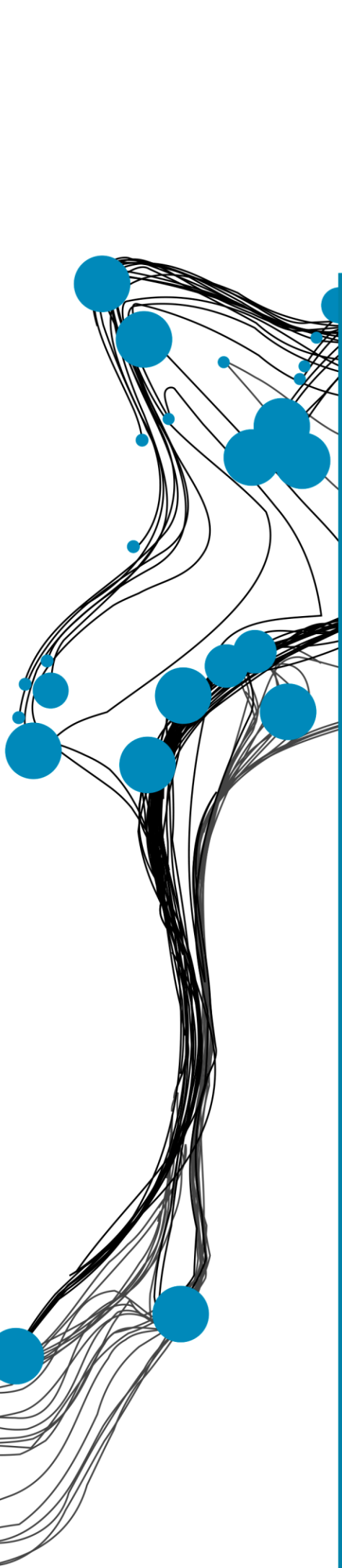

Faculty of Electrical Engineering, Mathematics and Computer Science

MASTER THESIS

July 16, 2020

Automatic container code identification using Machine Learning

Denisa-Valentina Licu

Under the supervision of:

Cofano Software Solutions B.V.
Leon de Vries MSc

UNIVERSITY OF TWENTE.
Dr. Mannes Poel
Dr.Ing. Gwenn Englebienne

UNIVERSITY OF TWENTE.

Contents

Automatic container code identification using Machine Learning

Denisa-Valentina Licu
Cofano Software Solutions/University of Twente
Enschede, The Netherlands
licu.denisa.valentina@gmail.com

ABSTRACT

With the accelerated development of the global industries and the increasing number of containers that pass through a single terminal every day, repetitive and slow parts of the shipping process need to be automatized in order to keep up with the market. Possible automations are numerous, but all start and make use of an automatic container code identification system, needed in order to replace the manual identification that takes place nowadays at most of the terminals. The goal of this research is to design and evaluate such a system and also investigate the effect of damage on the performance of the automatic identification. The proposed method is composed by two parts, a detection step and a recognition step and uses a Convolutional Neural Network and Gaussian heatmaps for text detection and a kNN classifier for character recognition. The results show a 100% detection rate for containers in good shape and 80% for damaged containers. The overall identification results show 40.4% of complete container codes identified for containers in good shape and 12.0% for damaged containers. When the proposed algorithm outputs a complete code, that is in 100% of the times the real container code, with 0% false positive rate. The proposed pipeline proved to be successful for the container identification task, facing some challenges due to the chosen recognition method, challenges that can be overcome increasing the recognition dataset. Regarding the damage, the proposed pipeline has some issues detecting heavily damaged characters at the extremities of a container code.

KEYWORDS

container code identification, machine learning, container code detection, container code recognition, CNN, deep learning, kNN

1 INTRODUCTION

Nowadays, the location on the planet, the soil and the weather conditions do not stop us to have the desired fruit that grows only at Ecuador or the special type of seeds that grow only in Asia, clothes designed at the other side of the globe or the cheapest electronics from China. We overcame the space and the climate constraints, we pack, ship and transport everything we want, wherever we want, using shipping containers. Thousands of containers pass through a single terminal every day and, as the accelerated development of global industries continues, the number will just increase. At most trading gates, a person reads and registers manually every single container that enters such a terminal. We all know that humans are prone to mistakes, their focus decreases during the day, and it is easy to incur errors. Moreover, the whole process is too slow to efficiently manage all those thousands of containers every day. With the development of the transportation industry, the competition among terminals has also been increasing, and it is essential for them to keep up with the development of the market. Automation of the repetitive and slow parts of the container management process can make the businesses more productive, more efficient, but also more cost effective. Possible improvements and automations are numerous, from automated localization inside a terminal's yard, more efficient container loading and unloading and transportation planning, but all those start and further make use of an automatic container code identification system. The goal of this research is to design and evaluate an automatic container code identification system using Machine Learning.

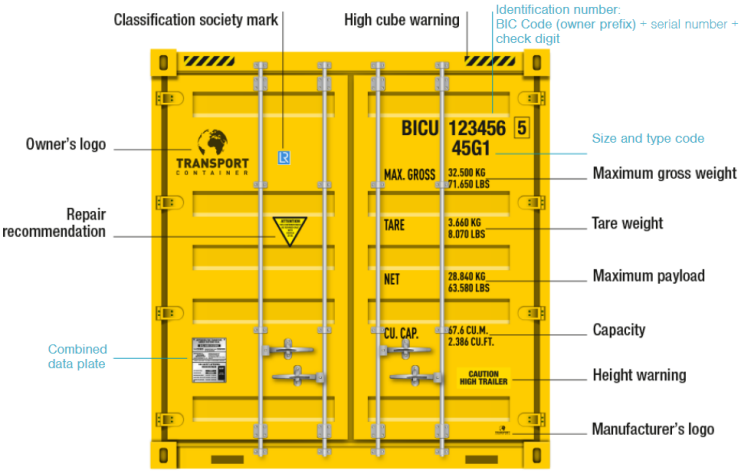

Figure 1: Back door view of a container. Source: [19]

1.1 Research questions

The focus of this research is solving the first part of the container identification task, namely the detection of the container code. There are 3 attributes that differentiate the container code from scene text in general: a container code is an alpha-numeric string, the characters are always in upper-case and the corrugated surface of the container body distorts the container code. Due to the specific challenges, this is where the state-of-art in natural scene understanding may have troubles when applied to this task. The second part of the task, the character recognition, is a largely studied problem among the researchers in the field.

The research question of this paper is "***To what extent is the automatic identification of the container code possible using machine learning ?***"

This research question is further spit in 4 sub-research questions:

- To what extent is the automatic container code detection from containers in good shape possible?

Figure 2: Examples of containers in good shape

- To what extent is the automatic container code identification of containers in good shape possible?
- To what extent is the automatic container code detection from damaged containers possible?
- To what extent is the automatic container code identification of damaged containers possible?

It is important to mention that most of the containers present damage, but for the sake of this experiment, a container is considered damaged when the container code printed on it is damaged. Thus, if a container is damaged, but the container code printed on it is not damaged, the container is considered in good shape. Figure 2 presents some examples of containers in good shape and Figure 3 presents examples of damaged containers.

2 BACKGROUND

2.1 Container management system

A shipping container or **an intermodal freight container** is a standardized steel box with strength suitable to withstand shipment, storage and handling, being designed to be moved from one transport modality to another, by road, rail or sea, without unloading and reloading. Those containers are reusable and they are used as a storage unit for moving all kinds of products and raw materials between different locations. They are the most used transportation method for long distance international trade all over the world. Currently, there are around 17 million intermodal freight containers in the world [130] and each container has an unique identification code printed on its 5 visible sides (door end , 2 sides, top and front end) according to the ISO standard 6346 [15]. The back door view is presented in Figure 1.

A container terminal is "a facility where containers are transferred between various means of transportation for further transport" [126]. The transfer can be between large ships and inland vehicles, such as trains and trucks, in the case of a maritime terminal and between barges (small ships), trains and trucks for the inland shipping terminals.

How does the **container management system** work? Each container is tracked through all the terminals it arrives at, using its unique identification code. Every time, at every terminal gate, a person has to inspect and manually type each container code into

Figure 3: Examples of damaged containers

an informatic system. Then the container can enter the terminal and be loaded onto the next means of transportation.

2.2 Container identification code

Each container has a unique code consisting of 4 capital letters, 6 numbers, and 1 check digit. All codes have to follow the ISO 6346 standard, an international standard for coding, identification and making of intermodal (shipping) containers. The standard is managed by the International Container Bureau (BIC - Bureau International des Conteneurs) and covers the serial number, owner, country code, and size of the shipping container. The structure of an ISO 6346 code is presented in Figure 4 and it contains [15] [42]:

(1) owner code: **3 capital letters**
(2) category identifier: **1 capital letter**
(3) serial number: **6 numeric digits** (assigned by the owner)
(4) **1 check digit**

The container **owner's code** must consist of 3 capital letters of the Latin alphabet, must be unique and must be registered at

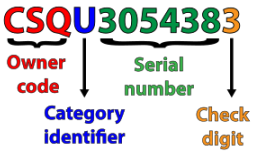

Figure 4: Container code structure according to ISO 6346. Source:[128]

the International Container Bureau in Paris to ensure uniqueness worldwide [128] [82].

The **category identifier** can have one of the following values[15]:

- J: for detachable freight container related equipment
- R: for reefer (refridgerated) containers
- U: for freight containers
- Z: for trailers and chassis

The **serial number** consists of 6 numeric digits, assigned by the owner and must be unique within that owner's fleet [128] [82].

The **check digit** consists of 1 numeric digit used to validate the transmission accuracy of the first 10 characters. The computation

C	S	Q	U	3	0	5	4	3	8	Calc.
13	30	28	32	3	0	5	4	3	8	
*	*	*	*	*	*	*	*	*	*	
2^0=1	2^1=2	2^2=4	2^3=8	2^4=16	2^5=32	2^6=64	2^7=128	2^8=256	2^9=512	
=	=	=	=	=	=	=	=	=	=	
13	60	112	256	48	0	320	512	768	4096	6185(a)
b) Division by 11: 6185/11=										562.27
c)Erase decimal digits:										562
d)Multiply by 11: 562*11=										6182
e) Check digit= 6185-6182=										**3**

Figure 5: Computing the check digit for the container code: CSQU3054383. Source:[111]

of the check digit is done in 3 steps, as follows, and a step-by-step example is presented in Figure 5:

(1) A numerical value equivalent is assigned to each letter of the alphabet, beginning with number 10 for the letter A (11 and all its multiples are not used) as presented in Figure 6. The numeric digits from the code keep their original value.

A	B	C	D	E	F	G	H	I	J	K	L	M
10	12	13	14	15	16	17	18	19	20	21	23	24
N	O	P	Q	R	S	T	U	V	W	X	Y	Z
25	26	27	28	29	30	31	32	34	35	36	37	38

Figure 6: Equivalent numeric value for each letter of the alphabet. Source:[111]

(2) Now, each of the 10 positions has a numeric value assigned. Each of these numeric values has to be multiplied by 2 at the power of the position in the code. Counting the position starts at 0, from left to right [111] as presented in Figure 7.

1.nbr	2.nbr	3.nbr	4.nbr	5.nbr	6.nbr	7.nbr	8.nbr	9.nbr	10.nbr
1	2	4	8	16	32	64	128	256	512

Figure 7: Corespondent multiplier considering the position. Source:[111]

(3) This step consists of multiple mathematical operations as follows:
   (a) Sum up all the numeric values multiplied by 2 at the power of the corresponding position from the second step
   (b) Divide this sum by 11
   (c) Use the floor function to round the result down to 0, making the result an integer
   (d) Multiply the result by 11
   (e) Subtract the result from (d) from the result from (a) and this result is the check digit (if the final difference is 10, then the check digit is 0). [111]

2.3 Natural Scene Text Understanding

In an era driven by visual information and with the continuous development of the low-price cameras, vision techniques are more and more used to solve from simple task to very complex problems.

Generally, an image can contain two kinds of content [49]:

- perceptual content: attributes such as colour, shape, intensity, texture
- semantic content: objects, events, text

Natural Scene Text is the text present in images taken in the outdoor environment. Very different from text in documents, natural scene text exhibits much higher diversity and variability. The text can vary in terms of languages, colours, sizes, fonts, orientations, shapes and aspect ratios [99], as illustrated in Figure 8. Figure 9 presents the properties of text and all of them can bring variation in natural scene text.

Figure 8: Variation in natural scene text. Source:[64]

Property		Variants or sub-classes
Geometry	Size	Regularity in size of text
	Alignment	Horizontal/vertical
		Straight line with skew (implies vertical direction)
		Curves
		3D perspective distortion
	Inter-character distance	Aggregation of characters with uniform distance
Color		Gray
		Color (monochrome, polychrome)
Motion		Static
		Linear movement
		2D rigid constrained movement
		3D rigid constrained movement
		Free movement
Edge		Strong edges (contrast) at text boundaries
Compression		Un-compressed image
		JPEG, MPEG-compressed image

Figure 9: Properties of text in images. Source:[49]

Moreover, scene text is affected by degradation, distortions and cluttered backgrounds. Many approaches of scene text understanding try to address a single issue, and just few approaches tackle a combination of them [86].

Natural Scene Text Understanding or **Text Information Extraction (TIE)** means the extraction of the text information from images and video. Such a system involves 5 steps as presented in Figure 10:

(1) **Text Detection**: there is no prior information if there is any text in the image. This step determines if there is any text in the current image/video frame.
(2) **Text Localization**: determines the location of the text in the given image/frame.
(3) **Text Tracking*** (just for videos): makes the localization easier in consecutive frames. It uses the position of the text in the previous frame/frames to find the text in the current frame. In this way, the searching is more optimal than searching every frame from scratch.
(4) **Text Extraction and Enhancement**: text components are segmented from the background and then enhanced.
(5) **Recognition**: transforms the pieces of image that contain text in plain text.

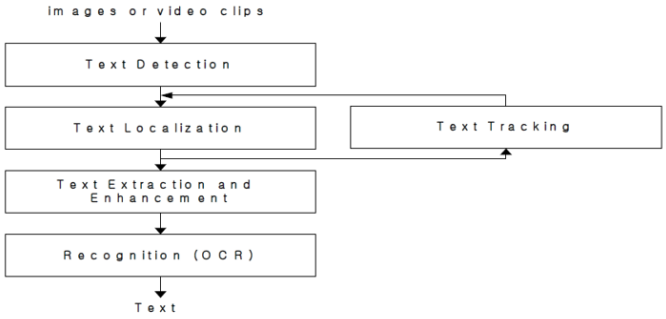

Figure 10: Architecture of a Scene Text Understanding System. Source:[49]

Because this research focuses on images, tracking will not be further analyzed in this research. Also, researchers often group the text detection and text localization in one single step called detection, and text extraction, enhancement and recognition again in one single step called recognition.

2.4 Step wise vs end-to-end approach

Regarding the type of architecture, there are 2 approaches to solve such a problem (Figure 11):

- **Step wise approach**: Independent steps solve different parts of the problem and then the output of one step is used as input for the next one and so on. Changes in one step do not affect the other steps.
- **End-to-end approach**: The steps represent a single entity, with one single input and one single output. Tunes and changes in one step influence all other steps and each step uses the knowledge from all the others steps.

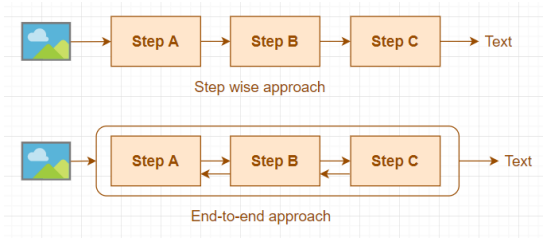

Figure 11: Step wise vs End-to-end approach

The type of pipeline can influence the end result in terms of accuracy, but also in terms of time performance. For example, if step A and B use the same features to compute their output, then an end-to-end approach can reduce significantly the overall time, computing the features just once, but also makes A and B to benefit from each others knowledge. On the other hand, if step A and B have to be unbiased by each other, a step wise approach has to be chosen.

