Tag recognition from panoramic scans of industrial facilities

Master of Science (Tech.) Thesis
University of Turku
Department of Computing
Software Engineering
June 2022
Emil Dahlberg

Supervisors:
Teijo Lehtonen
Mikko Yllikäinen

UNIVERSITY OF TURKU
Department of Computing

Emil Dahlberg: Tag recognition from panoramic scans of industrial facilities

Master of Science (Tech.) Thesis, 74 p.
Software Engineering
June 2022

CAD-based digital twins are commonly used by operators of process industry facilities to combine 3D models with external information and documentation. However, often a suitable model does not exist, and the plant operators must instead resort to laser scans with panoramic photos, which provide little to no metadata or information about their contents. Reading of equipment tags or other useful text from these scans could hugely increase their usefulness, as that information could be used to connect equipment to its documentation and other data. In this thesis, the feasibility of such extraction as a special case of deep learning text detection and recognition is studied.

This work contrasts practical requirements of industry with the theory and research behind text detection and recognition, with experiments conducted to confirm the feasibility of a potential application. It is found that the task is feasible from both business domain and deep learning perspectives. In practice, off-the-shelf text detection models generalize very well to the problem but integrating text recognition to build an end-to-end solution is found to require further work. End-to-end text recognition models appear promising in research, but rather uncommon in practical applications. Recent laser scans including color imagery are found suitable for the task and using them for recognition is found feasible; however, the usefulness of older scans remains unclear due to their poor quality. Deploying a successful practical solution is thus possible with modern scans but acquiring such scans may require collaboration with facility operators.

Keywords: text detection, text recognition, end-to-end text recognition, digital twin, deep learning

Contents

1 Introduction

Process industry companies operate and build facilities with a large amount of external maintenance, engineering, and operating information. Digital twins, which are effectively data-integrated 3D models of the facilities, are useful for these companies as a "one stop shop" that combines data from all their different systems. Cadmatic Oy provides such solutions to its customers along with its design tools.

However, for many older facilities, corresponding 3D models suitable for the digital twin purpose either were never made or are not available. This is especially common for older facilities built in the 20th century. To compensate, a common practice used to quickly create 3D data from an older pre-existing facility is by utilizing camera-equipped laser scanners, which produce panoramic images with depth information, which are then processed to point clouds, consisting of colored points in 3D space. However, compared to a 3D CAD model, point clouds do not include any useful domain information and metadata, such as the names and properties of pieces of equipment displayed in the scans.

With the evolution of deep learning in recent years, major advances have been made in the field of end-to-end text recognition. With its practical applications in computer vision, end-to-end text recognition has made automatically reading text from images depicting natural scenes feasible. Since scanning facilities produces data resembling 360-degree panorama images, it should be possible to combine these technologies - directly apply text detection and recognition solutions to the scans and gain information about equipment present in them.

Thus, the main purpose of this thesis is to study the feasibility of a machine learning solution that extracts labels and their positions from panoramic scans of industrial plants. The tasks of text detection and recognition are discussed separately: detection meaning finding text in an image, with text recognition meaning reading characters out of a tightly cropped image of text.

1.1 Thesis structure

This chapter is the introduction of the thesis. It is followed by Chapter 2, which sets the background and research questions: it explains the domain background of this work and seeks to provide a good overview of its goals and requirements - why the proposed solution is necessary and what special challenges are involved in reaching it. Parallel to this domain knowledge, Chapter 3 covers the basics of deep learning to give the reader a light overview of the inner workings of the algorithms described and utilized in this thesis. Chapter 4 then contains a literature survey of deep learning research with relevance to the task at hand. Chapter 5 combines the information of the previous chapters, providing an overview of how academic research relates to the problem domain, and builds a basic hypothetical architecture for the solution, with some further questions to be answered in later evaluation of methods. Chapter 6 brings some more insight to the problem with practical experiments, validating hypotheses from the previous chapter. Chapter 7 discusses these results and builds an overall view of the thesis as a whole, and lastly, Chapter 8 is the conclusion of the thesis.

2 Domain background

This chapter covers the starting point of the thesis: the background and reasoning as to why the subject of reading text from panoramic scans was chosen. Much of the background information covered in this chapter is not sourced from any academic research but is rather domain knowledge from personal findings at Cadmatic and interviews with employees involved with the development of Cadmatic software.

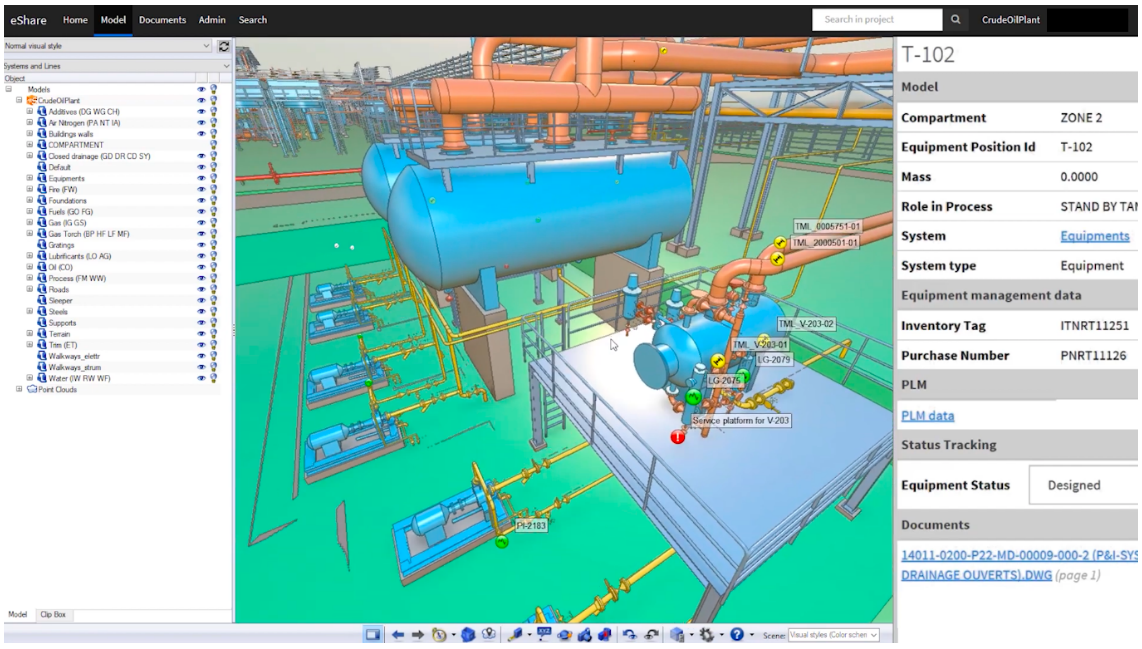
Digital twins, equipment labels, their usefulness and relation to laser scans are discussed in Section 2.1. This is closely followed by Section 2.2, which provides an initial starting point for applying computer vision to get useful data out of the scans. Lastly Section 2.3 covers the final questions that this thesis seeks to answer from a domain perspective.

2.1 Equipment labels in digital twins

There is no universally accepted precise meaning of *digital twin*, but it can be broadly defined as a digital mirroring of a physical entity [1]. In this thesis, digital twin refers to a 3D software representation of a facility with connections to external systems. In practice this means a collection of one or several CAD models or scans of the facility, accompanied with an information management solution that connects parts of the 3D representation to relevant data such as status, design documents and real-time data.

Cadmatic's information management solution in this space, eShare, as seen in Figure 2.1, is mostly based on 3D models created in Cadmatic or other CAD software. The idea is that all of the different systems and items in the facility are represented by parts of the CAD model, which have useful attributes and metadata. For example, these attributes can describe which pipeline a pipe section belongs to, or what system a piece equipment is part of. Embedded in the CAD model itself, this data is incredibly important when it comes to information management: for example, if the name of a pump is known, the CAD model of the pump can be connected to its technical drawings or related sensor data.

Typically, the system works by connecting attributes of the CAD model to documents and queries to external systems. However, sometimes such models are not available, and point clouds are used instead for this purpose.



[redacted]

T-102

Model	
Compartment	ZONE 2
Equipment Position Id	T-102
Mass	0.0000
Role in Process	STAND BY TAI
System	Equipments
System type	Equipment
Equipment management data	
Inventory Tag	ITNRT11251
Purchase Number	PNRT11126
PLM	
PLM data	
Status Tracking	
Equipment Status	Designed
Documents	

14011-0200-P22-MD-00009-000-2 (P&I-SYS DRAINAGE OUVERTS).DWG *(page 1)*

Figure 2.1: Cadmatic eShare is used to view CAD models with connected data

2.1.1 Point clouds and laser scans

While new facilities are nowadays first designed using CAD tools and the models then exported for digital twin use, this is a somewhat recent development considering the age of some of these facilities. In Cadmatic's experience, for many of the older facilities a 3D model suitable for information management either was not kept by the owner after the facility was built, or never existed in the first place. As a result, Cadmatic's customers often utilize point cloud and laser scan technologies to create model-like 3D representations of existing facilities when a suitable model is not available. An example of a point cloud can be seen in Figure 2.2.

Point clouds, as their name suggests, consist of a very large number of points in a 3D space. When created by a terrestrial laser scanner, the scanner typically measures the distances of points from its position and combines that data with images taken from the same place. This leads to the creation of full-color point clouds.

Unlike 3D models, point clouds have no domain-relevant metadata: for instance,

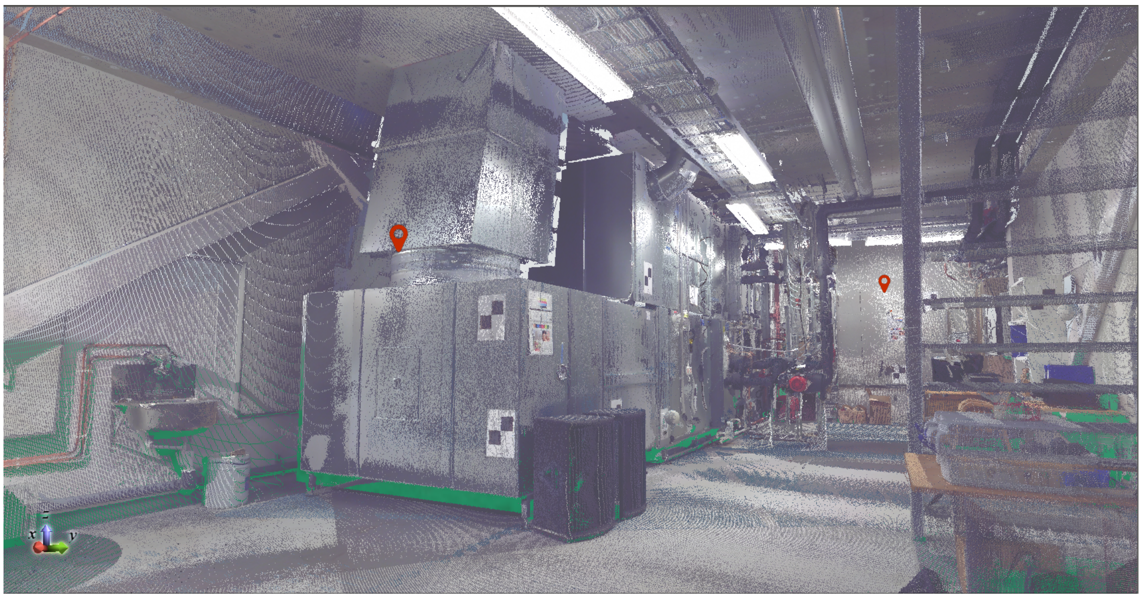


Figure 2.2: An example point cloud with red scanner position markers

there is no easy way to connect a pipe-like group of points to information about that pipe. This makes point clouds inherently less useful than 3D models, and ultimately limits the usefulness of information management software when applied to older facilities.

Historically, many different kinds of laser scans of varying levels of fidelity have been produced depending on scanner equipment and settings. Some of the scans are grayscale, while some are colorful; some are extremely high resolution with a lot of readable small text, while others are of very poor quality. In short, the quality and type of data varies wildly. However, from recent examples and interviews with scanner operators, it apppears that modern laser scanners produce full color scans of enormous resolution, even supporting high dynamic range imaging in a larger than traditional color space.

2.1.2 Physical equipment labels

According to employees at Cadmatic, it is typical that process industry facility machinery, equipment, pipes etc. are labeled for identification in some human-readable way. In practice this means that a small metal plate or tag with some identifying alphanumeric code is attached to a piece of equipment. Usually, this code is printed by a machine in a consistent typeface, but in some cases the code may be painted on or handwritten with a marker.

Extracting the locations and text of these labels in 3D scans would be extremely useful for information management purposes because it would make it possible to automatically detect machine locations in the laser scan 3D representation of the facility, making navigation within the 3D representation and connecting data much easier. This would bring point clouds closer to 3D models in usefulness, making it possible to connect external information of equipment to their point cloud representations.

Labels are usually somewhat visible in point cloud scans - the smallest and most faraway signs are unreadable, but close-by plates are often readable. An example label in a close-up image can be seen in Figure 2.3. A typical problem for scans like this appears to be coverage: imaging is performed with the goal of covering the entire facility for simple viewing and not reading text. Thus, even with the newest equipment the imaging may not be accurate enough for text detection simply due to the fact that individual scans are too few and far between.

Details of which labels exist as a whole - codes of machines, pipelines, and such - are usually known as part of the documentation of the facility. Thus, it can be assumed that a list of known labels exists or can be produced to aid recognition.

More information what kind of labels in terms of size, colors, orientation etc. could be found out by collaborating with Cadmatic's customers. While this was not possible for this thesis, it is an important area of consideration, as different countries, companies and even facilities have different methods of labeling and tagging equipment. The usefulness of a partially working solution should be evaluated, as well as any methods that could help add the missing data in failure cases such as pictures taken by hand.

At present, it is still unclear which kinds of labels are the most useful within

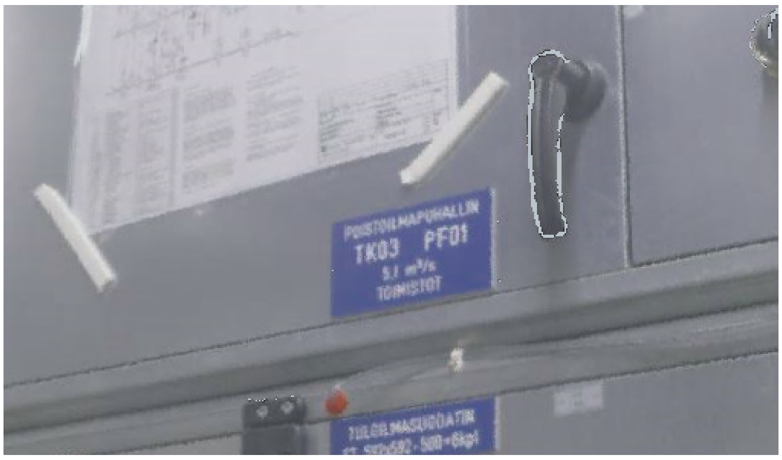


Figure 2.3: Poor quality label

facilities for connecting information, or how good the coverage of a facility should be for the solution to be useful.

2.1.3 Panorama representation

Many point clouds, such as those generated by laser scanning, can be represented as spherical 360-degree panoramas with depth information. When this depth information is stored separately, such panoramic point clouds can be stored directly as pictures using equirectangular projection. An example of this can be seen in Figure 2.4; note how the upper and lower parts of the image are most distorted. This representation may be produced directly by the laser scanner's own software or other software used for processing point clouds. It is trivial to generate perspective views, ie. "normal" looking pictures, of these 3D panoramas.

