case of generic object detection, text possesses many defining characteristics which allow the usage of more specialized methodologies for its detection.

2.3.1.1 Traditional Approaches

Most traditional text detection approaches are based on Connected Components Labelling (CCL), also referred to as Connected Components Analysis (CCA) [17]. These methods localize and extract candidate components through image segmentation by colour clustering, edge detection, region-growing, etc. The extracted components are then filtered out by manual classifiers based on hand-crafted features. Other approaches involve the use of sliding window techniques in which windows or regions of varying size and passed through the image and are classified as regions of text or not. Text regions are further enhanced with morphological operations or statistical modelling methods for pattern recognition such as Conditional Random Field, a special case of Markov Random Fields. These approaches analyse the image under the context of a specific language, specifically looking for features that match it. The Stroke Width Transform (SWT) [18] operator instead analyses the image looking for textual features, specifically looking for the stroke width, potentially of text, at each pixel. After computing the stroke width at each pixel, a modified CCA is used to group and classify the pixels. Figure 2.7 shows the steps performed by the Stroke Width Transform operator to detect text regions in an image.

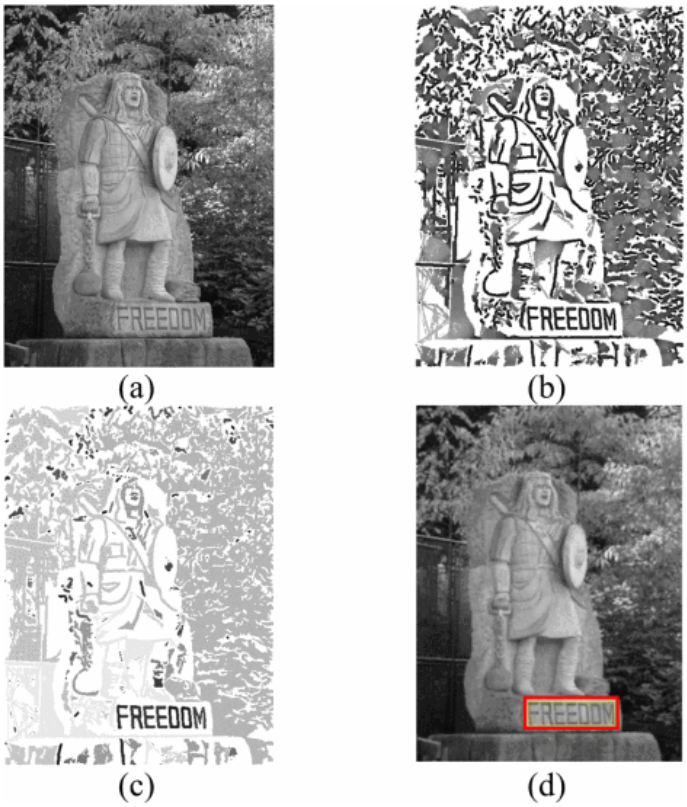


Figure 2.7: The Stroke Width Transform converts the image (a) from containing grey values to an array containing likely stroke widths for each pixel (b). This information suffices for extracting the text by measuring the width variance in each component as shown in (c) because text tends to maintain fixed stroke width. This puts it apart from other image elements such as foliage. The detected text is shown in (d) [18].

2.3.1.2 Deep Learning Approaches

Recent Deep Learning approaches for Text Detection are heavily inspired by the rapidly developing generic object detection algorithms. The primary change involves modifying the region proposal regression modules of generic detectors to localize text instead. These borrow the stacked convolutional layers that encode the image into feature maps which are then fed into a classifier to identify text locations. EAST (Efficient and Accurate Scene Text) [19] utilizes a FCN (Fully Convolutional Network) and integrates features from the different levels encoding the image as a single feature map and produces a harmonic mean (h-mean) of 83.27 in the IC15 dataset. CRAFT (Character Region Awareness for Text Detection) [20] achieves even better h-mean in the IC15 dataset with 89.8 using an innovative training network design that operates on the character neighbourhood level instead of the pixel level. Table 2.1 shows the performance of various Deep Learning text detection techniques on the ICDAR 2015 dataset. While this dataset focuses on scene text detection, in contrast to our digital document text detection, the principles applied are identical and the differences that do exist simplify the work of the text detector by the absence of illumination problems and having an orthographic perspective of the text.

Method	Recall	Precision	H-mean
Zhang et al. [21]	43	71	54
SSTD [21]	73	80	77
EAST [19]	78.3	83.3	80.7
CRAFT [20]	84.3	89.8	86.9
PMTD [22]	84.59	92.37	88.31

Table 2.1: Performance comparison of some text detection methods on the ICDAR 2015 dataset.

2.3.2 Text Recognition

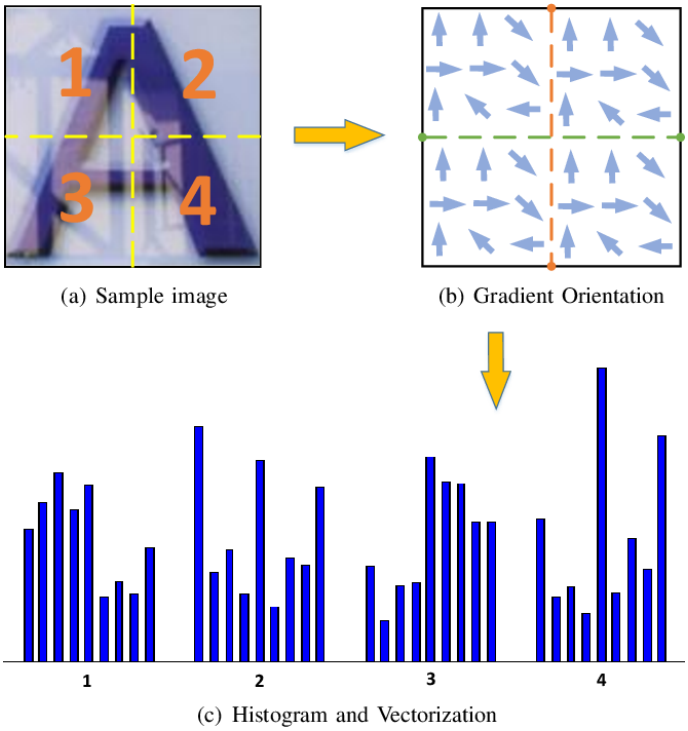
Text recognition takes well-defined regions known to have text and extracts the actual textual information present. Similarly to text detection, text recognition can be grouped into the larger set of generic object detection problems, while also benefiting from having unique characteristics to employ more specialized methodologies.

2.3.2.1 Traditional Approaches

Some traditional approaches for text recognition divide text recognition into several sub-problems to tackle individually, while other groups of research instead utilize complete feature-based methods.

Research into individual problems in text recognition involve text binarization to differentiate textual regions from the background, line segmentation to separate multiple lines, individual character segmentation to analyse individual characters and individual character recognition to finally extract the textual information.

Complete feature-based methods use feature descriptors such as Histogram of Oriented Gradients (HOG) and learning methods for classification such as Support-Vector Machines (SVM) [23] or random decision forests [24]. Figure 2.8 shows the HOG feature extraction of a character. Several enhancements to HOG for text recognition have emerged such as combining HOG with a spatial pyramid to encode the relative spatial layout of the character parts [25]. The aspect of including spatial information to character parts was also explored by Co-occurrence HOG [26] by counting the frequency of co-occurrence between pixel pairs.



(a) Sample image (b) Gradient Orientation

(c) Histogram and Vectorization

Figure 2.8: HOG feature extraction from a sample character [26].

2.3.2.2 Deep Learning Approaches

Deep Learning approaches to Text Recognition borrow much inspiration from generic object detection algorithms. While recognition of a single character can be achieved using a Convolutional Neural Network (CNN) [27], recognition of text of arbitrary length as a sequence of characters can be achieved with Recurrent Neural Networks (RNN). Tesseract 4 implements a Long Short Term Memory (LSTM) [28] system which is a type of RNN. Other approaches utilize Convolutional Recurrent Neural Networks (CRNN) [29] which combine a convolutional neural network (CNNs), a RNN and Connectionist Temporal Classification (CTC) loss as a scoring function. This method combines feature extraction, sequence modelling and transcription into a single system that does not need character segmentation. Figure 2.9 shows the architecture of a framework for text recognition, highlighting the role of a LSTM encoder and decoder.

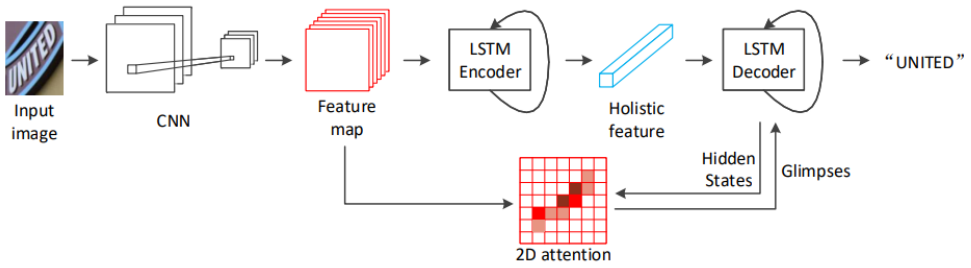


Figure 2.9: Overview of a framework for text recognition making use of LSTM models [30].

2.4 Object Detection

Object detection consists in the detection of instances of objects of a certain class in digital images and videos. Detection, as opposed to recognition or classification, involves defining a bounding box for the class instance in an image or video where the intended object is not necessarily the focus of the image and potentially with other objects of different classes also present. Object detection can be viewed as a generic problem to detect any kind of desired objects, however, studying the characteristics of the selected class allows the employment of more specialized techniques which can lead to better results.

2.4.1 Traditional Approaches

The first approaches to Object Detection utilized Template Matching [31], in which the image of the object desired to search, the template, is passed pixel by pixel through the scene image. For each pixel of the template and for each possible position in the scene image, by using a sliding window, a similarity metric is calculated. If the similarity is high enough it is considered that the template image is present at that location in the scene image. Different metrics may also be used to suit different use cases. Direct Template Matching approaches enforce many restrictions on the situations it can be used. The candidate image to match cannot be scaled or stretched in any way compared to the template image. Direct matching that compares only the colour will also be oblivious to similarities in shape or form.

More complex Template Matching techniques also try to search for rotated or scaled versions of the template image, however usually leading to a greatly increased number of computations. Recent approaches such as Best-Buddies Similarity (BBS) [32] and Deformable Diversity Similarity (DDIS) [33] have made template matching more dynamic to more complicated scenarios. Figure 2.10 shows, in the pink boxes, the results of using DDIS template matching with the green box.

Later approaches to Object Detection focus on the formulation of some way to represent the various features of images which can then be compared. These features can be corners, texture, colour properties, shape, etc. These features are built into a descriptor that contains the information of visual characteristics in the region. By comparing and matching descriptors it is possible to detect the presence of an object in an image.

The Harris Corner Detector is a popular corner detection operator that looks for regions that contain significant changes in all directions, this operator is rotation invariant but not scale-invariant, meaning it can distinguish corners even when the image is rotated but not when it is scaled. Figure 2.11 shows the corners detected by Harris on a chessboard. Scale-Invariant Feature Transform (SIFT) [34] is another technique capable of detecting local features but associated with a descriptor that is not only rotation invariant but also scale-invariant. Figure 2.12 shows the architecture of the SIFT algorithm. Later developments aimed to improve the performance of SIFT with Speeded-Up Robust Features (SURF) [35] and Binary Robust Independent Elementary Features (BRIEF) [36] by reducing the complexity of SIFT. Oriented FAST and Rotated BRIEF (ORB)

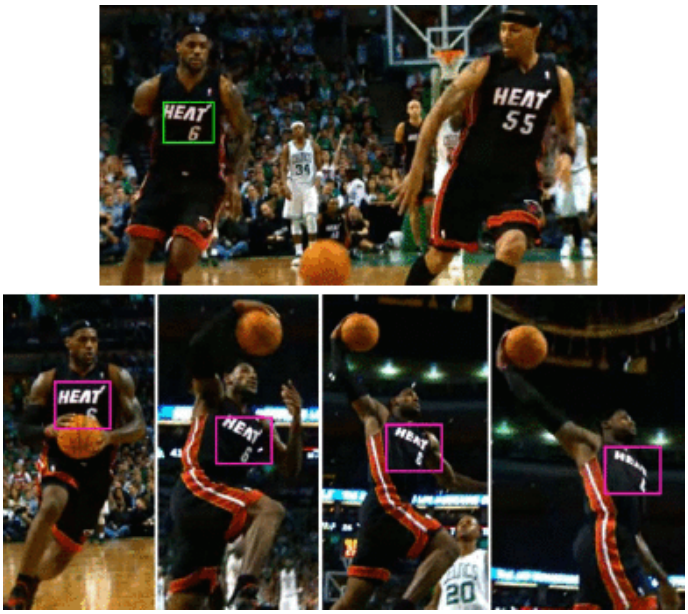


Figure 2.10: Template matching with DDIS [33].

[37] made modifications to the descriptors used by BRIEF and combined it with Features from Accelerated Segment Test (FAST) [38] key point detector. These methods performance depends on the circumstances in which they are applied but performance comparisons generally point to ORB as being the fastest algorithm and producing similar results to SIFT which obtains the best results [39].

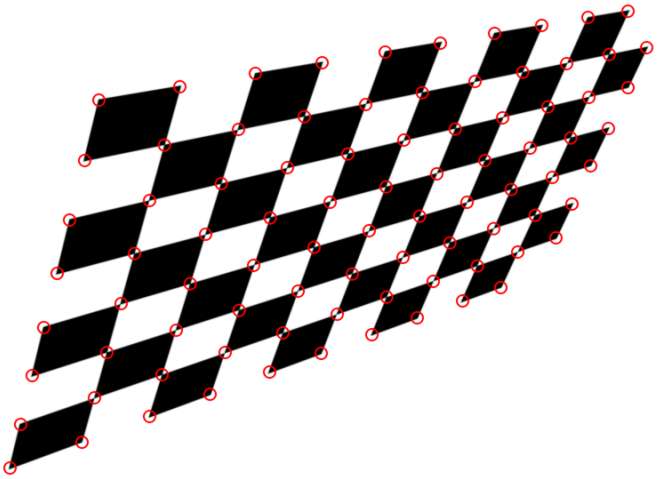


Figure 2.11: Harris corner detector applied on a chessboard [40].

The matching of two feature descriptors is based on their distance formula which depends on the descriptor used. Besides the calculation of the distance between two features, several enhancements have been developed to produce a finer match. Lowe's ratio test [41] takes the two best matches for a template Keypoint and calculates the ratio between them, if the value is below a certain amount the match is discarded. Other enhancements include the use of a symmetry

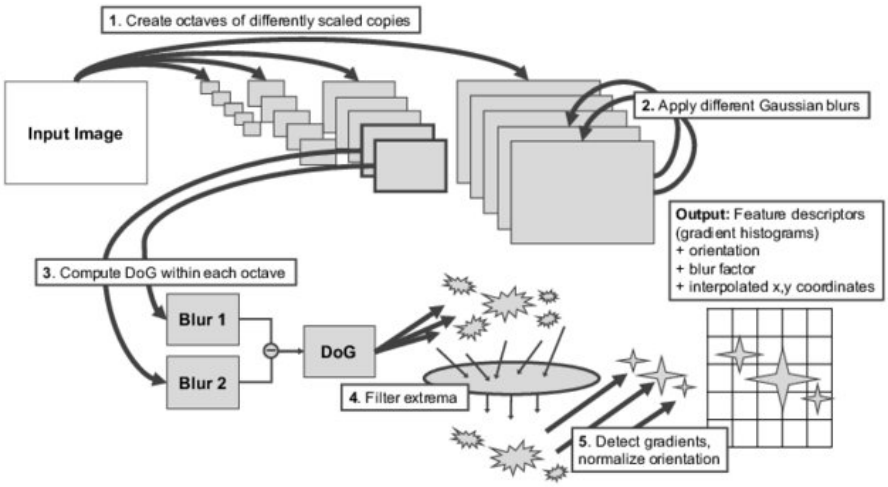


Figure 2.12: Steps of the SIFT algorithm [34].

test by using bidirectional matching and using techniques to find outliers in the matches, such as RANdom SAmple Consensus (RANSAC) [42].