2.5 Classic computer vision techniques vs machine learning

There are 3 main approaches to solve a computer vision problem [107]:

(1) **classic computer vision techniques**: using different properties of images such as colour and edges and applying filters and different thresholds.
(2) **standard machine learning**: applying different algorithms/ models/ architectures already proven to work for certain tasks such as YOLO (You only look once) for object detection.
(3) **specialized machine learning(deep learning)** : a lot of algorithms are useful for their generality (being able to use the same algorithm for the same task, in different conditions), but some problems have very specific demands and constraints. Those mostly use deep learning networks specially designed to fit a certain task such as EAST (Efficient and Accurate Scene Text Detector) for the text detection problem.

2.6 Methodology background

In this part, some background information about concepts, algorithms or methods used and referred further in the Related work or in the Methodology section will be briefly presented.

Bootstraping is a resampling technique used to estimate statistics on a population by randomly sampling a dataset with replacement [6].

Convolutional Neural Networks (CNN) is a class of neural networks which take an image as input and assigns importance to various aspects in the image, being able to differentiate one

from the other. A CNN is able to successfully capture the spatial and temporal dependencies in an image through the application of relevant filters [98].

R-CNN (Region-based Convolutional Neural Networks) combines rectangular region proposals with Convolutional Neural Network features [70]. R-CNN is a two-stage object detection algorithm. The first stage identifies a subset of regions in an image that might contain an object and crops them out of the image. The second stage takes each cropped region, extracts features using the CNN and with these features classifies the object in each region using a Support Vector Machine (SVM) classifier [23].

Fast R-CNN [27] uses a similar approach as R-CNN, but instead of feeding the cropped region proposals to CNN, it feeds the whole input image and generates a convolutional feature map. From this feature map, all region proposals are identified and their features are extracted. Fast R-CNN is faster than R-CNN because it does not have to compute features for every region proposal, it does it once for the whole image. Then, the corresponding features are extracted for each region, eliminating the multiple computations for overlapping parts between regions [23].

Faster R-CNN [100] instead of using an external algorithm for region proposals, as R-CNN and Fast R-CNN do, it generates proposals directly in the network, using a region proposal network (RPN) [70].

Mask R-CNN [34] extends Faster R-CNN, but in parallel with bounding box recognition it also predicts an object mask.

CLAHE (Contrast Limited Adaptive Histogram Equalization) is a technique used to enhance local contrast. CLAHE computes several histograms corresponding to distinct sections of the image, and uses them to redistribute the lightness over the image [125]. See Figure 12 for an example.

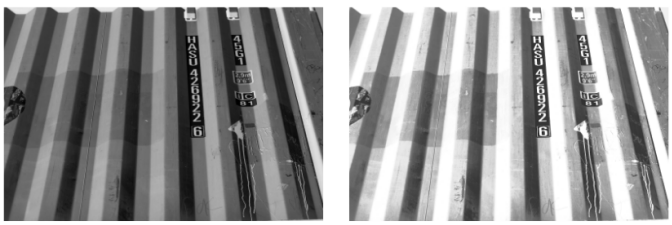

Figure 12: Contrast Limited Adaptive Histogram Equalization (CLAHE) example: a-initial image, b-image after applying the CLAHE technique

Delaunay triangulation is a particular way of joining a set of points to make a triangular mesh, trying to avoid skinny triangles [123].

EAST (Efficient and Accurate Scene Text Detector) is a natural scene text detection algorithm. The pipeline directly predicts words or text lines of arbitrary orientations and quadrilateral shapes in full images, using a single neural network, eliminating intermediate steps [149].

Histogram of oriented gradients (HOG) is a feature descriptor used in computer vision and image processing for the purpose of object detection [127]. The technique counts occurrences of gradient orientation in localized portions of an image. The HOG descriptor can serve as the feature vector in any machine learning classification scheme (SVM, NN, Logistic Regression) [114].

kNN (k-Nearest Neighbours) is a supervised machine learning algorithm that can be used to solve both classification and regression tasks. The kNN algorithm assumes that similar things are in close proximity [31]. It computes a distance metric between a new sample that has to be labeled and all the data points in the training set. It takes the first k neighbours with the shortest distances to the new point. In the case of classification, it computes the mode of those neighbours and, in the case of regression, it computes the mean. This value is the label of the new point. A graphical representation is presented in Figure 13.

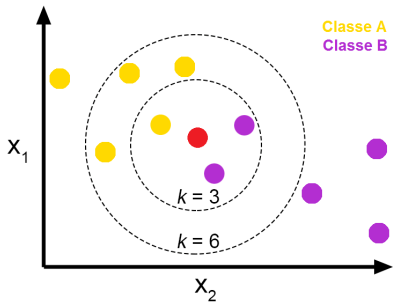

Figure 13: K-Nearest Neighbours (kNN) graphical representation. Source:[31]

K-means is an unsupervised machine learning algorithm used for clustering. Based on the provided features, the algorithm finds k groups in the data. It is an iterative algorithm that assigns each of the data points to one of the groups, based on feature similarity [26].

LeNet5 is one of the classic CNN architectures. It consists of two sets of convolutional and average pooling layers, a flattening convolutional layer, two fully-connected layers and finally a softmax classifier [91].

Maximally Stable Extremal Regions (MSER) is a method for blob detection in images. The MSER algorithm extracts a number of co-variant regions called MSERs. MSERs are connected areas in an image, with almost uniform intensity, surrounded by contrasting background. [69].

Random Forest is a supervised machine learning algorithm consisting of multiple individual Decision Trees that operate as an ensemble. Each decision tree is build using a part of the training data and a part of the features. When a new sample has to be classified, all the decision trees are used and each of them outputs a label. Using all those labels from all the individual trees, the random forest takes the mode of them and assigns that label to the sample. A graphical representation is presented in Figure 14.

Recurrent Neural Networks (RNN) is a class of neural networks with connections between nodes forming a directed graph along a temporal sequence. Unlike feedforward neural networks, RNNs can use their internal state (memory) to process sequences of inputs and exhibit temporal dynamic behavior [129].

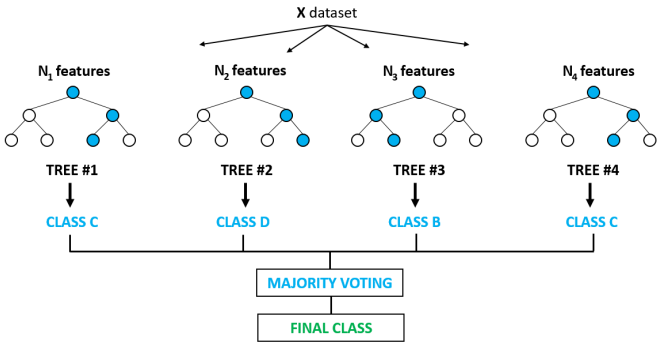

Figure 14: Random Forest graphical representation. Source:[87]

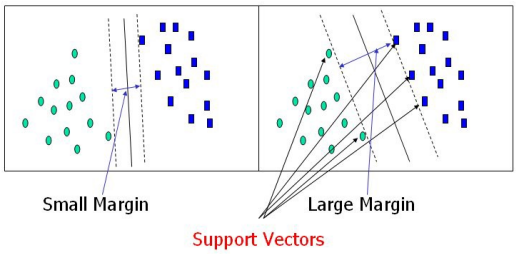

Figure 15: Support Vector Machine (SVM) graphical representation. Source:[24]

LSTM (Long Short Term Memory Networks) are a special kind of RNN, capable of learning long-term dependencies [81].

Single Shot Detector (SSD) is an object detection algorithm, that needs only one shot to detect multiple objects within an image. On the other hand, **Region Proposal Network** (RPN) based approaches such as **R-CNN** need 2 shots, one for generating region proposals and one for detecting the object of each proposal [112]. SSD discretizes the output space of bounding boxes into a set of default boxes over different aspect ratios and scales per feature map location. At prediction time, the network generates scores for the presence of each object category in each default box and produces adjustments to the box to better match the object shape. Additionally, the network combines predictions from multiple feature maps with different resolutions to naturally handle objects of various sizes. SSD completely eliminates proposal generation and subsequent pixel or feature re-sampling stages and encapsulates all computation into a single network [59].

Spatial Transformer Network (STN) [72] is a module which can be inserted into existing CNN architectures. It brings the advantage of actively spatially transform feature maps, without any extra training supervision or modification to the optimisation process.

Support Vector Machine(SVM) is a supervised machine learning algorithm used mainly for classification tasks, but it can also be used for regression problems. SVM creates a hyperplane which separates the data points into classes [88] [83]. It uses support vectors, data points from each class that are closer to the hyperplane and influence the orientation and position of the hyperplane [24]. A graphical representation is presented in Figure 15.

YOLO (You Only Look Once) [89] is a real-time state-of-art object detection algorithm where a single convolutional network predicts the bounding boxes and the class probabilities for these boxes. It takes an image and splits it into an SxS grid and within each of the grid it takes m bounding boxes. The network outputs a class probability and offset values for each bounding box. The bounding boxes having the class probability above a threshold are selected and used to locate the object within the image [23].

2.7 Datasets background

In this part, information about datasets referred in the Related work section will be briefly presented.

CTW-1500 (CTW) consists of 1000 training and 500 testing images. The images contain curved text instances and are annotated by polygons with 14 vertices [3].

ICDAR2003 (IC03) consisting of 249 images, containing texts in English. This dataset was used for the ICDAR 2003 Robust Reading Competition for scene text detection [43].

ICDAR2013 (IC13) consists of 229 images for training and 233 for testing, containing texts in English. This dataset was used for the ICDAR 2013 Robust Reading Competition for focused scene text detection. The annotations are at word-level using rectangular boxes [43].

ICDAR2015 (IC15) consists of 1000 training images and 500 testing images, containing texts in English. This dataset was used for the ICDAR 2015 Robust Reading Competition for incidental scene text detection. The annotations are at the word level using quadrilateral boxes [43].

ICDAR2017 (IC17) consists of 7,200 training images, 1,800 validation images, and 9,000 testing images, containing text in 9 languages. The text regions are annotated using quadrilateral boxes [43].

IIIT 5K-Word consists of 5000 images with text in both natural scenes and born-digital images. It exhibits variation in font, colour, size, layout and the presence of noise, blur, distortion and varying illumination. This dataset comes with three types of lexicons (small, medium, and large) for each test image [3].

TotalText consists of 1255 training and 300 testing images, containing curved text. The text regions are annotated by polygons and word-level transcriptions [3].

SynthText consists of 800K synthetic images. These images are obtained by blending natural images and text with random fonts, sizes, colours, and orientations [3].

SCUT-CTW1500 consists of 1000 training images and 500 test images, containing mainly curved text [3].

Street View Text (SVT) dataset is a collection of outdoor images with scene texts of high variability.

Total-Text consists of 1255 training images and 300 testing images, containing English curved text [3].

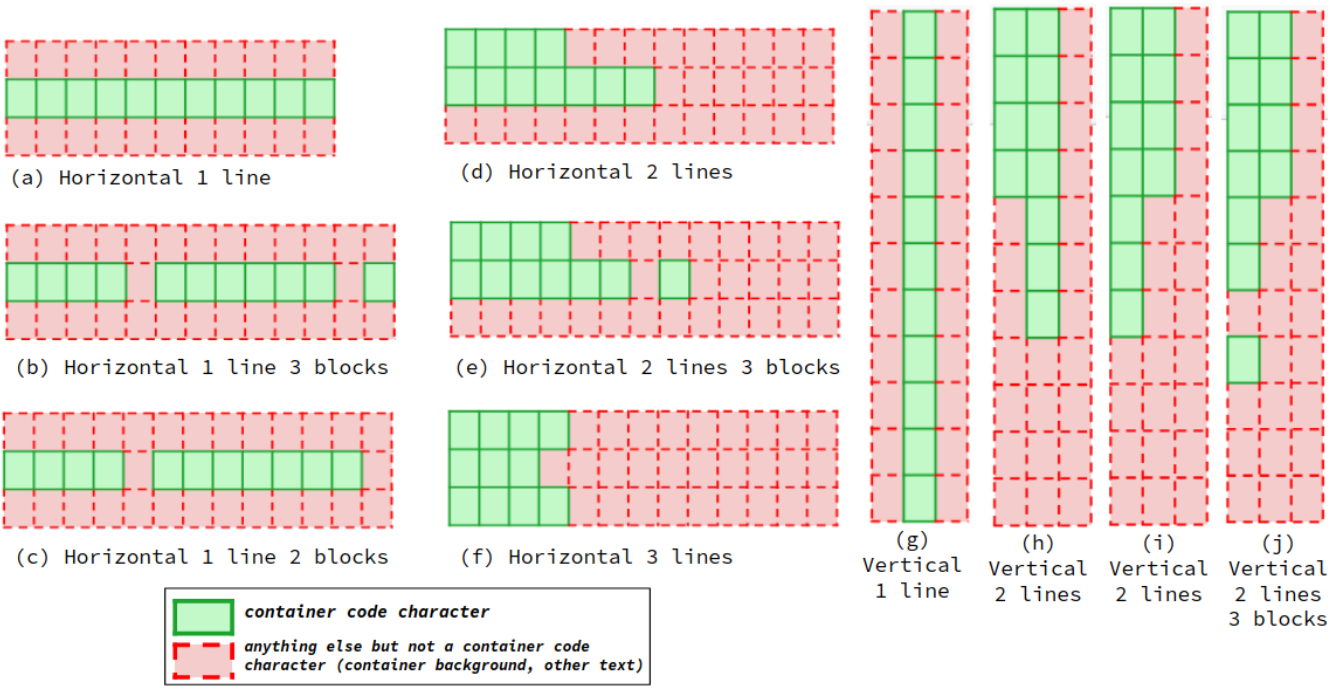

Figure 16: Different container code layouts

3 CHALLENGES

At a first glance, the container code identification task seems to be a classical digit and letter recognition problem, but it actually arises a long list of challenges. The ISO standard only regulates the pattern of the identification code, but does not regulate other parameters such as foreground and background colours, font size and style, character orientation and so on. On top of these unregulated parameters, environment conditions such as illumination and fog and camera related parameters such as blur and photometric distortions add extra complexity. Also, text printed on surfaces becomes damaged or contaminated because of rust and structural damage from rough handling and exposure to sea water. Thus, the main challenge is to design a system as versatile as possible to handle all the variability in daily life in real environmental conditions. Following, a list of challenges, grouped in categories is presented.

(A) **Business domain challenges**: the challenges induced by the different parameters of the container and container management system.

(1) **Container related challenges**: there is no prior information of text location, size and orientation.

- **unknown location of the code on the container body**: it is known that the identification code is somewhere in the right side of the container but this does not provide significant information given the multitude of possible layouts.
- **unstandardized layout and orientation** (horizontal/vertical): the code layout is not standardized and there are multiple layouts used, represented in Figure 16.
- **curvy material and corrugated text over it**: the corrugated surface of some containers makes the 2D projection of the text on a 3D surface to be distorted.
- **damage, mud, rust, peeling paint**: parts of the container and container code are damaged, scratched, partially or totally covered by mud or rust. Figure 17 presents some examples of characters with different degrees of damage.

Figure 17: Example of damaged container code characters.

- **occlusions**: parts of the container and container code are occluded by different objects such as stickers.
- **objects cluttered into the code**: metal pipes or logos are present between the characters of the code or really close to it.

- **unstandardized foreground** (text colour) **and background colours**: the ISO standard does not say anything about colours. Figure 18 presents a container where a part of the container and container code has even a different colour than the other part.

Figure 18: Example of a container.

- **reflexive/mat containers**: not all the containers have the same paint properties, and thus, some containers have reflective surface in the sunlight, being sometimes impossible to capture a photography from certain angles.