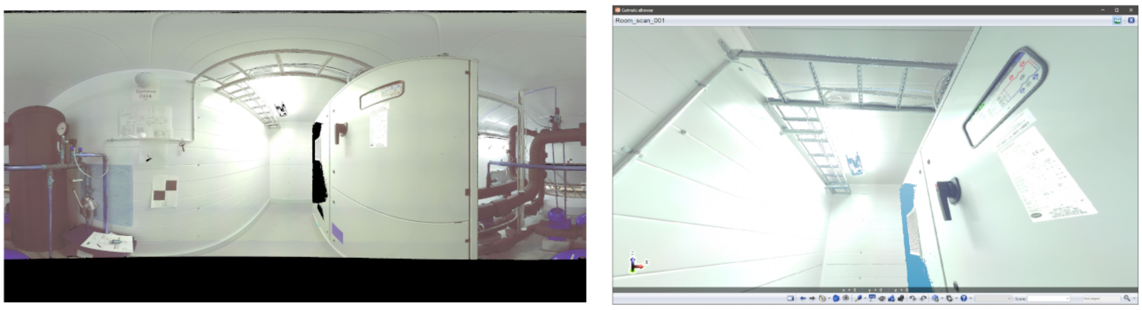


Figure 2.4: An example of an equirectangular panorama. Flattened equirectangular view on the left, with perspective view on the right.

At the horizon level, an equirectangular projection has relatively little distortion compared to alternatives such as cube maps. Cube maps map the panorama onto a cube, which leads to more seams and unpredictable distortion depending on the orientation of the cube map.

Despite its usefulness and ease of viewing, equirectangular mapping is not perfect for the purpose of storing panoramas. Because equirectangular mapping sets longitude as the X and latitude as the Y coordinate in the image, the upper and lower edges of the image contain a disproportionately high number of pixels compared to the center of the image, relative to the perspective area they represent. This means that a large part of the picture is effectively wasted on the top and bottom edges of the view, which usually do not contain any interesting information. In turn, the most interesting areas at horizon level use the least pixels compared to their true size, which is similarly counterproductive.

Regardless of its problems, equirectangular mapping is a commonly used projection, and representation as pictures makes image-based text detection and recognition easier to apply to these scans.

2.2 Applying deep learning

Since laser scans can easily be represented as pictures, it is most beneficial to use the various well-researched image-based methods of computer vision for their analysis. Specifically, detecting text from these scans can be seen as a special case of text detection and recognition in natural scenes, which are both established problems in deep learning. These methods are similar to optical character recognition in documents but target more challenging pictures.

As this is a new domain application of these methods, there are several technical challenges and special aspects to the problem that must be sorted and understood before a viable deep learning solution can be built.

2.2.1 File format and properties

The distortion caused by equirectangular mapping is somewhat problematic, as it stretches text and thus makes it more difficult to read. At the same time, panorama images are very large compared to the cropped and scaled images typically used in computer vision, leading to questions about whether these algorithms operating on small images are able to detect extremely tiny text.

Typically, text detection algorithms are optimized for somewhat large text, and object recognition algorithms target objects that take up considerable space in the image. In this specific application, however, the text is typically very small and difficult to detect. In general, detection of small objects is considered a difficult problem [2]. Thus, focusing on algorithms that excel on small object detection may be beneficial.

Given the small size of labels, existing scans might be of too low quality for the purposes of this thesis. This is not necessarily a problem for the final business case for this technology, as the solution would likely justify scanning the facilities with better scanner technology, as doing so is a one-time cost, and in many cases new scans must already be conducted periodically.

While depth data is likely not utilized by any off-the-shelf algorithm, it may prove useful in detecting whether text is obstructed, whether the surface of the detected text appears plausible, etc. For instance, large changes in depth could be used to disqualify certain text locations, or small changes used to detect smooth planes likely to contain text.

2.2.2 Performance and practicality

While many extensively researched applications of object detection such as self-driving cars or real time text translation are extremely performance critical, the application of extracting text from panoramic images is not so. This property adds a considerable computational advantage for building the model, as detection can be done at worst on a scale of minutes per image instead of milliseconds. At the same time, the number of images is limited and not a constant video feed, for example.

2.2.3 Lexicon and context

A constraint of this specific application of text recognition is that most of the text instances are not composed of actual words in any language. Instead, the text instances are alphanumeric codes used to identify equipment within facilities. This may have an effect on the detection and recognition tasks, as the text is noticeably unusual: for instance, a method recognizing only English words would be entirely useless.

In addition to their lack of language, the text instances are short and do not have any grammatical context. Some methods could use the context of words to deduce their meaning, similarly to machine translation. This kind of a model would most likely only cause problems, as for alphanumeric codes grammatical context does not exist. In addition, any model that typically deals with paragraphs of text may have trouble with the codes being equivalent to single words.

As noted in Section 2.1.2, the existence of an equipment label list implies a lexicon of the possible codes to be found in the facility is known. However, this list of words greatly differs from natural language lexicons, as codes often share prefixes or other parts, and several different codes may differ by just one character.

2.3 Research questions

Based on the previous background information, this section covers the questions that the following chapters seek to answer:

When it comes to industrial plants, it is unclear how well labeled different devices and items are. Valves appear to have small tags that are easily obscured, while larger elements such as fuse boxes have much larger and easily legible labels. Equipment labeling varies: some plants have all larger equipment well signified with easily readable plates, while some simply have hand-written markings with a marker, etc.

Focusing on the machine printed labels, is detecting and reading them possible even in theory?

Many existing scans are low resolution and captured at a distance, aiming for coverage rather than accurate data. How does the quality and distance of a laser scan affect the accuracy of text detection and recognition? Is it possible to use older, lower quality scans of facilities, or will we need new ones, or wait for the technology to improve?

Regarding applicability of current research, how well do state-of-the-art natural scene text detection and recognition algorithms lend themselves to this task? What approaches can be taken to combine and modify these methods to improve accuracy over directly applying previous research, and is this application one where existing methods and models will transfer easily?

3 Deep learning background

This section explains important machine learning concepts necessary for understanding this thesis. The first section provides a brief overview of machine learning, while later sections cover more specific key areas of deep learning discussed in this thesis.

3.1 Overview of machine learning

Typically, computer programs are defined in an explicit manner by writing out the logic of a desired application. In contrast to explicit programs, machine learning means the study and use of algorithms that come up with ways to make decisions or predictions on their own without explicit prior definition. By analyzing input data, a machine learning algorithm learns a model that produces output predictions. [3] (p. 20)

Machine learning excels at problems where the solution is not known or is unfeasible to formulate. For example, consider the problem of whether a given image contains a dog: easy for a human, but difficult to formulate to a computer. As problems like this can be solved with machine learning, computer vision is one of its most popular applications.

In this thesis methods of machine learning, and specifically deep learning are explored for the tasks of text detection and text recognition. The fundamental idea of a text detection and recognition method is to train a model - an algorithm and the parameters of said algorithm - on training data, and then use that model to predict the location and content of text in previously unseen images. In the tasks of text detection and recognition, that training data is images where the location and characters of text are known. This generated model is then used to detect and recognize text in new images: the ability of this model to make predictions on unknown data is called generalization.

3.2 Model training

To train a machine learning model for a specific task, a dataset of some kind is needed. In supervised learning such as that in this thesis, the training data consists of example input-output pairs which are then divided into different sets based on their purpose, as displayed in Figure 3.1. The training set is used for training the model, which in this case is an artificial neural network; the validation set is used for tuning the adjustable parameters of the network called hyperparameters; the test set is used to evaluate the model's final performance [3] (p. 86, 95). Artificial neural networks are described in more detail in the following section.

In contrast to supervised learning, unsupervised learning means training a model

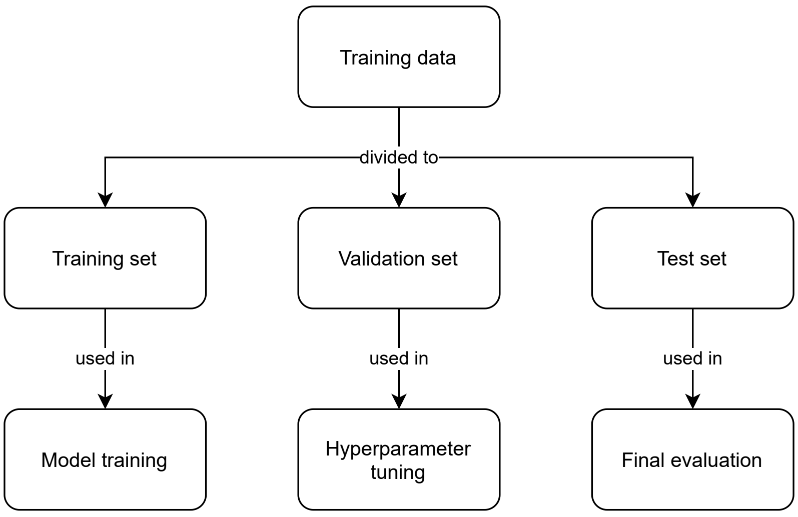


Figure 3.1: Data division

based on data without explicit input-output pairs. While such learning is not commonly used for text recognition, there are some methods incorporating weakly supervised learning, where the input data is limited or imprecise. For instance, if a method is trying to detect individual characters, doing so using training data where labels are word-level as opposed to character-level is weakly supervised learning. [4] (p. 6, 108-113)

3.3 Artificial neural networks and deep learning

Artificial neural networks are a fundamental concept in machine learning that draws its inspiration from the nervous systems of living organisms. Like nervous systems in nature, it too consists of a complex network of neurons that pass signals forward. [3] (p. 27) Since artificial neural networks are very good at learning complex tasks and modern graphics processing units have made it possible to process them in reasonable timeframes, they have been applied to a huge number of tasks, including computer vision.

In terms of terminology, deep neural networks are neural networks with many layers of neurons within them. The definition of "deep learning" itself doesn't require deep neural networks, but is nowadays used for all kinds of gradient based methods [5] (p. 261), including those of feedforward neural networks.

3.3.1 Feedforward neural networks

There are many different architectures of artificial neural networks, but the most historically notable one that most modern neural networks are derivatives of, is the multilayer perceptron. It is a type of feedforward neural network, which are neural networks where the connections between neurons do not form a cycle.

A simple multilayer perceptron consists of three layers of neurons: an input layer that represents the inputs of the network, an intermediate layer called the hidden layer, and an output layer representing the output. Each neuron in a layer gets its input from all the neurons of the previous layer and passes its output on to all neurons of the next layer. These connections between layers have weights. [5] (p. 138), [4] (ch. 5, 6)

Specifically, the output of any single neuron within the neural network is determined by the weighted sum of its inputs, which are the activation of previous layer's neurons, as well as a bias term. These are passed into a non<linear activation function such as a sigmoid function that determines what the output of the neuron should be. The weights of the connections between neurons effectively decide which connections from the previous level matter most for the calculation of that specific neuron. An example of a multilayer perceptron can be seen in Figure 3.2. [5] (section 6.1)

In artificial neural networks, training means updating the weights of connections between neurons, and the neurons' biases. When the neural network is trained on training data, a loss function indicating how wrong the network was in its predictions

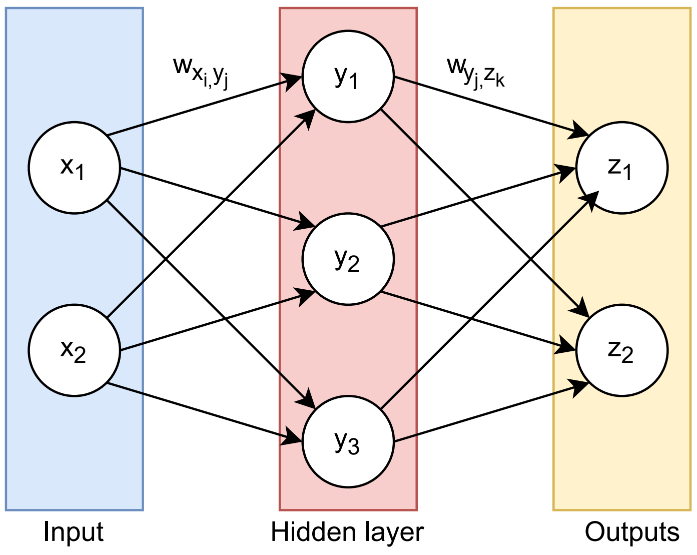


Figure 3.2: Feedforwad neural network layers

is evaluated. In order to reduce this error, the weights of the connections need to be updated in a beneficial way. [5] (section 6.2)

Weights of the neural network are updated using an algorithm called stochastic gradient descent, utilizing backpropagation. First, the backpropagation algorithm efficiently calculates the gradient of the error function regarding each weight: in other words, it figures out what kind of effect changing each weight has to the overall loss of the network. Next, stochastic gradient descent is used to update the weights in such a way that the error is lowered, according to the gradient. The end result of this process is that the training error is reduced, and hopefully the network's predictions are closer to expected. A similar algorithm is visualized in Figure 3.3. [5] (section 6.2), [3] (sections 4.3, 5.9)

3.3.2 Convolutional neural networks

Convolutional neural networks, CNNs for short, are neural networks containing convolutional layers. They take advantage of the mathematical operation of convolu-

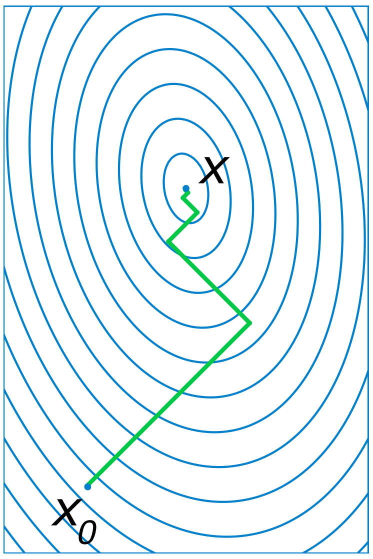


Figure 3.3: An example of gradient descent: the algorithm zigzags towads a local minimum at the center of the contour lines. Adapted from [6].

tion, which lends itself well for computer vision tasks. In concept, they are somewhat inspired by the biological visual cortex of the eye. On a basic level, a convolutional layer takes a tensor such as an image as input. On this input, convolutional filters are applied with weights and bias similar to a neuron in a feedforward network, resulting in an output called a feature map. The convolution can be parametrized in terms of size - how large the receptive field sliding window is, stride - whether the convolution should run for every position in the image or skip some, and padding which allows convolution to continue over the edges. [5] (ch. 12), [4] (ch. 13)

In computer vision tasks, convolutional layers are often used for feature extraction, stacked one on top of another with different parameters. The feature maps that these layers produce may then be fed to, for example, a fully connected neural network that finally classifies the features. One example of such a network can be seen in Figure 3.4.

In the context of this thesis, convolutional neural networks are used at every step of a text recognition pipeline, as they are the primary method used to generate features out of two-dimensional images.