Feature-based Object Detection can deal with scaling and rotation better than template matching. However, in circumstances where the scale and rotation of the template are known to match the ones present on the image, Template Matching may achieve equal or better performance than feature-based Object Detection.

2.4.2 Deep Learning Approaches

Deep Learning techniques for object detection have attracted a lot of attention and research since the surge in popularity after a CNN called AlexNet achieved breakthrough performance labelling images in the ImageNet challenge in 2012 [43]. There was however the need to select different regions of interest in the same image which may contain the objects to detect. With objects in different locations and of different scales inside the same image the number of regions to detect quickly increases. Later advancements aimed to reduce the number of regions by a variety of methods. Region Based Convolutional Neural Networks (R-CNN) [44] and later variations such as Fast R-CNN [45] aimed to fix this problem by limiting the number of regions to check to a manageable number with the first R-CNN variation using a selective search algorithm for regions and limiting them to 2000 regions. Improvements to R-CNN centred on removing the fixed region selection algorithm used by it which was instead replaced by a stage in the CNN, drastically reducing the number of regions to process. Problems with detecting small objects in images were later addressed by Feature Pyramid Networks (FPN) [46] which uses an image pyramid to access smaller sections of the image.

While the previous methods look at specific, identified regions of the image, You Only Look Once (YOLO) [47] trains on full images simultaneously predicting bounding boxes and their object class probabilities, which led to a significant increase of detection speed but some loss of

accuracy compared to the previous two-stage detectors. Figure 2.13 illustrates the main stages of the YOLO system. After the first version, several other versions of YOLO with better results in testing datasets were created. YOLO9000 or YOLOv2 [48] introduced batch Normalization layers after each Convolutional layer. YOLOv3 [49] improved detection for smaller objects by performing it at different scales with the use of a FPN. YOLOv4 [50] applied several state-of-the-art methods such as Bag of Freebies [51], CBN(Cross-iteration batch normalization) [52] and PAN(Pan aggregation network) [53] which gave it a considerable performance boost. Figure 2.14 compares the performance of YOLOv4 with other object detection architectures, most notably with YOLOv3, on the MS COCO dataset.

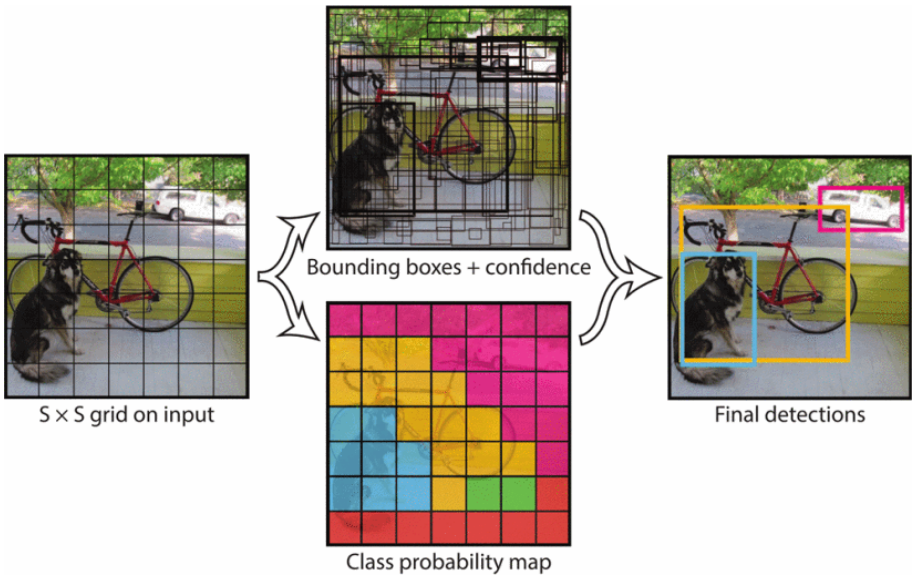


Figure 2.13: Main stages of the YOLO system [47].

Single Shot MultiBox Detector (SSD) [54] took the one-step object detector process of YOLO and introduced multi-reference and multi-resolution detection techniques which greatly enhanced performance. One-step object detection was later found to have severe class imbalance during training which was attributed as the culprit for the performance difference in comparison to two-step techniques like Faster-CNN. RetinaNet [55] addresses this problem by introducing a focal loss function which accelerates the process of deciding the confidence level on the correct class which led to a large performance gain. Figure 2.15 illustrates the architecture of the FPN adopted by RetinaNet.

Table 2.2 lists the median AP of the aforementioned object detection methods on the VOC 2012 TEST SET.

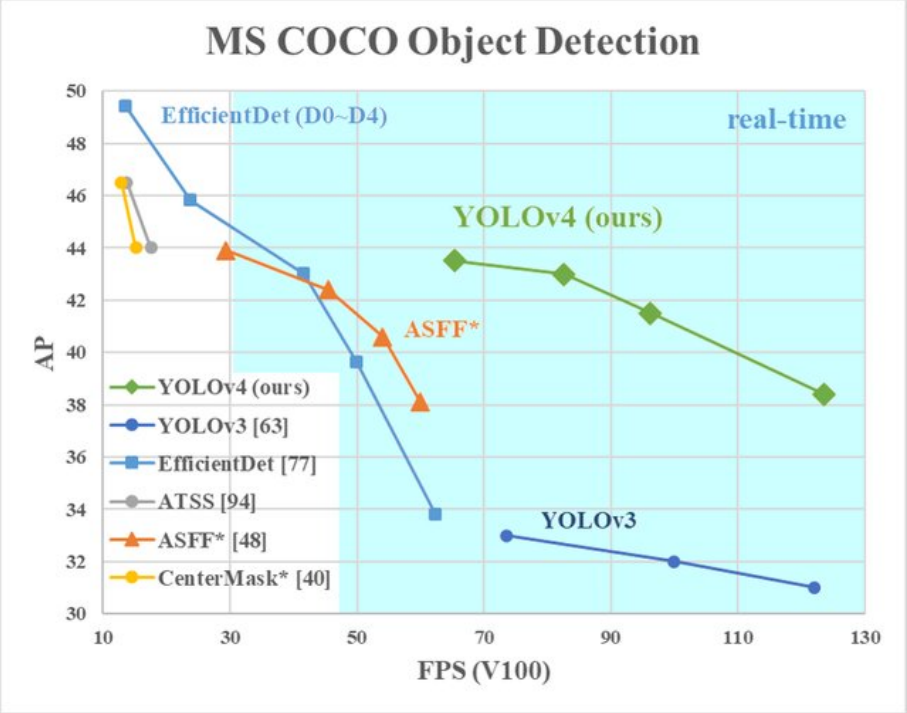


Figure 2.14: Performance comparison between YOLOv3 and YOLOv4 among other methods [50].

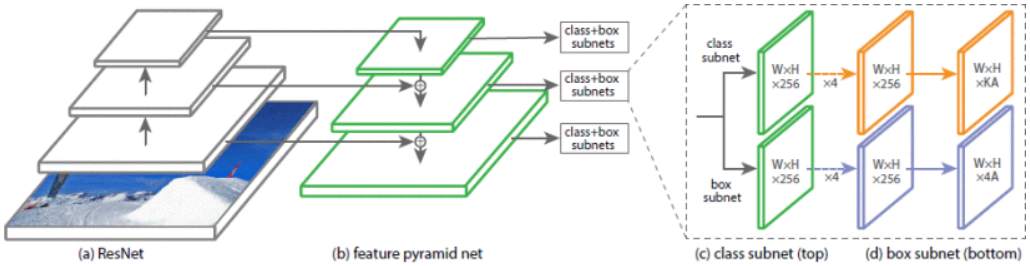


Figure 2.15: FPN adopted by RetinaNet [55].

Method	mAP
R-CNN [44]	53.3
Fast R-CNN [45]	68.4
Faster R-CNN [45]	70.4
YOLOv1 [47]	57.9
YOLOv2 [48]	78.2
SSD [54]	79.3
RetinaNet [55]	85

Table 2.2: Performance comparison of some object detection methods on the VOC 2012 TEST SET. mAP denotes the median AP across 20 categories (aero, bike, bird, boat, bottle, bus, car, cat, chair, cow, table, dog, horse, mbike, person, plant, sheep, sofa, train and tv) [56].

2.5 Similar applications

The detection and recognition of text and symbols in digital documents, from engineering drawings to floor plans, is a common problem. Recent approaches make use of deep learning techniques to obtain high accuracy models, though depending on the existence of a robust dataset that represents every class of symbol or text in sufficient quantity and in various circumstances.

Models pre-trained in CRAFT, EAST and Tesseract have been used to detect and recognize text in floor plans in similar conditions to the proposed problem [57][58].

CNN architectures inspired by YOLO or Faster RCNN have been used to detect and classify symbols in engineering and piping drawings [59][60].

The good results obtained in these papers highlight the capacity of Deep Learning techniques to obtain high accuracy solutions in problems identical to the ones studied in this thesis. Nonetheless, these papers alert to the need for a robust dataset with sufficient class distribution and conditions variation.

Chapter 3

Dataset Characterization

3.1 Source and Structure

The dataset consists of a series of welding blueprint images with their respective Alphanumeric References and Welding Symbols. It is comprised of the images of each blueprint and annotations about the position and content of every Alphanumeric Reference and Welding Symbol in them. The position is represented as the 2D coordinates of a point inside the Alphanumeric References or Welding Symbol, situated approximately at its centre. Both problems are composed of two subproblems - detection and recognition. A correct recognition is dependent on a correct detection, if the detection fails to include crucial information for the recognition it will almost certainly fail. The position specified in the ground truth must refer to a position that is essential to ensure a correct recognition. However, the inclusion of this point in the detection does not guarantee a correct detection. A failure in the recognition step may indicate a failure of the recognition itself or a faulty text detection.

3.2 Characteristics

The dataset includes 17 welding blueprints, containing 2893 Alphanumeric References and 1173 Welding Symbols. All of the blueprints contain Alphanumeric References, however, not all of them contain Welding Symbols. The dimensions of the blueprint images range from the smallest with 4678x3309 pixels to the largest with 14042x9933 pixels. These large dimensions pose some implementation considerations which will be discussed in Section 4.2.7.

3.2.1 Alphanumeric References

Alphanumeric References consist of a sequence of letters and numbers forming an alphanumeric code. The font used by these codes is the same for all blueprints. It is a sans-serif font and automatic font identification software identifies it as some variation inside the Arial font family. A uniform font across the entire dataset facilitates the process of text recognition but could misrepresent the problem. As the text is present in technical documents, in this case welding blueprints,

it is safe to assume that the fonts used will generally be a Sans Serif Type Style. Deep Learning text recognition techniques like Tesseract are trained with a set of fonts, typically formal fonts, ideal for OCR. Tesseract documentation provides a list of all of the fonts used to train the LSTM system, in which the Arial font family is present.

The size of the characters present in the Alphanumeric References varies between blueprints, with some having 16 pixels of height and others 24 pixels. The size of the characters may have an impact on the performance of text recognition.

The stroke width of the text of the Alphanumeric References is always 2 pixels or more. An insufficient thickness could hinder text detection and recognition. The drawing lines of the blueprints have varying thickness and in some cases have the same thickness as the text which may be problematic if a segmentation of both is required.

Some of the Alphanumeric References in the blueprints are displayed at an angle; 54 Alphanumeric References are displayed at a non-square angle and 64 Alphanumeric References are vertical. Non-horizontal Alphanumeric References represent 4% of the dataset and require preprocessing before having text recognition applied to them.

Most Alphanumeric References are surrounded by a continuous line around them, referred to as the Reference Frame. This line can obstruct text detection and may be necessary to remove. Some other Alphanumeric References are surrounded by a non-continuous line in a dashed form, representing 10% of the dataset.

The alphanumeric codes present in the Alphanumeric References are composed of capital letters and numbers. The Alphanumeric References of one of the blueprints of the dataset also contain the dash character (-). This particular blueprint has 79 Alphanumeric References, representing 4% of the dataset. Table 3.1 shows the number of instances of each character in the text of Alphanumeric References. Some characters are completely absent from the dataset while others are much less represented than the most occurring ones. Certain character mismatches are common in OCR, such as between the character 0 (zero) and the capital letter O. As some characters are absent, certain mismatches could be missing in this dataset. There is no indication that certain characters are always absent from the Alphanumeric References text even in other blueprints outside the used dataset.

The length of the text in the Alphanumeric References varies, with the smallest at 9 characters and the longest at 29. The length is associated with the number of lines of the Alphanumeric References as the length of each line varies between 9 and 16 characters and the longest Alphanumeric References are composed by multiple lines.

Most Alphanumeric References have only a single line but 106 have two lines. It is important that text detection and recognition correctly associates the two lines into the same Alphanumeric Reference while aware that they are different lines.

Character	Instances
0	3309
1	5577
2	5354
3	1773
4	2835
5	1249
6	1835
7	1022
8	1641
9	655
A	2864
B	389
C	377
D	0
E	0
F	1195
G	0
H	572
I	0
J	0
K	0
L	2424
M	0
N	3
O	0
P	3383
Q	0
R	879
S	222
T	340
U	0
V	8
X	50
Y	0
W	0
Z	357

Table 3.1: Instances of each character in Alphanumeric References.

3.2.2 Welding Symbols

Welding Symbols consist of several elements that inform the operator about the type and location of the weld as well as information that supplements the weld symbol. The drawing and format of these symbols is defined by the ISO 2553 standard [61]. Our problem centres on the detection and recognition of the weld type. Figure 3.1 illustrates the various parts of a weld symbol, in green the

weld location, in red supplementary information about the weld, in orange the weld reference and in blue the type of weld which is what is desired to identify. All of the Welding Symbols of the dataset include a back or backing weld symbol which is the arrow shape behind the actual weld symbol, illustrated by the dashed blue box in Figure 3.1.

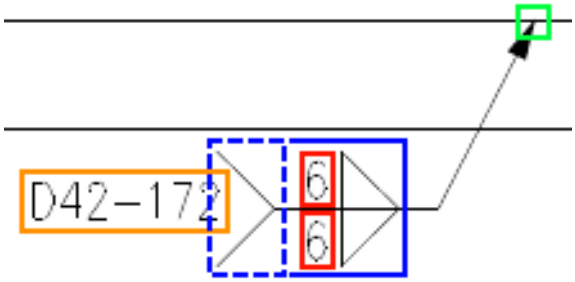


Figure 3.1: Parts of a Welding Symbol, in green the weld location, in red supplementary information about the weld, in orange the weld reference and in blue the type of weld which is what is desired to identify.

The space between the actual weld symbol and the back or backing weld depends on the length of the supplementary information in red. Figure 3.2 shows two instances of the same Welding Symbol class with different space between the back or backing weld and the actual weld symbol, depending on the length of the supplementary information. In this case, the distance between the two components is 26 pixels on the first case and 35 pixels on the second. The detection and recognition system will need to be able to handle these different representations of the same Welding Symbol class.

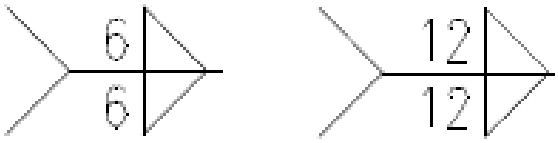


Figure 3.2: Different symbol dimensions based on length of supplementary information.

The dataset contains 24 different Welding Symbols. Table 3.2 shows the numbers of instances of each Welding Symbol class in the Alphanumeric References and a sample image of the symbol shape. Certain Welding Symbols classes such as 1, 3, 4 and 7 are much more represented than the others. In the case of symbol classes 8 and 15 there is a single instance, meaning that the instance used as a template will also be the only test in the entire dataset.