(2) **Environment related challenges**:
- **light**: different light conditions are present in the environment in different locations and different times of the day (and the lack of light during the night). Moreover, some containers have a part of the code in the sunlight and a part in the shadow, creating uneven light at the same time, on the same surface.
- **weather conditions**: fog , rain, snow

(3) **Text related challenges**: the text is also unstandardized by the ISO standard and presents different variable parameters such as:
- font
- colour
- size
- stroke width
- occluded/missing characters

(B) **Data driven/technical domain challenges**: the challenges given by the data handling and technical parameters

(1) **Image acquisition** (camera):
- resolution, image quality, noise, blur
- camera orientation, viewing angle
- image distortions/transformations: due to the camera quality, but these occur also when a truck stops closer/further to the camera, more in the front/more in the back, in the moment of image acquisition

(2) **The lack of a public dataset**: currently, there is no public dataset for the container identification task. This means first, the need of gathering and labelling a dataset, and second, the results of this research are not comparable with other models that solve the same problem because they each use a different dataset.

(3) **Fast processing, close to real time**: the identification algorithm has to be fast enough to make the whole container management process faster. It is not desired for a container to spend at the terminal's gate more than 2-3 minutes for identification.

4 RELATED WORK

4.1 Natural Scene Text Understanding

In this subsection, text detection consists of text detection and text localization steps presented in Section 2 and text recognition consists of text extraction, enhancement and recognition steps presented in the same section.

There are a few survey papers that show an overview of the methods used for Natural Text Scene Understanding, classifying those methods form different perspectives:

- Zhang et. al (2013) [144] have mainly focused on papers/methods related to scene text detection, but ignored methods on text recognition.
- Uchida et al. (2014) [113] reviewed methods/papers for text detection and recognition for images and videos, but for text in images in general, mostly text from documents, not natural scene text.
- Kim et al. (2003) [49] and Ye et al. (2015) [86] present also an overview for text detection and recognition, but for text in images in general, not natural scene text.
- Zhu et al. (2016) [141] tried to extend the work from [144] and [113] with a special emphasis on the latest advances in the scene text understanding area at the moment of the study.
- Thillou et al. (2012) [9] present a natural scene text understanding overview focused on the extraction and segmentation methods.
- Long et al. (2018) [99] is the latest overview and presents methods before and in the Deep Learning Era, focusing on the Deep Learning Era.

4.1.1 ***Before the Deep Learning Era.*** The methods used before the Deep Learning Era mainly extract and use low and mid-level hand crafted features with repetitive pre and post-processing steps. Those methods are constrained by the limited representation of hand crafted features and mostly do not handle intricate circumstances and blurred images [99].

A. Detection

There are 2 main methods used for detection before the deep learning era [99]:

(1) **Connected Component Analysis** (CCA): method that extracts candidate components using a variety of ways and then groups those components into bigger and bigger constructions, until a certain point, when the constructions are classified as text or non-text. The non-text constructions are filtered out and the text constructions are left. Epshtein et al. [21], Huang et al. [39], Jain et al. [46], Neumann and Matas [77], Yao et al. [134], Yi and Tian [13], Yin et al. [139], all use Connected Component Analysis for scene text detection in their research.

(2) **Sliding Window** (SW): windows of different sizes slide over the image and each window is classified as a text/ non-text segment. Then all text segments are grouped further into characters using various methods such as morphological operations (Lee et al. [47]), Conditional Random Field (CRF) (Wang et al. [119]) or graph based methods (Coated et al. [14] and Wang et al. [121]).

Table 1 presents some of the detection algorithms with their strengths and weaknesses, application domains, techniques and reported performance.

B. Recognition

Table 2 presents some of the recognition algorithms with their strengths and weaknesses, application domains, techniques and reported performance.

The most used approach for text recognition before the Deep Learing Era is the feature based approach. Shi et al. [103] and Yao et al. [136] used character segments based algorithms for recognition, Rodriguez et al. [93] [92], Gordo et al. [29] and Almazan et al. [1] used label embedding to perform directly matching between images and strings, Busta et al. [8] used the stroke as feature for recognition and Phan et al. [84] used character key-points as features for their algorithm.

Another widely used approach is to split the recognition problem in multiple sub-problems which include text binarization (Lee et al. [54], Mishra et al. [74], Wakahara et al. [116], Zhou et al. [150]), text line segmentation (Ye et al. [137]), character segmentation (Nomura et al. [79], Roy et al. [97], Shivakumara et al. [105]), single character recognition (Chen et al. [10], Sheshadri et al. [101]) and word correction(Karatzas et al. [48], Mishra et al. [73], Wachenfeld et al. [115], Weinman et al. [122], Zhang et al. [20]).

C. End-to-end

There have been a few efforts to tackle the scene text understanding problem using an end-to-end approach before the Deep Learning Era [78], [119]. Table 3 presents two end-to-end approaches with their strengths and weaknesses, application domains, techniques and reported performance.

4.1.2 ***The Deep Learning Era.*** The Deep Learning Era brings not only changes and advancements in the algorithms used, but has totally changed the way researches approach the problems, enlarging the scope of research by far. Besides the advantage of automatic feature learning, that saves researchers from the repetitive work of designing and testing large amount of potential hand-crafted features, the deep learning paradigm brought a blossoming expansion for the auxiliary tasks around the deep learning algorithms. Thus, the deep learning era created the opportunity for new viewpoints, faster and simplified pipelines, synthetic data generation algorithms and more others [99].

A. Detection

The input for text detection is the whole image and the output is a cropped text instance image which contains 1 word or 1 text line. There are 3 main trends for text detection:

(1) **Pipeline Simplification**: Most of the methods before the deep learning era and some early methods that use deep learning have multi-step pipelines. More recent methods have much simpler pipelines that simplify the training process [99].
- **Multi-step methods**: the text detection task is split in multiple steps([135], [146], [33]). Yao et al. [135] used a convolutional neural network to predict if each pixel in the image belongs to a character, if it is inside the text region and also predict the text orientation around this pixel. Next, they connected positive responses into characters or text regions. Then, they applied Delaunay triangulation for characters from the same text region, grouping characters into text lines using graph partition based on the predicted orientation [99]. A similar approach is used by Zhang et al. in [146] to first build a dense map indicating the pixels that are within text line regions, then for each text line, all character candidates are extracted using MSER [78]. Those character candidates show information about the scale and orientation of the text line. In the end, a bounding box is used to isolate the final text line candidates.
- **Simplified pipelines**: recent methods of scene text detection use a 2-step pipeline made off by an end-to end neural network and a post-processing step. These methods use techniques from object detection considering the text as a special type of object [99]. There are 2 main approaches:
  - **Anchor based methods** are inspired from a general object detection network called SSD (Single Shot Detector) [59]. A representative work by Liao et al. [57] adapts the SSD network to fit different variations in orientation and aspect rations of text lines. EAST [149] is a version of the standard anchor-based box prediction method that made a difference into the field of text detection with its very simplified pipeline, efficiency and speed [99].
  - **Region proposal methods** mainly follow the standard object detection framework of R-CNN [27], [28], [100] where initially a set of regions that could contain text are proposed and then a neural network classifies them as text/non-text. Rotation Region Proposal Networks [68] follow and adapt the standard Faster RCNN framework. The standards axis aligned rectangles are replaced by rotating region proposals to fit text of arbitrary orientations. Further R2CNN [140] adapts the method to solve the aspect ratios variations.

(2) **Different Prediction Units**: A very big difference between text detection and object detection in general is that text presents homogeneity and locality. This means that any part of a text is still text, properties that are not valid for object detection in general. This fact was the starting point for a new branch of text detection methods, that first only predict sub-text components and then assemble then into a text instance. Considering the granularity of the text, there are 2 main levels of prediction granularity:
- **text instance level**: follows the general object detection routine, first a region proposal network produces initial and coarse guesses of text instances and then a refinement network classifies these as text/non-text and corrects their localization. This method is used by [16], [37], [140], [57], [58], [62], [68], [145], [149].
- **sub-text level**:

Algorithm	Strength	Weakness	Application domain	Technique used	Performance
Zhong et al. [148] (1995)	can detect text in natural images	only simple images	text localization in colour images	Horizontal spatial variance was utilized to roughly localize texts and then colour segmentation was performed within the localized regions to find texts	10 seconds processing time, no performance reported
Jain et al. [46] (1998)	can detect text in natural images	only simple images and relies on manually defined rules	text location in images and video frames	They decomposed images into several non-overlapping components by colour clustering, grouped components into text lines through component analysis, and then removed non-text components according to geometric rules.	accuracy lies between 72% and 99,2% depending on the type of images.
Kim et al. [50](2003)	can detect text in natural images and videos	only simple images and only applicable to horizontal text	text detection in images	They trained a SVM classifier to classify each pixel by directly using the raw pixel intensity as local feature. Text areas were sought via adaptive MeanShift in probability maps	2.4 % miss rate, 71.5% false detection rate and 0.72 seconds average processing time per image on their dataset and 0.22 precision, 0.28 recall, 0.22 F-measure for detection on ICDAR 2003 dataset
Li et al. [55] (2000)	can detect and track text in videos	only horizontal text	text detection and tracking in videos	Images are decomposed by using the mean of wavelet coefficients, and the first-order and second-order moments as local features	no performance reported, just examples
Chen et al. [11] (2004)	can detect text in complex images and it is fast	only horizontal text	fast text detection	The detector is a cascade Adaboost classifier, in which each weak classifier is trained from a set of features. The feature pool includes mean strength, intensity variance, horizontal difference, vertical difference, and gradient histogram	97,2% detection rate and 93% recognition rate, 0,6 precision , 0.6 recall, 0.58 F-measure for detection on ICDAR 2003 dataset
Lyu et al. [65] (2005)	can detect text in videos and can handle multilingual text	only horizontal text	multilingual text detection in videos	They proposed a coarse-to-fine multi- scale search scheme. The scheme used properties such as strong edge and high contrast of texts to distinguish between text and non-text regions	91,1% detection rate, 90,8% detection accuracy, 0.25 seconds processing time
Wang et al. [120] (2010)	can detect text in complex natural images	only horizontal text and requires a lexicon for each image	specific words location in natural scenes	Firstly, single characters are detected by sliding window. Then, possible combinations are scored according to the structural relationships between characters. Finally, the most similar combinations are selected from the given list as the output results.	51,5 % recognition accuracy for the CDAR03-CH dataset
Epshtein et al. [21] (2010)	can detect text in complex natural images, can handle multilingual text and it is reasonably fast	only horizontal/ nearly horizontal text and relies on manually defined rules	natural scene text detection	They use the property that characters have nearly constant stroke width and propose a new image operator: Stoke Width Transform(SWT). This operator provides an easy way to recover character strokes from edge maps and is able to efficiently extract text components of different scales and directions from complex scenes	0,73 precision, 0,6 recall, 0,66 F measure, 0.94 seconds processing time

Neumann et al. [76](2010)	can detect text in complex natural images and it is reasonably fast	only horizontal/ nearly horizontal text	text localization and recognition in real-world images	They proposed a text detection algorithm based on Maximally Stable Extremal Regions (MSER). This algorithm extracts from the original images MSER regions as candidates, and eliminates invalid candidates using a trained classifier. At a later stage, the remained candidates are grouped into text lines through a series of connection rules.	0.59 precision, 0.55 recall, 0.57 F-measure for detection and 0.42 precision, 0.39 recall and 0.4 F-measure for recognition on the ICDAR 2003 dataset
Yi et al. [13] (2011)	can detect text of different orientations and can handle multilingual text	only simple natural images and relies on manually defined rules	tilted text detection in natural images	Firstly, the image is divided into different regions according to the distribution of pixels in colour space, and then regions are combined into connected components according to the properties such as colour similarity, spatial distance and relative size of regions. Finally, non-text components are discarded by a set of rules.	0.71 precision, 0.62 recall, 0.62 F-measure for detection on the Robust Reading Dataset
Shivakumara et al. [106] (2011)	can detect text of different orientations	produces text blocks instead of words or lines and relies on manually defined rules	multi-oriented text detection	The method extracted candidate regions by clustering in the Fourier-Laplace space and divided the regions into distinct components using skeletonization. However, these components generally do not correspond to strokes or characters, but just text blocks.	0,76 precision, 0,86 recall, 0.81 F-measure, 7,8 seconds processing time on ICDAR 2003 dataset
Yao et al. [134] (2012)	can detect text of different orientations, can handle multilingual text and it is reasonably fast	relies on manually defined rules	detection of texts of arbitrary orientations in natural images	Based on Stroke Width Transform(SWT) [21], this algorithm is equipped with a two-level classification scheme and two sets of rotation and rotation-invariant features specially designed for capturing the intrinsic characteristics of characters in natural scenes	0.69 precision, 0.66 recall, 0.67 F-measure for detection on ICDAR 2003 dataset
Huang et al. [39] (2013)	can detect text in complex natural images and can handle multilingual text	only horizontal text and relies on manually defined rules	text localization in natural scene images	Tehy presented a new operator based on Stroke Width Transform (SWT), called Stroke Feature Transform (SFT).In order to solve the mismatch problem of edge points in the original Stroke Width Transform, SFT introduces colour consistency and constrains relations of local edge points, produc- ing better component extraction results.	0,81 precision, 0,74 recal, 0,72 F-measure for detection on ICDAR 2005 dataset
Huang et al. [40] (2014)	can detect text in complex natural images and has excellent performance	only horizontal text	scene text detection	They proposed a novel framework for scene text detection, which integrated Maximally Stable Extremal Regions (MSER) and Convolutional Neural Networks (CNN). The MSER operator works in the front-end to extract text candidates, while a CNN based classifier is applied to correctly identify true text candidates and separate the connections of multiple characters in components.	0.84 precision, 0.67 recall, 0,75 F-measure for detection on ICDAR 2005 dataset

Table 1: Comparison between existing text detection methods.

Algorithm	Strength	Weakness	Application domain	Technique used	Performance
deCampos et al. [109] (2009)	can recognize characters in natural images	only simple images and it is sensitive to font variation	character recognition in natural images	The problem is addressed in an object categorization framework based on a bag-of-visual-words representation. They assess the performance of various features(Geometric Blur, Scale Invariant Feature Transform, Spin image, Maximum Response of filters and Patch descriptor) based on kNN and SVM classification.	55,26% recognition accuracy
Mishra et al. [75](2012)	can recognize words in natural images and it is insensitive to font variation	sensitive to font variation	scene text recognition	This method uses sliding window to detect possible characters, and treat the detection results as bottom-up information. It also uses top-down information that comes from the statistics of a large dictionary. The bottom-up and top-down information are integrated in a unified model through Conditional Random Field (CRF).	73,26% recognition accuracy on Street View Text dataset and 81,78% recognition accuracy on ICDAR 2003 dataset
Mishra et al. [73] (2012)	can recognize words in natural images	relies on large lexicon and it is sensitive to font variation	scene text recognition	They proposed a new text recognition method based on the algorithm in [75]. This method introduces an error correction model, which take full advantage of higher order prior information, further boosting the recognition accuracy.	recognition accuracy of 68% on Street View Text dataset and 72,01% on ICDAR 2003 dataset
Novikova et al. [80] (2012)	can recognize words in natural images	relies on large lexicon and it is sensitive to blur and occlusion	text recognition in natural images	They proposed to characterize character appearance and the relationship between characters via a unified probabilistic model. Unlike the algorithm in [75], character candidates are extracted using MSER. This method adopts Weighted Finite-State Transducers as the probabilistic model and searches the most likely word by an efficient reasoning algorithm.	82,8% recognition accuracy for ICDAR 2003 and 72,9% for Street View Text dataset
Rodriguez-Serrano et al. [93] (2013)	can recognize words in complex natural images, it has retrieval based word recognition and it is efficient	requires a lexicon for each image	text recognition	They explored a new approach for text recognition, in which label embedding was utilized to directly perform matching between strings and images, by- passing pre- or post-processing operations.	76,1% recognition accuracy on IIIT-5k dataset
Shi et al. [103] (2013)	can recognize words in complex natural images and it is insensitive to blur, occlusion, font variation	relies on manually designed character models and requires detailed annotations for parts	scene text recognition	They proposed a part-based tree-structured model to recognize characters in cropped images.	79,3% recognition accuracy on ICDAR 2003, 82,87% on ICDAR 2011 and 73,51 on Street View Text dataset

Yao et al. [136] (2014)	can recognize words in complex natural images, can learn character parts from training data and it is insensitive to blur, occlusion, font variation	inefficient	scene text recognition	They proposed a novel representation, called Strokelets, which consists of a set of multi-scale mid-level elements . Strokelets can be automatically learned from character level labels and are able to capture the structural properties of characters at different granularities. Moreover, strokelets provide an alternative way to identify individual characters and compose a histogram feature to describe characters	80,33% recognition accuracy on ICDAR2003 and 75,89% on Street View Text dataset

Table 2: Comparison of existing text recognition methods.