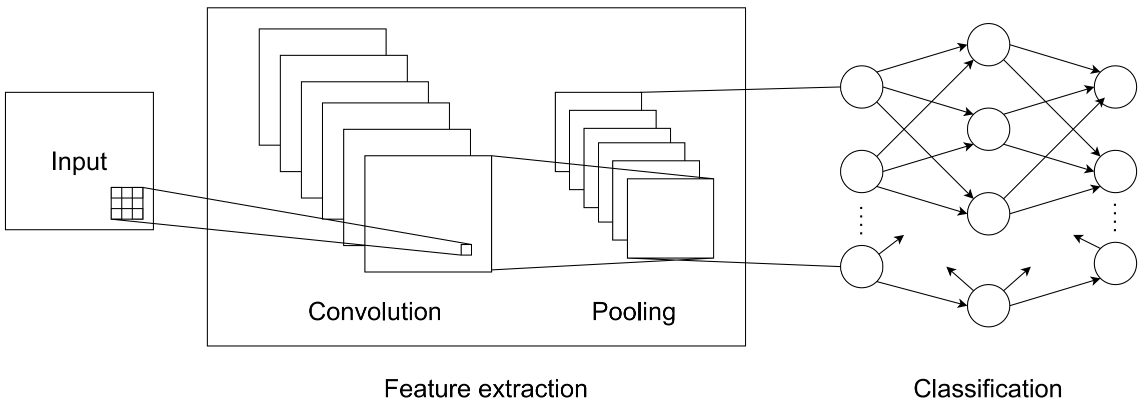


Figure 3.4: Example convolutional neural network

3.3.3 Recurrent neural networks and others

A considerable downside of feedforward networks is that they have no memory and thus are incapable of "remembering" things: they simply take an input and produce an output without regard to what came before or after it in a sequence. *Recurrent neural networks*, also known as RNNs are a class of neural network that solve this problem by processing data sequentially and passing data from one time step to the next. These networks are thus capable of learning dependencies between different elements of the sequence. For instance, in the case of text recognition, a word can be processed letter by letter and data about the previous letters can be passed to the current letter. The most notable downside of RNNs that many later improved designs have attempted to fix is that the farther apart elements of the sequence are, the less likely they are to affect each others' outputs. [4] (ch. 14) [5] (ch. 13)

One such recurrent neural network architectural component is the Long short-term memory (LSTM) unit [7]. With a more elaborate design than its predecessors, it can maintain its hidden state for much longer than a typical RNN. Attention and transformers are further improvements to RNN-based architectures.

As recurrent neural networks are particularly useful when processing sequential or time-series data, in the context of this thesis, they are most useful in the task of text recognition.

3.4 Relevant other applications of deep learning

Some common applications of deep learning are image classification, object detection and natural language processing. While the applications usually share the same deep learning basis, each of them takes a different direction in its research and methodology. These tangentially related technologies build the basis on which modern text detection and recognition works are built.

3.4.1 Related technologies

An image classifier takes an image as input and assigns it to one out of several defined categories. One of the most significant breakthroughs in deep learning was the 2012 work of Krizhevsky et al. [8] winning the ImageNet image classification competition by utilizing a deep convolutional neural network. Much of the research related to deep learning in computer vision since has been based on these same concepts. [9] (p. 2353)

An object detector, on the other hand, is tasked with finding instances of some specific object in an image. Typically, this is visualized by drawing a bounding box on the objects found in the image. This application is closely related to image classification - in fact, one of the most important object detection works was Girshick et al. [10] adapting the aforementioned work by Krizhevsky et al. [8] to work in an object detection context. Object detection itself is closely related to text detection. [11] (p. 2)

Natural language processing is considerably wider of a concept, and concerns many subjects related to spoken and written language such as machine translation or handwriting recognition. Compared to the previous two problems, which are mainly related to computer vision and thus related to two-dimensional images, many language processing problems are rather seen as sequence-to-sequence problems. [12] Most recent improvements to text recognition seem to emerge from this field.

3.4.2 Relation to text detection and recognition

Text detection can be considered a subtask of object detection [13]. As a result, much of the research on object detection has implications on text detection, and many text detection methods themselves are adapted directly from object detectors.

Text recognition, however, is closer to a natural language processing task. In the past, text recognition was typically viewed as a task of separating each word into characters and classifying those characters. Nowadays, however, it is mainly viewed as a sequence-to-sequence task. Many architectures from other sequence-to-sequence tasks like handwriting recognition are nowadays utilized in text recognition [13]. CTC, attention and transformers are examples of such architectures that are covered later in this thesis.

3.5 Evaluation of models

Text detection and recognition methods are evaluated through statistical measures that quantify their performance. For text detection, the most common statistics used are *precision*, *recall* and *F-score*.

Precision answers the question "out of all the predictions produced by the model, how many are true positives". The value is represented as a decimal number. For instance, if 50 of 100 predictions are correct, the resulting precision value is 0.5. Recall, on the other hand, represents how many of all instances of the ground truth are detected. For example, if the ground truth contains 100 instances and 50 are detected, the resulting recall is 0.5. [4] (p. 294)

Alone, either of these two measures does not fully describe the usefulness of the model. For instance, high precision numbers can be attained even if almost no text instances are detected; vice versa, very high recall values can be attained while producing lots of wrong extra detections. F-score combines these two measures to one number, calculated as a harmonic mean of precision and recall. As it places an equal weight to both numbers, it cannot alone represent the usefulness of the model for all situations, but as a single number it gives a better overview than the two individual measures. [4] (p. 294-295)

Text recognition is generally measured with a single measure of accuracy, which is effectively the recognition rate: it represents the fraction of the total instances that the recognition model output correctly. Another measure that is especially used in handwriting recognition is *Character Error Rate* (CER), which measures the number of insertions, substitutions and deletions required for the output and ground truth to match. This is based on Levenshtein distance, which is a simple edit measure proposed by mathematician Vladimir Levenshtein in 1966 [14]. This algorithm counts the number of insertions, deletions and substitutions between two text strings and produces a total number of them. Dividing this distance by the number of characters results in the character error rate.

4 Related work

Text recognition is probably most familiar to those who have used optical character recognition tools for scanning paper documents or used license plate readers. End-to-end scene text recognition is concerned with a task that is a superset of these problems: reading any kind of text in any kind of setting. Typically finding and reading text are considered two different problems: text detection and recognition, with separate research dedicated to each task.

In the last few years, research on text detection and recognition has advanced significantly with the development of deep learning methods. While previously these tasks were dominated by entirely or partially handcrafted features and algorithms, recent methods utilize deep learning from end to end, producing dramatically improved results. With parallel processing becoming more and more accessible, deep learning is becoming more and more viable for industrial use.

The aim of this chapter is to provide a theoretical background for the thesis as a whole: a basic literature review of academic research relevant to end-to-end text recognition. It focuses on methods developed for the separate tasks of detection and recognition as well as those meant for end-to-end recognition. Almost all of the text detection and recognition methods described are designed to work on 2D images from a camera, as that is the most researched and most common real-world application of these methods. Very little public research exists on applying these concepts further to panoramas or pictures with additional depth information; thus most of the research covered in this chapter ignores those aspects of the problem and focuses on flat images instead.

4.1 Overview of related work

Scene text recognition methods are divided into three main categories in almost all of recent research: text detection, text recognition and end-to-end text recognition which combines the other two [13]. This section explores these three topics and other relevant research about data and models regarding text and images.

Text detection means finding the position and orientation of text in a picture. Specifically, this usually means localizing individual words or lines of text within a picture and fitting a bounding box around each of them. In natural scenes, the detected words are usually a relatively small part of the whole image. For example, a picture of a storefront with a sign on top could be used as input for text detection.

Text recognition, on the other hand, means reading text from a cropped image where text has already been located and correctly identified. In practice text recognition methods are not usually applied on perfectly cropped and rectified images, as text is always surrounded by some space in the image requiring some kind of rectification. Still, these images are much tighter cropped than those used in text detection.

End-to-end methods perform both text detection and recognition - they are end-to-end in the sense that they take in an image and output the text within the image. They either combine separate detection and recognition approaches to one single method or use a unified solution to solve both problems at once. Their main advantage is being trainable end-to-end, meaning that there is no need for the training of two separate models. Generally, the sharing of feature maps between the stages means better integration between the detection and recongition stages, which should logically lead to better end-to-end performance than combining two separate methods.

Figure 4.1 shows the difference between text detection and recognition as concepts. End-to-end text recognition combines both of the two concepts in one.

Most works today focus on either detection or recognition separately. Thus, developing a useful end-to-end solution requires thorough inspection of research both in detection and recognition as separate tasks. Some competitive end-to-end solutions have been proposed, but they largely adapt concepts from the two separate stages, and less research has been conducted on the process as a whole compared to the two separate steps. For a more comprehensive history of scene text detection and recognition, a 2020 survey by Long et al. [13] compiles the most influential methods and their development.

4.1.1 Choice of research topics

Since many existing papers on text detection and recognition focus on some subset of the detection or recognition problem such as curved or distorted text, not all of the them are directly relevant to this specific application of text recognition. For instance, while curved text is unlikely to appear in the input data of this thesis, it is one of the main challenges new text recognition research tends to focus on

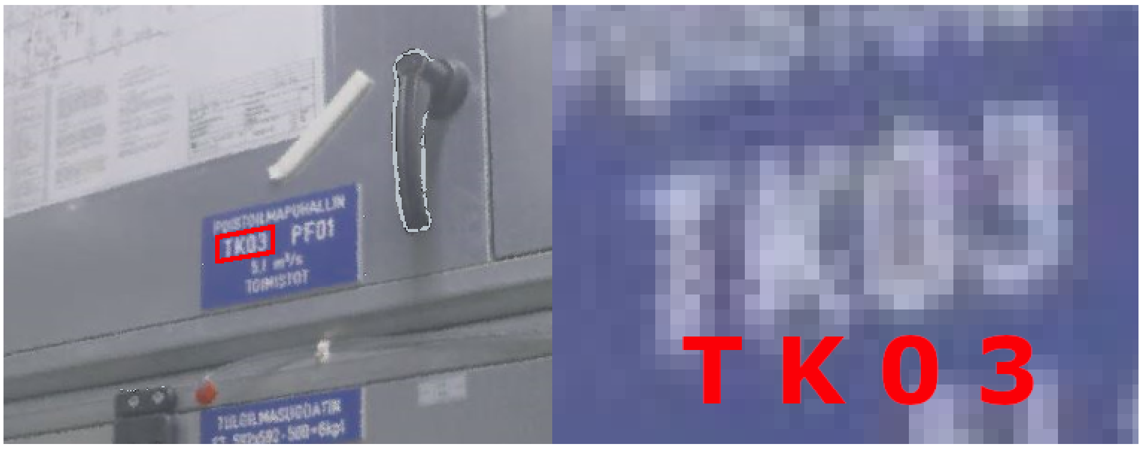


Figure 4.1: Text detection (left) vs. text recognition as tasks (right)

[13], [15]. Although typically not included in the domain of scene text recognition, another example of a very specific application is real time car license plate detection, where many assumptions can be made about the background, layout, orientation and contrast of detectable text in the license plate, and strict performance requirements apply. Thus, methods to solve these specific challenges often rely on assumptions that do not hold for label detection in facilities. In addition, many proposed solutions are optimized for performance on a frames-per-second level on some available GPU, which is not necessary for the purposes of this thesis, as the detection and recognition are not performed in real time.

One potentially significant difference between current research and the desired application is that research is focused on "natural scenes" that are generally less cluttered and simpler than facility interiors, and often on texts much larger than those of small labels. The dataset images typically include text on large traffic signs or storefronts, where text is typically large and perhaps more diverse typographically, while the labels in facilities are typically very small, just large enough to be readable in the panoramas.

4.2 Text detection methods

Since text detection can be considered a special case of object detection, virtually all current text detection methods are based on general object detection research on some level, typically incorporating the newest advances in that field. Many of these methods draw from bounding box regression -based methods, while some perform image segmentation. More advanced methods extract text-specific features and take special properties of text such as the division to characters into account.

Text detection methods are difficult to categorize clearly, as they largely borrow from each other. First, methods that just adapt bounding box object detectors are discussed, after which methods doing more elaborate segmentation approaches are introduced.

4.2.1 Object detection -based

Typically, an object detection -based detector will encode convolutional feature maps of the input image and predict the location of text within them using a classifier, based on which a bounding box is then fitted to the image. These methods' greatest advantage is their simplicity: they are well researched and relatively simple to implement and train end-to-end due to the popularity of object detection research.

To better explain these methods, some background explanation of the popular methods regarding object detection itself is necessary. The task of object detection can be defined simply: does a given image contain instances of objects belonging to some known category of objects, and if so, where in the image are they located and what is their area [11]. Essentially the task is to label some known categories of objects within an image, as seen in Figure 4.2.

Modern deep learning -based object detection got its start from the R-CNN method by Girshick et al. [10]. The idea of the original method is to extract regions of interest from the image, run those through a convolutional feature extractor, and use a classifier on the results. These are then merged back to one image by prioritiz-

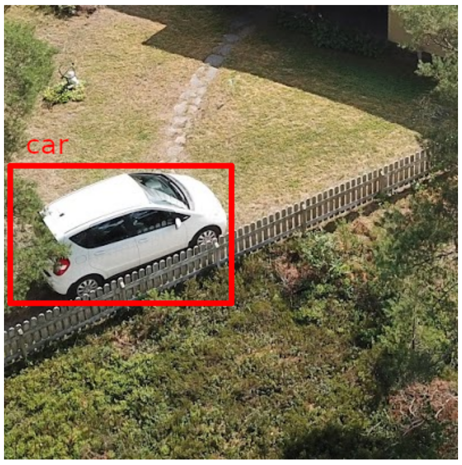


Figure 4.2: Object detection (bounding box regression) example

ing higher-scoring regions and then improved with bounding box regression. Later methods based on this work then improved end-to-end trainability, performance and many other qualities of the detector [16][17].

Some examples of notable object detectors previously adapted to text recognition are YOLO of Redmond et al. [18], SSD of Liu et al. [19] and Mask R-CNN [17]. One of the most significant modern text detectors, EAST, proposed in 2017 by Zhou et al. [20], uses a similar architecture. Extremely fast and accurate compared to the state of the art at the time, it used a U-shaped convolutional network [21] which merges feature maps from different feature extraction levels gradually. While newer methods may surpass it in performance, its relative simplicity, performance and ease of implementation has lead to it becoming very popular to re-implement, and several public and open source implementations are available. TextBoxes++ by Liao et al. [22] is another similar well regarded text detector. Based on their earlier work called TextBoxes, it is based on the SSD object detector [19].

While these methods are very simple, efficient and well researched, their detection does not work very well for arbitrarily shaped or long text compared to segementation-based methods [13]. Most current bleeding edge research is more complex and text-specific in nature. Still, these methods seem very popular in practice due to their simplicity and competitive detection results.

4.2.2 Segmentation -based

In segmentation-based methods, typically the different areas of an image containing text are detected on a pixel level and processed into separate text instances. This is arguably a better method than the previously described region-based text detection methods, as a fundamental property of text is that it can be split into several parts - for instance, a sentence can be split into words - and the remaining parts are still considered text. While general object detection treats blocks of text as objects, segmentation treats text as more of a class of content within the image.

Figure 4.3 showcases an example of object segmentation, differing from bounding box regression in Figure 4.2. In this case a car is separated from its background on a pixel level.

Segmentation of text may happen on different levels. The most basic level of segmentation is text segmentation, in which each pixel is classified as text or not text. More complex methods may find character or multi-character segments within text, closer to multi class segmentation.

Baek et al. [23] propose a method called CRAFT, which performs character level segmentation. It localizes and groups individual characters within images into text instances. However, this is somewhat challenging as real-world datasets have a word-level rather than character level data, requiring weakly supervised learning.

The Textsnake method proposed by Long et al. [24] represents text as a sequence of overlapping discs that are joined together. The algorithm specializes at detecting free form and distorted text instances. It works by first segmenting the image to find text regions, and then fits a center line and discs to each instance.

As another example of segmentation, Wang et al. propose a Pixel Aggregation

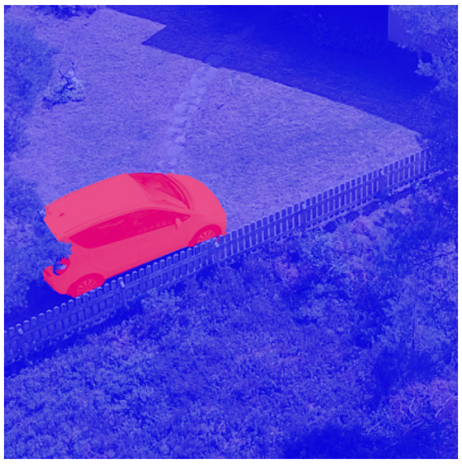


Figure 4.3: Image segmentation example: the "car" part of image in red, with background in blue.