Table 3.2: Instances of each Welding Symbol class in Alphanumeric References

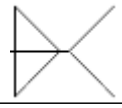
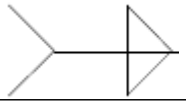
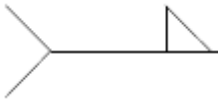
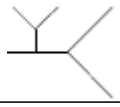
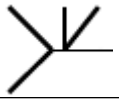
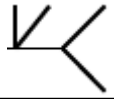
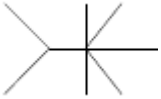
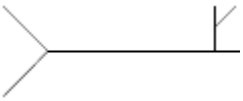
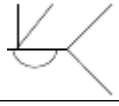
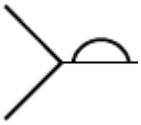
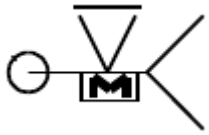
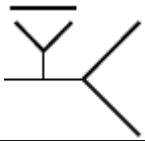
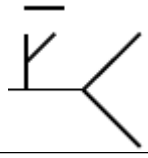
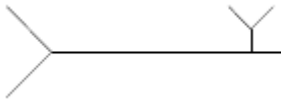
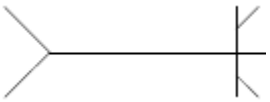
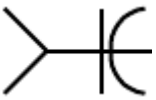
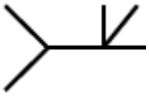
Welding Symbol	Instances	Sample
1	400	
2	2	
3	333	
4	149	
5	34	
6	4	
7	132	
8	1	
9	6	
10	2	
11	38	
12	10	
13	11	
14	13	

Table 3.2: Instances of each Welding Symbol class in Alphanumeric References

Welding Symbol	Instances	Sample
15	1	
16	2	
17	2	
18	8	
19	2	
20	3	
21	3	
22	5	
23	7	
24	3	

The size of Welding Symbol instances depends on the class and the length of the complementary information. Different blueprints also have slight differences in the size of instances of the same symbol class. Figure 3.3 illustrates two instances of the same symbol class with very different dimensions. The scale factor between the two instances is approximately 2 and this scaling is shared by the other symbol classes between the two blueprints of these instances.

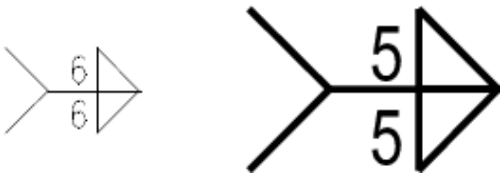


Figure 3.3: Different Welding Symbol scales for the same instance.

Instances of the same symbol class can have different stroke thicknesses. The thickness along the Welding Symbol may also not be uniform. Figure 3.4 shows three instances of the same symbol class with different stroke thickness and the case of the second instance which does not have a uniform thickness, its horizontal line in the centre only has a thickness of 1 pixel compared to the 2 pixels of the rest of the shape.

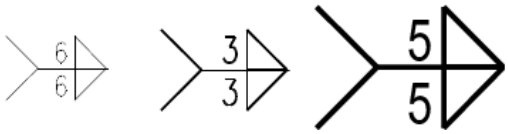


Figure 3.4: Different Welding Symbol thicknesses for the same class.

Certain shapes of a symbol class are also present in other symbol classes. In some cases, the shape of a symbol class can be a sub-shape of another class. Figures 3.5 and 3.6 display the similarities between symbol classes 7 and 1 and 3 and 4, respectively, with the difference between the symbols in red. When detecting instances of these classes, the surplus of information, in these cases related to the existence or absence of the triangle shape, will complicate the selection between the two matching candidates in the pair.

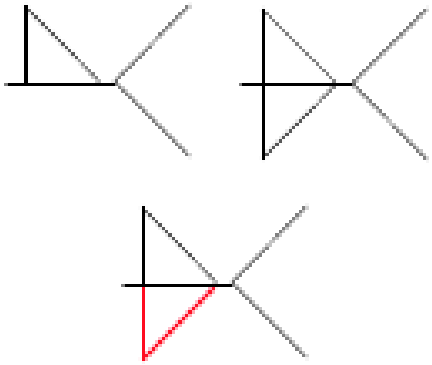


Figure 3.5: Instances of symbol class 7 and 1 with the difference in red.

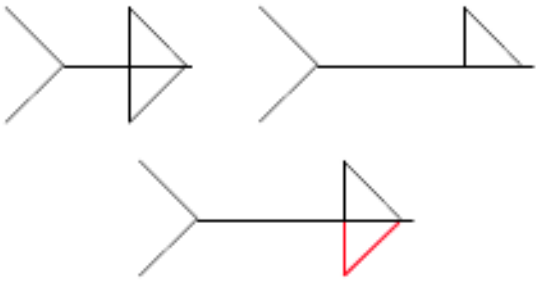


Figure 3.6: Instances of symbol class 3 and 4 with the difference in red.

Chapter 4

Proposed Solution

Pre-trained deep learning models for text detection and recognition will be used for the Alphanumeric Reference problem. Of the available techniques, EAST, CRAFT and Tesseract were selected for their proven results and availability. Pre-processing and output refinements will be required when applying these techniques which will be tackled using traditional computer vision techniques which will deal with specific situations.

Due to the limitations of the dataset identified in Chapter 3, a traditional computer vision approach using feature matching was elaborated for Welding Symbol Detection. Pre-processing techniques are used to make the variations of Welding Symbol classes between blueprints as close as possible and a robust feature matcher, using several enhancements to the basic descriptor matching, is used to obtain finer matches.

The proposed solutions for each of the problems have their own pre-processing steps specifically designed for them. However, they also share a common pre-processing step that prepares the input blueprints for certain techniques. Figure 4.1 gives an overview of the shared pre-processing step, which is used by both solutions, and that is elaborated in Section 4.1.

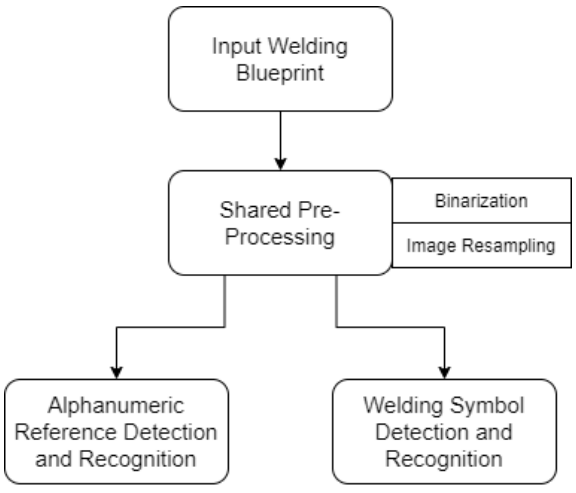


Figure 4.1: Overview of the Shared Pre-Processing step.

4.1 Shared Pre-Processing

The welding blueprints in the dataset are provided in Portable Document Format (PDF) as they are intended for printing. The first step is to convert them to an easily manipulable image format such as Portable Network Graphics (PNG). These images are obtained in greyscale with a white background and elements in black with colour blending around them. Certain techniques to be used such as Morphological Operations or Connected Components Analysis require a binarized image. As the blueprints consist of a pure white background with elements in black, a binarization is applied with a threshold of 254 for the range 0 to 255. This ensures that every detail in the blueprints is preserved in black with only the background remaining white. In a dataset that uses scans of printed blueprints, potentially having a non-uniform background, a more dynamic binarization would need to be used.

4.1.1 Image Resampling - Interpolation

Certain pre-processing techniques will require rotation and scaling of images. These processes can cause loss of image quality which can severely hinder recognition techniques. When performing these techniques, interpolations are applied that aim to minimize the quality loss. A simple Nearest-neighbour interpolation will select the pixel which is closest to the rotated or scaled position and does not consider the values of neighbouring points. This can lead to incorrect pixel selection and a concentration of value in a single pixel that should be spread among itself and its neighbours. Bilinear and Bicubic interpolations take into consideration the neighbourhood of the target position spreading the value to those pixels. Bilinear interpolation applies to the 4 pixels around the sample point in a 2x2 grid while Bicubic interpolation analysis 16 pixels in a 4x4 grid. Both methods yield better results than Nearest-neighbour interpolation while the difference between them is negligible for text recognition. The different results of the interpolation techniques will be analysed in Section 4.2.3, upon Alphanumeric Reference rotation. Figure 4.2 compares the aforementioned interpolation techniques. The black dots correspond to the point being interpolated, and the red, yellow, green and blue dots correspond to the neighbouring samples. Their heights above the ground correspond to their values.

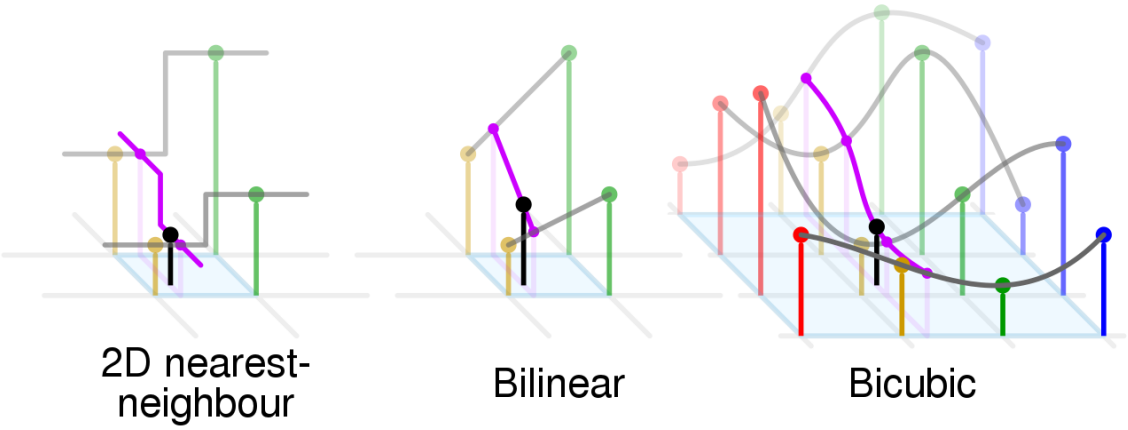


Figure 4.2: Comparison of interpolation techniques. [62]

4.2 Alphanumeric Reference Detection and Recognition

A flowchart of the proposed solution for Alphanumeric Reference Detection and Recognition is illustrated in Figure 4.3. It is split into 5 main steps: image pre-processing; text detection; text box refinement; text recognition and text refinement.

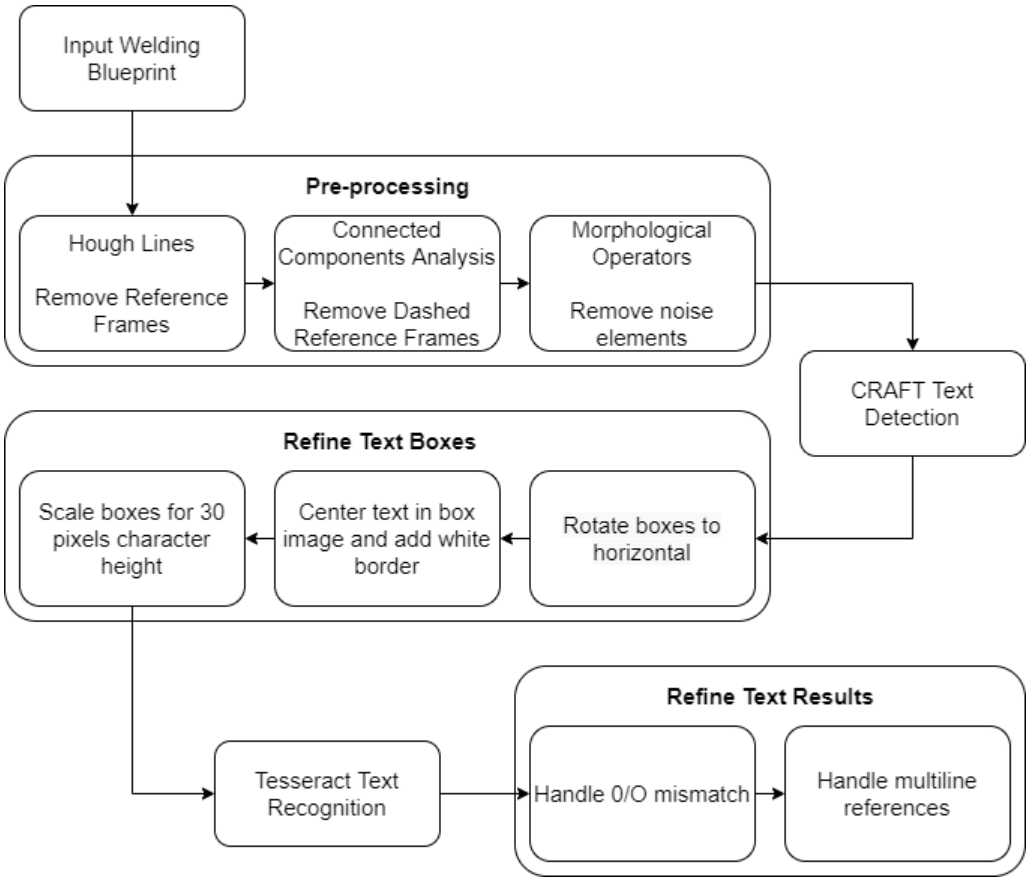
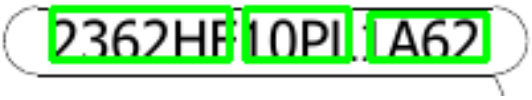


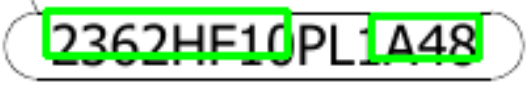
Figure 4.3: Alphanumeric Reference Detection and Recognition Flowchart.

4.2.1 Pre-processing

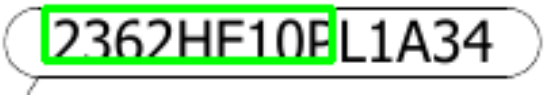
The deep learning techniques selected for text detection were EAST and CRAFT, which were analysed in Section 2.3.1.2. They were selected due to their good results, availability and documentation. At first, Deep Learning techniques were applied directly to the blueprint. This produced poor results with many of the Alphanumeric References not being properly detected as is illustrated in Figures 4.4a, 4.4b and 4.4c. The green rectangles denote the text regions identified by EAST.



(a) Split Alphanumeric Reference Detection (EAST).



(b) Vertically misaligned Alphanumeric Reference Detection (EAST).



(c) Partial Alphanumeric References Detection (EAST).

Figure 4.4: Cases of incorrect reference text detection (EAST)

In Figure 4.4a the Alphanumeric Reference was properly detected but split into multiple regions which would not constitute a problem for text recognition as long as the regions are properly combined by considering their proximity. In Figure 4.4b a bigger problem emerges as not only is the Alphanumeric Reference split into multiple text regions but the text regions identified intersect the text and omit part of the characters which could be problematic for the consequent text recognition. In Figure 4.4c another problem emerges as the Alphanumeric Reference is only partially detected which would make its proper recognition impossible. By analysing the problems identified, it was hypothesized that the Alphanumeric Reference frame could be interfering with the text detection algorithms, especially the horizontal line above the text. As such, the methods identified in Section 2.2 were applied to remove the Alphanumeric Reference frame and obtain better text detection results.

In the case of Alphanumeric References like Figures 4.4a, 4.4b and 4.4c the frame around the reference has a lower thickness compared to the thickness of the text, also referred to as the stroke width of the text. With this difference in thickness, it is possible to apply Morphological operators which remove the frame or the text to be individually analysed, in our case, for text detection.

In some blueprints, the thickness of the Alphanumeric Reference frame is the same as the thickness of the text which would make it impossible to remove one of the components from

the other using Morphological Operators. Additionally, this difference between the thickness of the reference frame and the text would need to be known for each blueprint to select the correct kernel size for the Morphological operations. Figure 4.5 shows an Alphanumeric Reference with a difference in thickness between the reference frame and the reference text while Figure 4.6 shows a reference whose parts of the frame have the same thickness as the text, not allowing it to be separated using morphological operators. Instead, more dynamic solutions were explored which can more easily adapt to differences in this relation.