Algorithm	Strength	Weakness	Application domain	Technique used	Performance
Wang et al. [119] (2011)	good performance on street view images and it is robust	only applicable to horizontal texts and it requires a lexicon for each image	end to end scene text recognition	They consider characters as being a special type of objects and perform object detection using a nearest neighbor classifier trained using HOG features [17]. Further, they group the identified characters into words using a Pictorial Structure (PS) based model [22]	54% character classification accuracy on Chars74K-15 dataset and 64% on ICDAR dataset
Neumann and Matas [78](2013)	fast and good performance on ICDAR 2011 Robust Reading dataset	unable to detect single or two-letter words and only applicable to horizontal text	end-to-end scene text recognition	They used a decision delay approach and keep multiple segmentations of each character until the last stage when the context of each character is known. They used extremal regions for the character segmentations and a dynamic programming algorithm for decoding the recognition results [99].	85,4% precision, 67,5% recall and 75,4% F-measure for text localization and 39,4% precision, 37,8% recall for text recognition on ICDAR 2011 dataset

Table 3: Comparison of existing end-to-end methods.

- **pixel-level**: an end-to-end fully convolutional network predicts for each pixel belongs to a text instance or not. Then post-processing methods group pixel into text instances, using different algorithms to separate pixels from 2 adjacent instances: Deng et al. [18] add link predictions to each pixel, while Wu and Natarajan [133] classify each pixel as being text, border or background, assuming that border separates well text instances. Other researches that use pixel level prediction granularity for text detection are [33], [135] and [146].
- **components-level**: refers to a local region or text instance, sometimes containing one or more characters. Lyu et al. [67] proposes corner localization and detects all 4 corners of each text instance and uses those predict which components are part of the same instance. Wang et al. [117] use a clustering based method to cluster pixels using their colour consistency and edge information. The clusters are further used to extract characters and predict text instances. Another approach of component-level methods is Connectionist Text Proposal Network (CTPN) [110], [131], [151] that uses the idea of anchors and recurrent neural networks for sequence labelling. They usually consists of a CNN network and a RNN in top of it. Each position in the feature map represents features in the region specified by the corresponding anchor. The RNN labels each group of features as text/non-text [99]. Other researches using component level granularity are [32] and [102].

(3) **Specific Targets**: Most of the current detection researches tackle unique difficulties of the scene text detection problem, being designed for special purposes such as:

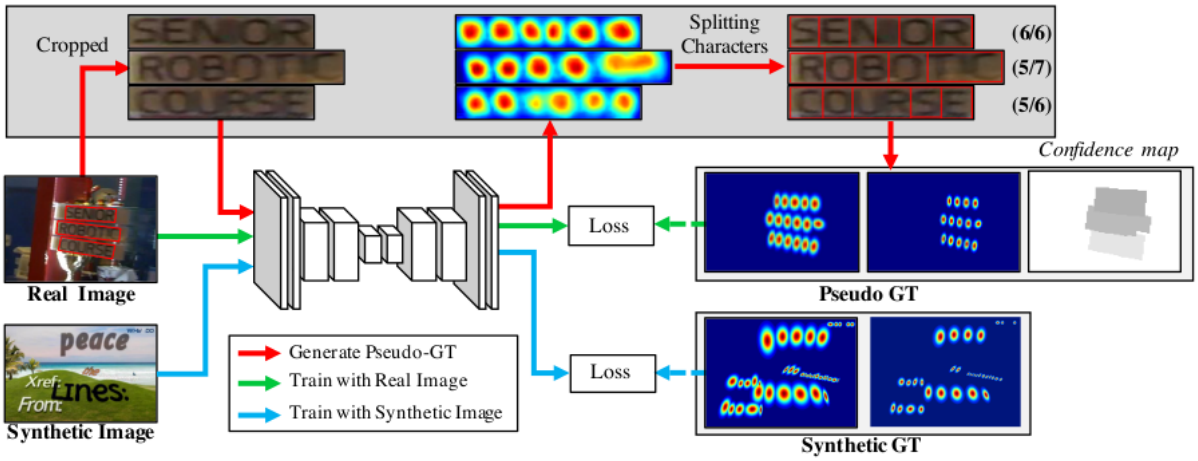

Figure 19: The overall training pipeline for the method proposed by Baek et al in [3]. Training is carried out using both real and synthetic images in a weakly-supervised fashion. (GT=ground truth) Source:[3]

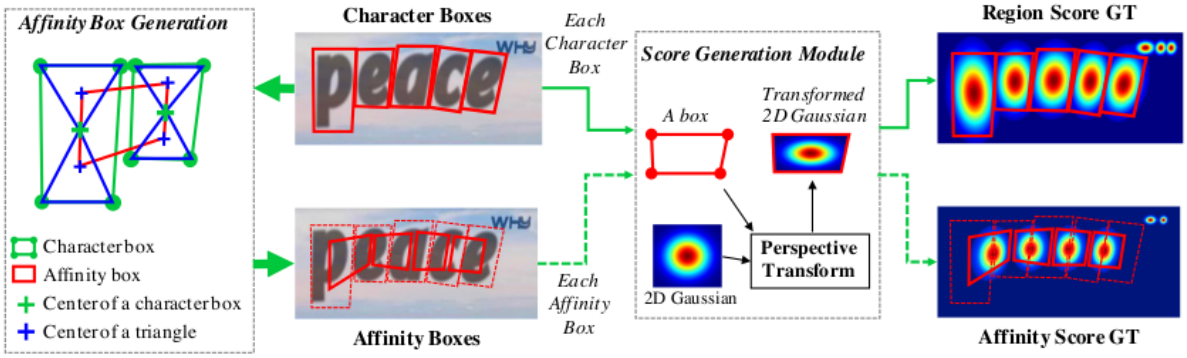

Figure 20: Ground truth(GT) generation procedure used by by Baek et al. in [3]. They generate ground truth labels from a synthetic image that has character level annotations. Source:[3]

- **long text**: general object detection systems fail to detect text, because, unlike general object, text is present in various aspect ratios. Thus, several researches have been conducted to solve the problem of long text detection (Jiang et al. [140], Lyu et al. [67], Shi et al. [102]).
- **multi-orientated text**: skewed and rotated text is common in real world, but the text detection is rotation-sensitive affecting the performance. Several methods tackled this problem: [140] , [57], [58], [62], [118], [68], [149].
- **text of irregular shapes**: Curved text makes the classic rectangle bounding boxes inefficient due to the large amount of background sent towards recognition. Some of the researches that tackle this problem are: [143], [66], [34].
- other purposes: **speedup** (Zhou et al. [149]), **easy instance segmentation** (Deng et al. [18], He et al. [33], Polzounov et al. [85] , Wu et al. [133]), **retrieving designed text** (Rong et al. [94]), **complex background** (He et al. [35]).

The last couple of months brought some advancements in the field of natural scene text detection. Table 4 presents these with their strengths and weaknesses, application domains, techniques and reported performance.

- Baek et al. [3] proposed a framework which effectively detects text area by exploring each character and affinity between characters. They train a deep neural network to predict character regions and the affinity between characters, their architecture being presented in Figure 19 .
  To overcome the lack of individual character level annotations, their proposed framework exploits both the given character-level annotations for synthetic images and the estimated character-level ground-truths (GT) for real images acquired by the learned interim model. Figure 20 presents the ground truth generation procedure that they used.
- Huang et al. [41] proposed a new Mask R-CNN [34] based text detection approach which can robustly detect multi-oriented and curved text from natural scene images in a

Algorithm	Strength	Weakness	Application domain	Technique used	Performance
Baek et al. [3] (2019)	high flexibility in detecting complicated scene text images, such as arbitrarily-oriented, curved, or deformed texts	multi-language issues	arbitrary shape text detection in natural scenes	They trained a deep neural network to predict character regions and the affinity between characters, their architecture being presented in Figure 19.	see Figure 21
Huang et al. [41] (2018)	can detect multi-orientate and curved text	slow and it struggles with skewed nearby long text-lines	multi-oriented and curved text detection in natural scenes	They proposed a new Mask R-CNN based text detection. To enhance the feature representation ability of MaskR-CNN, they propose to use the Pyramid Attention Network (PAN) as a new backbone network of Mask R-CNN.	see Figure 21
Wu et al. [132] (2019)	can handle complex backgrounds and can detect text of arbitrary shapes	slow	irregular text detection in natural images	They proposed a pixel-wise method named TextCohesion, which splits a text instance into 5 key components:a Text Skeleton and 4 Directional Pixel Regions.	see Figure 21

Table 4: Comparison of the latest text detection researches.

unified manner. To enhance the feature representation ability of Mask R-CNN for the text detection tasks, they propose to use the Pyramid Attention Network (PAN) as a new backbone network of Mask R-CNN. Their Mask R-CNN based text detection network is composed of 4 modules: 1) A PAN backbone network that is responsible for computing a multi-scale convolutional feature pyramid over a full image; 2) A region proposal network (RPN) that generates rectangular text proposals; 3) A Fast R-CNN detector that classifies extracted proposals and outputs the corresponding quadrilateral bounding boxes; 4) A mask prediction network that predicts text masks for input proposals.

- Wu et al. [132] proposed a pixel-wise method named TextCohesion, which splits a text instance into 5 key components: a Text Skeleton and 4 Directional Pixel Regions. Their goal is to detect irregular text instances. First they predict Text Skeleton(TS) and Directional Pixel Regions(DPRs) and then the post-processing links TS and DPRs to reconstruct the text. All TS are verified by a Confidence Scoring Mechanism.

Method	ICDAR 2013			ICDAR 2015			ICDAR 2017		
	P	R	F	P	R	F	P	R	F
Baek et al. [3]	**97.4**	**93.1**	**95.2**	89.8	**84.3**	**86.9**	**80.6**	68.2	73.9
Huang et al. [41]	-	-	-	**90.8**	81.5	85.9	80	**69.8**	**74.3**
Wu et al. [127]	-	-	-	-	-	-	-	-	-

Method	CWT1500			TotalText		
	P	R	F	P	R	F
Baek et al. [3]	86	81.1	83.5	87.6	79.9	83.6
Huang et al. [41]	86.8	83.2	85.8	-	-	-
Wu et al. [127]	**88**	**84.7**	**86.3**	**88.1**	**81.4**	**84.6**

Figure 21: Performance comparison (%) of the latest text detection researches on 5 different datasets

B. Recognition

While the input for text detection is the whole image, the recognition phase takes as input the output of the detection phase. This is a cropped part of the original image which contains 1 character, 1 word or 1 line of text. Traditional text recognition methods split the recognition task into 3 sub-tasks: image pre-processing, character segmentation and character identification. Due to the challenges posed by the character segmentation that constrains the performance of the whole recognition process, the current methods avoid the character segmentation. There are 2 major approaches:

(1) **Connectionist Temporal Classification (CTC)**: is "a type of neural network output and associated scoring function, for training recurrent neural networks (RNNs) such as LSTM networks to tackle sequence problems where the timing is variable" [124]. Shi et al. [4] propose for scene text recognition an architecture that stacks CNN with RNN. First, the convolutional layers extract a sequence of features from the input image. Second, the recurrent layers predict for each frame a label distribution. Third, the connectionist temporal layer or the transcription layer translates the per-frame predictions into the final label sequence. Gao et al. [25] and Yin et al. [138] adopt a similar approach but they don't use RNN, but just stacked CNNs. Gao et al. [25] use the CNN layers to capture the contextual dependencies of the input sequence, approach characterized by lower computational complexity than RNN. Yin et al. [138] simultaneously detect and recognize characters by sliding the text line image with character models, which are learned end-to-end on text line images labeled with text transcripts [99].

(2) **Attention Mechanism**: Lee et al. [53] propose a RNN with attention modeling (R2AM) for lexicon-free scene text recognition. The attention-based mechanism performs soft feature

selection for better image feature usage [99]. Cheng et al. [12] proposed Focus Attention Networks (FAN) to attenuate the attention drift problem in the existing attention based methods. Liu et al. in [63] propose an attention based encoder-decoder which achieves state of art accuracy consuming less computational resources than the previous methods. In another research, Liu et al. [60] present a hierarchical attention mechanism (HAM) which consists of a recurrent RoI-Warp layer and a character-level attention layer. They adopt a local transformation to model the distortion of individual characters, resulting in an improved efficiency, and can handle different types of distortion that are hard to be modeled by a single global transformation [99].

(3) **Other methods**: Jaderberg et al. [44], [71] perform word recognition on the whole image holistically. They train a deep classification model just on data produced by a synthetic text generation engine, and achieve state-of-the-art performance on some benchmarks containing English words only. But application of this method is quite limited as it cannot be applied to recognize long text sequences as phone numbers.

Figure 22 presents a performance comparison between the text recognition methods presented in this section.

Method	ICDAR 2003				ICDAR2013
	50	None	50k	Full	None
Shi et al. [4]	98.7	89.4	**95.5**	97.6	86.7
Gao et al. [25]	98.7	89.2	-	96.7	88
Lee et al. [52]	97.9	88.7	-	97	90
Cheng et al. [12]	**99.2**	**94.2**	-	97.3	
Liu et al. [62]	98.8	93.1	93.8	**97.9**	**92.9**
Liu et al. [59]	-	91.5	-	-	90.8
Jaderberg et al. [43]	98.7	89.6	93.4	97	81.8

Method	IIIT5k			SVT	
	50	1k	None	50	None
Shi et al. [4]	97.6	94.4	78.2	96.4	80.8
Gao et al. [25]	99.1	**97.9**	81.8	**97.4**	82.7
Lee et al. [52]	96.8	94.4	78.4	96.3	80.7
Cheng et al. [12]	**99.3**	97.5	**87.4**	97.1	**85.9**
Liu et al. [62]	97	94.1	87	96.1	-
Liu et al. [59]	-	-	83.6	-	84.4
Jaderberg et al. [43]	95.5	89.6	-	93.5	71.7

Figure 22: Scene text recognition accuracies (%) on general benchmarks. "50" and "1k" are lexicon sizes, "Full" indicates the combined lexicon of all images in the benchmark, and "None" means lexicon-free.