Network [25] that specializes in arbitrary shaped text. It uses a lightweight backbone and utilizes "Feature Pyramid Enhancement Modules", extracting multi-level features.

SegLink by Shi et al.[26] uses the SSD object detector by Liu et al. [19] to detect text segments, between which it predicts links to connect them as instances.

4.3 Text recognition methods

The purpose of text recognition is to convert a tightly cropped image of text, such as the output of a text detection algorithm, to a sequence of characters. How well the image is cropped before being fed to text recognition varies - on the hand-labeled academic datasets the space in each text instance may be minimal, while object detectors often produce much more varied results.

As text is read from one side to the other, the input data of text recognition can be considered sequential: text consists of sequences of words, words consist of sequences of characters, and characters consist of sequences of lines of pixels. There are several neural network architectures specifically focusing on sequential data, and many can be applied to text recognition as a result.

4.3.1 Overview of recognition architectures

There are several relevant network architectures from other domains that frequently appear in text recognition research. These are often related to recurrent neural networks seen in machine translation and other natural language processing tasks, as text recognition is seen as a sequence-to-sequence problem. [13]

Probably the most common text recognition-related architecture from natural language processing is the encoder-decoder architecture, which uses one RNN to encode the input sequence to a vector, which is then decoded by another RNN.

Originally the work of Sutskever et al. [27], this architecture comes from machine translation.

An evolution of the encoder-decoder architecture is the addition of the attention mechanism, which was first described by Bahdanau et al. [28] for machine translation. It allows better modeling of dependencies and alignment between two sequences, as each element of the input sequence produces its own context vector rather than a single common one. This improves the network's ability to "remember" elements of the sequence farther away from the current output, and "pay attention" to previous relevant parts of the sequence. Attention is also very commonly used in text recognition, and builds an especially strong implicit language model.

Transformers, as introduced by Vaswani et al. [29], improve again upon attention. Transformer-based models, instead of relying on an encoder-decoder model, base the entire architecture on the attention mechanism itself. Transformers have been recently adapted to text recognition by Xue et al. [30].

In a text recognition application, all of these RNN-based methods learn dependencies between different parts of the input text image. This means that not only do they learn which part of the image stands for each letter, but they also learn dependencies between characters within a single word. This is useful for recognition of natural language text, as it allows for better construction of words. For this reason, RNN-based models are considered to contain an implicit language model.

For a decent overview of the landscape of modern text recognition, in 2019 Baek et al. [31] compiled a benchmark of recent deep learning models for text recognition and pointed out problems with the methodology of previous text recognition model comparisons, such as the inconsistency of datasets used for training. They reduce most text recognition methods to one simple pipeline of transformation, feature extraction, sequence modeling, and prediction.

4.3.2 Character level features

Character level methods generally attempt to divide the word of the input image to a sequence of characters, and then classify those characters individually. Especially many of the early conventional text recognition methods were based on character segmentation or detection [13]. While not generally used today on its own, using character level information has several benefits over word level data, most notably language independence. For the particular task covered in this thesis, this property is important.

A considerable practical problem in the implementation of character segmentation methods is that most existing text detection and recognition datasets have word-level annotations. This means that character positions must be somehow inferred using weakly supervised learning rather than the supervised learning that is possible in word-level methods.

Some modern methods work on a character level. Char-Net by Liu et al. [32], for instance, uses a character level encoder. It encodes features on both word and character level, introducing a bespoke recurrent RoiWarp layer and using character level attention. Liao et al. [33] on the other hand utilize a semantic segmentation network with a character-level attention mechanism. It appears that character-level features are generally not used as commonly as they once were, as most have adopted a word level encoder-decoder architecture instead [13].

4.3.3 Word level features

Word level features are typically extracted using methods developed for sequential data. These sequence-to-sequence methods do not view text as a problem of classifying individual characters but recognizing pieces of text as sequences that map to certain words. This means that they are almost always language-specific, as their word detection greatly depends on their dataset.

Within the four stage pipeline by Baek et al. [31], transformation normalizes the input against inconsistent orientations and translations of text according to Spatial Transformer Network by Jaderberg et al. [34] Feature extraction is then used to produce feature maps from the input, typically using a convolutional or recurrent neural network classifier backbone. Sequence modeling considers the sequentiality of text to predict characters based on each other instead of individually. Lastly, encoder-decoder based prediction, typically using *Attention* or *CTC*, is performed.

CTC stands for Connectionist Temporal Classification [35]. It is a sequential decoding model that can map inputs and outputs regardless of their alignment and is often employed in other sequential tasks such as audio or handwriting processing. What this means is that CTC is able to assign labels to sequences that are not segmented where character locations and boundaries are not clear or known. It has been used in several algorithms, such as that by Shi et al. [36].

Attention, on the other hand, comes from the field of neural machine translation [28]. It learns alignment between the input and output sequences, where the attention mechanism helps ensure the passing of data between steps of the encoder. Attention has been utilized by several methods, such as that by Cheng et al. [37] When used, it creates a stronger implicit language model than a typical RNN-based recognition model, as it learns stronger relationships between different parts of the input sequence. This property reflects its strength in machine translation tasks. In addition, Xue et al. [30] adapted transformers, which improve upon attention, for text recognition.

4.4 End-to-end text recognition methods

End-to-end text recognition, also known as end-to-end text spotting, usually combines detection and recognition algorithms to one solution. There are two main kinds of end-to-end methods: those consisting of two separate model steps and those consisting of one step with two integrated stages. Systems with two separate models first detect text using one model, crop images out of the source image and then run recognition on the cropped images. This makes the two steps entirely separate and not end-to-end trainable. Meanwhile, one-step methods instead input the feature maps from the detection step to the recognition step, creating an end-to-end trainable single model. Recent research on end-to-end text recognition is focused on the latter method, while separate methods appear to be more common in practical applications based on the number of blog posts, public models and works like that of Mai et al. [38].

4.4.1 General end-to-end text recognition

One of the most significant recent end-to-end text detectors is FOTS by Liu et al. [39]. It combines an EAST-like detection branch with a new rectifying step called RoIRotate and lastly a CTC recognition branch, providing a single end-to-end trainable model. Another notable method is Mask TextSpotter by Lyu et al. [40] which uses a Region Proposal Network [41] to feed a Fast R-CNN [41] bounding box regression branch and a mask branch for word and character segmentation.

CharNet by Xing et al. [42], not to be confused with the previously mentioned Char-Net text recognition method, is another end-to-end solution. It uses a similar ResNet backbone to FOTS, but instead uses separate character recognition and text detection branches. The basic idea is that one branch recognizes characters in the image while the other detects lines of text. The character segmentation -style approach suffers from the same problems as other character segmentation methods, namely the fact that character-level training data is rare and thus weakly supervised learning must be deployed.

4.4.2 License plate recognition

License plate recognition is one of the areas of research that end-to-end text recognition has been applied to. From the point of view of this thesis, license plate recognition is interesting because it is arguably an easier version of the equipment label detection problem. Exactly as in the label task, license plate recognition does not deal with words of any language, but alphanumeric codes consisting of Latin characters. In comparison, however, license plate text instances are bound to be more homogenous in terms of background, size and typeface, and plate recognition systems are subject to stricter runtime performance bounds. These systems are likely useful for this thesis, as they are practical implementations of text detection/recognition that can be deployed on their own rather than purely academic exercises that only concern a part of the full end-to-end pipeline.

Some modern license plate recognizers use a method similar to encoder-decoder based text recognition methods [43][44]. Many methods, especially older ones, use character segmentation instead, which very few new text recognition models adopt compared to the encoder-decoder architecture and its derivates [13].

4.5 Adaptation to small text

Since text detection adapts methods from object detection, it makes sense to study small object detection to find out how the small size of text could affect model performance. The definition of small object detection varies, but one common definition originating from the COCO dataset [45] defines small objects as those under 32x32 pixels in size. Compared to general object recognition tasks, adapting models especially to small objects studied comparatively little, as most mainstream models focus on objects that consist of a larger area of the image.

In their review, Tong et al. [2] found that current deep learning methods used for small object detection are still quite far from humanlike performance despite being useful for a multitude of applications. They note a distinct lack of datasets and models specifically crafted for this task. They cover some common approaches to the problem, such as feature pyramids, which are also commonly used in text detection.

4.6 Adaptation to panoramas

While there is not much research on end-to-end text recognition on panoramas, there is some relevant research involving general object detection. Applying object detection to 360-degree imagery suffers from one large problem: both models of most existing methods and the datasets used to train their models assume a perspective view rather than a 360-degree view. The two most common intuitive but flawed solutions to these problems are either to apply the algorithm directly onto the distorted equirectangular image or to generate many perspective-correct views of the panorama and process them individually. The first of these solutions is cheap in terms of computing power, but may yield bad results due to image distortion, especially near the most distorted top and bottom parts of the images; the second takes distortion into account but needs much more computing power.

Yang et al. [46] combine these methods by running object detection on multiple overlapping stereographic projections. Better ways to handle panoramas have been suggested by Su and Grauman [47][48] regarding models for transforming networks trained on planar data to work on panoramic data.

One interesting similar application of 2D computer vision on panoramas is the bullseye detector by Mai et al. [38] In their research, panoramas from ships are used to detect "bullseyes" which are rather large signs with text on them, very similar to this thesis. In their case, cube maps were used instead of an equirectangular projection. They used EAST for text detection and Tesseract for text recognition.

4.7 Synthetic training data

When it comes to having little to no training data, it is possible to automatically generate large datasets for end-to-end text recognition as described by Gupta et al. [49] in their work on SynthText. In SynthText, a large dataset of images without text is taken, and text is then inserted on top of the images to generate a large dataset of natural images with known text and labels. Many models are pretrained with their synthetic dataset and reportedly produce decent results - even if the dataset still requires finetuning with better, more realistic data for specific use cases. While the SynthText dataset itself is often used by many text detection and recognition methods, the generation algorithm itself is not. Based on this information, it appears that this work could also be adapted for different purposes with different text typefaces and transformations.

5 Applying research to the domain

This chapter covers the technologies introduced in Chapter 4 and assesses how they fit the requirements and details covered in Chapter 2. The overall architecture of a potential solution is discussed first as a whole, after which subsequent sections go into more detail about each step of the solution and the challenges and proposed solutions regarding them based on literature.

5.1 Overall architecture

The goal of the method as a whole is to find the locations of known texts from data produced by laser scanners, specifically alphanumeric codes consisting of Latin characters. As covered previously in Chapter 4, image-based methods should be first chosen over any 3D data or point cloud-based methods due to the popularity of relevant research and the inherent two-dimensionality of text.

Within image-based end to end text recognition, the options then are to choose either separate text detection and text recognition steps, or a combined end-to-end text recognition method sharing features between the stages. Based on literature surveyed previously in Chapter 4, a combined model should perform better when trained end-to-end on training data accurately representing the scenario. However, most of existing practical methods and research are based on separate detection and recognition steps. As noted in Chapter 4, end-to-end models appear promising in their research results yet seem much rarer and less commonly used. This makes comparing the two types of architecture difficult as a whole: the one-step option should be better in theory, but it is difficult to find practical solutions compared to the two-step option. Thus, the focus of this thesis is on two separate steps.

As a sufficient amount of domain-specific training data is not available, the detection and recognition models rely on training on standard datasets. For quick experimentation, the available methods are typically distributed with pretrained models deriving from these datasets. This means that the models have been already trained on some dataset - typically one that is much larger and more generic than if one was compiled from facilities with tags in them. How well these common detection and recognition models generalize to the facility data at hand is unclear. It is known that their performance can be improved with additional training: finetuning the model with manually labeled data that better corresponds to the target.

The initial idea is to utilize a standard model for both detection and recognition steps. At this point, the greatest challenge is finding models with satisfactory performance, and making them work together efficiently. Then, later, the models could be trained further with domain specific data once acquired.

Once a text instance is extracted from an image, it must be processed and paired somehow with an item on the list of possible labels, which we can assume exists based on Chapter 2. With such a list, it is possible to utilize a filtering step that selects the most likely label out of the known labels list, but only if the recognition output is close enough to the exactly correct sequence. Once the label text is known, this is then fed to the last part which localizes the text and its position in the three-dimensional panoramic point cloud - however, this last part seems trivial and is algorithmically out of scope for this thesis.

The overall initial architecture of the problem solution, as seen in Figure 5.1, thus consists of four processing steps: text detection and thus localizing the text, text recognition which means reading the found text, filtering text instances, and lastly outputting the final match and its location.

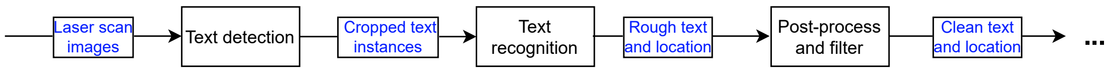


Figure 5.1: Overview of a text recognition pipeline

5.2 Text detection

As covered previously in Chapter 4, there are many different types of text detection methods. Most should be valid for this task, as the text instances to be detected should be short codes, which resemble single words which are a common case in general text detection. There is no need to be able to process, for example, longer paragraphs or entire pages of text that might exist in other applications like document OCR.

Typically research on text detection is conducted on the same few academic datasets. These collections of images usually contain text in "natural scenes", which means text found in city streets, for example. While this is a very different environment to the interior of a facility, with many different types and orientations of text, there are also similarities - both concern pictures taken with a camera rather than a scan of a document as one might see in document OCR, for example.

Most of the text instances found in the facility data are on a horizontal level, and on a flat background. Some text instances, such as those on pipelines or tanks, may be sideways or on a curved surface. Therefore, methods which produce simple bounding boxes as their detection results will likely yield decent results, but methods capable of operating on more complicated instances should be preferred if possible.

5.3 Text recognition

While in principle most text recognition methods should be fit for the purpose, the most fitting text recognition methods should be ones that do not contain an implicit or explicit language model, as in the encoder-decoder models using attention. Many of the state-of-the-art methods most accurate on benchmark datasets are able to reach such high performance by essentially "skimming" words like this, as the language model corrects for any small mistakes. Such a property is likely not useful for this task, as the target data is not in English or any other language. As a result, methods that excel on these datasets likely would possibly not generalize very well to the target data.

Although the text is not of any language, the lexicon of the recognition step is known from the list of possible labels. As a result, there is no need for the algorithm to output exactly correct text. Especially with character-level predictions, it should be possible to evaluate how likely each word on the list is to map to each word found in the panoramas. If such character-level predictions do not exist, it is still possible to treat the output as a black box and take the output words and compare their similarity.

5.4 Filtering and post-processing

The purpose of the filtering and post-processing step is to inspect the recognized text instances and process relevant texts while discarding irrelevant or erroneous ones.

A typical problem with an end-to-end text recognition pipeline is that some output characters get easily confused with optically similar ones: the letter "O" might become the number "0", the letter "I" gets mixed up with the lowercase "l" and so on. In addition, sometimes the recognition model misses letters or inserts additional ones. This means that if one wants to find a known text from an image - for example, one from an equipment list - the text matching algorithm must accept a degree of error, designed with awareness of "similar" looking texts. The easiest and most basic solution for this problem is using some kind of edit distance. Edit distance, as mentioned in Chapter 3, is part of the character error rate measurement.

Edit distance can be utilized without touching the internals of the algorithm at all, treating it as a black box. However, for models that output character-level predictions, it may be useful to also analyze those predictions rather than the output word.

5.5 Training data

In this section methods of transforming data to improve the method are discussed. Typically, deep learning methods do not require a great deal of manual preprocessing unlike traditional methods when it comes to the data itself. Rather, most of the effort is spent on acquiring, labeling and synthesizing more training data.