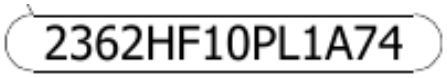


Figure 4.5: Alphanumeric Reference with different thickness between reference frame and text stroke width.

Figure 4.6: Alphanumeric Reference with same thickness between reference frame and text stroke width.

4.2.1.1 Removing Alphanumeric Reference Frame - Hough Line Transform

To remove the horizontal lines above and below the text, constituting the frame of the Alphanumeric Reference, the Hough Line Transform was applied to detect and eliminate these. To limit the line detection to the reference frame, without interfering with the text, the selected Hough lines were required to have significant length, longer than characters, and to be continuous, to not consider lines combining line segments from characters. Figure 4.7 shows the Alphanumeric Reference from Figure 4.4a with Hough lines detected in red. After removing these lines, applying EAST text detection produces Figure 4.8 with the text now properly detected.

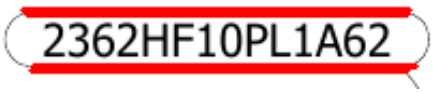


Figure 4.7: Alphanumeric Reference of Figure 4.4a with Hough lines detected in red.

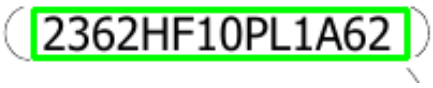


Figure 4.8: EAST detection on the Alphanumeric Reference of Figure 4.4a after Hough lines removed.

4.2.1.2 Removing Dashed Alphanumeric Reference Frame - Connected-components Analysis

Some Alphanumeric Reference frames are not continuous lines but instead dashed lines, as is illustrated in Figure 4.9.

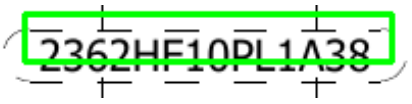


Figure 4.9: Vertically misaligned dashed frame Alphanumeric Reference detection.

Alphanumeric References with a dashed frame can not have their frame removed by using the Hough Line Transform as it would require line candidates to allow line segment combinations which could lead to situations where the combinations of character segments of the text could be identified as a line. Figure 4.10 illustrates an Alphanumeric Reference with a dashed frame having Hough Line Transform applied to it while allowing non-continuous lines. The reference frame is correctly detected and drawn over but so are character segments of the reference text.

Figure 4.10: Hough Line Transform applied allowing non-continuous lines over a dashed Alphanumeric Reference frame.

Therefore, to deal with these lines, Connected-Component Analysis is performed to identify and remove connected-components of small area, in pixels, or with a rectangular shape. The area threshold to remove connected-components is selected as an amount that is too small to represent text, in which the text would be too small to be readable. A connected-component as a rectangular shape is one when its connected-component area, in pixels, is the same as the area of its bounding box. The rectangular shapes remove the dashed frames and other noise in the blueprint without interfering with the text. Figure 4.11 shows an Alphanumeric Reference with a dashed frame with connected-components with low areas marked in blue and rectangular shapes in red. Removing these clears most of the reference frame.

Figure 4.11: Alphanumeric Reference with a dashed frame marked with connected-components with low area in blue and rectangular shapes in red.

Removing connected-components with an area, in pixels, larger than a maximum defined value was also considered to remove the reference frames entirely even when continuous. However, the relation between the area of the reference frame and of the text is not always the same, in some situations the characters of the reference text are connected which creates connected-components

with an area that approximates that of the frame, depending on the stroke width of the text. Figure 4.12 illustrates the case of an Alphanumeric Reference where the characters are too close to each other and form a connected-component. A maximum allowed connected-component area would remove the elements in red, including reference characters.

Figure 4.12: Alphanumeric Reference with characters too close to each other forming a single connected-component from multiple characters.

4.2.1.3 Removing Small Noise Elements - Morphological Operators

As the last step of pre-processing the image, a Closing morphological operation - an Erosion followed by a Dilation - is applied with a 3x3 kernel. This is used to remove the remaining single-pixel elements that can interfere with text detection or recognition. As analysed in Chapter 3, the text of the Alphanumeric References always has at least 2 pixels of thickness so this step will not damage the text itself.

4.2.2 Alphanumeric Reference Detection

With the pre-processing complete, having removed the Alphanumeric References frame, the blueprint image is now ready for text detection. Figure 4.13 shows a region of a blueprint with various Alphanumeric References with problematic characteristics such as varying frame thickness and text with the same thickness as the frame, Figure 4.14 shows the same region after pre-processing.

Text detection techniques are applied, to the pre-processing result, to produce bounding boxes around the text. CRAFT is chosen over EAST for producing more refined bounding boxes centring the text and without gaps. Figures 4.15a and 4.15b show the text boxes detected by CRAFT and EAST, respectively, on the example shown in Figure 4.14. Note that this implementation of EAST is not making use of the bounding box rotation and is only being used to assess performance on square-angle Alphanumeric References.

The boxes produced by CRAFT properly centre around the text and groups the Alphanumeric References code together which facilitates text recognition. Other textual information in the blueprint is also detected such as measurements and other instructions. Knowing that Alphanumeric References are alphanumeric codes with some length, bounding boxes with a very different aspect ratio - the ratio between their width and their height - can be discarded.

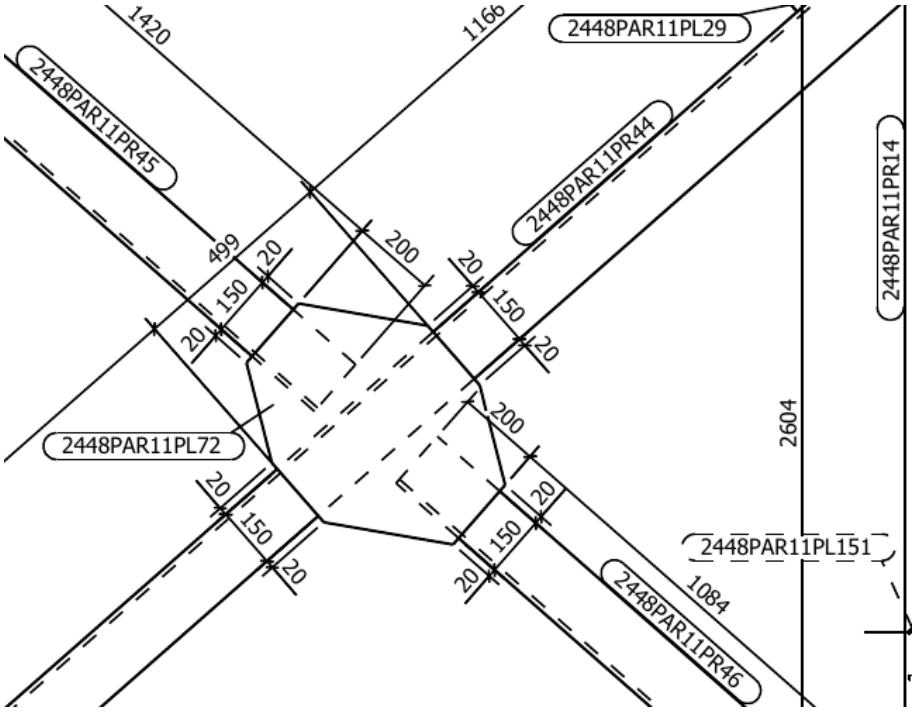


Figure 4.13: Region of a blueprint with Alphanumeric References with problematic frames with varying frame thickness and text with the same thickness as the frame.

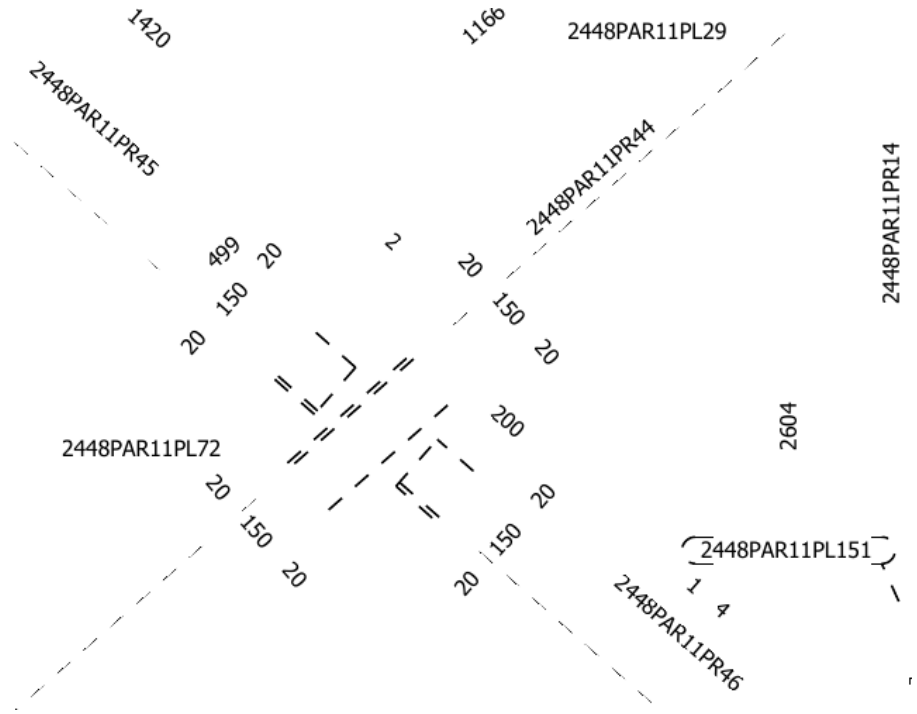
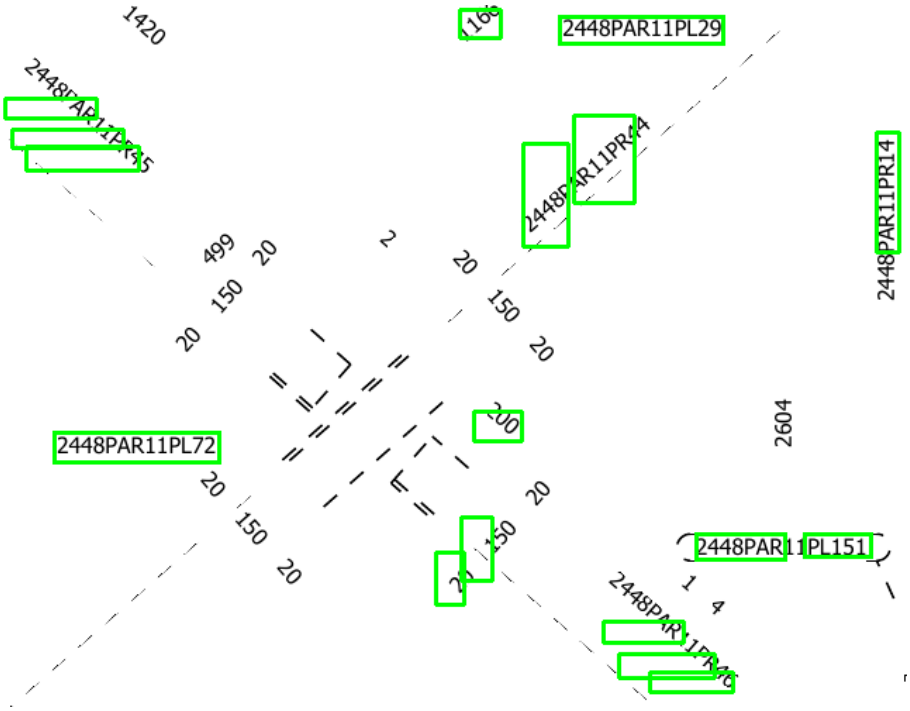
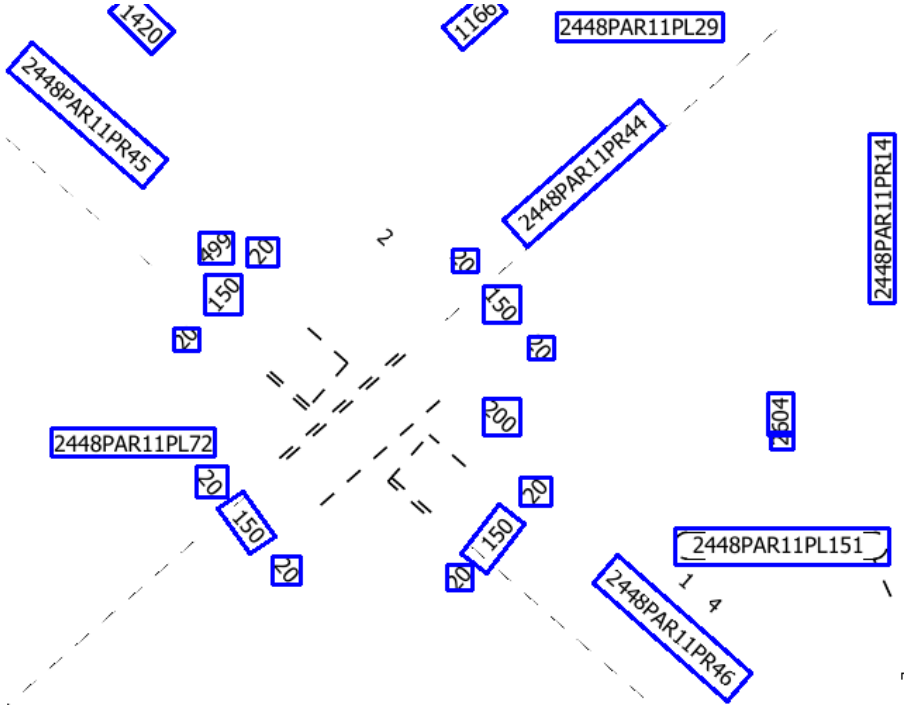


Figure 4.14: Region of a blueprint with Alphanumeric References with problematic frames, after pre-processing.



(a) Text boxes detected by EAST.



(b) Text boxes detected by CRAFT.

Figure 4.15: Text detection techniques on the example shown in Figure 4.14

4.2.3 Rotating Detected Text Boxes to Horizontal

The majority of Alphanumeric References are presented horizontally, however, some are displayed vertically and others even at non-square angles. While deep learning text detection methods are

designed to deal with these situations, especially after noise elements around the Alphanumeric Reference have been removed, text recognition methods require the text to be presented horizontally, so a rotation has to be calculated and applied on the text detection bounding box. When a non-square rotation is performed, the choice of the interpolation method is important to ensure the resulting image has good quality for text recognition. Figure 4.16 shows an Alphanumeric Reference in a non-square angle and Figure 4.17 shows that Alphanumeric Reference rotated using Nearest-neighbour, Bilinear and Bicubic Interpolation. Nearest-neighbour produces artefacts and warps pixels to wrong positions while Bilinear and Bicubic Interpolations try to approximately preserve the positions of the intensity values. Bicubic Interpolation is used as it retains the intensity of the pixels slightly better than Bilinear Interpolation which lightens the pixels compared to the original image.

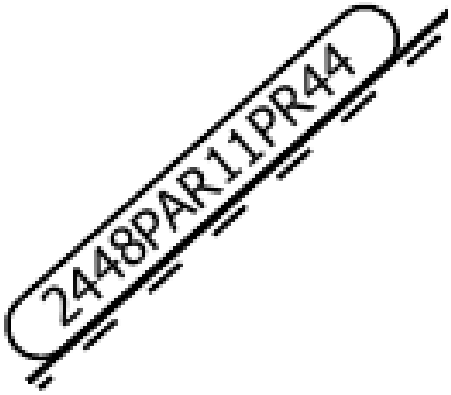


Figure 4.16: Alphanumeric Reference in non-square angle.