C. End-to-end

Recently, instead of a text detection and a text recognition separate systems, a **text spotting system** combines text detection and recognition into a single system [99]. The early work first uses a detection model to generate text proposals and then a recognition model to recognize the text instances such as in the researches [30], [45] and [57]. Liao et al. use in [57] an SDD based text detector and CRNN for text spotting. Jaderberg et al. propose in [45] Edge Box proposals and a trained aggregate feature detector to propose candidate word bounding boxes. Further, the bounding boxes are filtered and rectified , the result being sent to the recognition model proposed in [71]. The main disadvantage of the 2 step method is the error propagation between the detection and the recognition models, that affects the overall performance. Recently, more end-to-end trainable networks tackle this problem [5], [7], [36], [56], [61]. Bartz et al. propose in [5] a STN (Spatial Transformer Network) [72] to circularly attend to each word in the image and them recognize them separately. There are no word bounding boxes used in the training. Li et al. [56] use an encoder-decoder based recognition model as substitution for the object classification module in Faster-RCNN [100]. Liu et al. [61], Busta et al. [7] and He et al. [36] proposed an unified detection and recognition system with a very similar overall architecture which is made of a detection branch and a recognition branch. Liu et al. [61] and Busta et al. [7] adopt EAST [149] and YOLOv2 [90] as their detection branch and have a similar text recognition branch in which text proposals are mapped into fixed height tensor by bilinear sampling and then transcribe in to strings by a CTC-based recognition module [99]. He et al. [36] used also EAST for detection, but introduced character spatial information as explicit supervision in the attention-based recognition branch.

Table 5 presents these end-to-end text spotting methods with their strengths and weaknesses, application domains, techniques and reported performance. Figure 23 presents a performance comparison between those methods. There are two protocols used for evaluation: end-to-end and word-spotting. End-to-end needs to recognize all the words precisely, no matter whether the dictionary contains these strings. On the other hand, word-spotting only examine whether the words in the dictionary appear in images, making it less strict than the end-to-end protocol for ignoring symbols, numbers and words whose length is less than 3. Also, for both evaluation protocols, there are 3 specific lists of words used as lexicons for reference in the test phase, named: "Strong", "Weak" and "Generic". "Strong" lexicon provides 100 words per-image including all words that appear in the image. "Weak" lexicon includes all words that appear in the entire test set. "Generic" lexicon is a 90k word vocabulary.

4.2 Container Code Recognition

There are 3 attributes that differentiate the container code from scene text in general: a container code is an alpha-numeric string, the characters are always in upper-case and the corrugated surface of the container body which distorts the container code. There have been several researchers trying to solve the container code recognition problem:

- Verma et al. [2] proposed an end-to-end pipeline that uses Region Proposals generated based on Connected Components (CC) for text detection in conjunction with Spatial Transformer Networks (STN) for recognition. Their proposed architecture consists of 4 major blocks as presented in Figure 24:
  - Image Processing
  - Extraction of region proposals using CC
  - Classification using STNs

Algorithm	Strength	Weakness	Application domain	Technique used	Performance
Bartz et al. [5] (2017)	simple architecture	can not detect text in arbitrary locations in the image, can detect a fixed number of maximum words	end-to-end scene text spotting	They used a single deep neural network, that learns to detect and recognize text from natural images, in a semi-supervised way. Their network integrates and jointly learns a spatial transformer network(STN) [72], which can learn to detect text regions in an image, and a text recognition network that takes the identified text regions and recognizes their textual content.	78% recognition accuracy on French Street Name Signs(FSNS) dataset and 95,2% sequence recognition accuracy on Street View House Number (SVHN) dataset
Li et al. [56] (2017)	can handle text of different aspect ratios	can not handle multi-orientated text	end-to-end text spotting	They used an encoder-decoder based recognition model as substitution for the object classification module in Faster-RCNN [100].	see Figure 23
Liu et al. [61] (2018)	can handle text of different orientations and it is fast	requires a lexicon	end-to-end fast text spotting	They used EAST [149] for detection and for recognition text proposals are mapped into fixed height tensor by bilinear sampling and then transcribed in to strings by a CTC-based recognition module [99].	see Figure 23
Busta et al. [7] (2017)	fast	sensitive to blur and can not handle small text	end-to-end text spotting	Similar architecture as Liu et al. in [61], but they used YOLOv2 [90] for detection instead of EAST.	see Figure 23
He et al. [36] (2018)	can handle complex backgrounds	requires a lexicon	end-to-end text spotting	Similar architecture as Liu et al. in [61]. They also used EAST [149] for detection, but introduced character spatial information as explicit supervision in the attention-based recognition branch.	see Figure 23

Table 5: Comparison between existing end-to-end text spotting methods

Word-Spotting	ICDAR 2013			ICDAR2015		
Method	Strong	Weak	Generic	Strong	Weak	Generic
Li et al. [55]	-	-	-	**94.16**	**92.42**	**88.2**
Liu et al. [60]	**95.94**	**93.9**	**87.76**	87.01	82.39	67.97
Busta et al. [7]	92	89	81	58	53	51
He et al. [36]	93	92	87	85	80	65

End-to-end	ICDAR 2013			ICDAR2015		
Method	Strong	Weak	Generic	Strong	Weak	Generic
Li et al. [55]	-	-	-	91.08	**89.81**	**84.59**
Liu et al. [60]	**91.99**	**90.11**	84.77	**93.55**	79.11	65.33
Busta et al. [7]	89	86	77	54	51	47
He et al. [36]	91	89	**86**	82	77	63

Figure 23: Performance comparison of text spotting methods showing the F-measure in percentage

– Sequence generation
First the input image is pre-processed (scaling and binarization) making the separation of the characters easier, then using CCs text region proposals are generated. Those proposals go as input into the classification networks. They used 2 classification networks, 1 for digits and 1 for letters. Further, they compare the first 4 characters generated by the STN Classification with all the codes from the ISO Code Directory to find which character sequence is valid. After having a valid sequence of the first 4 characters, they use heuristics to find the other part of the code, trying to find a code that fits the ISO pattern. Their experiments show an 100% coverage for detection and 99,64% accuracy for the recognition of the separate characters and 95% accuracy for the recognition of the whole code. Their solution seems to have an impressive performance, but they used just 19 images for testing that, from my point of view, it is not enough to consider the results significant. Also, they weren't able to compare their solution with prior solutions because each of them was conducted using different datasets.

- Mei et al. [52] proposed a different approach for the container code identification task, detecting and recognizing each character separately and them combining them to form

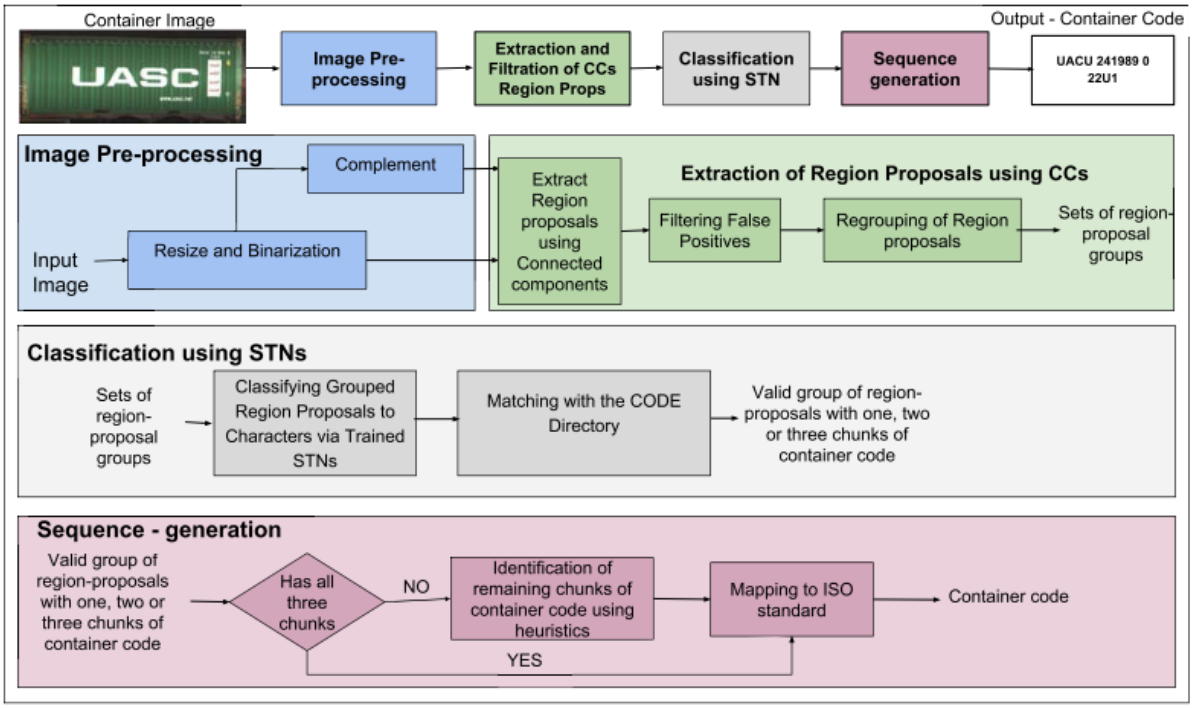

Figure 24: Container Code Recognition Pipeline proposed by Verma et al. Source:[2]

a code. First, the characters are isolated, then template matching(TM) based on hand crafted features is used for detection and 2 LeNet5 trained separately, one for digits and 1 for letters are used for recognition. Then the characters recognized are combined according with their credibility and the verification rule of container codes. They use real container images for training, in contrast with the previous work that used synthetic images for training and real ones just for testing. Moreover they use all 4 sides of each container and apply the described algorithm for all 4 side images. Further they combine the results using the 4 sides and using heuristics and the check digit, they choose the best character sequence. They report a 91,9 % accuracy after their experiments combining results from all 4 sides of the container in the final result.

- Yoon et al. [142] studies the effect of using multiple container sides images on the recognition accuracy. They use Connected Component Analysis (CCA) for character segmentation and Support Vector Machine (SVM) for recognition. The SVM algorithm uses 44 classes , 10 for numbers, 24 for letters (O and I are considered as 0 and respectively 1) and 10 for numbers with surrounding rectangles (for the check digits). Then they proposed a character-level integration and decision-level integration of the results. The character-level integration produces a new code fabricated from the recognized codes of the planes. The decision-level integration finally selects one among six codes including the new code from the character-level integration [142]. Figure 25 presents the processing flow of the character-level integration and an example for each step. They reported a 70% accuracy recognition from a single view and 96% accuracy for integration of 5 container views.
- Koo and Cha [51] proposed another method which uses vertical edge information, a spatial structure window, and texture clustering. The vertical edge information is extracted using a top-hat transform. The candidate region and type of ISO-Code is obtained using a Spatial Structure Window (SSW) which wraps around the vertical edges. The ISO-Code is extracted using texture clustering by the K-Means algorithm which is then recognized by a Back-propagation Neural Network (BP) [51]. The results yielded a recognition rate of 98.39%.
- Rozinat et al. [147] proposed a container code extraction algorithm. They used template matching, but preceded by a series of filters in order to reduce noise. The flow diagram that they propose is presented in Figure 26. Their method can segment out container code characters properly with a ratio above 96%.
- Kumano et al. [104] proposed a character recognition scheme based on a dynamic design method for recognizing differing character string layouts in container marks or numbers. Field tests have been conducted and they obtained a recognition rate of 92.8%.
- Chen et al. [38] use computer vision techniques to tackle the container code identification task reporting 87.95 % recognition rate on a 83 images test set.

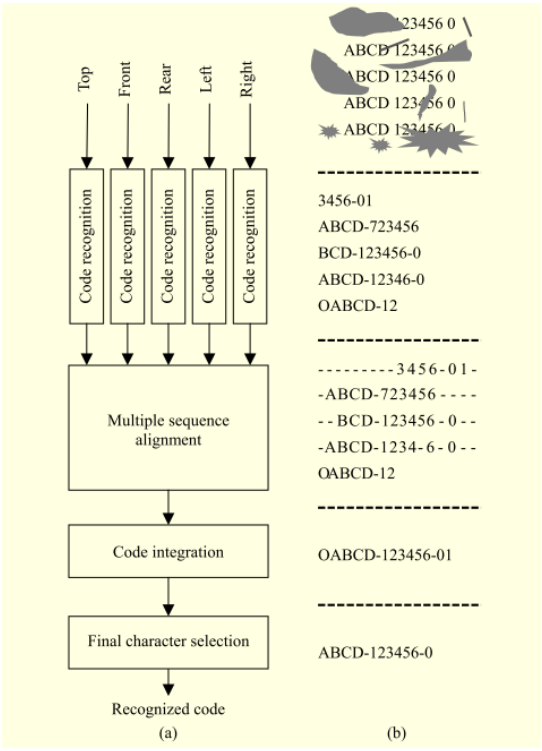

Figure 25: Character-level integration of the code recognition results: (a) processing flow and (b) example results of each step. Source:[142]

Table 6 presents these methods with their strengths and weaknesses, techniques and reported performance.

As a small conclusion of the related work findings, each of the related researches are based on some assumptions, achieve impressive performances on certain challenges and fail to deal with other challenges. After the container management system domain analysis, it turns out that most of the time, the container code is damaged and this is the biggest problem that even humans face when they have to identify the containers. Thus, this research will study the effect of the damage on the performance of the container code identification. Table 4 and Table 6 present the most relevant approaches for this research.

5 DATASET AND METHODOLOGY

5.1 Methodology

Considering that most of the specific challenges posed by the container code influence the detection of the code and that the character recognition is a largely studied problem, the proposed pipeline is split in 2 parts: detection of the container code and recognition of the container code characters. A standard simple recognition algorithm is chosen, while a new approach is used for container code detection, approach based on a recent research by Baek et al. [3].

Thus, the proposed pipeline is presented in Figure 30 and consists of 2 main parts:

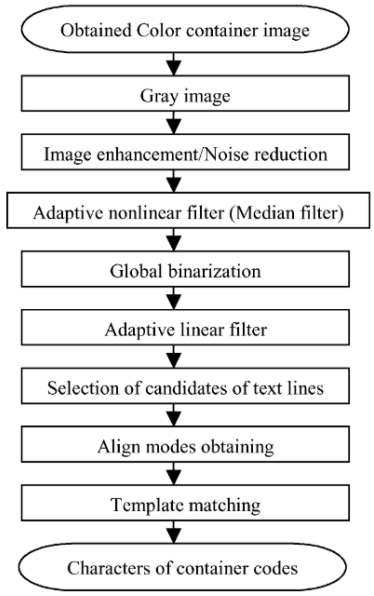

Figure 26: Flow diagram of the system proposed by Rozinat et al. Source:[147]

Code Characters	Detection Rate	Recognition Rate	Overall Rate (Detection+Recognition)
Alphabets	100%	98.96%	98.96%
Digits Only	100%	100%	100%
Complete Code	100%	99.64%	99.64%

Figure 27: Results reported by Verma et al. in [2]. Source:[2]

Method		Accuracy (%)
Recognition from a single view	Top	56.33
	Front	47.71
	Rear	71.45
	Left	42.45
	Right	39.51
Proposed algorithm using multiple views without character-level integration		93.33
Proposed algorithm using multiple views		96.20

Figure 28: Results reported by Yoon et al. in [142]. Source:[142]

(1) Detection
(2) Recognition

5.1.1 ***Detection.*** The detection phase has 3 steps as illustrated in Figure30:

(1) Pre-processing: affine transformation of the input image
(2) CNN processing

Algorithm	Strength	Weakness	Technique used	Performance
Verma et al. [2] (2017)	can handle text spatial transformations such as translation, scale and rotation	requires a lexicon for the first 4 characters, relies on manually defined rules and works only when the code is perfectly visible	Their proposed architecture consists of 4 major blocks as presented in Figure 19:Image Processing,Extraction of region proposals using CC,Classification using STNs and Sequence generation.	see Figure 27 and Figure 29
Mei et al. [52] (2017)	can deal with missing characters using multiple sides of the containers	uses a predetermined range of character size and relies on manually defined rules	They proposed a framework based on convolutional neural network (CNN) and template matching (TM)	see Figure 29
Yoon et al. [142] (2016)	can deal with missing characters using multiple sides of the containers	works only when the code is perfectly visible	They used Connected Component Analysis (CCA) for character segmentation and Support Vector Machine (SVM) for recognition.	see Figure 28 and Figure 29
Koo and Cha [51] (2013)	can handle light variations	works only when the code is perfectly visible	They proposed a method which uses vertical edge information, a spatial structure window (SSW), texture clustering and a Back-propagation Neural Network (BP).	see Figure 29
Rozinat et al. [147] (2005)	can handle noisy images and light variations	can not handle distorted characters	They used template matching, preceded by a series of filters in order to reduce noise	see Figure 29
Kumano et al. [104] (2004)	can handle light variations	works only when the code is perfectly visible	They proposed a character recognition scheme based on a dynamic design method for recognizing differing character string layouts in container marks or numbers.	see Figure 29

Table 6: Comparison between existing container code recognition methods

Method	Detection+Recognition rate
Verma et al. [2]	99.64
Mei et al. [51]	91.9
Yoon et al. [137]	96.2
Koo and Cha [50]	98.39
Rozinat et al. [142]	96
Kumano et al. [100]	92.8

Figure 29: Performance comparison between the existing container code recognition algorithms (%)

(3) Post-processing: threshold applied on the regional score map

Pre-processing: Before going as input to the CNN network, the input image is transformed in order to reduce camera distortions. An affine transformation is used to reduce the skewness of the container code, this process being visually illustrated in Figure 30.