5.5.1 Utilization of depth data

For an application of text detection and recognition, the availability of depth data is highly unusual. The only relevant application of depth data in end-to-end text recognition is that in SynthText [49], as mentioned previously in Chapter 4, where depth data generated from original images is utilized to improve the placement of text on them for augmentation purposes.

Theoretically depth data could be used in a few different ways: as a part of the input data of the text detection algorithm, thus including it in the detection model itself, or in a more conventional way in postprocessing, using it to filter out text instances with large amounts of depth variance, for example. Using depth data as input for the text detection algorithm could be useful, as smooth and even surfaces are more likely to contain text compared to surfaces with great variance, and the algorithm could well utilize this information for more accurate predictions if given good training data. This further highlights the problem of not having proper training data fit for the task.

As text is inherently two-dimensional, the third dimension provided by depth data is likely not very relevant for text recognition, unlike general object detection where it could be very beneficial for detecting three-dimensional objects. Still, it is possible that a detection model could learn what kind of surfaces the labels tend to be on, and even in the text recognition phase it might be useful input indicating when the detected text box is badly placed, what kind of transformation might have to be applied to detected text to straighten it out, or whether the area is useful at all. Given the rarity of depth data, none of these potential applications have been researched comprehensively.

5.5.2 Dataset

Ideally the training data for a task such as this would be a large collection of labeled panoramas that are identical to those the model is intended to be used on. However, in this specific use case, the data format is very specific and extremely difficult to gather a sizable dataset for, as there is no simple way to collect an entirely representative dataset of the problem without consulting tens of customers for their data and hand-labeling it all. As a result, automatic data generation should be utilized along the customer data.

Based on manual inspection of existing customer data, the most significant potential problem with laser scans is the quality of text within them. Since the labels are often very small and the scans are not originally meant for reading signs and text, the resulting cropped pictures of text acquired from the detection step are often too poor in quality to be recognized properly by a human.

In practice the quality and sparsity of scans taken from facilities is a problem that could be largely solved by commissioning new scans with better equipment. Based on interviews with Cadmatic's employees, given the value that a potential solution would bring, the costs associated are likely reasonable business-wise. It is very rare that the text to be detected is truly too small to be found in a scan taken at short range. Even using non-panoramic pictures or entirely different hardware for the task is considered viable.

5.5.3 Augmentation

It is possible to generate synthetic data for the purpose of training the model. As mentioned previously in Chapter 4, the synthetic text generation algorithm SynthText as described by Gupta et al. [49] requires depth and segmentation maps produced based on the images. In this case, interestingly, real depth maps can be utilized for such an algorithm for generating synthetic training data.

The usefulness of augmented training data for this purpose is not known, however. In most papers, it seems the SynthText dataset is used to generate a baseline model which is then finetuned with an industry dataset. Whether augmented data is useful for finetuning remains unclear, as the dataset used for finetuning should most closely resemble the real input data.

5.6 Further questions

Based on these findings, a few questions are raised that could be answered by practical experiments.

First, is it possible to attain good results with out-of-the box text detection and recognition methods? In principle, most text detection methods should be applicable to the task, yet a subset of recognition methods, namely those without strong language models, should be effective. How should those methods be configured and combined?

Second, is small text a problem for text detection methods? In the past, consistently detecting small objects has been problematic for object detectors. Perhaps the same holds true for text. Most of the data consists of very small text instances.

Last, as for building a practical solution, which parts of building a practical solution are most likely to be problematic? What should be the focus of continued development?

6 Experiments with pretrained models

This chapter is about practical trials and experiments conducted to ensure that solving the problem is realistic, and to assess how accurate current methods potentially are at the facility text recognition task. These insights should be useful in determining how to continue development further after the thesis. The experiments performed are mostly based on a small custom dataset of flat images with labeled text instances, and out-of-the-box pretrained models.

First, some initial experimentation is covered, attempting to discover the nature of the problem; then, different methods related to the step of text detection are explored with analysis on their results, especially regarding changing the confidence threshold; lastly, methods related to text recognition are covered with some analysis as to what challenges lie in combining the two separate steps.

6.1 Initial experiments and problem discovery

The initial period of experimentation focused on problem discovery and quick evaluation of out-of-the-box solutions. In this phase, the main purpose of the work was to understand the general state of text detection and recognition at the moment, as well as gauge the practical difficulty of the problem. Some of the most popular individual methods were thus applied to the test images in order to gain first insights.

Initial attempts to utilize text recognition were conducted using Tesseract OCR [50], which is an optical character recognition system in development since the 1980s primarily meant for text documents, recently mainly worked on by Google. Both conventional OCR methods as well as models based on deep learning are included in the software, with various options on how to segment the text and which model to use. When an attempt was made to use it as an end-to-end solution on a few "easy" handpicked loosely cropped pictures of text, the results were very poor - only the simplest text instances were recognized in any degree, and the examples that were had tons of mistakes. It became clear at this point that Tesseract could not be used in a "plug and play" fashion for the desired purpose, and that it would require configuration based on the provided user manual [51]. Even when different settings such as alternative segmentation modes and models were tried, no meaningful results could be obtained unless the text was unusually large in size and laid on a solid background. Thus, it seemed Tesseract was especially sensitive of background clutter as well. As a result, no further experiments were conducted. Tesseract has been mainly developed for scanning documents, and even its more modern LSTM models do not seem to generalize well to more complex images containing text.

Next, a pretrained model [52] of EAST [20] was evaluated for the text detection step. The original idea was to combine the EAST detector with Tesseract as the recognition algorithm to form an initial script for end-to-end experiments. A similar setup is used by Mai et al. [38] In this application Tesseract, despite not being good for the end-to-end task, was assumed to be able to produce better results when fed tighter cropped and perhaps preprocessed text instances, as there were many examples online of this being attempted successfully. In the end, when combined, many text instances were detected in a promising manner by EAST, but Tesseract OCR operating on them still did not produce convincing results. This implied that "basic" text detection was likely easily workable, while better and more elaborate methods would be required for recognition.

Some small initial experiments was also conducted on available pretrained end-to-end and detection models, mainly FOTS [39] and PGNet [53]. The basic results of these trials were that many text instances were detected and even recognized somewhat accurately, confirming that some useful degree of end-to-end accuracy should be attainable with more recent and elaborate technologies. PGNet's results were especially promising on oddly shaped and rotated text instances, which make accurate detection and recognition quite difficult.

Similar to Tesseract, a brief attempt was made with the pretrained recognition models from the GitHub repository [54] of the text recognition comparison compiled by Baek et al. [31]. Again, the initial results seemed very poor as much of the text in the dataset images is of very low resolution and did not seem worth pursuing. Evaluating models from the MMOCR repository [55] produced similarly poor results.

The main learnings of initial experiments were that deep learning -based text detectors looked promising in general, but whether they could produce bounding boxes of sufficient accuracy remained unclear. Text recognition, on the other hand, remained an open question, and understanding the reasons for its continued failure could be key for implementing an end-to-end solution. Conventional and most popular recognition methods would likely not perform well enough, and rather a closer study of state-of-art methods would be necessary to attain acceptable results.

It also seemed that the lack of domain-specific training data could become be a significant limitation, as deep learning methods require a large amount of representative training data to realize their full potential. This meant that development would need to be focused on out-of-the box pretrained models, existing datasets and possibly augmented data.

6.2 Test dataset and analyzing old scans

Some existing laser scan data from Cadmatic's customers was briefly analyzed regarding whether or not the data seemed good enough for text detection and recognition. While the specifics of that data are confidential and cannot be shared as part of this thesis, this analysis concerns panoramas scanned from more than 15 facilities over many years. The criteria for acceptance were as follows: any included image had to be in color, text must be visible in the image, and the found text must be human-readable.

When these criteria were applied to the data, recent high-resolution imagery was found to be most suitable for text recognition, with older scans less likely to be included. By far the most common reasons for non-inclusion of older scans were poor quality and lack of readable text instances; the former problem also worsens the latter. In a few cases, the scan seemed monochrome without color data; while text detection from these scans might be possible given sufficient resolution, they were left out of scope due to their obsolescence and likely future irrelevance.

It was found that when it comes to existing, older scans, this selection criteria easily weeds out the vast majority of laser scans. While this result seemed initially concerning, it was quickly discovered that laser scans produced by modern scanners could be much higher quality than the old data lead on. These insights about modern laser scans were verified through interviews with Cadmatic's parent company's employees responsible for performing these laser scans and inspecting examples of scans they had performed: the resolutions of scans could be the likes of 10,000 x 20,000 pixels with HDR imaging, as opposed to older ones closer to a scale of 2,000 x 4,000.

Out of these feasible panoramas, a small hand-picked custom dataset was compiled as a test set, containing 22 screenshots and 94 hand-labeled text instances. These images are perspective projections taken from panoramas, mostly consisting of signs and labels of small text in typical scenarios. Most of the images contained several instances of text.

6.3 Text detection

With the dataset compiled, several out-of-the-box text detection models were compared to get a basic understanding of which methods would be most fitting for the use case. These models were tried out-of-the-box pretrained on standard datasets, most of the models coming from MMOCR [55] which includes several text detection models, but a public implementation of EAST [52] was also evaluated. This gave some of the methods quite an uneven playing field, as the same datasets had not been used for training all of the models. As showcased on the study on text recognition by Baek et al [31] for instance, this kind of inconsistency can quickly lead to totally invalid comparisons when it comes to evaluating the relative performance of different methods. As such, this data should probably not be used to aid in definitely choosing a specific model, but rather which models are most promising for further trials, and whether the task is feasible at all.

6.3.1 Methodology

Detecting text instances is somewhat different compared to detecting objects, as any instance of text consists of smaller instances of text. For example, a paragraph may contain several sentences, themselves containing several words, which themselves contain letters. Text is more of a class of content within the image than a specific object. Therefore, given an area of text, it can be argued that there are multiple different but equally correct ways to detect the text instances: multiple ground truth text instances may be detected as one instance, and one ground truth instance may be detected as multiple. Both of these ways to handle multiple words could result in valid detections as seen in Figure 6.1. As a result, there are many different methods of scoring text detection, several which are covered and improved upon by Liu et al. [56]. However, in this case, an extremely simple if somewhat flawed method is used for this comparison due to its simplicity:

1. A prediction is considered valid if ground truths cover more than half of the prediction box area. Predictions which fall outside of this criterion are considered false positives. This means that if over 50% of the covered area is text, the detection is considered valid.

2. A text instance is considered detected, if valid ground truths cover more than half of its area. Any text instances satisfying this rule are considered true positives. Instances with no detections are considered false negatives.

There are a few reasons as to why this detection criteria, which combines one-to-many detection and many-to-one detection, is useful. First, it allows for a quick binary classification of both the detections and the ground truths. This makes it easy to quantify the performance of the methods in an easily understandable manner: for instance, accuracy and recall are easily calculable. A second upside is that it allows for the evaluation of ensemble methods, as additional detection boxes from different methods can be added to the result without much downside.

This set of detection criteria however does have several shortcomings as well. For instance, methods which detect multiple lines of text as one instance will receive the same score as methods which have more granularity in their detection, despite

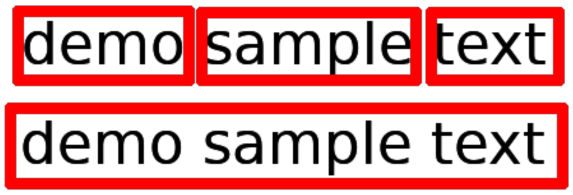


Figure 6.1: Example of two correct ways to label the same text

the difference in real world usefulness, as most recognition methods work on the assumption the input text is in form of a line. Such a failure case is showcased in Figure 6.2, where one prediction "detects" two text instances while covering neither sufficiently.

A related downside of this detection scoring is that it is not affected by the accuracy of the produced bounding boxes beyond the initial filtering; the detections are not scored based on their usefulness. These criteria also do not take multiple detections of the same text instance into account.

Still, almost none of the text in the custom dataset is laid out multiline, and when looking over the detections, none of the methods seem to fall into these traps in any significant way.

While each method was evaluated with an out-of-the-box model, all of the methods also included an adjustable confidence threshold parameter that would filter out the less likely detections. Experiments were mainly conducted with the default setting of this parameter (0.5), but different settings were also compared in some cases. In addition to the detection rate, the sizes of detected and undetected text instances were also output and compared to each other to get an idea about whether text size is a relevant problem for detection models.

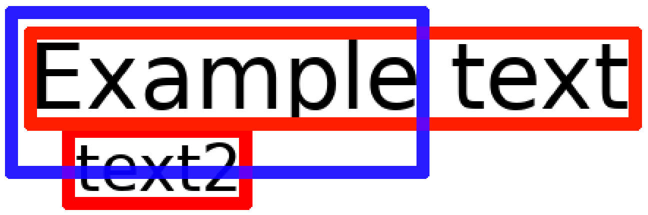


Figure 6.2: Two ground truths (red) can be erroneously "detected" by one prediction (blue).

All of the models used are listed in Table 6.1. Each model was run on the same set of images and scored based on the aforementioned criteria. These methods come with a common adjustable parameter - confidence threshold, which is used to discard instances less likely to be text.

Model name	**Training set**	**Source**
DRRG	CTW1500	[55]
DBNet	IC15	[55]
FCENet	CTW1500, IC15	[55]
Mask R-CNN	CTW1500, IC15, IC17	[55]
PSENet	CTW1500, IC15	[55]
TextSnake	CTW1500	[55]
janzd-east	IC15+13	[52]

Table 6.1: List of models evaluated, with their sources

6.3.2 Results and analysis

The recall of the methods with confidence threshold set 0.01 was compared first, as seen in Figure 6.3. In this case, it seemed that FCENet, TextSnake, EAST and especially Mask R-CNN performed the best when using a low confidence threshold, all of them scoring over 50 detections on one of their variants. However, especially TextSnake and Mask R-CNN produce a very notable amount of disjointed false positive areas despite their high detection rates. Overall, these results indicate that text detection methods have strong potential for this task, even without domain-specific training data.

With the thresholds set low, the results imply that extremely high detection rates can be attained at the cost of even higher false positive rates. As adjusting the confidence threshold affects both the rate of true and false positives, the choice of a good confidence threshold value will depend on how detrimental false positives are

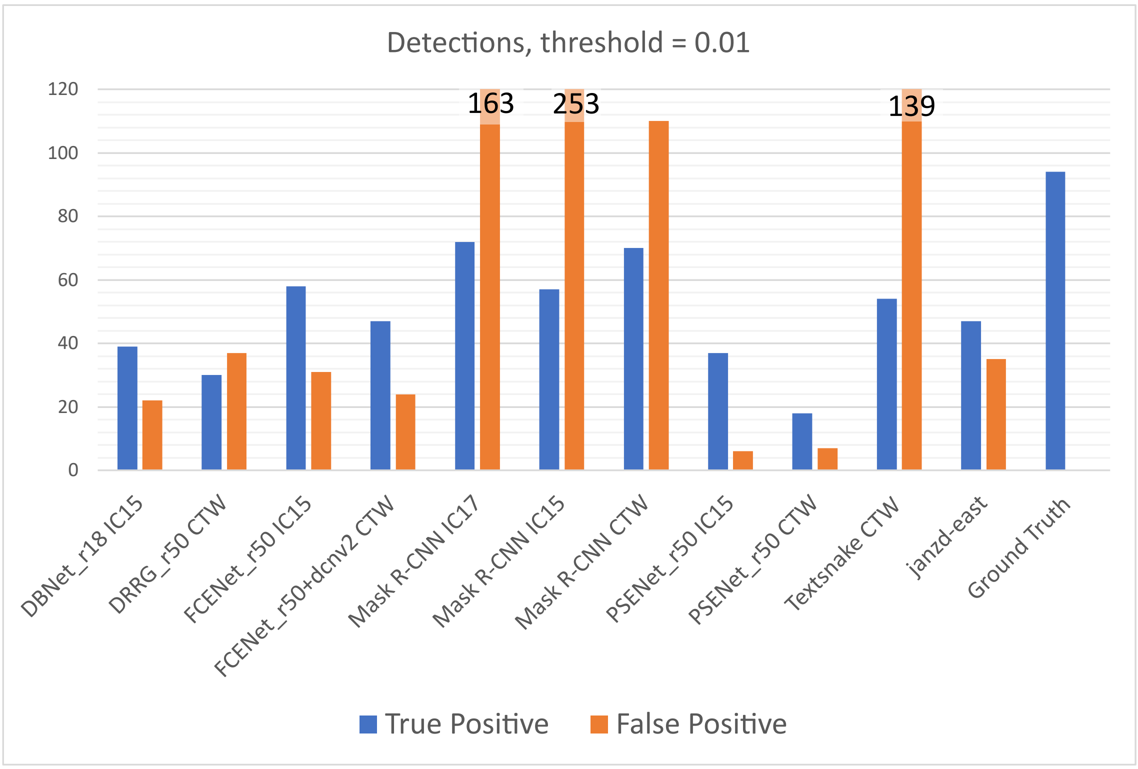


Figure 6.3: Absolute number of detections, 0.01 threshold

to end-to-end recognition as a whole both in terms of performance and recognition errors.