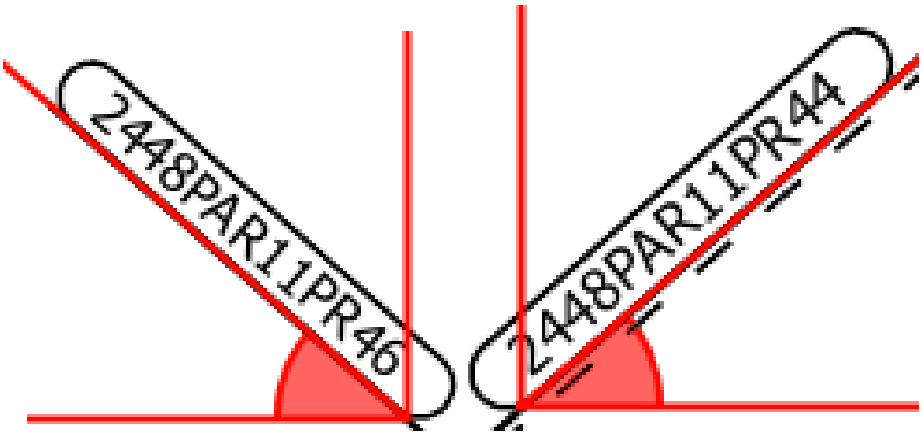
(a) Nearest-neighbour Interpolation.

(b) Bilinear Interpolation.

(c) Bicubic Interpolation

Figure 4.17: Alphanumeric Reference rotated to horizontal with Nearest-neighbour, Bilinear and Bicubic Interpolation.

The text of non-horizontal Alphanumeric References is always oriented to be readable so the rotation is always performed using the smallest angle the reference makes with the horizontal. Vertical Alphanumeric References have the text oriented to the right so they are rotated in that direction. Figure 4.18 shows two Alphanumeric References angled in different directions with text oriented to be readable, with the angle of rotation highlighted.



(a) Rotated clockwise. (b) Rotated counter-clockwise.

Figure 4.18: Alphanumeric References rotated clockwise and counter-clockwise.

4.2.4 Refine Text Boxes

Before applying text recognition with Tesseract, several refinements can be made to improve results. Tesseract documentation recommends various image conditions to ensure the best results in text recognition, these are verified and applied now [63].

4.2.4.1 Binarization

Tesseract internally binarizes the image during its text recognition process. However, if the image does not have a uniform background, it recommends a preemptive binarization, handled by the developer, which deals with the problematic background. In our case, the background is uniformly white so no further adjustment is required.

4.2.4.2 Noise Removal

Preemptive noise removal is also recommended, especially around the text. In our case, the noise around the text has already been removed for text detection and our images do not have other forms of global noise such as granularity effects from scene capture or artefacts from compression during PDF to PNG conversion.

4.2.4.3 Centring Text

Tesseract recommends that the text is centred inside the bounding box which is not always the case. In order to centre the text, horizontal and vertical scanlines are swept in both directions to find the first and last row and column with a non-white pixel. These pixel positions are then used to remove the borders around the text, ensuring it is best centred in the bounding box.

4.2.4.4 Border around text

Tesseract also recommends that text has a minimum border of approximately 10 pixels around it for better recognition. While potentially uneven borders were removed in the previous step, a uniform border of 10 pixels is now added on the result of the removal.

4.2.4.5 Character Height

Tesseract text recognition performs differently depending on the character size of the image, specifically the character height [64] as Figure 4.19 illustrates. While it is desirable to upscale very small characters, the error rate also increases when upscaling characters already at a given height. The actual ideal character height depends on the font, however, Tesseract recommends a character height of approximately 30 pixels to produce the best results. The height of the characters in a detected text box is sampled by obtaining the bounding box of a character using Tesseract, if the height is different from 30 pixels the image is scaled to obtain that character height. The scaling is performed using Bicubic Interpolation to maintain the quality of the image.

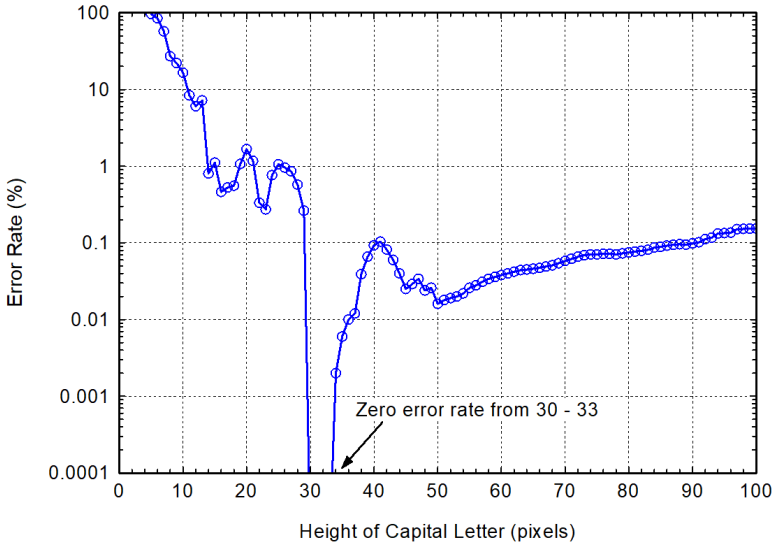


Figure 4.19: Relation between Error Rate in relation to height of capital letters [64].

4.2.5 Text Recognition

With the text boxes detected and refined with Tesseract recommendations, Tesseract text recognition is now applied to them.

By default Tesseract, expects a page of text to perform text recognition. This assumption allows Tesseract to apply several internal optimizations to obtain better results. However, in our case, the text desired to identify is on a small region and typically of a single line. For this, Tesseract has different Page Segmentation Methods (PSM), Figure 4.20 shows all the available PSM modes. PSM 7 is used with which Tesseract treats the image as a single text line. As mentioned

in Section 4.2.6.2 of the dataset analysis, some Alphanumeric References contain multiple lines, these will be treated and split manually, in the next step, to deal with multiline references whose lines are too close.

```
 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.
```

Figure 4.20: Tesseract Page Segmentation Methods. [63]

4.2.6 Refine Text Recognition

While occasional character mismatches are expected in text recognition, certain recurring situations were identified as the main sources of lower accuracy in certain blueprints.

4.2.6.1 O and 0 mismatch

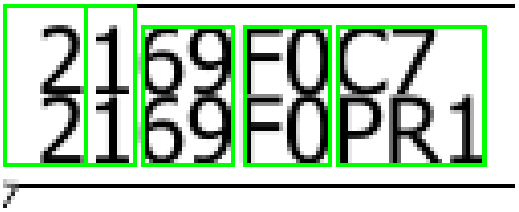
Tesseract may confuse certain characters due to their shape similarity. Pre-processing the text box with the various enhancements suggested by Tesseract documentation helps minimize these occurrences but may not avoid them entirely. In our dataset, Tesseract frequently confused 0 (zero) with the capital letter 'O'. Figure 4.21 illustrates an example of an Alphanumeric Reference with the number 0 that gets incorrectly recognized as the character O. To deal with this a manual analysis of the character is performed whenever Tesseract identifies a 0 (zero) or the capital letter 'O'. In the font used by the alphanumeric codes, as with most fonts, the 0 (zero) has less width than the capital letter 'O'. With this information, the ratio between the height and length of the character is calculated to decide if it represents a 0 (zero) or a capital letter 'O' based on a threshold ratio value calculated from sample characters of the font present in the blueprints.

Figure 4.21: Alphanumeric Reference with 0 (zero) character incorrectly recognized as the capital letter O.

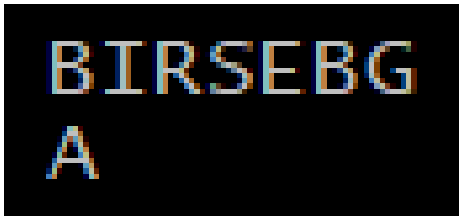
4.2.6.2 Multiline Alphanumeric References

Most Alphanumeric References are composed of a single line with an alphanumeric code. However, some references have multiple lines representing two or more parts of the alphanumeric code. Tesseract is designed to read full pages or blocks of text making it suitable to read these multiline references without a problem. However, some of the multiline Alphanumeric References are displayed with the lines too close to each other, often without a single white line to separate them, which causes Tesseract to interpret the multiple lines as a single line of text and mismatching many characters. Therefore there is a need to separate the lines before sending them to Tesseract for recognition. Figure 4.22b shows the character boxes produced by Tesseract when recognizing the text of Figure 4.22a and Figure 4.22c shows the resulting text recognition.

(a) Multiline Alphanumeric Reference.



(b) Multiline Alphanumeric Reference with Tesseract's bounding boxes connecting the characters of different lines.



(c) Text recognized by Tesseract on Multiline Alphanumeric Reference of Figure 4.22a.

Figure 4.22: Multiline Alphanumeric Reference with text lines too close.

Despite not having a single white line separating the Alphanumeric Reference lines, it is possible to notice a difference in intensity when the characters of different lines touch, caused by blending. By analysing the intensity of the image using horizontal scan lines it would be possible to detect the pixel where the lines touch. However, these conditions do not apply to all the multiline Alphanumeric References. In some cases, the pixel where the text lines touch has the same intensity as the rest of the characters, in others, due to variation of the thickness of the characters, the intensity of the characters is not uniform which would cause lines in the middle of the text to be detected as the separation. A more dynamic approach was then considered. As the height of the characters of the Alphanumeric References is uniform in a blueprint, statistical analysis of this height is used to deduce the number of lines of a given reference. By calculating the mean and standard deviation of the height of all the potential Alphanumeric References, step ranges corresponding to the number of lines are defined. This allowed to correctly separate the lines in most of the cases and recognize them individually with Tesseract.

4.2.7 Implementation Details

Welding blueprints often have very large dimensions. Deep Learning techniques such as the analysed EAST and CRAFT utilize increasing amounts of Random-Access Memory(RAM) as the size of the image to process increases. To be able to apply the developed system to images of any size, a pre-processing step was designed to split the original image into a grid of sections that can be supplied to EAST or CRAFT without very high RAM requirements. This step takes a reasonable predefined section size which must be guaranteed to operate within the available RAM of the system. To detect Alphanumeric References that may not be fully contained inside any of the defined sections, such as a reference that is split by two neighbouring sections, a maximum reference size is also provided. This value defines the size of side sections that are placed between each of the main sections. The size of these side sections is enough to fit an Alphanumeric Reference, according to the provided maximum reference size. Finally, square sections are also created on the corners of the main sections to detect Alphanumeric References that intersect the corner where four main sections meet. Figure 4.23 shows a section of a blueprint with side sections in green on the sides and corner sections in red on the corners. If the original image size is not a multiple of the provided main section size the remaining space is used to form sections as well. In case the size of these remaining sections is not large enough to accommodate an Alphanumeric Reference in accordance with the maximum reference size provided, then these sections are extended to have at least the maximum reference size. This ensures that all Alphanumeric References in a blueprint are fully contained by at least one section to then be detected and recognized.

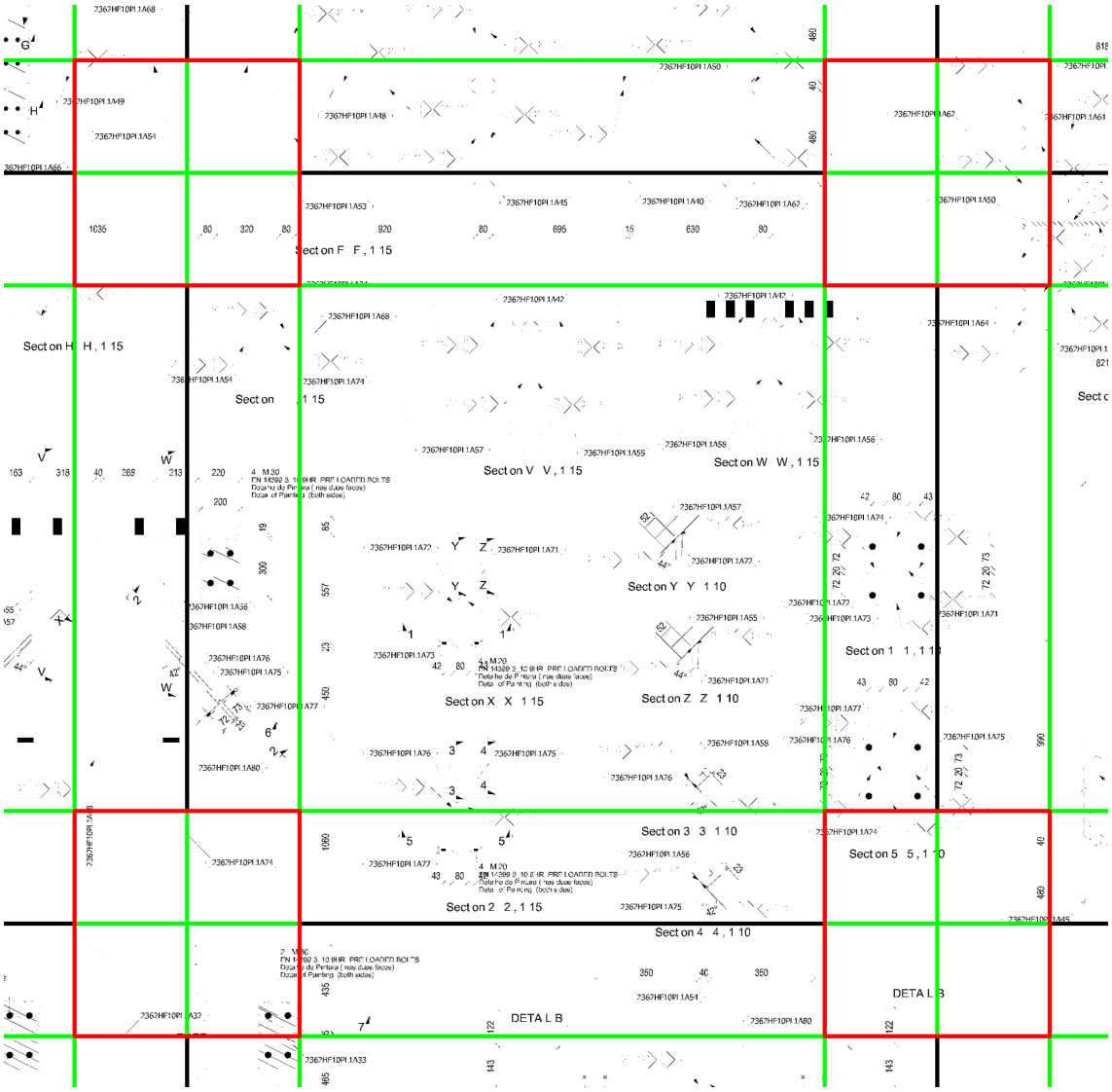


Figure 4.23: Sections of a Blueprint.

4.3 Welding Symbol Detection

The training of a deep learning model for Welding Symbol Detection and classification would require a substantially larger and more varied dataset than the one available. Due to the simplicity of the Welding Symbols, and the similarities between them, a robust feature matcher is used to ensure the selection of the correct matches. This similarity will also cause multiple symbol classes to be considered valid candidates for the same region of the blueprint, which brings the necessity of using various metrics to make the correct candidate selection. Like with Alphanumeric Reference detection and recognition, traditional computer vision techniques will be used to pre-process the input and refine the output.

Flowchart of the proposed solution for Welding Symbol Detection is illustrated in Figure 4.24. Feature detectors and descriptors are used on the blueprint and on Welding Symbol templates

which are then matched inside a sliding window using a robust feature matcher. Finally, as many candidates for the same sliding window emerge, a candidate selection is made based on a variety of metrics.