CNN processing: A fully convolutional neural network (CNN) architecture based on the approach proposed by Baek et al. in [3] is used for detection. The first part is based on VGG-16[108] network and overall is similar to U-net[95]. The architecture is illustrated in Figure 31. All layers use as activation function ReLu. The network was trained using the Adam optimizer and binary crossentropy as loss function. In order to prevent overfitting, dropout with a rate of 0.5 is used after each Max Pooling layer and after each concatenation operation.

The CNN was trained using the dataset described in Section 5.2.1.

The input of the CNN is a RGB container image with a size of 256x256x3.

The output of the CNN are 2 score maps of size 128x128x1: a region score map and an affinity score map. The region score map represents the probability that each pixel is a centre of an character and the affinity score map represents the probability that each pixel is the center of the space between 2 adjacent characters. The probability scores are encoded using a Gaussian heatmap, rather than binary pixel values. Heatmaps have been used before in other applications such as pose estimation and are really recently proved by Baek et al. [3] to work on text detection due to their scale invariance, flexibility when dealing with regions that are not rigidly bounded and focusing on intra and inter character level rather than on the whole text instance. Considering that the container code

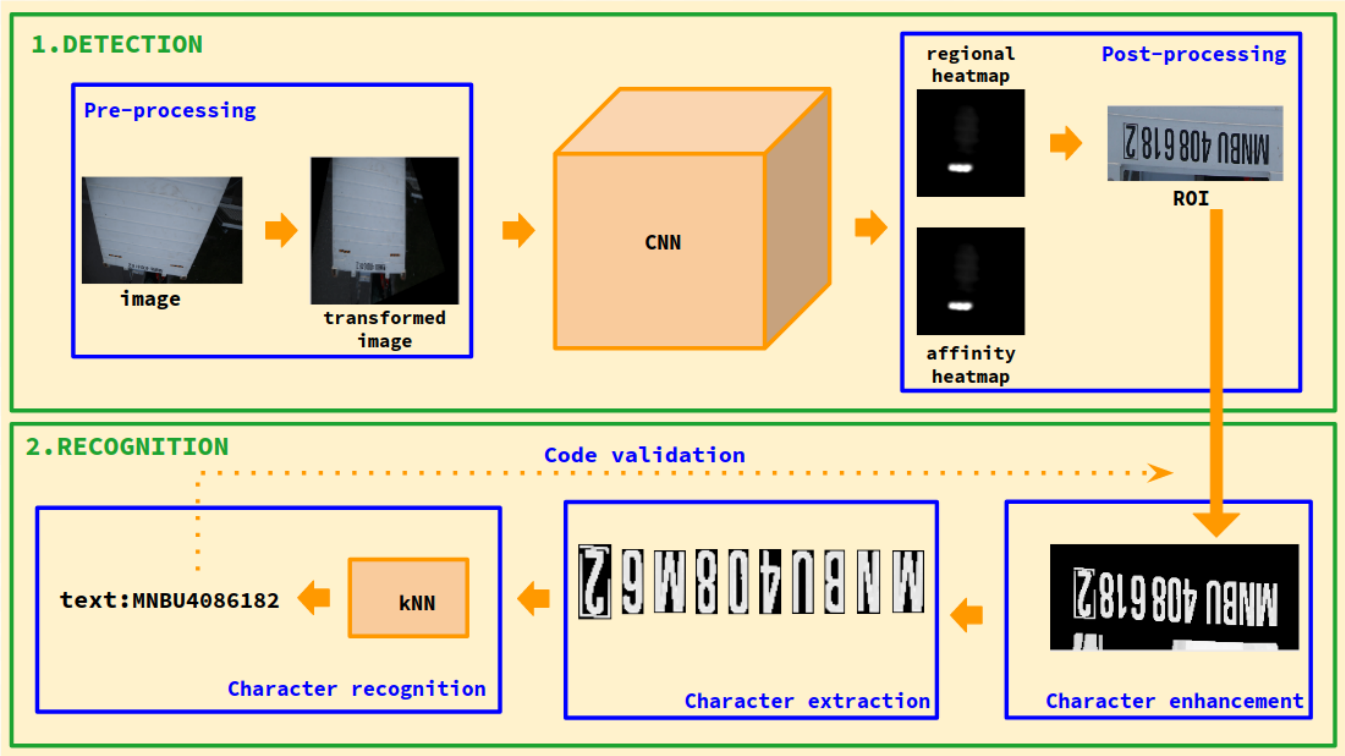

Figure 30: The pipeline of the proposed solution

characters are distorted, the following process is followed in order to generate the heatmaps [3]:

- create a 2 dimensional Gaussian isotropic map that will be used further for all the characters
- for each character:
  - use the character coordinates to create a character box
  - compute a perspective transform between the prepared Gaussian map and the character box
  - using the perspective transform, warp the Gaussian map to the box area

The process is illustrated in Figure 20. For the affinity heatmaps the same process is applied, but instead of the character box, the boxes are defined as presented in Figure 20, using 2 adjacent characters.

The region score map is used further in the pipeline, the affinity score map is only used in the training phase, in order to get more insight about the text, namely about the space between characters.

Post-processing: The regional map represents score probabilities for each pixel to be the center of a character and thus, it can be used to locate the text in the image. Applying a threshold over this map results in isolating the Regions of Interest (ROIs). Each ROI will be a piece of the original image that contains text. Thus, one or more text ROIs will be the output of the detection phase as illustrated in Figure 30.

5.1.2 ***Recognition***. The recognition phase has 4 steps:

(1) Character enhancement
(2) Character extraction
(3) Character recognition
(4) Code validation

The **character enhancement** consists of the following image processing operations:

(a) convert into grayscale
(b) increase contrast using the CLAHE technique
(c) enhance the white colour
(d) apply a mask to find the white colour
(e) apply dilation with a 2x2 seed
(f) obtain the inverted image from the grayscale image from step (a). Then repeat all the steps from (b) until (e) on the inverted image.

The colour of the container code text is not known and using both, the original image and the inverted image, together with a mask for white colour, both dark and light text are enhanced.

Character extraction: Using the outputs of the step (e) for both, the original image and the inverted image, all the objects in the images are extracted and each of them will represent a region of interest (ROI). Each ROI is a part of the image resulted after dilation, not part of the original image.

Figure 32 illustrates the character enhancement and extraction steps.

Character recognition: In the literature, a K-Nearest Neighbours (kNN) algorithm is the most promising classifiers for this

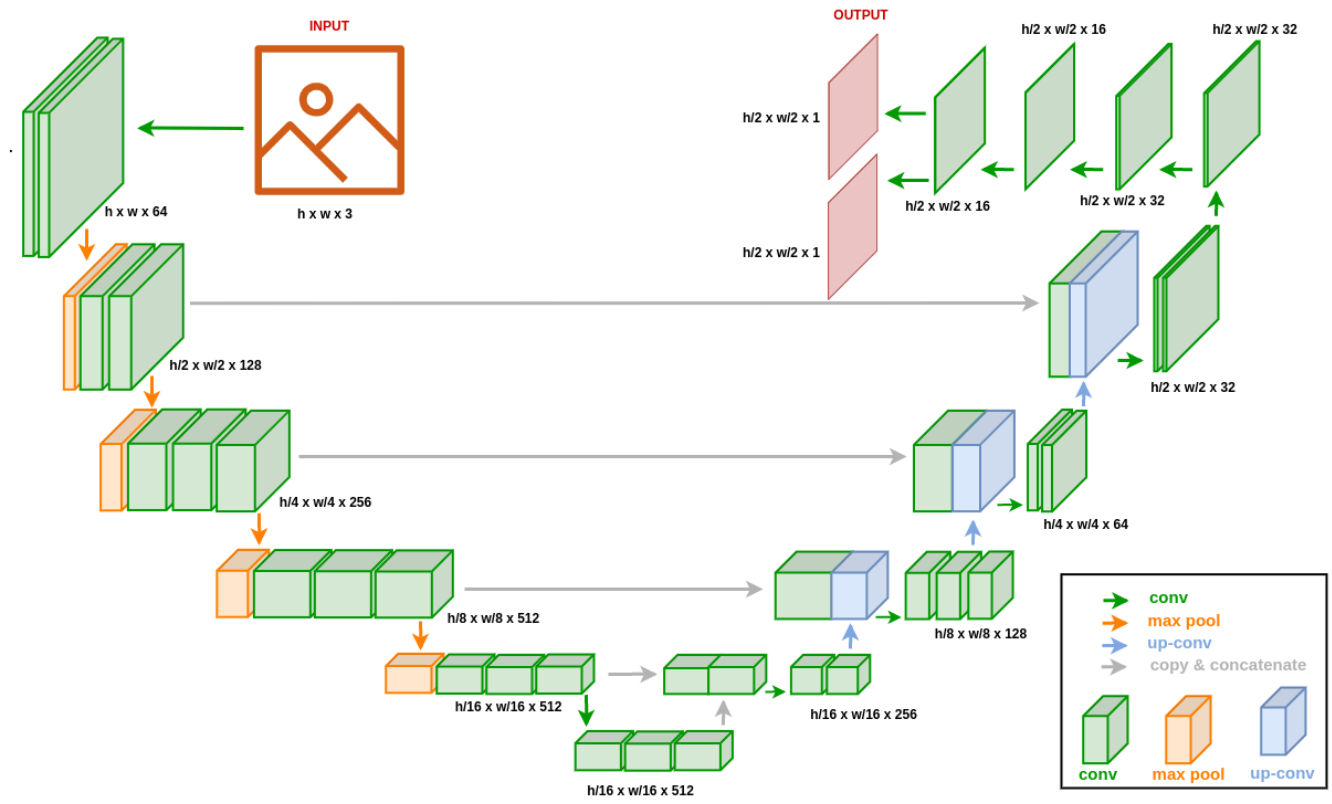

Figure 31: The CNN architecture for detection

task, and thus, a kNN classifier is used for character recognition. Each ROI from the previous step follows the next process:

(1) the ROI is resized at 128x128 pixels
(2) the ROI is flattened
(3) a kNN classifier is used and a text character is assigned to the ROI

The kNN was trained using the dataset described in Section 5.2.3. The parameters used are the following:

- k=1 neighbour
- Euclidian distance

Using only the closest neighbour (k=1) was chosen due to the highly imbalanced dataset. Some classes have a really low amount of samples (even 1 sample for 2 classes), while others have a high number of samples (more than 50).

Code validation: The input of this step is represented by a list of ordered characters. This character list is tested in order to verify if it meets all the requirements of a container code:

- It fits the ISO code format: 4 letters and 7 digits.
- It verifies the check digit: applying the algorithm described in Section 2 on the first 10 characters results the 11th character.

If the character list meets all the requirements, then a complete container code is found and returned. If the character list does not meet all the requirements, then the ROI is rotated till the container code is found. Considering the position of the camera and the possible orientations of the container and container code, there are 3 rotation angles that will be used, in this order: 180 degrees, 270 degrees and 90 degrees. If, after checking all the 4 perspectives, a complete container code is not found, then all the incomplete container code candidates are evaluated. If there are more candidates, then the one that fits the most of the ISO code is returned. A schematic representation of the code validation step is presented in Figure 33.

If the detection phase returns more than one ROI, then each ROI, one by one, follows the same process. Assuming that each image contains only one container code, if at any point a complete container code is found, the process stops and any other ROIs left are not evaluated anymore. If a complete code is not found in any of the ROIs, then all the incomplete candidates are combined and the best one from the total is returned. Figure 34 illustrates a case when 2 ROIs are returned by the detection phase.

5.1.3 ***Evaluation***. The evaluation can be split in 3 phases:

- Detection evaluation
- Recognition evaluation
- Overall evaluation

The overall evaluation will be done using a customized performance metric called the performance vector, illustrated in Figure 35. The results of the overall pipeline fall in one of the 2 categories:

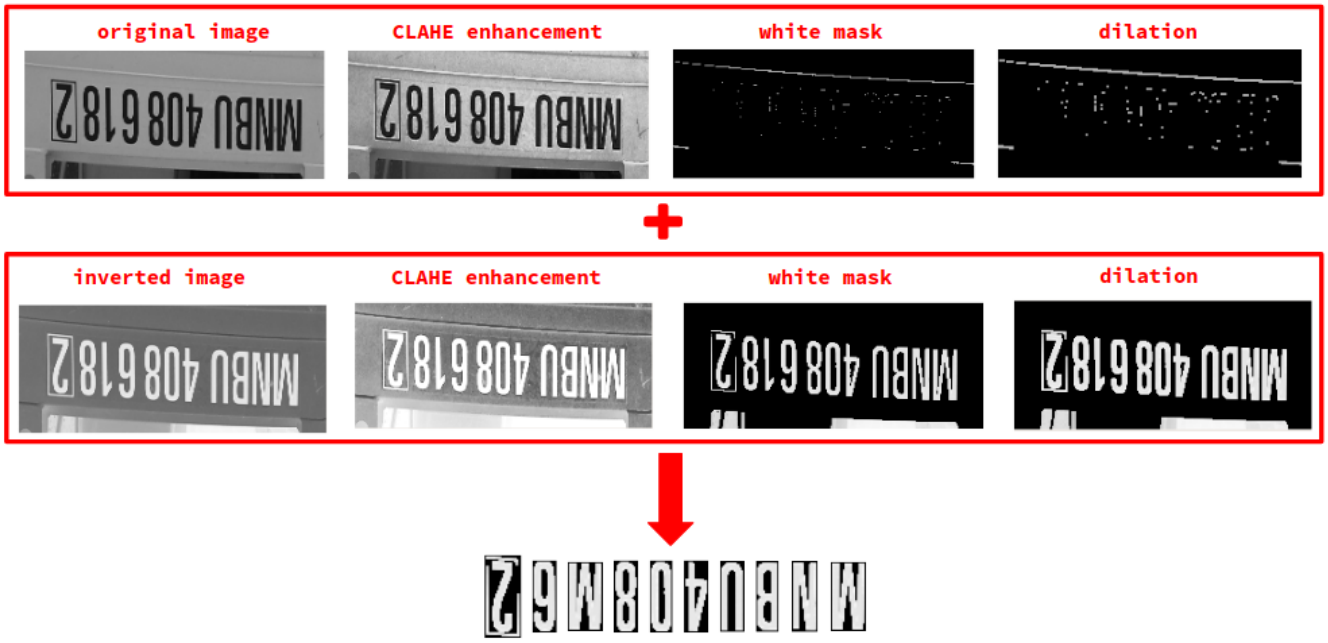

Figure 32: Character enhancement and extraction

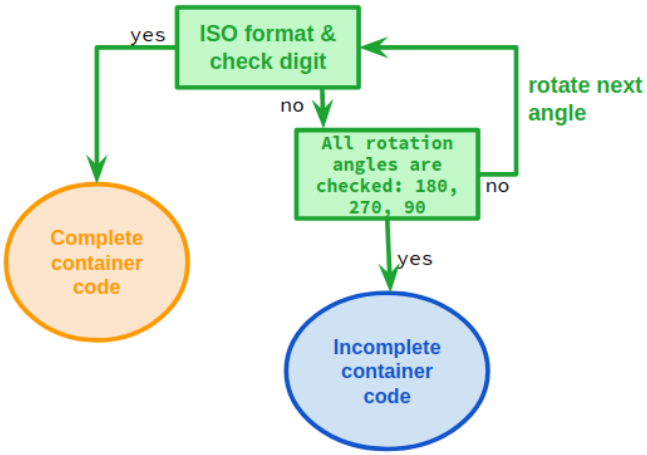

Figure 33: Code validation

- category 1: A complete container code
- category 2: An incomplete container code

The performance vector follows the same approach being composed by 7 indicators:

- complete and correct: the number of containers for which the algorithm found a complete container code and the code is the real one
- complete but wrong: the number of containers for which the algorithm found a complete container code, but the code is not the real one
- incomplete 10 ch: the number of containers for which the algorithm found 10 correct characters matching the real code
- incomplete 9 ch: the number of containers for which the algorithm found 9 correct characters matching the real code
- incomplete 8 ch: the number of containers for which the algorithm found 8 correct characters matching the real code
- incomplete 7 ch: the number of containers for which the algorithm found 7 correct characters matching the real code
- incomplete < 7 ch: the number of containers for which the algorithm found less than 7 correct characters matching the real code

The overall evaluation shows how many times the proposed algorithm is right or wrong and it is influenced by the mistakes made by both, the detection and the recognition phases.