Adjusting the confidence threshold

Figure 6.4 displays the effect that changing the confidence threshold has on recall. This subset of models has been chosen to showcase how differently the threshold can affect different models: Mask R-CNN, an object detection -derived model, scales all the way down to 0.01, while the others seem mostly unaffected after threshold 0.4-0.5. Generally, it can be seen that if the model scales like Mask R-CNN, the default setting of 0.5 is by no means optimal for reaching maximum recall. As long as false positives can be dealt with, much better results could be attained with lower thresholds.

Many methods seem entirely unaffected by thresholds lower than 0.5. Some of

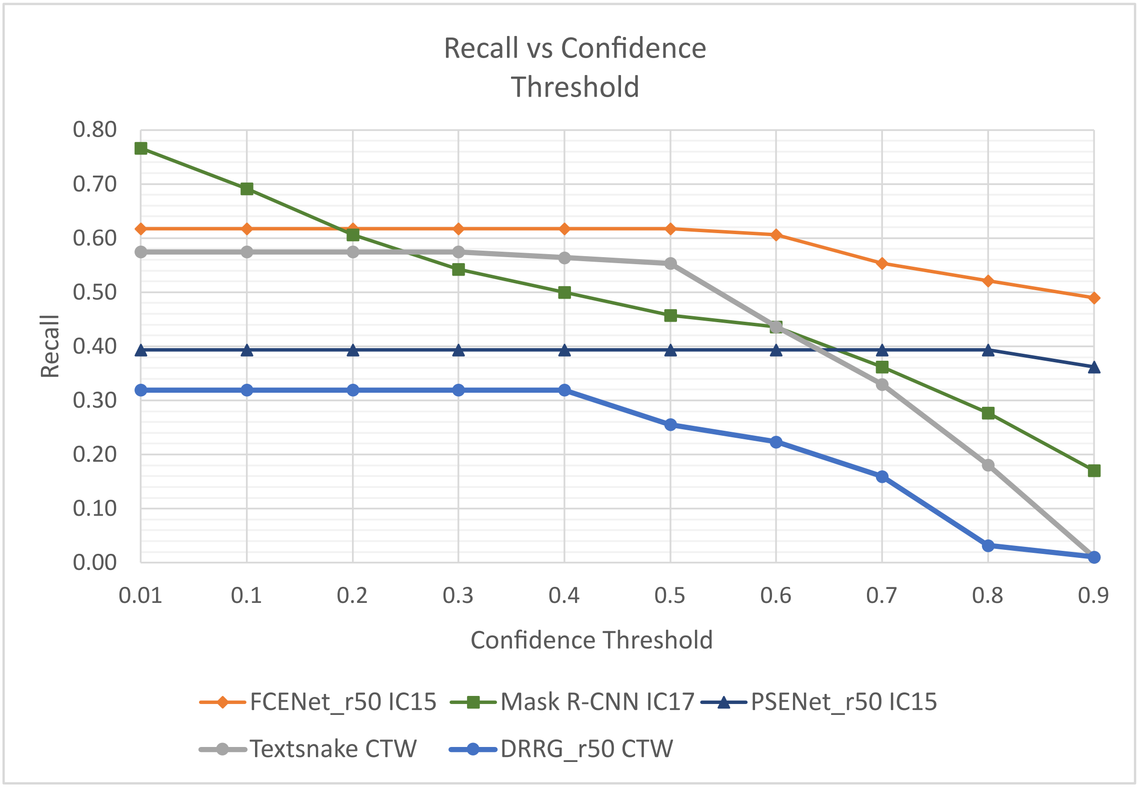


Figure 6.4: How different methods' recall responds to changing the threshold

these methods, such as DRRG, have internal thresholds or other parameters which do not act on the level of the entire detection. Similar parameters can probably be adjusted for those methods to get results better matching those of Mask R-CNN, but for the sake of simply studying feasibility, individual methods were not studied in such detail.

F-score, a measure that gives equal weight to precision and recall, can be used to measure how confidence threshold affects the results as whole. Figure 6.5 shows an example of how changing the confidence threshold affects precision, recall and F-Score all at once, in this case for Mask R-CNN, and how choosing the right value clearly depends on false positive tolerance as high recall values are accompanied with low precision.

Because in this specific application it is assumed an equipment list exists and can be used to filter the resulting text, having wrongly labeled text instances is not considered a huge detriment to the model. By far the largest downside would be the addition of hundreds of useless text instances that could possibly slow down the processing time.

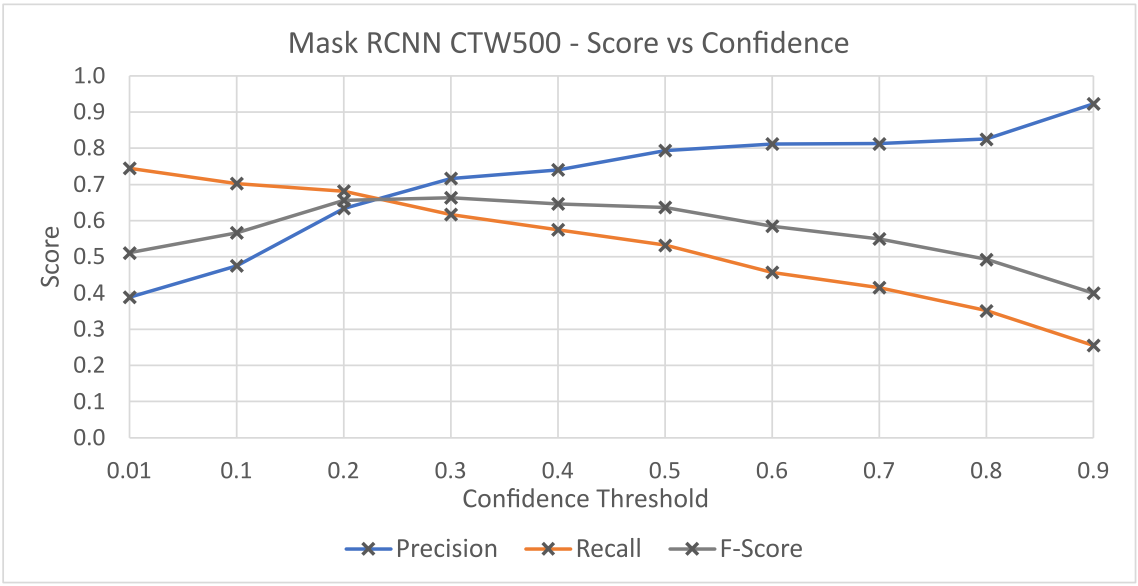


Figure 6.5: How Mask R-CNN responds to changing the threshold

Using an ensemble method

As an experiment as to whether these methods are generally detecting the same or different text instances, a small experiment on the possible effectiveness of ensemble learning was performed. Several methods were combined to one ensemble. The models chosen were the highest performing models at a threshold of 0.5 - Mask R-CNN CTW, FCENet CTW and TextSnake primarily. When combined to an ensemble, the collection of models beat each individual model in recall while losing precision, although not as badly as Mask R-CNN CTW at 0.01 threshold. These results can be seen compared to Mask R-CNN in Figure 6.6.

It was found that most instances found by low-recall methods would find the same text instances as the high-recall ones, with combining them only raising the total toll of true positives by few to none. However, when combining the best methods in the 50-70 detections range out of 94, a result over 80 appears to be attainable with an ensemble as long as hundreds of false positives are acceptable or possible to filter

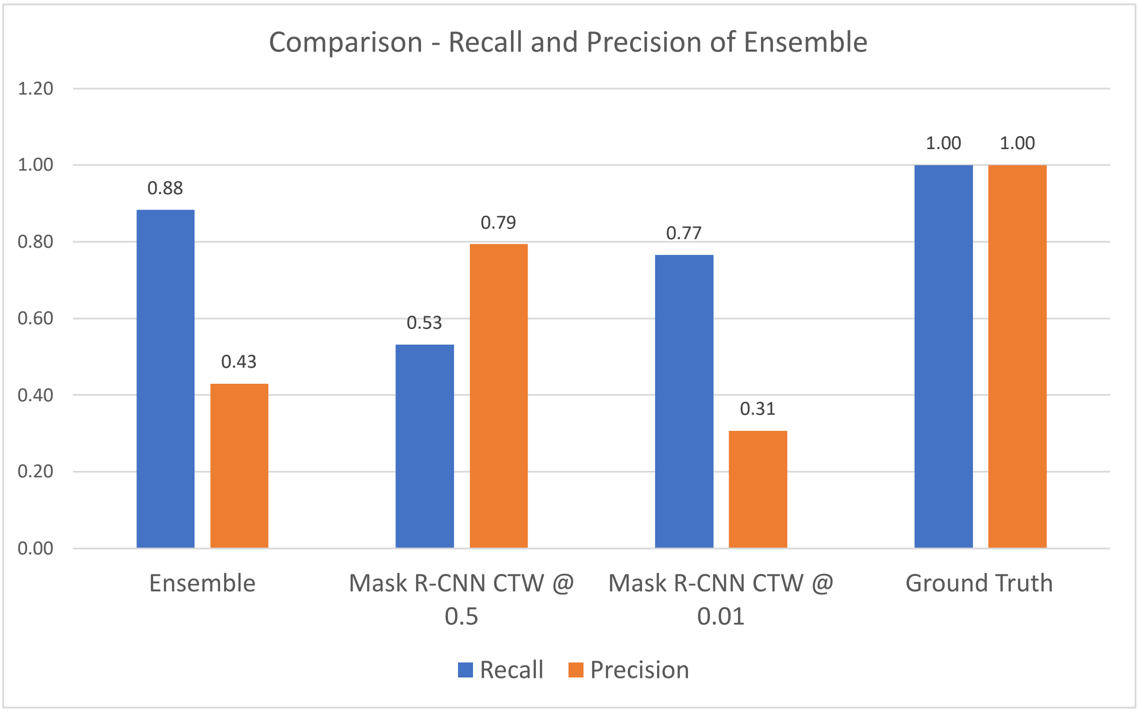


Figure 6.6: Metrics of the ensemble method compared to Mask R-CNN

out somehow. These results indicate that combining several different methods to an ensemble may lead to slightly higher detection counts than using a single method.

Dataset selection

The models displayed large differences in their performance depending on which dataset they were trained on. No specific universal "best match" dataset could be determined from these results, as which dataset behaved best varied, but CTW1500 and IC17 datasets performed especially well compared to IC15. This data is inconclusive, as many other datasets or a combination of them could be used for training. An example of direct dataset comparison is shown in Figure 6.7, where Mask R-CNN produced best results with the two aforementioned datasets.

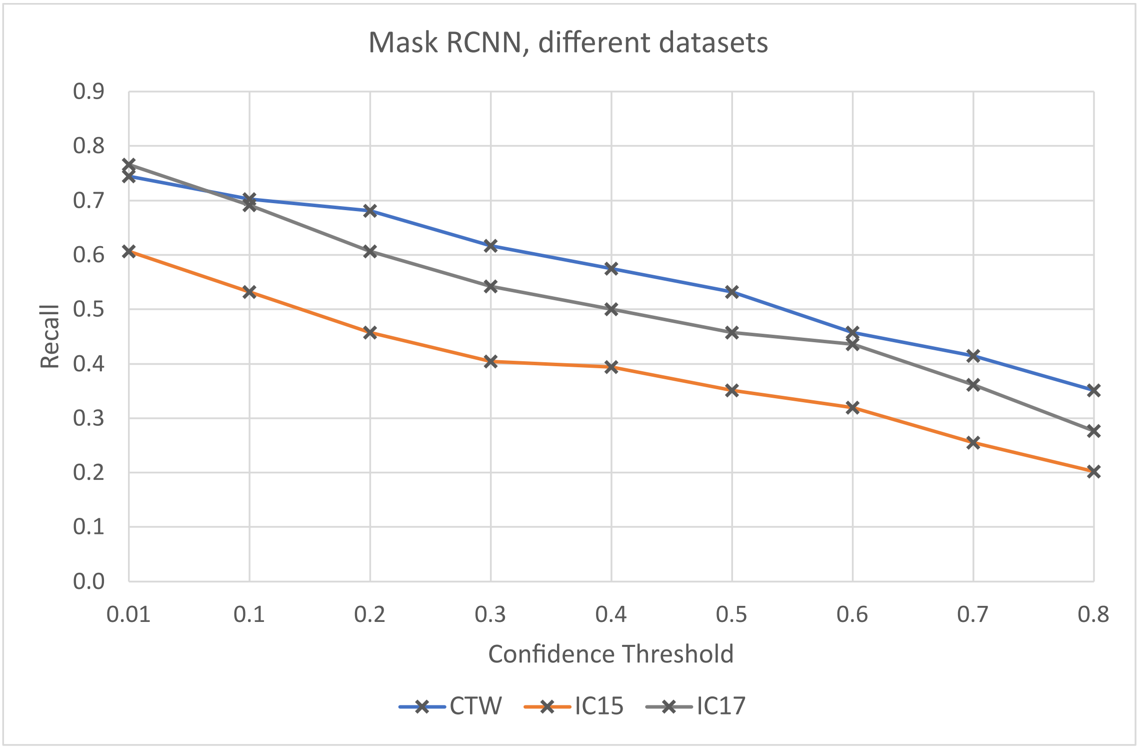


Figure 6.7: Comparison of Mask R-CNN results on different datasets

6.3.3 Breakdown by size

In Figure 6.8 EAST, Mask-RCNN, TextSnake and the ensemble method are represented as distributions of text instance size in pixel area, with true positives and false negatives shown separately. These results indicate that the smaller text instances tend to be more difficult to detect as large false negatives seem rare. Therefore, small text detection still remains an important part of solving text detection in facilities.

The ensemble method failure cases are those that are not detected by any method of the ensemble. It can be seen that the distribution of undetected text instances is much different and on a lower range than that of the text boxes in general.

These results indicate that small text detection is still problematic in modern text detection approaches when using standard datasets of text. It also implies that for such data where small text instances are very common, special measures should be taken to improve small text detection. This is especially true for public pretrained text detection models which are trained on academic datasets that tend not to include small text instances.