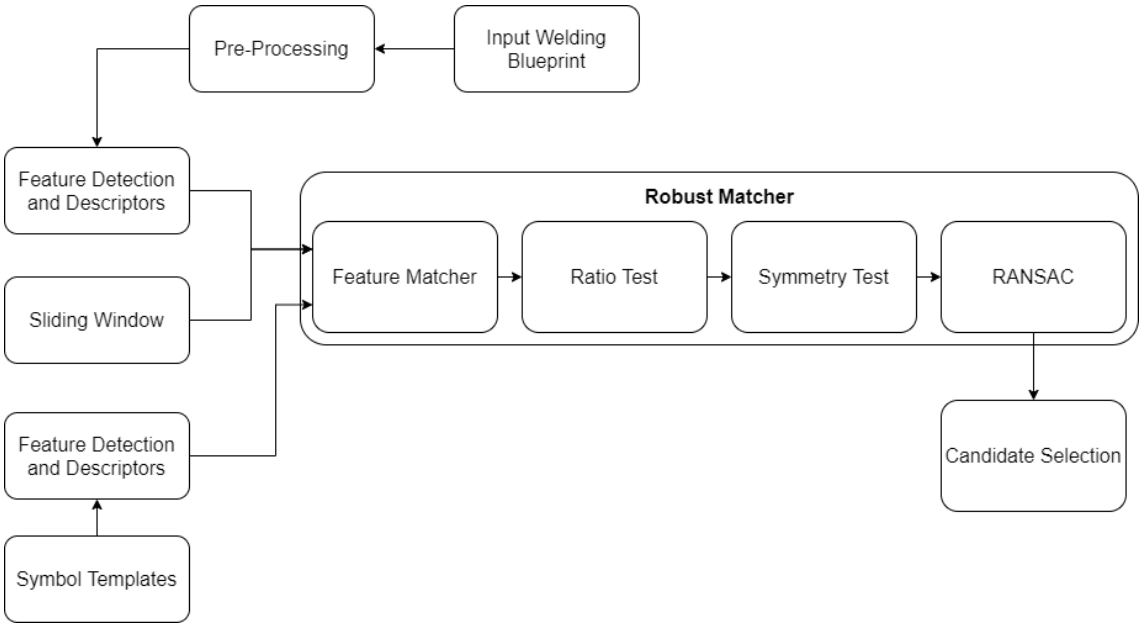


Figure 4.24: Welding Symbol Detection Flowchart.

4.3.1 Pre-processing

As with Alphanumeric Reference Detection and Recognition, pre-processing techniques are used to facilitate the core functioning of the solution. Aiming to use feature matching, it is important to remove noise elements around the Welding Symbols, specifically the characters around them, which can create mismatches or influence the descriptor of correct matches. For this, connected components analysis is used to remove elements with an area lower than welding symbols, which will include the characters around it. As the pixel area of the symbols and of the individual characters are very different, a threshold value is picked from a sample blueprint. Figure 4.25 shows a Welding Symbol of class 1 with textual elements close to it which are removed by this step.

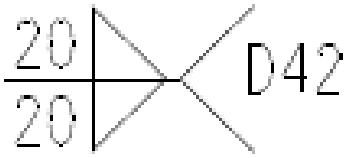
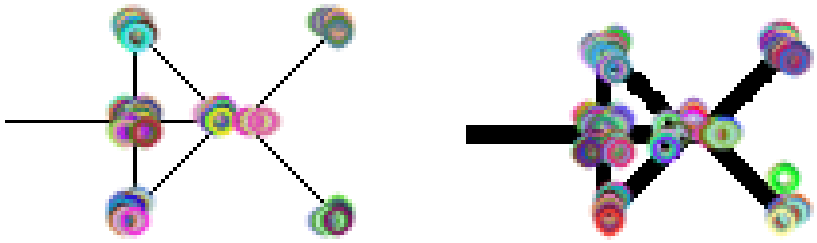


Figure 4.25: Welding Symbol of class 1 with text around it.

4.3.1.1 Welding Symbol Thickness

Welding Symbols of the same class are not always displayed with the same thickness. This difference between the template symbol and the symbols in the blueprint can result in features being detected at different locations, in a different quantity, which hinders the matching process. Figure 4.26 shows the result of Harris corner detection on a regular thin symbol and on an untreated thick symbol.



(a) Harris corners in regular Welding Symbol of class 1.

(b) Harris corners in thick Welding Symbol of class 1.

Figure 4.26: Comparison of Harris corner detection in regular symbol of class 1 and a thicker version.

Figure 4.27 shows the matches between an instance of class 1 with 1-pixel thickness with an instance of the same class with higher thickness. The number of features detected for the 1-pixel thickness instance is 136, however, only 47 matches are produced matching with the thicker version which detects 154 features. Metrics that analyse the absolute or relative number of matches would consider this matching unlikely to be correct.

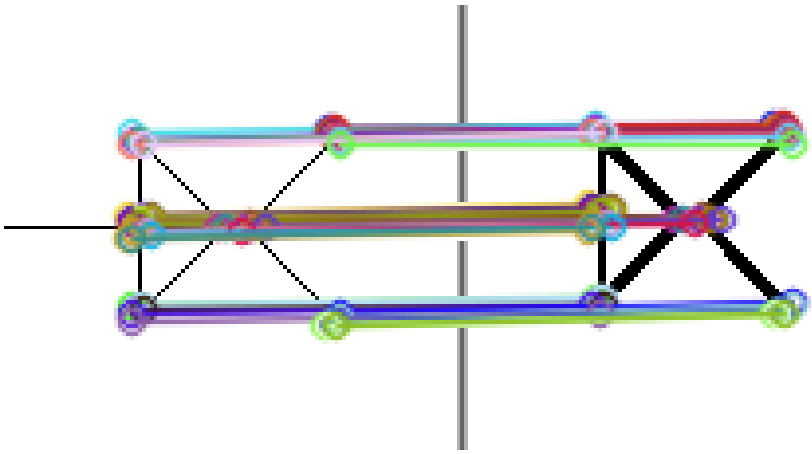
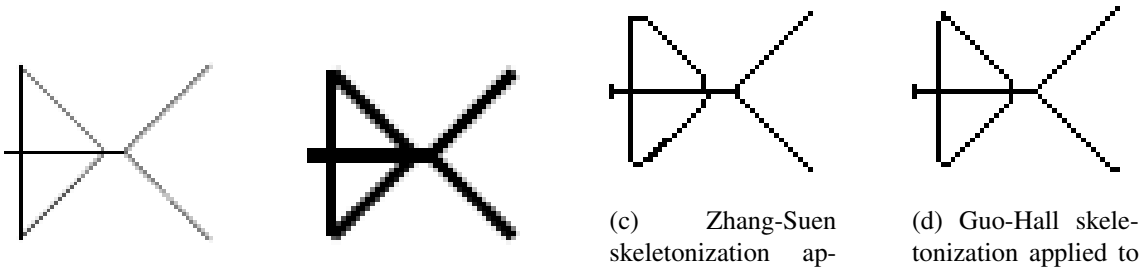


Figure 4.27: Matches between regular and thick Welding Symbol of class 1.

To uniformize the Welding Symbols with the same thickness, a skeletonization process is applied using the Guo-Hall algorithm [65]. The Zhang-Suen algorithm [66] was also experimented with but produced worse results. Figure 4.28a shows an instance of a symbol of class 1 with a stroke thickness of 1, Figure 4.28b shows an instance of the same class of symbol but with higher stroke thickness. Figures 4.28c and 4.28d show the Zhang-Suen algorithm and the Guo-Hall algorithm, respectively, applied to Figure 4.28b. As can be observed, the Zhang-Suen algorithm rounds the acute angles of the symbol and in certain areas produces a thickness higher than 1, creating characteristics that may be identified by feature detectors. The Guo-Hall algorithm correctly reduces the thickness of the Alphanumeric Reference in Figure 4.28b to 1 while retaining a very similar shape to the sample symbol of thickness 1 shown in Figure 4.28a.



(a) Regular Welding Symbol of class 1.

(b) Thick instance of class 1.

(c) Zhang-Suen skeletonization applied to the instance of Figure 4.28b.

(d) Guo-Hall skeletonization applied to the instance of Figure 4.28b.

Figure 4.28: Transforming thick variant of class 1 into regular instance.

Figure 4.29 shows Harris corner detection on Figure 4.28d. The position, number and distribution of Keypoints now more closely match the ones obtained for an instance of the symbol originally with a thickness of 1, such as in Figure 4.26a.

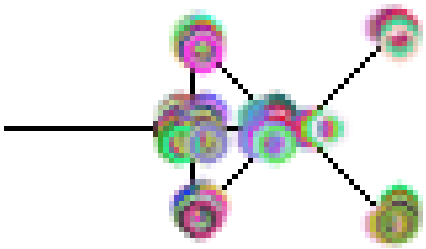


Figure 4.29: Harris features in thick Welding Symbol after Guo-Hall skeletonization.

Figure 4.30 illustrates the matches between an instance of class 1 with 1-pixel thickness and an instance of the same class with a higher thickness that was skeletonized using the Guo-Hall algorithm. The number of matches is now 97, doubling the previous 47 without skeletonization. This allows metrics that analyse the absolute or relative number of matches to have more confidence in the correct Welding Symbol candidate.

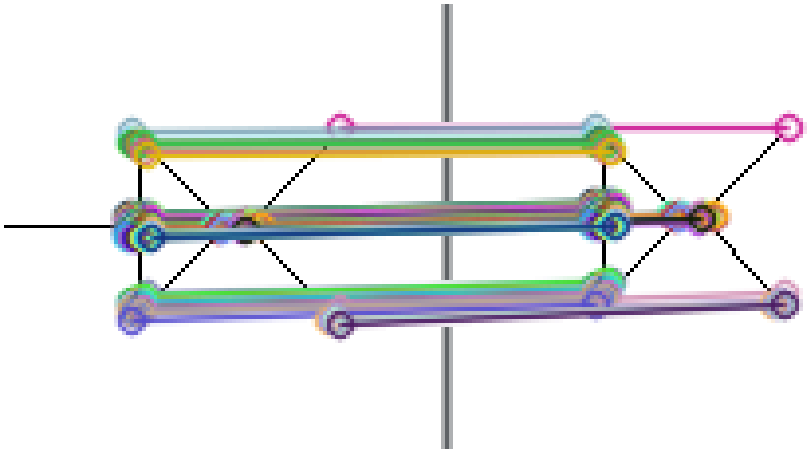


Figure 4.30: Matches between regular and skeletonized Welding Symbol of class 1.

4.3.2 Feature Detection

Several feature detectors were experimented with, aiming to facilitate correct candidate selection. Different feature detectors aim to detect different types of features.

- Harris detects line ends and line intersections (as corners)
- Features from accelerated segment test (FAST) detects angled strokes and line ends
- AKAZE detects line ends, line intersections and corners
- ORB uses a modified version of FAST that detects primarily angled strokes
- Adaptive and Generic Accelerated Segment Test (AGAST) detects line ends and certain angled intersections

Figure 4.31 shows the features detected by the methods enunciated, on a Welding Symbol of class 1.

A modified version of the Harris corner detector, referred to as Harris Best, was also experimented in an effort to bring more relevance to quantified clusters of features over the saturation of features in certain areas of the Welding Symbols. This modified version takes the points obtained from regular Harris corner detection, clusters them together and in every cluster only keeps the best N Harris points. The number of Harris points kept in each cluster, N, must be smaller than the number of points at any cluster of any Welding Symbol, while also sufficient to ensure that

Figure 4.31: Harris, FAST, AKAZE, ORB and AGAST features detected on Welding Symbol 1.

the Robust Feature Matcher has enough points to perform RANSAC. Under the assumption that the best Harris points of each cluster best represent the most relevant features of each cluster, this modified method could reduce the impact of the number of matches on candidate selection and bring more value to dispersed matches.

4.3.3 Feature Descriptor

Welding Symbols in blueprints are always oriented horizontally but may be displayed with various different scales. The descriptor for feature matching is selected taking these conditions into consideration - being invariant to scale but not to rotation. Respecting these requirements, the BRIEF descriptor was selected.

4.3.4 Robust Feature Matcher

A robust feature matcher [42] was implemented to match the template and the candidate window. This method matches each feature of the template with the two closest features on the candidate window, that is, with the smallest distance measurement in the descriptor. Lowe's ratio test [41] is then applied to discard matches that are not sufficiently different in distance. This process is applied by matching the candidate window to the template and also in the reverse, matching the template to the candidate window. This allows to then apply a symmetry test, only accepting match pairs when their points are the best matching feature of the other point of the pair. Finally, RANSAC is used as a strategy to produce a homography of the bounding box of the Welding Symbol while removing outlier matches.

As analysed in Chapter 3, while the shape of the same symbol class in different instances remains the same, the space between certain elements of the symbol varies between the width of one or two characters. For this, it is important to consider a sufficient allowed reprojection error when applying RANSAC. The symbols in the blueprints have sufficient separation between them to accommodate a large reprojection error without starting to mix feature matches of multiple instances. The size of the sliding window also limits how many symbols are captured at any given slide. Taking this into account, a reprojection error of approximately the width of three characters is used.

4.3.5 Sliding Window

A sliding window is passed through the blueprint to match each window with the various Welding Symbol classes templates. The sliding window must be large enough to encompass instances of the symbol class with a larger scale than the template. Chapter 3 analyses the scale difference between instances of the same symbol class in different blueprints. The size of the sliding window considered is two times larger than the template size.

The step of the sliding window must also be small enough to ensure that every instance of a Welding Symbol in the blueprint is fully contained by at least one window. A step of one pixel ensures this, under the penalty of a very large number of computations. As the size of the sliding window used is surplus to requirements, a sliding window step of eight is used to keep runtime lower.

4.3.6 Candidate Selection

Multiple matches with different Welding Symbol classes are obtained from a single sliding window. It is then required to select the correct candidate. Several metrics were explored to make the right candidate selection.

Quantifying the matches between two Welding Symbol candidates can provide information about which may be the correct one. They can be compared by the absolute or relative number of matches as well as by the minimization of wrong matches, by picking the candidate with the least RANSAC outliers.

The following Match Metrics are considered:

- Matches: select the candidate with the highest number of matches
- Match Ratio: select the candidate with the highest ratio between the number of matches and the number of features of the template
- Minimize Outliers: select the candidate with the least amount of outlier matches after RANSAC.

Certain feature detectors, such as the Harris corner detectors, accumulate a lot of Keypoints in the same region of the image as can be observed in Figure 4.31. Different instances of the same Welding Symbol class may produce very different amounts of Keypoints at these regions of concentrated features. This can throw off metrics that analyse the number of matches.

To minimize the impact of the number of features matched in the same region, metrics that analyse the number of clusters from matched features were experimented with. Features are clustered together using Density-Based Spatial Clustering of Applications with Noise (DBSCAN).

The epsilon value, the maximum distance between two samples to be considered in the neighbourhood, is estimated from the Welding Symbol templates. These templates are generally equal or smaller than the Welding Symbol instances on the various blueprints of the dataset. This should ensure the epsilon value can deal with scaled-up versions of the template symbols.

Generally, it is recommended that the minimum number of samples to form a cluster should be 3 or more when expecting the data to contain noise points. In our problem, different feature detectors produce different amounts of features at certain regions, to ensure the clustering works even for those that produce a smaller amount or more isolated features, the minimum samples to consider a cluster is set to 1.

Certain Welding Symbols with a high density of feature inducing characteristics (edges, corners, blobs or ridges) produce clusters that cover a large area of the symbol. Instances of such Welding Symbols with minor changes to their shape may lead to these large clusters being separated into multiple smaller ones which hinders the performance of cluster metrics.

The following Cluster Metrics are considered:

- Clusters: select the candidate with the highest number of match clusters
- Cluster Ratio: select the candidate with the highest ratio between the number of template feature clusters and the number of match clusters

To evaluate candidates based on the dispersion of their matches, metrics that evaluate the shape of homography with RANSAC were implemented. These consider that the quality of the homography may be associated with a correct candidate selection. As a perfect resulting homography would be represented by a rectangle in our problem, resulting from translation and scaling transformations, the metrics make comparisons with its bounding box and the angles at the vertices.

The following Shape Metrics are considered:

- Rectangle: select the candidate which produces a box closest to a rectangle, with the lowest ratio between the area of the shape and the area of its bounding box.
- Perimeter: select the candidate which produces a box closest to a rectangle, with the lowest ratio between the perimeter of the shape and the area of its bounding box.
- Angles: select the candidate whose homography angles are closest to 90 degrees.
- Standard Deviation: select the candidate with the least difference between the Keypoint coordinates standard deviation of the match and of the template.
- Range: select the candidate with the least difference between the range of Keypoint coordinates of the match and of the template.