The recognition is evaluated using the confusion matrix of the kNN classifier. The accuracy will be reported as well, but considering that the classes are highly imbalanced, the confusion matrix is the best indicator for the performance and also to see what kind of mistakes the classifier makes.

The output of the detection will be visually inspected. The accuracy of the CNN was proved to be irrelevant, as presented further in the Discussion section.

Besides the visual inspection, a bounding box based evaluation called Intersection over Union (IoU) or Jaccard index is used. This method measures the similarity between the real and the predicted ROI using the overlapping and the union areas as illustrated in Figure 36.

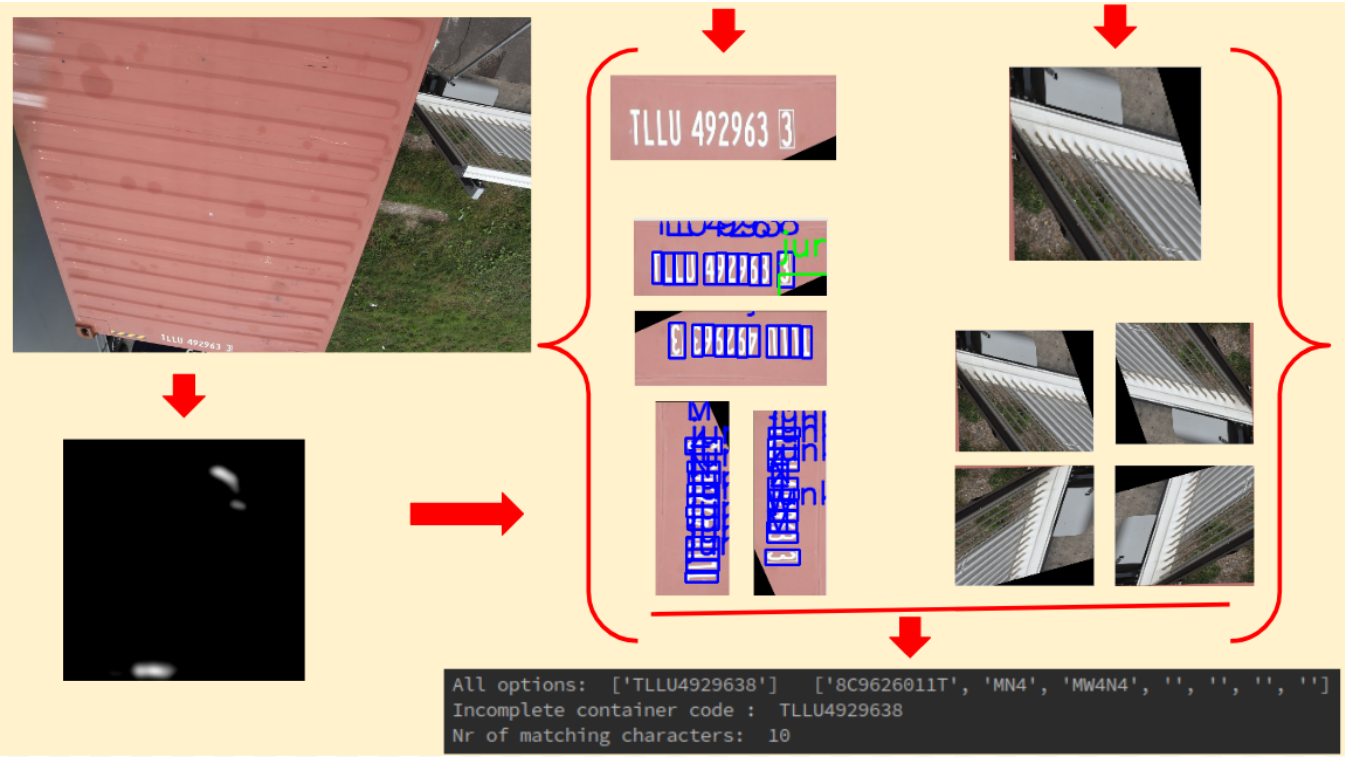

Figure 34: Schematic example when multiple text ROIs are returned by detection

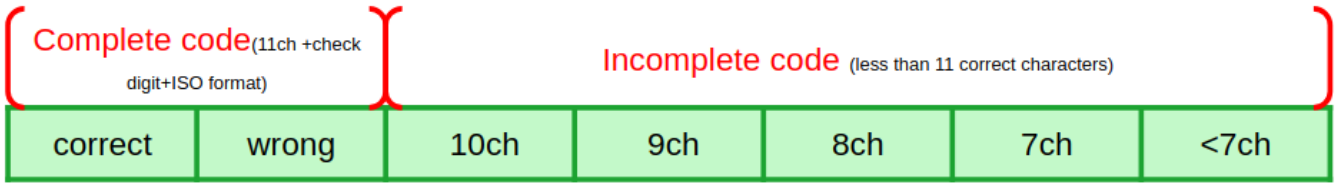

Figure 35: The performance vector

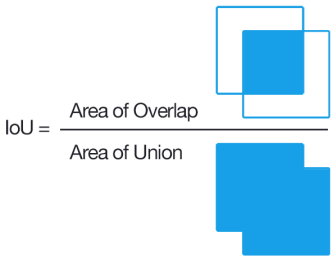

Figure 36: Intersection over Union (IoU) formula. Source: [96]

Another way to evaluate the quality or added value of the proposed detection method, is using the proposed recognition method and performance vector described before, but 3 different detection methods:

(1) the proposed detection
(2) the state-of-art EAST text detector
(3) a manually ROI detection

The comparison between the performance vectors of 3 detection methods and the same recognition method will indicate if the proposed detection method brings any added value for this task.

In order to see the impact of the damaged containers for the automatic container identification, the evaluation will be done separately for containers in good shape and damaged containers.

The overall evaluation was done using the dataset described in Section 5.2.2.

5.2 Datasets

In this section, different datasets used will be presented.

5.2.1 ***CNN container dataset.*** This consists of 112 images of containers, each of them with a visible container code, some containers being in a good shape and others having different degrees of damage. To gather these data, Cofano Software Solutions placed a camera setup at one of their clients, a container terminal from The Netherlands. The images are in the RGB colour space and each image has a resolution of 4000 x 6000 pixels. The images are taken from the top, thus each image displays the top part of a container with a complete/partially visible container code. The images are taken when trucks leave or enter the terminal with containers and they have to stop on the weighing bridge placed near the entrance gate. Figure 38 shows the weighing bridge near the entrance gate and Figure 37 presents the image acquisition setup. The pole has a box in top and the camera is placed in the box. This setup was placed at the entrance gate, near the weighing bridge.

Figure 38: Weighing bridge near the entrance gate

Figure 37: Image acquisition setup

All the container code characters were manually annotated with 4 points representing the 4 corners of each character. Considering that the characters are distorted, the resulted bounding boxes are not rectangular, but mostly diamond shaped. This resulted in 44 points (11 characters x 4 points per character) as annotation for each image. These point coordinates are used to create the regional and affinity maps needed to train the CNN detection architecture described in Section 5.1.1.

Because the available number of real container images was limited, an **image augmentation** process was chosen to increase the amount of images for the CNN training. The following processing steps were applied to each image in order to create 10 new images:

- rotation with a random angle between 15 and 345
- the scale remains constant
- colour alteration: one of the colour channels RGB or all 3 channels are randomly chosen and for each of the chosen channels or channel a random value between 0 and 30 is added/subtracted from that channel's value

In this way, some diversity in terms of the container code position in the image, but also colour variety is provided, obtaining a dataset of 1120 images (112 real containers x 10).

Considering that the character annotation of the real images and heatmap creation is done before augmentation, the same rotation operations are also applied to the corresponding regional and affinity heatmaps, but the colour alterations are applied only to the container images since that does not influence the heatmaps.

5.2.2 ***Overall evaluation container dataset.*** Later in the process, a new set of container images was acquired, in the exactly same conditions as the container dataset used to train the CNN model. This dataset was used to evaluate the performance of the proposed solution. The containers were split in:

- containers in good shape
- damaged containers

The containers were assigned manually in one of the categories. The dataset contains 52 containers in good shape and 26 damaged containers. It is important to mention that most of the containers present damage, but for the sake of this experiment, a container is considered damaged when the container code printed on it is damaged. Thus, if a container is damaged, but the container code printed on it is not damaged, the container is considered in good shape. Both categories, damaged and in good shape containers, were handled in the same manner, but used in different performance evaluation tests.

In comparison to the dataset used for the CNN training, these containers were not annotated at character level. Instead, they were

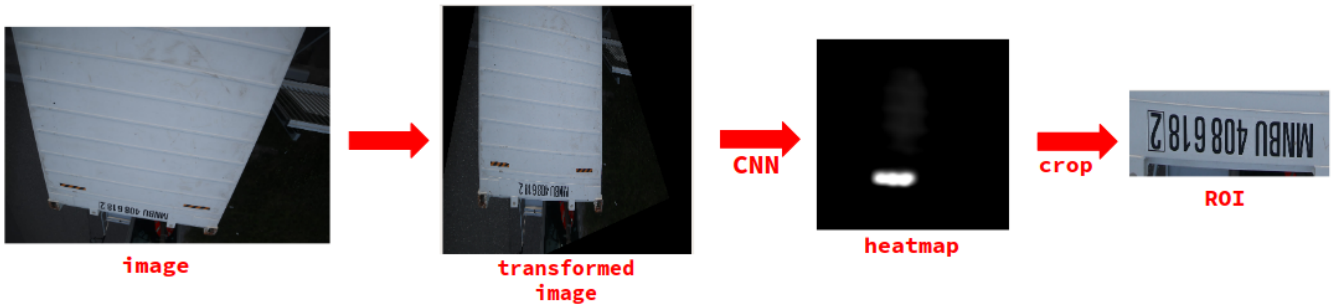

Figure 39: Detection result example for a container in good shape

manually and visually inspected and labeled with the container code they contain, in order to be able to verify to what extent the code returned by the proposed solution matches the real code.

In a normal situation, only one container dataset is needed, both with character level annotations and container code label, split further for different purposes. Considering that the data was received in stages, the need for manually annotations and the time constraints, only the required annotations were applied to each dataset.

5.2.3 ***Character recognition dataset.*** This dataset containts 1393 images of digits and capital letters split in 28 classes:

- 10 digits: 0,1,2,3,4,5,6,7,8,9
- 17 capital letters: A, B, C, D, E, G, H, L, M, N, P, R, S, T, U, W, Z (some letters are not present in the dataset because they did not occur in any of the 112 containers analysed)
- junk: images with scratches, rust, wholes, poles, all other object found on a container body that are not text

The same containers used to train the CNN model were also used to extract the characters for this dataset. The character enhancement procedure described in Section 5.1.2 was applied to each image and then the characters were manually extracted from the dilated images and labeled with the corresponding character tag.

70% of this dataset was used for training and 30% for testing.

6 RESULTS

6.1 Detection results

Both, the output of the CNN network and the resulted ROIs were visually inspected for the containers in good shape and damaged containers as well. Figure 39 presents the step by step outputs for a container in good shape and Figure 40 illustrates the detection results for containers with different properties such as small or big text, a different code layout, damaged text and different light conditions. The visual inspection showed that for all the containers in good shape the container code was correctly detected, resulting in a 100% detection rate. Sometimes multiple ROIs were detected as for the container presented in Figure 40 case (a), but one of them was, in all cases, the container code region. Regarding the damaged containers, in 80% of the cases, the container code region was correctly identified.

The resulted average of the IoU index for the containers in good shape is 95% and 80% for damaged containers.

6.2 Recognition results

Figure 41 presents the performance of the kNN character classifier in terms of accuracy and confusion matrix. The accuracy is 95.42%. For the confusion matrix, the rows are the real classes and the columns are the predicted classes. For example, the first class, class 0, has 32 samples, 29 are predicted in the right class, 2 as class 6 and 1 as class 9.

6.3 Overall results

Further, the overall performance results are presented for the 3 different detection methods, all combined with the same recognition method. The performance results are reported using the performance vector described in Section 5.1.3.

6.3.1 ***Proposed detection.*** Figure 42 and 44 show the performance for containers in good shape and Figure 43 and 45 for damaged containers.

Figure 42 shows that in 100% of the cases when the proposed pipeline found a complete container code, the resulted container code was actually the real one. Thus, every time when the propose pipeline outputs a complete container code, this is 100% the real container code. Figure 44 illustrates that the proposed algorithm found a complete code for 40,4 % of the 52 containers in the test set.

All the orange variations of colour in Figure 44 belong to the incomplete code category. In 28.8% of the cases, 10 out of 11 correct characters are found, in 7.7% of the cases 9 out of 11 characters are found and so on. In 5.8% of the cases or namely 3 containers out of a total of 52, the proposed algorithm found less than 7 correct characters and these containers fell also in the incomplete code category.