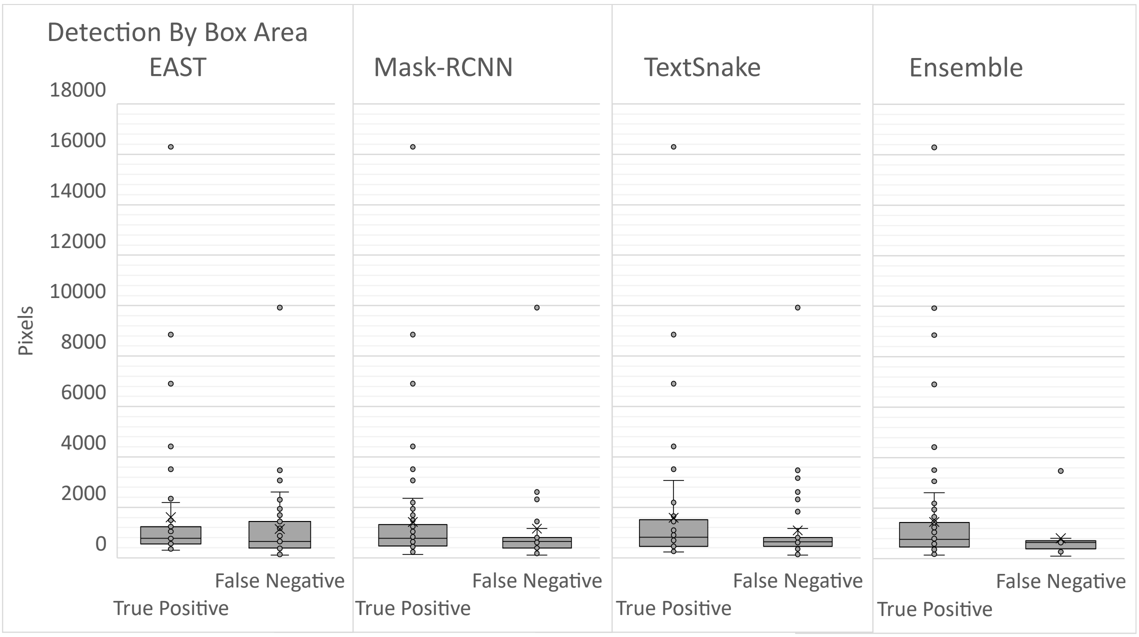


Figure 6.8: Distribution of detection box sizes for each method

6.3.4 Possible improvements

The greatest limitation of the models themselves was the requirement of using pre-trained models due to a lack representative training data of the problem domain. A dataset of labels within an industrial plant simply does not exist; even simple information about how these facilities look inside is not openly shared information but had to be acquired through interviews and reviewing data from Cadmatic's customers. Thankfully, any pictures of text in varied and cluttered scenes are likely good enough for a detection training dataset that would produce decent results when applied to facility data.

While the algorithms find many text instances, combining data from multiple algorithms requires some kind of processing, as simply taking the box output is not good enough. Non-maximum suppression [11] (p. 40) is a common approach taken to combine detection boxes to one instance, but whether or not this is suitable for text is unclear.

The dataset did not contain any vertical text instances, although study of other Cadmatic materials suggests those do exist in facilities. Since the text instances were horizontal, detection performance on rotated text could not be assessed. Some detection methods are supposedly better at detecting different orientations of text, producing irregular-shaped detection areas rather than bounding boxes [13]. In this use case, as the processing time is not badly constrained, the same image can be processed in several rotations from the very beginning.

The results do not measure the detection boxes' suitability for text recognition. The detections output by the models are significantly less accurate than the ground truth boxes and datasets typically used for measing recognition models. Poorly cropped detection boxes would lead to lower end-to-end accuracy due to more errors in recognition phase.

6.4 Text recognition

Like text detection, text recognition was also evaluated. The idea of text recognition experimentation is similar to that of text detection: inspect state-of-art and industry standard methods and evaluate the results.

First, the significance of preprocessing is discussed as the intermediate step between detection and recognition - what kinds of processing could be applied to detection outputs to increase the likelihood of recognition success. After that, some publicly available methods are tried on the custom dataset mentioned previously, and the results are analyzed. Finally, some ideas are offered as to how to improve the results.

6.4.1 Preprocessing

As a separate step preceding text recognition, text detection outputs text instances which are effectively coordinates of 2D boxes containing text. Before being fed to the text recognition method, these boxes have to be properly extracted out of the image, rectified and/or otherwise preprocessed to reduce distortion and irrelevant data that may worsen recognition accuracy.

End-to-end methods can also have such rectifying steps between the detection and recognition branches. FOTS [39], for example, includes a bespoke built-in method, RoIRotate, taking this into account.

Conventional models like Tesseract are heavy on image preprocessing. Tesseract does thresholding internally, trying to binarize the input image [57]. Similar preprocessing steps may be useful for deep learning models, especially if the preprocessing makes newly encountered examples appear more like training data.

In this phase of experimentation, custom preprocessing was not deployed between the detection and recognition phases although some text recognition models include such methods.

6.4.2 Initial experimentation

The initial experimentation covered many different kinds of text recognition models. Most modern text recognition models follow an encoder-decoder architecture, which tend to learn rather strong language models. In theory, it should be best to use a model that does not learn implicit assumptions about language. However, sequence-to-sequence models that learn language information are clearly the state of the art at the moment, as they produce best results on typical text, and as such consist most of the evaluation. In this phase, text instances directly output from a detection method were used - specifically those output by Mask R-CNN in the previous sections.

Preliminary experiments indicated that this time that pretrained models within MMOCR [58] would not produce sufficient results on text instances, as most of the text instances fed to the models did not at all resemble their ground truth counterparts. This result was surprising, given the high accuracy of the models on academic datasets.

Models from the previously mentioned GitHub repository [54] of Baek et al. [31] were also evaluated in this phase, with the models achieving similar performance to MMOCR. However, it was notable that rather often the implicit language model learned by the model was attempting to read the alphanumeric codes as English language words instead.

Results of text recognition on text instances output directly by the detector seemed very poor - almost always entirely unrecognizable. Although these results were not recorded in a statistical manner, it was clear from manual evaluation that utilizing the recognition models as-is on data output by the detector would not be viable. The bounding boxes from the detection step would likely need to be more accurate, or require some kind of preprocessing for improved performance; thus, later experiments were conducted with hand-labeled text instances instead.

6.4.3 Text recognition comparison

A second comparison was conducted in order to gain statistical insights about the viability of recognition assuming perfect detection results. Its findings could help pinpoint whether the poor results of the previous experiments are caused by the inaccuracy of detection or the recognition models themselves, and to evaluate whether there are notable differences in the accuracy of older and newer methods on real-world data. The used methods are listed in Table 6.2 with their original training datasets and sources.

The models of CRNN [55] and Tesseract [50] were chosen to represent the older technologies often used in practice. They contain LSTM-based models, which were common previously before the attention mechanism became popular. In direct comparison, SVTR [59], ABINet [55], Master [55] and TPS-ResNet-BiLSTM-Attn (TRBA) [54] are chosen to represent technology closer to the current state-of-the-art, deploying attention- and transformer-based models. All of these models are measured using publicly available code and pretrained models. Most of them are trained with the same few synthetic datasets.

As the results of initial experiments were poor, a more rigorous experiment setup was created to measure text recognition methods. First, a cleaner recog-

Model name	**Pretrained Dataset/Settings**	**Source**
Tesseract v4	Default settings	[50]
CRNN (MMOCR)	Syn90k	[55]
SVTR (PaddleOCR)	MJSynth, SynthText	[59]
ABINet (MMOCR)	Syn90k, SynthText	[55]
MASTER (MMOCR)	Syn90k, SynthAdd, SynthText	[55]
TPS-ResNet-BiLSTM-Attn	MJSynth, SynthText	[54]

Table 6.2: List of recognition models evaluated, with their sources

nition dataset was created based on the detection dataset utilized in the previous section, with ground truths directly cropped out of the images rather than being extracted by a detection model. Then, the produced text instances were fed to the recognition models, with a case-insensitive Character Error Rate and recognition rate (accuracy) measured from the model output in comparison to the ground truth.

In this specific experiment, ABINet and SVTR produced the best results at 32.5 and 33.7 CER respectively. As expected, the Tesseract model produced by far the worst result, 92.9 CER with 0% recognition rate, with others falling in-between. Even the best model, SVTR, only produced the exact correct result in 28% of cases, with ABINet and MASTER close behind at 22% and 17% respectively. These results can be seen in Figure 6.9. Interestingly, SVTR managed a higher recognition rate in this experiment compared to ABINet despite the two methods' similar CER results.

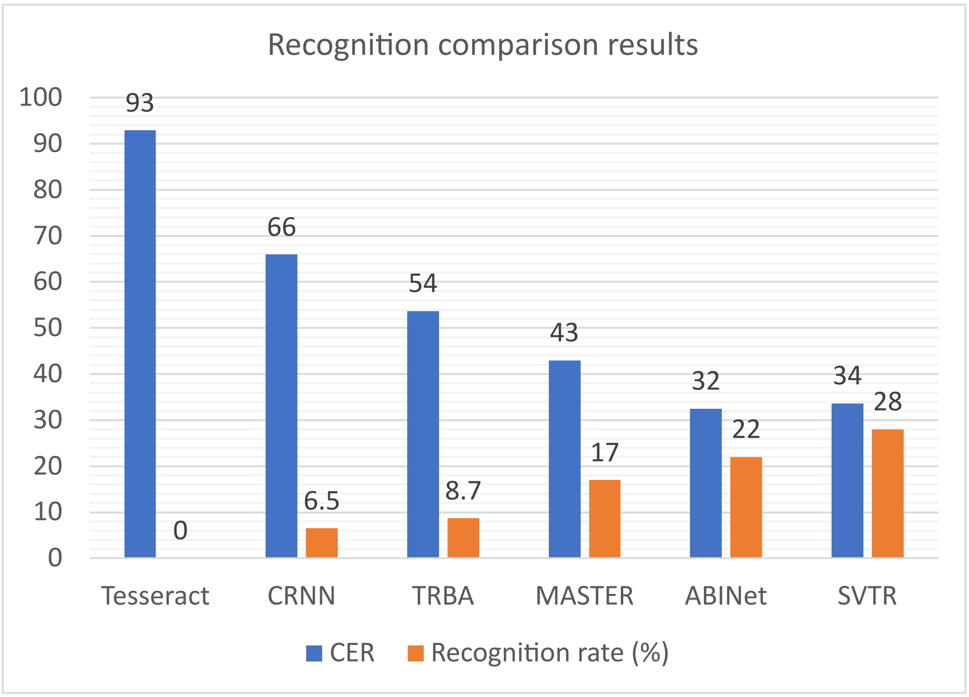


Figure 6.9: Results of the recognition comparison, with measured character error and recognition rates.

6.4.4 Analysis of the results

At first glance, the results indicate that modern methods produce much improved results compared to older ones - attention-based models scored much higher than those using LSTMs, with ABINet and SVTR proving especially strong. Interestingly, ABINet includes an explicit language model, which was an aspect theorized to detract from recognition results on such non-language labels, yet it was one of the top performers. It is possible that the language model contributed to its lower recognition rate compared to SVTR, but the difference in results is fairly insignificant and difficult to judge based on one experiment.

Something to note regarding these results is that equipment labels are often extremely similar to each other, often differing only by one character. For example, when an image containing the code "PF01" was fed to the TRBA model, it produced the prediction of FFOT - optically similar, but entirely different in CER due to the numbers replaced with English characters. When the best models are compared, the results of such individual detections can vary greatly: the SVTR model recognizes "PF01" as "P0OP" while ABINet produces the exact correct prediction.

Therefore, simply studying the CER or recognition rate may not be sufficient in ensuring that the most important data is retained, but practical trials are required to ensure that the models can discriminate between similar alphanumeric codes.

Overall, in these experiments detection and recognition models produce decent results individually as far as academic benchmarking is considered. However, when realistic detection results are fed to recognition models, their accuracy ends up being worse than expected. There are a few possible reasons for this.

First, the recognition models have been trained with datasets very dissimilar to detection results. Their original training data is extremely tightly cropped and hand-labeled. The accuracy of these bounding boxes is much higher than those of text detectors, which tend to produce inaccurate detections. This helps explain why pretrained models from Baek et al. seemed to have slightly improved results in comparison to the alternatives, as their models specifically add rectification to the recognition pipeline. It also explains why FOTS - a method that uses shared feature maps with an internal rectification step - was earlier able to attain somewhat promising results despite utilizing an encoder-decoder architecture for recognition just like the failed separate recognition methods.

Second, it is unclear how well academic datasets and their text generalize to data from the facility environment. The training datasets are often synthesized or from environments very different to facilities, with the text instances themselves also greatly differing from the facilities. Especially notable is that the training datasets tend to contain text instances that are larger in size but more typographically diverse.

6.4.5 Possible improvements

Given the combined results where both detection and recognition steps seem decent in academic benchmarks on their own but not when combined in practice, the most likely performance bottleneck is the integration between the two. Most likely the data passed from detection needs to be processed further for the recognition step to work correctly. End-to-end recognition methods were not evaluated thoroughly, but they solve this very problem - there is tight integration between the detection and recognition parts of the pipeline, and they can be trained end-to-end on the same dataset.

The largest drawback of this entire approach is the inability to use our own dataset for training a model. Possibly huge improvements could be attained from compiling a better labeled dataset of text in industrial plants, and then using that dataset to train an end-to-end model leveraging modern developments. The synthetic data used for training the recognition could also be modified to appear more like text found in facilities.

If a two-step method was preferred instead, best results could most likely be reached with text recognition methods that contain preprocessing and rectification steps and are thuss less prone to errors due to inaccurate bounding boxes. At the very least, the text recognition models should be trained with data that resembles the output of a real text detector rather than the perfectly cropped bounding boxes often seen in academic datasets. These improvements could be reached by either generating the training dataset from the output of the detector or augmenting synthetic errors similar to those of detectors into the training data.

7 Analysis and discussion

The purpose of this chapter is to provide an overview of the results of the thesis and describe how they should be interpreted: what next steps should be taken, what aspects of a potential solution should be paid most attention to, and what research topics could be studied in the future. It presents the main findings of the thesis.

7.1 Practical solution and business case

In terms of the business case, a single glaring problem remains: it is not perfectly understood how comprehensive coverage of equipment the found text labels provide. Even if the recognition was perfect, would the resulting solution be useful? There are a few unanswered questions that contribute to this problem.

The first question is whether the label coverage provided by the laser scans themselves is good enough. As discussed in Chapter 2, a real facility likely contains many labels that are not visible in scans: for instance, valve tags, which are mostly small metallic plates with poor contrast, cannot typically be seen in the images. Even in cases where labels are "good" in concept, they are often too far away from the scanner to read. If a more complete ground truth list existed, consisting of the labels of the entire facility rather than just ones visible in the scans, one could better assess how many of the required labels are actually visible in the panoramas. At the moment, it is not possible to estimate how many of the total facility labels are included in the scans.

The second question remaining from Chapter 2 is whether the labels that can be recognized are relevant. While it is clear that only a subset of the facility's labels can be read, it is not exactly known how relevant this subset of labels is for the customers' use case. In practice, only the text instances with largest size and highest contrast are visible and human-readable in these scans. Many of these large texts consist of warning signs and other information that is not useful for human identification of equipment. If it turns out that the small unreadable valve tags are more useful than the readable large labels, the usefulness of the solution is clearly limited.

To answer these questions and gauge how useful a potential solution would be, customers should be asked to participate in the project. The relevance of text labels visible in panoramas would be easiest evaluated through a study of several facilities and their point clouds: for instance, it would be insightful to compare an annotated map of a facility to the point cloud scans taken from the same plant to see how well they match. Including people familiar with the customer facilities is vital to evaluating the solution's eventual end-to-end usefulness regardless of how good its eventual technical implementation is.