Combinations of these metrics were experimented with, based on their confidence value. Considering the cases the metric is bounded from 0% to 100% - Match Ratio, Cluster Ratio and Rectangle metrics - this value can be interpreted as the confidence of the metric that the candidate is correct. Metrics can then be combined for different ranges of confidence, for example, a low confidence value of Relative Clusters may indicate another metric is better suited.

4.4 Alphanumeric Reference Navigation - Proof of Concept Application

As mentioned in Section 1.2, a proof of concept application for multi-blueprint Alphanumeric Reference navigation was also developed. This application showcases the benefits of digitalizing the blueprints, allowing the user to identify Alphanumeric References that are used in multiple blueprints and navigate between them. The user can navigate the blueprint, panning and zooming in and out, and see, identified in red, the Alphanumeric References that appear in multiple blueprints. Clicking on a red Alphanumeric Reference changes its colour to green and displays information about the selected Alphanumeric Reference, such as the text identified and the blueprints it is present in. Additionally, all other instances of the same Alphanumeric Reference also have their colour changed to green. Figure 4.32 shows blueprint 2 with an Alphanumeric Reference selected in green, along with another instance of the same Alphanumeric Reference also in green. Figure 4.33 shows blueprint 3 with another instance of the same Alphanumeric Reference highlighted in green.

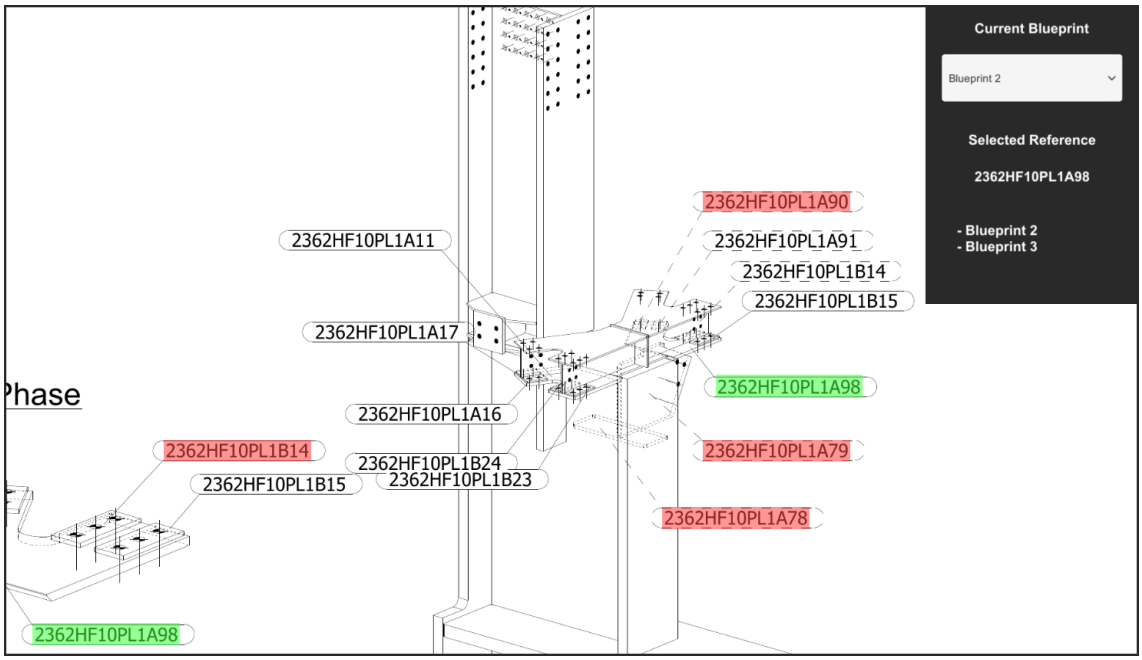


Figure 4.32: Multi-blueprint Alphanumeric Reference selected in green.

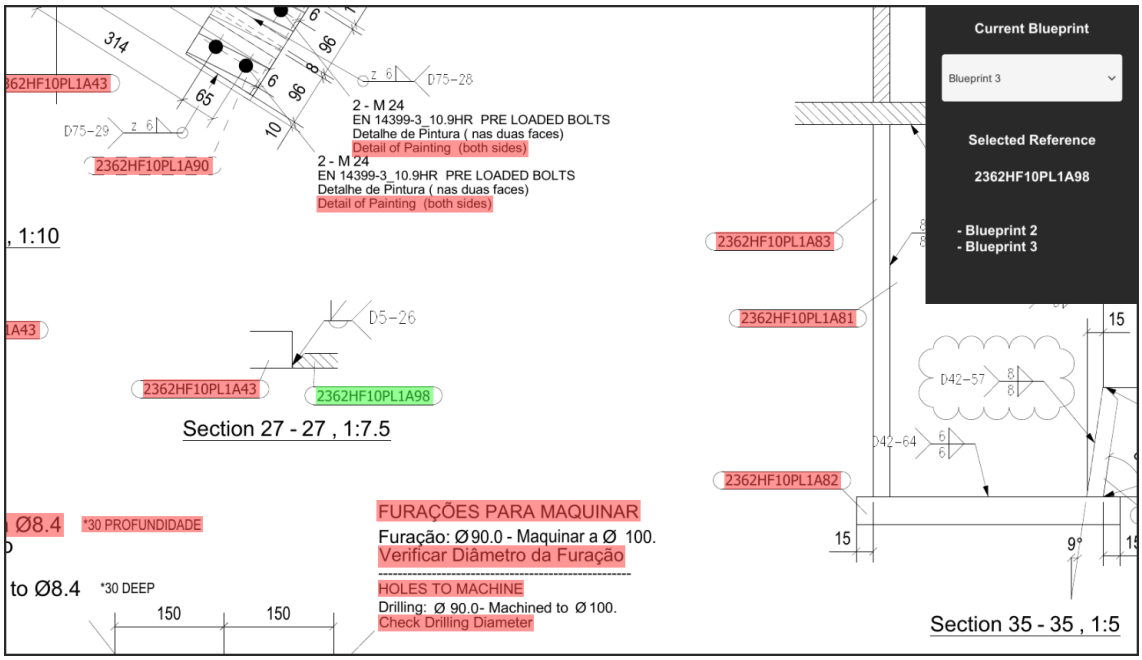


Figure 4.33: Multi-blueprint Alphanumeric Reference selected in green also present in Blueprint 3

Chapter 5

Results

Tests on individual blueprints and on the entire dataset were performed to evaluate the accuracy of the proposed solutions for: (i) Alphanumeric Reference Detection and Recognition; (ii) Welding Symbol Detection.

5.1 Alphanumeric Reference Detection and Recognition

The results of the proposed solution on blueprint 1 with and without various cumulative enhancements are shown in Table 5.1. A direct application of text detection and recognition resulted in 78% accuracy on blueprint 1. Angled Alphanumeric References rotated to the horizontal improved the accuracy to 82%, Alphanumeric Reference Frame removal improved to 85%, Tesseract recommendations improved the results to 98.33% and Text Recognition refinements improved the final accuracy to 98.75%.

Direct	+ Rotated Alphanumeric References	+ Alphanumeric Reference Frame Removal	+ Tesseract Recommendations	+ Text Recognition Refinements
78%	82%	85%	98%	98.75%

Table 5.1: Accuracy with various enhancements on blueprint 1.

While in blueprint 1 the aforementioned text recognition refinements did not have a significant role in improving the results, this was the case for blueprints that were impacted by character mismatches and multi-line Alphanumeric References with overly close lines. Notably, blueprint 12, which had an accuracy of only 5.17%, improved to 93.10% after text recognition refinements as is shown in Table 5.2.

Table 5.3 shows the results of the proposed solution on the various blueprints and on the entire dataset as well as the weight of each blueprint on the dataset, the relative number of Alphanumeric References it has in relation to the full dataset.

All enhancements except Text Recognition Refinement	+ O/0 Analysis	+ Multi line Analysis
5.17%	81.03%	93.10%

Table 5.2: Accuracy with Text Recognition Refinements on blueprint 12.

Blueprint	Percentage of total Dataset	Accuracy
1	8.30%	98.75%
2	4.18%	99.17%
3	6.98%	94.55%
4	2.77%	96.25%
5	6.19%	88.27%
6	7.20%	94.93%
7	16.56%	89.77%
8	2.77%	81.25%
9	2.07%	100%
10	11.48%	99.10%
11	4.56%	87.12%
12	2.00%	93.10%
13	7.50%	96.31%
14	1.14%	100%
15	3.63%	95.24%
16	2.73%	94.94%
17	9.65%	97.52%
Total	-	94.50%

Table 5.3: Accuracy of Alphanumeric Reference Detection and Recognition solution on the full dataset.

5.2 Welding Symbol Detection

The accuracy of the solution with combinations of various candidate selection metrics and feature detectors on blueprint 1 are shown in Tables 5.4, 5.5 and 5.6. Blueprint 1 was selected for its large amount of Welding Symbol classes most prevalent in the dataset, such as classes 1, 3, 4 and 7.

Harris, FAST and Harris Best obtained the best results by selecting the candidate with the most matches while AKAZE, AGAST and ORB had higher accuracy considering the highest Match Ratio. This may be explained by the different amount of features detected by these two groups of feature detectors, with Harris and FAST detecting large amounts of keypoints and AKAZE, AGAST and ORB detecting fewer, more specific features, as was discussed in Section 4.3.2. The Outliers metric, which selects the candidate with the least amount of RANSAC outliers, did not have good results with any feature detector, this may be due to a low number of outliers produced in each match, making it an undistinctive characteristic of different candidate matches.

		Match Metrics		
		Matches	Match Ratio	Outliers
Feature Detector	Harris	88.44%	65.99%	44.90%
	FAST	**95.24%**	68.71%	6.12%
	AKAZE	82.31%	90.48%	25.17%
	AGAST	46.26%	65.31%	20.41%
	ORB	70.75%	85.03%	53.06%
	Harris Best	92.92%	70.07%	12.93%

Table 5.4: Accuracy on blueprint 1 using various Feature Detectors with Match Metrics.

Cluster metrics generally obtained better results picking the candidate with the highest cluster ratio than simply picking the one with the highest number of clusters. The FAST detector notably had very low accuracy for both cluster metrics, this is likely due to the close proximity of the keypoints produced by it which run through the Welding Symbol and form a single cluster.

		Cluster Metrics	
		Clusters	Cluster Ratio
Feature Detector	Harris	74.15%	81.63%
	FAST	2.04%	8.84%
	AKAZE	74.15%	89.12%
	AGAST	40.14%	47.62%
	ORB	40.82%	1.36%
	Harris Best	37.41%	61.90%

Table 5.5: Accuracy on blueprint 1 using various Feature Detectors with Cluster Metrics.

Shape metrics, despite using different approaches, failed to obtain valuable accuracy results. Metrics that try to quantify the quality of the homography - Rectangle, Perimeter and Angles metrics - obtained generally similar results with low accuracy. This may be an indication that close candidates for the same sliding window produce homographies with similar quality. The metrics that try to evaluate the spread of the matches - Standard Deviation and Range metrics - obtained slightly better results but still inferior to other types of metrics previously analysed. The range metric specifically may be susceptible to the variability of space between Welding Symbol elements discussed in Chapter 3 and illustrated in Figure 3.2.

		Shape Metrics				
		Rectangle	Perimeter	Angles	Standard Deviation	Range
Feature Detector	Harris	56.46%	52.38%	57.82%	54.42%	69.39%
	FAST	22.45%	6.12%	20.41%	47.62%	34.01%
	AKAZE	42.18%	12.93%	46.26%	57.14%	44.90%
	AGAST	39.46%	22.45%	40.14%	51.70%	33.33%
	ORB	60.54%	16.33%	61.22%	45.58%	44.22%
	Harris Best	6.80%	4.76%	6.12%	29.25%	43.54%

Table 5.6: Accuracy on blueprint 1 using various Feature Detectors with Shape Metrics.

Custom metrics that try to perform decisions based on multiple metrics confidence level, specifically those that are constrained to a range - the Match Ratio, Cluster Ratio and Rectangle metrics that range from 0% to 100% - were also considered, as elaborated in Section 4.3.6. However, these ranged metrics were found to have a high confidence level even when they picked the wrong candidate. This limits any decision logic to be made and ultimately lead to no valuable custom metric being found. These custom metrics that function as decision trees could perform better if their decision nodes analysed more specific characteristics or elements of a candidate instead of the candidate as a whole.

With the various metrics and feature detectors analysed, the selection fell on using the FAST feature detector with the BRIEF feature descriptor for feature matching and making the candidate selection using the Matches metric, selecting the candidate with the most matches.

Table 5.7 shows the results of the proposed solution on the various blueprints and on the entire dataset as well as the weight of each blueprint on the dataset, the relative number of Welding Symbols it has in relation to the full dataset.

Blueprint	Percentage of total Dataset	Accuracy
1	12.53%	95.24%
3	10.49%	93.50%
4	4.60%	51.85%
5	18.33%	81.86%
6	10.14%	70.59%
7	16.28%	73.82%
8	3.15%	89.19%
10	20.72%	65.02%
12	3.75%	93.18%
Total	-	78.09%

Table 5.7: Accuracy of Welding Symbol detection solution on the full dataset.

Table 5.8 shows the accuracy of the solution for each Welding Symbol class. As discussed in Chapter 3, the bulk of the dataset is concentrated in 4 Welding Symbol classes - 1, 3, 4 and 7 - which represent 86.45% of its total. Pairs 1 and 7 and 3 and 4 of these symbol classes have also been found to have many visual similarities, with the shape of one class being a part of the other class shape. With this, a large part of the accuracy of the system amounts to its ability to correctly distinguish between these pairs.

Figure 5.1 shows the relative confusion matrix for Welding Symbol detection of the proposed solution as a heatmap, with darker green representing higher values. This confusion matrix evidences the aforementioned mismatches between certain symbol classes such as 1 and 7, 4 and 3 or 18 and 5.

Welding Symbol	Instances	Matches	Accuracy
1	400	377	94.25%
2	2	2	100%
3	333	318	95.50%
4	149	81	54.36%
5	34	1	2.94%
6	4	2	50.00%
7	132	66	50.00%
8	1	1	100.00%
9	6	1	16.67%
10	2	2	100.00%
11	38	12	31.58%
12	10	10	100.00%
13	11	7	63.64%
14	13	4	30.77%
15	1	1	100.00%
16	2	2	100.00%
17	2	2	100.00%
18	8	2	25.00%
19	2	1	50.00%
20	3	1	33.33%
21	3	3	100.00%
22	5	4	80.00%
23	7	6	85.71%
24	3	3	100.00%
Total	1173	916	78.09%

Table 5.8: Accuracy for each Welding Symbol class in the full dataset.