7.2 Technological basis

As for technology, the right way to deal with panoramic images should be decided. Panoramas, as described in subsections 2.2.1, and 4.4.6, can be naively processed as singular large equirectangular images, but depending on the method, this may be inefficient or even impossible due to their large size. As there is no standard solution to this problem, it can be approached in several ways: for instance, stereographic and cube map projections have been previously utilized in text and object detection in various contexts. Another approach is to take perspective-corrected "shots" out of panoramas, which can provide with a tradeoff of worse overall processing time for easier text detection. On its own, equirectangular projection is especially poor for labels situated at the top or bottom of the image as detection models are trained on typical perspective images.

Implementation of the machine learning system should start with the most well known and most implemented state of art methods. In Chapter 6 EAST and Mask R-CNN were found to have good detection performance and likely usable out of the box for this purpose. Text recognition provided much more trouble, with CRNN-based methods appearing most promising for text recognition.

As noted in Chapter 6, EAST especially appears to be the most commonly implemented and utilized text detection algorithm based on the number of practical solutions, blog posts and implementations available online. There are several open-source and out-of-the-box available models for text detection use. For text recognition or OCR, a similar "popular model" that generalizes well appears not to exist. Tesseract is commonly seen in practical contexts, but as it relies heavily on conventional image processing and is mostly meant for scanning documents, its performance does not appear to be sufficient. This is especially notable in cases where the text has poor contrast on a mixed background. Encoder-decoder methods, including those using attention, were found to be most promising and most studied based on the research behind them, but meaningful results could not be attained. However, the results of recent research in this space suggest that attaining decent results is possible with more effort put into customizing the solution. It should be noted that the used text recognition model should not have a strong explicit or implicit language model, as those are likely to worsen results on non-English alphanumeric codes.

It was also found that end-to-end models with recognition architectures similar to standalone recognition models performed better than separate recognition steps. This could mean that integrating separate text detection and recognition methods is a bottleneck for end-to-end accuracy, and that more research should be conducted on integrating them together.

Once text has been read to a text form, it is likely to have multiple errors. Its similarity must be compared to the ground truth, and thus a measure of similarity must be developed to check which expected text instance it represents. For this purpose, edit distance is the easiest and most common algorithm. Likely the easiest type of edit distance would be Levenshtein distance, optionally with some weighting taking optical similarity into account. More elaborate edit distance schemes may provide slightly better results.

The work so far indicates that a solution should be technologically feasible, but implementing a working solution requires more work and practical knowledge of the topic.

7.3 Possible future research topics

While results with out-of-the-box methods likely will not be satisfactory for the use case, fortunately there are some promising directions to study to improve the results.

While the detection results are already quite good, they could be improved to aid recognition. Using a combination of several different methods, or finetuning the model using facility-specific data should be considered. In Chapter 6, the results implied that it is especially important for the subsequent more difficult recognition step that the text boxes output by the detector are accurately cropped and oriented; thus, improving detection results should always improve recognition results as well. Logically, the detection step could be also improved using depth data, but this has not been studied.

On the other hand, if recognition performance is not good enough, this may be down to several factors. As these models perform very well on academic datasets, the first thing to do is to study what the differences are between the practical text instances and the academic ones, and whether these can be corrected by including another intermediary step in the recognition pipeline. Like with detection, finetuning with a dataset specific to labels in facilities could also drastically improve recognition performance.

In one line of research, these "conventional" separate methods could be evaluated against end-to-end models that share feature maps. As covered in Chapter 4, recent end-to-end models are easier to train and are potentially superior in performance in comparison to separate detection and recognition steps. At the moment, building a solution with separate detection and recognition steps is easier from a practical standpoint due to more available implementations.

A constant problem with applying deep learning to this application has been a lack of training data specific to labels in facilities. One path of research would be to study whether synthetic generation of training data on top of images with no text would be useful for this application. The inclusion of depth data in laser scans should make it easier to superimpose convincing text images.

8 Conclusion

Scanning labels out of panoramic laser scans from facilities is a new application of end-to-end text recognition. In concept, its purpose is to find the names of equipment contained in the scans, bringing point clouds closer to CAD 3D models in their usefulness in connecting to external data in information management software.

End-to-end text recognition is a subfield of deep learning combining the tasks of text detection and recognition. Text detection and recognition themselves borrow elements from several other areas of machine learning, such as object detection and natural language processing. Research on the individual subjects is very widely available, yet very little of it is specific to practical combined applications such as this one. The problem is atypical compared to other text detection and recognition applications, as panoramas are used rather than perspective images, and the text to be recognized is relatively smaller and on a more cluttered industrial background. However, the task does have some significant advantages as well, most notably the knowledge of the list of text labels in advance.

Adapting text detection and recognition to the task of locating labels in industrial facilities appears to be entirely feasible. In practical experimentation, it was discovered that the greatest challenge in its development is applying text recognition to the detection results. However, end-to-end methods produced promising results on this front, indicating the feasibility of the technology. Problems with text recognition were made worse by the lack of task-specific training data and full reliance on pretrained models, which were typically trained on English language data rather than the alphanumeric codes seen in facilities.

On the business side of the problem, assuming that the technical problems are resolved, the exact end-to-end feasibility of the solution is still unclear. For instance, how much special effort will be required from scanning teams on-site to improve recognition quality, and whether there's enough useful information visible in the images, are yet to be known. Modern scanners produce scans of sufficient quality, but the older existing scans tend to be too blurred to be useful.

Overall, these results indicate that recognizing text from facilities is feasible with some obstacles in the way, some technical and some business-related. The technical obstacles can likely be overcome with deep learning expertise, while the business obstacles can be cleared with more collaboration with the potential customers that would benefit from this functionality. Several future research topics are suggested, most importantly improving text recognition results, but also the potential use of domain-specific datasets and synthetic data.

References

[1] W. Kritzinger, M. Karner, G. Traar, J. Henjes, and W. Sihn, "Digital Twin in manufacturing: A categorical literature review and classification", *IFAC-PapersOnLine*, vol. 51, no. 11, pp. 1016–1022, 2018.

[2] K. Tong, Y. Wu, and F. Zhou, "Recent advances in small object detection based on deep learning: A review", *Image and Vision Computing*, vol. 97, p. 103 910, 2020.

[3] I. Goodfellow, *Deep learning* (Adaptive computation and machine learning), eng. Cambridge, Mass: MIT P., 2016, ISBN: 9780262337373.

[4] A. Wichert, *Machine learning : a journey to deep learning with exercises and answers*, eng. Hackensack, New Jersey: World Scientific, 2021, ISBN: 981-12-3406-X.

[5] P. Rivas, *Deep Learning for Beginners*, eng, 1st edition. Packt Publishing, 2020, ISBN: 1-5231-3641-3.

[6] "File:conjugate gradient illustration.svg, wikimedia commons", [Online]. Available: `https://commons.wikimedia.org/wiki/File:Conjugate_gradient_illustration.svg`, (accessed: 26.05.2022).

[7] S. Hochreiter and J. Schmidhuber, "Long short-term memory", *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[8] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks”, *Advances in neural information processing systems*, vol. 25, 2012.

[9] W. Rawat and Z. Wang, “Deep convolutional neural networks for image classification: A comprehensive review”, *Neural computation*, vol. 29, no. 9, pp. 2352–2449, 2017.

[10] R. Girshick, J. Donahue, T. Darrell, and J. Malik, “Rich feature hierarchies for accurate object detection and semantic segmentation”, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 580–587.

[11] L. Liu, W. Ouyang, X. Wang, *et al.*, “Deep learning for generic object detection: A survey”, *International journal of computer vision*, vol. 128, no. 2, pp. 261–318, 2020.

[12] D. W. Otter, J. R. Medina, and J. K. Kalita, “A survey of the usages of deep learning for natural language processing”, *IEEE transactions on neural networks and learning systems*, vol. 32, no. 2, pp. 604–624, 2020.

[13] S. Long, X. He, and C. Yao, “Scene text detection and recognition: The deep learning era”, *International Journal of Computer Vision*, pp. 1–24, 2020.

[14] V. I. Levenshtein *et al.*, “Binary codes capable of correcting deletions, insertions, and reversals”, in *Soviet Physics Doklady*, Soviet Union, vol. 10, 1966, pp. 707–710.

[15] X. Chen, L. Jin, Y. Zhu, C. Luo, and T. Wang, “Text recognition in the wild: A survey”, *ACM Computing Surveys (CSUR)*, vol. 54, no. 2, pp. 1–35, 2021.

[16] R. Girshick, “Fast r-cnn”, in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1440–1448.

[17] K. He, G. Gkioxari, P. Dollár, and R. Girshick, “Mask r-cnn”, in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 2961–2969.

[18] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You only look once: Unified, real-time object detection”, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 779–788.

[19] W. Liu, D. Anguelov, D. Erhan, *et al.*, “Ssd: Single shot multibox detector”, in *European Conference on Computer Vision*, Springer, 2016, pp. 21–37.

[20] X. Zhou, C. Yao, H. Wen, *et al.*, “East: An efficient and accurate scene text detector”, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 5551–5560.

[21] O. Ronneberger, P. Fischer, and T. Brox, “U-net: Convolutional networks for biomedical image segmentation”, in *International Conference on Medical Image Computing and Computer-Assisted Intervention*, Springer, 2015, pp. 234–241.

[22] M. Liao, B. Shi, and X. Bai, “Textboxes++: A single-shot oriented scene text detector”, *IEEE Transactions on Image Processing*, vol. 27, no. 8, pp. 3676–3690, 2018.

[23] Y. Baek, B. Lee, D. Han, S. Yun, and H. Lee, “Character region awareness for text detection”, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 9365–9374.

[24] S. Long, J. Ruan, W. Zhang, X. He, W. Wu, and C. Yao, “Textsnake: A flexible representation for detecting text of arbitrary shapes”, in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 20–36.

[25] W. Wang, E. Xie, X. Song, *et al.*, “Efficient and accurate arbitrary-shaped text detection with pixel aggregation network”, in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 8440–8449.

[26] B. Shi, X. Bai, and S. Belongie, “Detecting oriented text in natural images by linking segments”, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 2550–2558.

[27] I. Sutskever, O. Vinyals, and Q. V. Le, “Sequence to sequence learning with neural networks”, *Advances in neural information processing systems*, vol. 27, 2014.

[28] D. Bahdanau, K. Cho, and Y. Bengio, “Neural machine translation by jointly learning to align and translate”, *arXiv preprint arXiv:1409.0473*, 2014. arXiv: `1409.0473`.

[29] A. Vaswani, N. Shazeer, N. Parmar, *et al.*, “Attention is all you need”, *Advances in neural information processing systems*, vol. 30, 2017.

[30] C. Xue, S. Lu, S. Bai, W. Zhang, and C. Wang, “I2c2w: Image-to-character-to-word transformers for accurate scene text recognition”, *arXiv preprint arXiv: 2105.08383*, 2021.

[31] J. Baek, G. Kim, J. Lee, *et al.*, “What is wrong with scene text recognition model comparisons? dataset and model analysis”, in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 4715–4723.

[32] W. Liu, C. Chen, and K.-Y. K. Wong, “Char-net: A character-aware neural network for distorted scene text recognition”, in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[33] M. Liao, J. Zhang, Z. Wan, *et al.*, “Scene text recognition from two-dimensional perspective”, in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, 2019, pp. 8714–8721.

[34] M. Jaderberg, K. Simonyan, and A. Zisserman, “Spatial transformer networks”, *Advances in neural information processing systems*, vol. 28, pp. 2017–2025, 2015.

[35] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, “Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks”, in *Proceedings of the 23rd International Conference on Machine Learning*, 2006, pp. 369–376.

[36] B. Shi, X. Bai, and C. Yao, “An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition”, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 11, pp. 2298–2304, 2016.

[37] Z. Cheng, F. Bai, Y. Xu, G. Zheng, S. Pu, and S. Zhou, “Focusing attention: Towards accurate text recognition in natural images”, in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 5076–5084.

[38] A. Mai, M. Bilinski, and R. Provost, “Method to perform 3D localization of text in shipboard point cloud data using corresponding 2D image”, in *2020 IEEE Eighth International Conference on Communications and Electronics (ICCE)*, 2021, pp. 433–438. DOI: 10.1109/ICCE48956.2021.9352083.

[39] X. Liu, D. Liang, S. Yan, D. Chen, Y. Qiao, and J. Yan, “Fots: Fast oriented text spotting with a unified network”, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 5676–5685.

[40] P. Lyu, M. Liao, C. Yao, W. Wu, and X. Bai, “Mask textspotter: An end-to-end trainable neural network for spotting text with arbitrary shapes”, in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 67–83.

[41] S. Ren, K. He, R. Girshick, and J. Sun, “Faster r-cnn: Towards real-time object detection with region proposal networks”, *Advances in neural information processing systems*, vol. 28, pp. 91–99, 2015.

[42] L. Xing, Z. Tian, W. Huang, and M. R. Scott, "Convolutional character networks", in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 9126–9136.

[43] H. Li, P. Wang, and C. Shen, "Toward end-to-end car license plate detection and recognition with deep neural networks", *IEEE Transactions on Intelligent Transportation Systems*, vol. 20, no. 3, pp. 1126–1136, 2018.

[44] H. Li and C. Shen, "Reading car license plates using deep convolutional neural networks and LSTMs", *arXiv preprint arXiv:1601.05610*, 2016. arXiv: `1601.05610`.

[45] T.-Y. Lin, M. Maire, S. Belongie, *et al.*, "Microsoft coco: Common objects in context", in *European Conference on Computer Vision*, Springer, 2014, pp. 740–755.

[46] W. Yang, Y. Qian, J.-K. Kämäräinen, F. Cricri, and L. Fan, "Object detection in equirectangular panorama", in *2018 24th International Conference on Pattern Recognition (ICPR)*, IEEE, 2018, pp. 2190–2195.

[47] Y.-C. Su and K. Grauman, "Learning spherical convolution for fast features from 360 imagery", *Advances in Neural Information Processing Systems*, vol. 30, pp. 529–539, 2017.

[48] Y.-C. Su and K. Grauman, "Kernel transformer networks for compact spherical convolution", in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 9442–9451.

[49] A. Gupta, A. Vedaldi, and A. Zisserman, "Synthetic data for text localisation in natural images", in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 2315–2324.

[50] "Tesseract-ocr github repository", [Online]. Available: `https://github.com/tesseract-ocr`, (accessed: 15.03.2022).

[51] “Tesseract user manual”, [Online]. Available: `https://tesseract-ocr.github.io/`, (accessed: 15.03.2022).

[52] “Janzd/east github repository”, [Online]. Available: `https://github.com/janzd/EAST`, (accessed: 15.03.2022).

[53] P. Wang, C. Zhang, F. Qi, *et al.*, “Pgnet: Real-time arbitrarily-shaped text spotting with point gathering network”, *AAAI. AAAI*, pp. 2782–2790, 2021.

[54] “Clovaai/deep-text-recognition-benchmark github repository”, [Online]. Available: `https://github.com/clovaai/deep-text-recognition-benchmark`, (accessed: 15.03.2022).

[55] “Mmocr github repository”, [Online]. Available: `https://github.com/open-mmlab/mmocr`, (accessed: 15.03.2022).

[56] Y. Liu, L. Jin, Z. Xie, C. Luo, S. Zhang, and L. Xie, “Tightness-aware evaluation protocol for scene text detection”, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 9612–9620.

[57] “Tesseract user manual: Improving the quality of the output”, [Online]. Available: `https://tesseract-ocr.github.io/tessdoc/ImproveQuality.html`, (accessed: 15.04.2022).

[58] Z. Kuang, H. Sun, Z. Li, *et al.*, “Mmocr: A comprehensive toolbox for text detection, recognition and understanding”, in *Proceedings of the 29th ACM International Conference on Multimedia*, 2021, pp. 3791–3794.

[59] “Paddleocr github repository”, [Online]. Available: `https://github.com/PaddlePaddle/PaddleOCR`, (accessed: 11.06.2022).