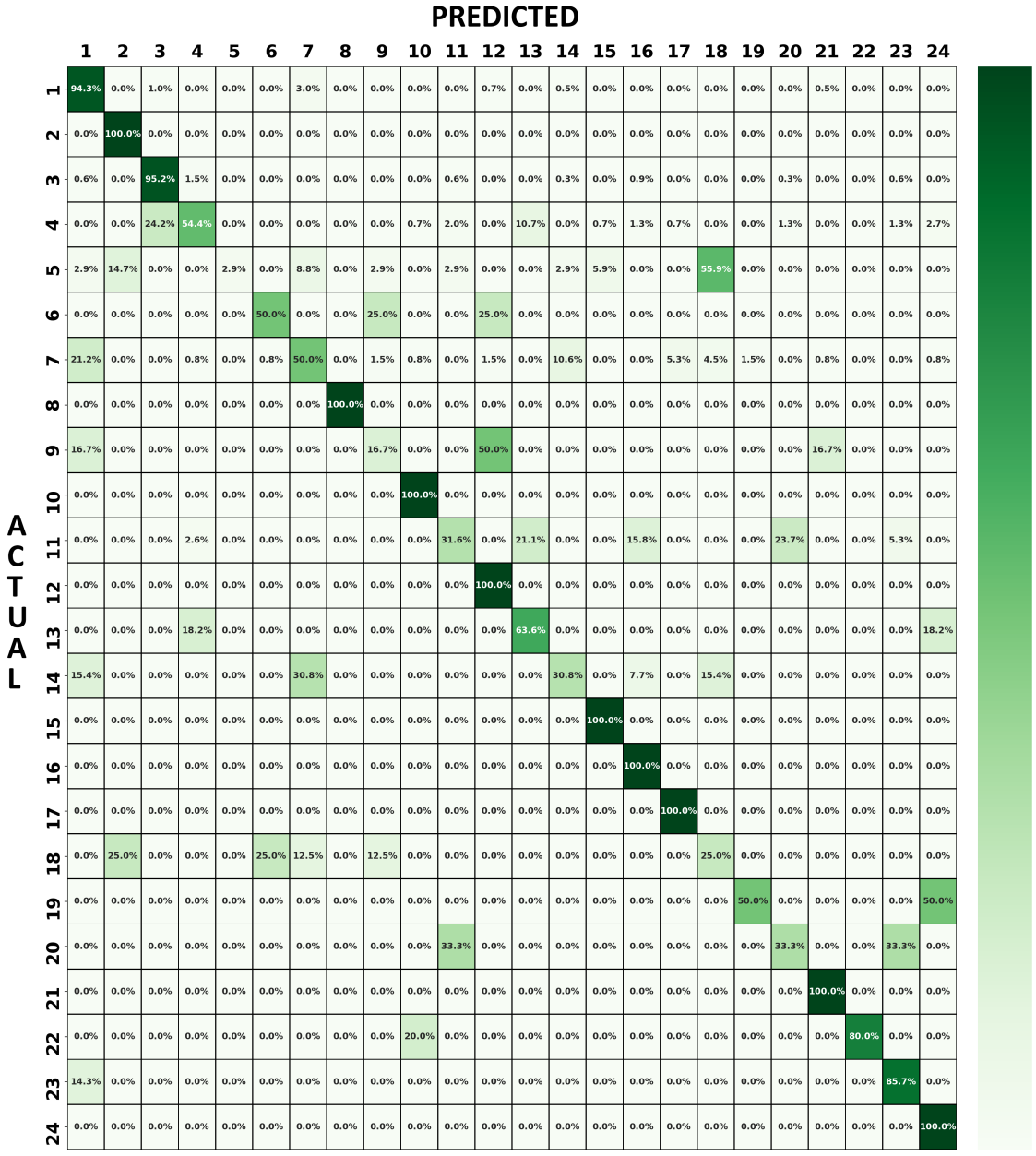


Figure 5.1: Confusion matrix.

Chapter 6

Conclusions and Future Work

This dissertation proposes a solution for the problems of: (i) Alphanumeric Reference Detection and Recognition; (ii) Welding Symbol Detection. These problems bring out the necessity to print large welding blueprints and have welding operators manually register their progress and observations in the physical copy of the blueprint. Additionally, operators are required to manually read, interpret and search the blueprints, an error-prone and repetitive process. Solving these problems means eliminating this necessity, reducing costs, improving safety and simplifying work.

As a real-world application with safety concerns, it is important that both solutions offer a high level of accuracy, in the case of Alphanumeric References that indicate the material and shape used and in the case of Welding Symbols that indicate the type of weld to perform.

6.1 Conclusions

The proposed solution for Alphanumeric Reference Detection and Recognition, combining deep learning tools for text detection and for text recognition, aided by pre-processing techniques and post-processing refinements, obtained good results for the test dataset with an accuracy of 94.50%. The studied dataset covers a good variety of scenarios and conditions in which Alphanumeric References are inserted in. The distribution of characters in the dataset is unbalanced, with low amounts of some and the complete absence of others. While text recognition is handled by a proven tool such as Tesseract, the introduction of certain missing characters in the dataset can cause problems, like the one encountered for the zero and capital letter "O" mismatch, which had to be handled manually.

The proposed solution for Welding Symbol Detection, utilizing traditional computer vision techniques of feature extraction and matching, aided by pre-processing techniques, obtained an accuracy of 78.09% on the test dataset, a fairly good result considering the methods used but not sufficient for a real-world application, especially considering the safety concerns. The dataset contains a large variety of symbol classes but most of them have a very small number of instances with the bulk of the dataset concentrated in only four classes. These four classes also happen to form two of the identified pairs of symbol classes where one of the elements is a sub-shape of

the other. With the solution aiming to obtain the best accuracy under this dataset, naturally, the solution is also shaped by it, meaning that the choices made are largely influenced by these four classes and their relation and not as generically for any symbol class which had a lower impact on the final accuracy. Additionally, an approach using feature matching and candidate selection depends on the Welding Symbol classes that are being considered, adding new ones will make the candidate selection process more difficult and possibly reduce the accuracy of the system.

6.2 Future Work

Future work in Alphanumeric Reference Detection and Recognition could go over the test of a dataset that includes the absent characters. This could bring up more cases of frequent character mismatches which would need to be handled individually. With a larger dataset, it may also be worth considering training a text detector with the characteristics of our dataset, which brought difficulties to the pre-trained models, which were handled with traditional computer vision techniques. While these techniques properly alleviated these difficulties, they are susceptible to failure under new conditions not represented in the current dataset.

Deep learning techniques should be tested for symbol detection. This will however require a larger and more balanced dataset of Welding Symbols, aided by data augmentation techniques which may be able to easily generate new data considering the simplicity of the symbols. Further work on the proposed solution with feature matching can try to explore better ways for candidate selection, combining new metrics that analyse specific characteristics of the Welding Symbols instead of the entire symbol. This, however, with the drawback that adding more symbol classes will only further complicate any candidate selection decision-making.

References

[1] M. Dinham and G. Fang. Weld seam detection using computer vision for robotic arc welding. In *2012 IEEE International Conference on Automation Science and Engineering (CASE)*, pages 771–776, 2012.

[2] Luciane Soares, Átila Weis, Bruna de Vargas Guterres, Ricardo Rodrigues, and Silvia Botelho. Computer vision system for weld bead geometric analysis. pages 292–299, 04 2018.

[3] Pablo Salazar, Gino Torres, Manuel Olivares, and Pedro Sariego. A vision based system for industrial robotic welding path correction. 11 2010.

[4] Yue Liu, Xiaohong Li, Dahai Ren, Shenghua Ye, Bengang Wang, and Jie Sun. Computer vision application for weld defect detection and evaluation. In Shenghua Ye, editor, *Automated Optical Inspection for Industry: Theory, Technology, and Applications II*, volume 3558, pages 354 – 357. International Society for Optics and Photonics, SPIE, 1998.

[5] Chris Tensmeyer and Tony Martinez. Historical document image binarization: A review. *SN Computer Science*, 1, 05 2020.

[6] Nobuyuki Otsu. A threshold selection method from gray-level histograms. *IEEE Transactions on Systems, Man, and Cybernetics*, 9(1):62–66, 1979.

[7] Wikimedia Commons. File:pavlovsk railing of bridge yellow palace winter.jpg — wikimedia commons, the free media repository, 2021. [Online; accessed 9-February-2021].

[8] D.H. Ballard. Generalizing the hough transform to detect arbitrary shapes. *Pattern Recognition*, 13(2):111–122, 1981.

[9] Wikimedia Commons. File:hough-example-result-en.png — wikimedia commons, the free media repository, 2020. [Online; accessed 9-February-2021].

[10] H. Samet and M. Tamminen. Efficient component labeling of images of arbitrary dimension represented by linear bintrees. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 10(4):579–586, 1988.

[11] Wikimedia Commons. File:screenshot-pixel region (figure 3).png — wikimedia commons, the free media repository, 2020. [Online; accessed 23-June-2021].

[12] R. M. Haralick, S. R. Sternberg, and X. Zhuang. Image analysis using mathematical morphology. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-9(4):532–550, 1987.

[13] Wikimedia Commons. File:binary image erosion.png — wikimedia commons, the free media repository, 2020. [Online; accessed 9-February-2021].

[14] Wikimedia Commons. File:binary image dilation.png — wikimedia commons, the free media repository, 2020. [Online; accessed 9-February-2021].

[15] Kai Wang, Boris Babenko, and Serge Belongie. End-to-end scene text recognition. In *2011 International Conference on Computer Vision*, pages 1457–1464, 2011.

[16] Shangbang Long, Xin He, and Cong Yao. Scene text detection and recognition: The deep learning era. *International Journal of Computer Vision*, 129, 01 2021.

[17] Lifeng He, Xiwei Ren, Qihang Gao, Xiao Zhao, Bin Yao, and Yuyan Chao. The connected-component labeling problem: A review of state-of-the-art algorithms. *Pattern Recognition*, 70, 04 2017.

[18] B. Epshtein, E. Ofek, and Y. Wexler. Detecting text in natural scenes with stroke width transform. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 2963–2970, 2010.

[19] X. Zhou, C. Yao, H. Wen, Y. Wang, S. Zhou, W. He, and J. Liang. East: An efficient and accurate scene text detector. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2642–2651, 2017.

[20] Y. Baek, B. Lee, D. Han, S. Yun, and H. Lee. Character region awareness for text detection. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9357–9366, 2019.

[21] P. He, W. Huang, T. He, Q. Zhu, Y. Qiao, and X. Li. Single shot text detector with regional attention. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 3066–3074, 2017.

[22] Jingchao Liu, Xuebo Liu, Jie Sheng, Ding Liang, Xin Li, and Qingjie Liu. Pyramid mask text detector, 2019.

[23] Corinna Cortes and Vladimir Vapnik. Support-vector networks. In *Machine Learning*, pages 273–297, 1995.

[24] Cong Yao, Apratim Purakayastha, Baoguang Shi, and Wenyu Liu. Strokelets: A learned multi-scale representation for scene text recognition. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, page 4042–4049. IEEE, June 2014.

[25] Zhi Rong Tan, Shangxuan Tian, and Chew Lim Tan. Using pyramid of histogram of oriented gradients on natural scene text recognition. In *2014 IEEE International Conference on Image Processing (ICIP)*, pages 2629–2633, 2014.

[26] Shangxuan Tian, S. Lu, B. Su, and C. Tan. Scene text recognition using co-occurrence of histogram of oriented gradients. *2013 12th International Conference on Document Analysis and Recognition*, pages 912–916, 2013.

[27] S. Albawi, T. A. Mohammed, and S. Al-Zawi. Understanding of a convolutional neural network. In *2017 International Conference on Engineering and Technology (ICET)*, pages 1–6, 2017.

[28] V. Frinken, F. Zamora-Martínez, S. España-Boquera, M. J. Castro-Bleda, A. Fischer, and H. Bunke. Long-short term memory neural networks language modeling for handwriting recognition. In *Proceedings of the 21st International Conference on Pattern Recognition (ICPR2012)*, pages 701–704, 2012.

[29] H. Li, P. Wang, and C. Shen. Towards end-to-end text spotting with convolutional recurrent neural networks. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 5248–5256, 2017.

[30] Hui Li, Peng Wang, Chunhua Shen, and Guyu Zhang. Show, attend and read: A simple and strong baseline for irregular text recognition. *CoRR*, abs/1811.00751, 2018.

[31] B. R. Meijer. Rules and algorithms for the design of templates for template matching. In *[1992] Proceedings. 11th IAPR International Conference on Pattern Recognition*, pages 760–763, 1992.

[32] Shaul Oron, Tali Dekel, Tianfan Xue, William T. Freeman, and Shai Avidan. Best-buddies similarity - robust template matching using mutual nearest neighbors, 2016.

[33] I. Talmi, R. Mechrez, and L. Zelnik-Manor. Template matching with deformable diversity similarity. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1311–1319, 2017.

[34] D. G. Lowe. Object recognition from local scale-invariant features. In *Proceedings of the Seventh IEEE International Conference on Computer Vision*, volume 2, pages 1150–1157 vol.2, 1999.

[35] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. volume 3951, pages 404–417, 07 2006.

[36] Michael Calonder, Vincent Lepetit, Christoph Strecha, and Pascal Fua. Brief: Binary robust independent elementary features. volume 6314, pages 778–792, 09 2010.

[37] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski. Orb: An efficient alternative to sift or surf. In *2011 International Conference on Computer Vision*, pages 2564–2571, 2011.

[38] E. Rosten and T. Drummond. Fusing points and lines for high performance tracking. In *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*, volume 2, pages 1508–1515 Vol. 2, 2005.

[39] Ebrahim Karami, Siva Prasad, and Mohamed Shehata. Image matching using sift, surf, brief and orb: Performance comparison for distorted images. 11 2015.

[40] Wikimedia Commons. File:harris corners detected on chessboard.png — wikimedia commons, the free media repository, 2020. [Online; accessed 10-February-2021].

[41] David Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60:91–, 11 2004.

[42] Robert Laganière. *OpenCV 2 Computer Vision Application Programming Cookbook*, pages 233–246. Packt, (s)econd edition, 2014.

[43] Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton. Imagenet classification with deep convolutional neural networks. *Neural Information Processing Systems*, 25, 01 2012.

[44] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Region-based convolutional networks for accurate object detection and segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38:1–1, 12 2015.

[45] Ross Girshick. Fast r-cnn, 2015.

[46] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection, 2017.

[47] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 779–788, 2016.

[48] Joseph Redmon and Ali Farhadi. Yolo9000: Better, faster, stronger, 2016.

[49] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement, 2018.

[50] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. Yolov4: Optimal speed and accuracy of object detection, 2020.

[51] Zhi Zhang, Tong He, Hang Zhang, Zhongyue Zhang, Junyuan Xie, and Mu Li. Bag of freebies for training object detection neural networks, 2019.

[52] Zhuliang Yao, Yue Cao, Shuxin Zheng, Gao Huang, and Stephen Lin. Cross-iteration batch normalization, 2020.

[53] Wenhai Wang, Enze Xie, Xiaoge Song, Yuhang Zang, Wenjia Wang, Tong Lu, Gang Yu, and Chunhua Shen. Efficient and accurate arbitrary-shaped text detection with pixel aggregation network, 2020.

[54] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg. Ssd: Single shot multibox detector. *Lecture Notes in Computer Science*, page 21–37, 2016.

[55] T. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár. Focal loss for dense object detection. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2999–3007, 2017.

[56] Z. Zhao, P. Zheng, S. Xu, and X. Wu. Object detection with deep learning: A review. *IEEE Transactions on Neural Networks and Learning Systems*, 30(11):3212–3232, 2019.

[57] Jason Ravagli, Zahra Ziran, and Simone Marinai. Text recognition and classification in floor plan images. In *2019 International Conference on Document Analysis and Recognition Workshops (ICDARW)*, volume 1, pages 1–6, 2019.

[58] Laura Jamieson, Carlos Francisco Moreno-Garcia, and Eyad Elyan. Deep learning for text detection and recognition in complex engineering diagrams. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pages 1–7, 2020.

[59] Eyad Elyan, Laura Jamieson, and Adamu Ali-Gombe. Deep learning for symbols detection and classification in engineering drawings. *Neural Networks*, 129:91–102, 2020.

[60] Yuxi Zhang. Cnn-based symbol recognition and detection in piping drawings, Aug 2019.

[61] Welding and allied processes — symbolic representation on drawings — welded joints. Standard, International Organization for Standardization, Geneva, CH, March 2019.

[62] Wikimedia Commons. File:comparison of 1d and 2d interpolation.svg — wikimedia commons, the free media repository, 2020. [Online; accessed 26-June-2021].

[63] Tesseract recommendations, 2021. https://tesseract-ocr.github.io/tessdoc/ImproveQuality.html, Accessed on 22.06.2021.

[64] Willus Dotkom. Tesseract character height, 2018. https://groups.google.com/g/tesseract-ocr/c/Wdh_JJwnw94/m/24JHDYQbBQAJ, Accessed on 22.06.2021.

[65] Zicheng Guo and Richard W. Hall. Parallel thinning with two-subiteration algorithms. *Commun. ACM*, 32(3):359–373, March 1989.

[66] T. Y. Zhang and C. Y. Suen. A fast parallel algorithm for thinning digital patterns. *Commun. ACM*, 27(3):236–239, March 1984.