The distribution over the 2 big categories is the following:

- 40,4% for the complete container code category
- 59,6 % for the incomplete container code category

Figure 40: Detection results for containers with different code properties: (a) small text, (b) big text, (c) different layout, (d) damaged text, (e) and (f) different light conditions

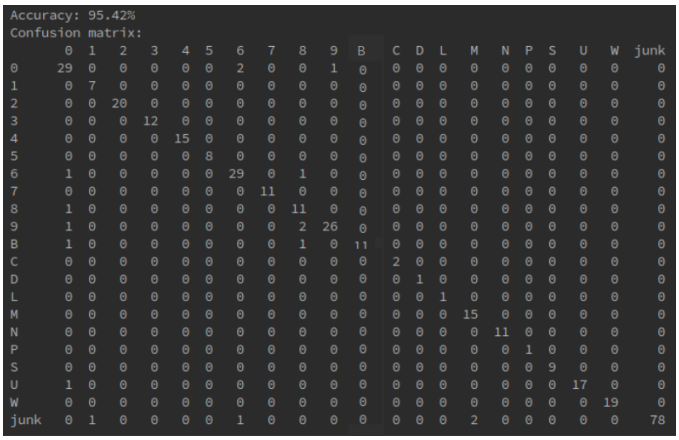

Figure 41: The performance of the kNN character classifier (The rows of the confusion matrix are the real classes and the columns are the predicted classes)

Total	C correct	C wrong	I 10ch	I 9ch	I 8ch	I 7ch	I <7ch
52	21	0	15	4	6	3	3

Figure 42: The performance vector for the containers in good shape using the proposed detection method (C=complete code and I=incomplete code)

total	C correct	C wrong	I 10ch	I 9ch	I 8ch	I 7ch	I <7ch
26	3	0	4	7	6	3	2

Figure 43: The performance vector for the damaged containers using the proposed detection method (C=complete code and I=incomplete code)

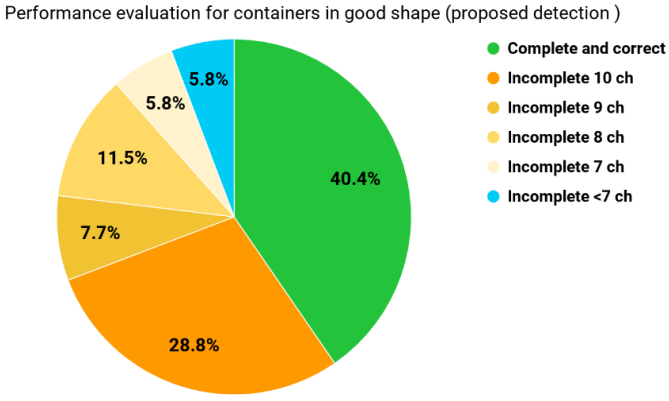

Figure 44: Graphically representation of the performance vector for containers in good shape using the proposed detection

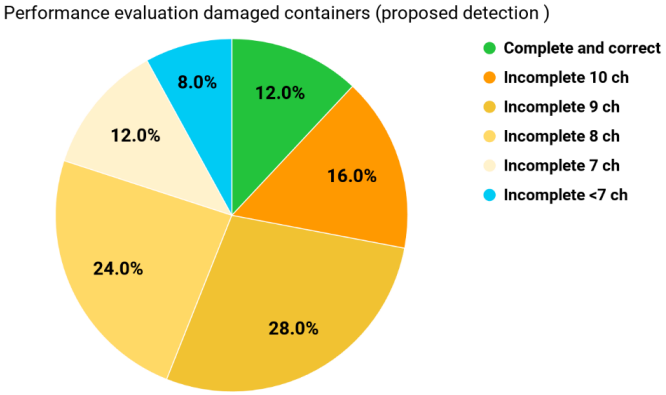

Figure 45: Graphically representation of the performance vector for damaged containers using the proposed detection

total	C correct	C wrong	I 10ch	I 9ch	I 8ch	I 7ch	I <7ch
52	0	0	0	0	1	2	49

Figure 46: The performance vector for the containers in good shape using EAST as detection method (C=complete code and I=incomplete code)

total	C correct	C wrong	I 10ch	I 9ch	I 8ch	I 7ch	I <7ch
26	0	0	0	0	0	0	26

Figure 47: The performance vector for the damaged containers using EAST as detection method (C=complete code and I=incomplete code)

Total	C correct	C wrong	I 10ch	I 9ch	I 8ch	I 7ch	I <7ch
52	21	0	15	4	6	3	3

Figure 48: The performance vector for the containers in good shape using a manual detection (C=complete code and I=incomplete code)

total	C correct	C wrong	I 10ch	I 9ch	I 8ch	I 7ch	I <7ch
26	3	0	4	7	6	3	2

Figure 49: The performance vector for the damaged containers using a manual detection (C=complete code and I=incomplete code)

For the damaged containers Figure 45 illustrates the distribution of performance classes. The same applies here as well, when the proposed solution finds a complete container code, that is in 100% of the case the real container code. The distribution over the 2 big categories for the damaged containers is the following:

- 12,0% for the complete container code category
- 88,0 % for the incomplete container code category

6.3.2 ***EAST detection*** . Figure 46 shows the performance for containers in good shape and Figure 47 for damaged containers.

6.3.3 ***Manual detection*** . Figure 48 shows the performance for containers in good shape and Figure 49 for damaged containers.

7 DISCUSSION

7.1 Detection

It is clear from Figure 40 that the detection method proposed can successfully deal with the container code challenges described in Section 3, being able to locate the container code in all images. At a close look at Figure 40 it can be seen that the proposed method can deal with variations in text size and font, different colours for background and text, different container code layouts, damaged text and different light conditions. Cases (e) and (f) show that the detection is possible at all times during the day, from the morning (e) till the evening (f). It is easy to remark the grass in image (e) and how this grass is not visible in image (f), due to the reduced lighting conditions during the evening. Experiments during the night were not conducted because the container terminal where the dataset was gathered does not operate during the night. Considering that during the night an external source of light is needed in order to be able to acquire images, the lightning variation is eliminated. Controlling the light source will make the light constant in all the images and the proposed detection method is expected to have a better performance, but this has to be further analysed for a certain conclusion.

Another information provided by the heatmap, besides the location of the potential container code, is the **intensity** of the highlighted region. This tells how certain the algorithm is that there is a container code. This is better illustrated when case (b) and case (c) from Figure 40 are compared. In case (b), where the container code is bigger and in perfect shape, the algorithm is more confident that the container code is there than in case (c) where the text is smaller and the container is partially damaged. The algorithm is able to find the container code in both cases, but with different degrees of certainty.

The threshold used to extract the text ROI from the image takes the histogram of the regional heatmap into consideration. Applying always a high threshold will only result in finding the perfect container code regions with high certainties such as case (b) from Figure 40 and missing the other ones such as case (c). On the other hand, always choosing a low threshold means, in some cases, that a really big part of the image has to be passed into the recognition step, fact that is not desired. This scenario will happen applying a low threshold in case (e). Using an adaptive threshold based on the histogram of the regional map assures that all the container code regions are found, be they with higher or lower certainties.

The value of the **IoU index** of 95% for containers in good shape shows, besides the visual inspection, that the proposed detection method can successfully locate the container code. At a close analysis between the predicted and ground truth regions used to calculate the IoU index, it was observed that the difference is made by a small amount of pixels around the container code text. The manual labelling does not take the same amount of pixels around the text every time, while the algorithm does. Thus, the predicted and ground truth bounding boxes are not 100% identical, but contain the container code region 100% of times.

The value of the IoU index for the containers with different degrees of damage of 80% shows that the proposed method can deal with damaged containers as well. It can be seen in Figure 40 case (d) that if the damaged characters are somewhere in the middle of the code, the whole container code region is correctly extracted. On the other hand, there are containers where the first or last characters are damaged. In these cases a partial container code region is extracted, causing the IoU index to drop to 80%.

The **accuracy** of the CNN detection model is 0.3%. Considering that the accuracy is calculated based on pixel values, that the real heatmaps are annotated on character level and the predicted heatmaps highlight the container code as a whole, the accuracy is not relevant as a performance metric in this case. While the CNN model does not output a character delimitation, but a whole code delimitation, this outcome is actually better than character delimitation. The desired output of the detection phase at the beginning of the research was character delimitation, but considering the limited amount of samples in the training set, the model did not have enough data to learn from. The resulted output proved to be more useful than the desired one, making the post-processing for the recognition part easier. It is known that the container code has 11 characters and has to respect the ISO standard, namely 4 letters followed by 7 digits. Having the text regions as a whole makes it easier to discriminate which is a container code region and which regions contain other text. Figure 50 shows an example of a container image with the corresponding real and predicted regional heatmap. The accuracy for this image is 0.5%, but it can be clearly seen that the CNN model has successfully located the container code. To be noted that in Figure 50 the container image and the real heatmap correspond to the original image and the predicted heatmap corresponds to the transformed container image. The comparison is made between all parts involved using the transformed version, but for illustration purposes, the raw version of each of them was chosen.

Making a comparison between the overall results using 3 different detection methods and the same recognition method presented in Figures 42, 45, 46, 47, 48 and 49, it results that the proposed detection method performs exactly as good as a manual detection. Considering that a manual detection is the best it can be achieved, it is proven that the automatic detection method proposed is the best that can be achieved to automate this task under the given conditions. This applies for both, the containers in good shape and the damaged containers. At a manually inspection, a difference is observed between the manually detected regions and predicted regions by the proposed detection method for damaged containers. As it was explained before, if some characters are highly damaged at the extremities of a container code, the proposed detection is

Figure 50: The real and predicted regional map for a container

not able to retrieve these in the ROI area. On the other hand, a manual detection will include them as well. Considering the degree of damage, the recognition phase will not be able to find the right character, and the end results will be the same as not including that character in the ROI in the first place. This is why the end results are identical for the manual detection and the proposed detection.

Regarding the EAST text detector, it seems unsuitable for this task. EAST is known as being state-of-art in scene text detection, but the experiments conducted on this dataset bring the conclusion that EAST can not deal with the challenges posed by the present dataset. Figure 51 presents 2 containers and the text regions detected using EAST. It can be observed that the detector cuts through the text taking only parts of the characters in the ROI. It also highlights false positives, as it can be seen in the second example in Figure 51. A major difference between the proposed detector and the EAST detector is the fact that EAST takes each chunk of the container code as a different text ROI and the proposed method takes the text region as a whole. On the containers where multiple text pieces are present, using the proposed method makes it faster to check if the text ROI is a container code or not. Considering that the post-processing can be done to put together all the separate pieces if EAST correctly detects all the container code pieces and for the consistency of the comparison with the manual and the proposed detection, a manual post-processing of putting together the chunks was done to obtain the results of the experiments presented in Figure 46 and Figure 47. Regarding the processing time, EAST is incontestably faster than the proposed detection. The processing time for EAST is on average 1 second and for the proposed detection is on average 45 seconds.

7.2 Recognition and the overall pipeline

In terms of performance of the whole proposed pipeline, Figure 44 shows the distribution of the results for containers in good shape and Figure 45 for damaged containers. As for the detection results, the recognition results and the intermediate ones were also manually inspected in order to identify the aspects that make the proposed solution fail. Figure 52 shows 4 example of containers, their corresponding detection results and overall results after recognition. Cases (a) and (b) show containers in good shape and cases (c) and (d) damaged containers. In cases (a) and (b) the characters are very clear and still the recognition phase is not able to predict the good characters. In this way, a perfect detection followed by a bad recognition ends with a completely useless end result, such as case (a) in Figure 52.

Manually inspecting all the containers resulted in identifying two reasons for a failed recognition:

(1) The first reason, and the one that happens most of the time, is that the kNN classifier does not predict correctly the characters. Considering the dataset used and the fact that not all the possible characters are even present in the training dataset, this problem can easily be solved, adding more samples. In this way, the biggest limitation of the proposed recognition, but also of the whole proposed pipeline is solved. The focus of this research is the detection of the container code region, and thus, a limited amount of time was dedicated to the recognition phase. Taking into consideration the high amount of research in the field of character recognition and also the easy, but time consuming way to improve the proposed solution, further improvement of the kNN classifier was considered not relevant.
(2) The second reason that happens in few of the cases is the character enhancement and extraction. In this part, the proposed algorithm highly relies on image processing techniques and some characters are not correctly extracted. Some future work may consider training the proposed CNN architecture used for detection with a bigger training set, fact that will lead to a character separation directly as output of the detection phase. Then a clustering techniques can be used to see which characters are close together and form the container code and which form other text on the container body. In such a situation, the post processing of the detection phase and the character enhancement and extraction of

Figure 51: Examples of the container code detection using EAST

the recognition phase, illustrated in Figure 30, will be eliminated. This will lead to a simplified pipeline, with almost no manually defined rules.

For the damaged containers, the damaged characters are most of the time extracted but classified as junk.

The accuracy of the kNN classifier is 95.42%, but considering that the classes are highly imbalanced, the confusion matrix is a better indication of the performance. Some of the letters (F, I, J, K, O, Q, Y, X, V) are not present in the training set because they were not present in the 112 containers used to create the dataset. Also, part of the letters (A, E, G, H, R, T, Z) are not present in the confusion matrix because these classes had less than 5 samples each and they were all used in the training set. Thus, the confusion matrix only shows the performance for only 21 classes. It can be seen in the confusion matrix in Figure 41 that most of the samples from the test set are predicted in the right class, a 6 is predicted as 0 and two 0's as 6, a 0 is predicted as 9 and two samples from the junk class are predicted as M, two 9's and one 6 are predicted as 8 and one U is predicted as 0. In general class 0 has the most false positives and class junk the most false negatives, meaning that some junk such as scratches or rust is seen as useful characters.

Looking at Figure 45 and Figure 49 it can be seen that summing up all the numbers from all the categories results 25 and the total is 26. This is because one of the containers does not fall in any of the categories mentioned. This container has all the 11 characters correctly identified, but due the to highly degree of damage between the characters, some damaged pieces of container background are seen as characters. In this way, all the container code characters are present in the end result, but between them, some extra fake characters are inserted, making impossible to know which are actually real characters and which is just damaged background looking like a character.

It can also happen that 11 characters are found, but only some of them are correct, as in Figure 34. In this case, the ISO format requirement is fulfilled, but the check digit requirement is not. This means that the code will fall in the incomplete code category, even if it has 11 characters in total. It is impossible to know which, out of the 11 characters, are the real ones, but as long as not all the requirements are fulfilled, it is known that the container is not complete.

Considering the aspects presented earlier about the performance of the detection and the reasons why the recognition is failing, the overall performance of the proposed pipeline can be highly increased.

7.3 Added value, limitations and future work

The added value, limitations and future work of this research concern the following aspects:

- As most of the previous researches, either in natural scene text understanding or container code identification, this research was also focused on specific aspects. All previous researches considered containers in perfect shape and did not consider the case of damaged containers. The business domain analysis showed that most of the containers in real life present a certain degree of damaged. Even the containers classified in this research as being in good shape present rust, scratches or other types of damage, but in a low proportion, being considered in good shape. The containers considered as damaged in this research present damage on the container code area and damaged text, from partial damaged characters

Figure 52: Examples of containers, the corresponding detection results and the overall results: (a) and (b) are containers in good shape, (c) and (d) are damaged containers

to completely missing characters. Thus the present research brought to attention the impact of the damage to the performance of the automatic container code identification.

- Most of the previous researches in container code identification require lexicons and the proposed solution does not use such a lexicon. It would be useful in the cases when the algorithm can find a big part of the code, to search in a database containing all the possible container codes, to find the complete code. In the situation of the current research, such a database was not available, but it can be used as future work to improve the results. Considering the performance obtained without using a lexicon and the extra time needed to find a match, not using a lexicon can be seen as an advantage.
- The advantage of scale invariance, flexibility in dealing with regions that are not rigidly bounded and the focus on intra and inter character rather than on the whole text instance, brought by the use of heatmaps instead of bounding boxes. This method proved to be successful for the container code detection task and none of the previous researches in container code identification used heatmaps for detection.
- Most of the related researches in container code recognition use 2 networks for character classification, one for digits and one for letters. This leads to an extra need of manually defined heuristics about the place of letters and digits on the containers body and missing one of the characters from the order in the ISO format brings extra complexity in the rules. The proposed recognition uses a very simple kNN classifier that has the potential to be sufficient for this task, if a bigger training set containing all the characters is acquired.
- Considering that the focus of this research was from the beginning the detection part and the comparison made with the manual detection, it can be said that the proposed detection method is the best as it can be for the given task, under the given conditions.
- The container code is printed on 5 sides of the container (door end, 2 sides, top and front). The top side of the container was used in this research and this choice has one main advantage and one disadvantage. The advantage is that the top side presents less to none other text instances than the container code, but the disadvantage is that this side is more exposed to damage than all other sides. Rain, snow and water stay longer on the top part, leading to more rust on the top part than on the other sides. Also, stacking the containers on top of each other brings scratches and more damage to the top part of the containers. Even if having more damage is overall a disadvantage, in the context of studying the effect of the damage on an automatic container code identification method, the top part can be seen as the best choice to study this effect. Future work can use the proposed method on multiple views of the same container, fact that will lead to a better performance. Characters that are damaged on one side can be visible on other sides, leading together to a complete container code.
- The pre-processing step from the detection phase applies an affine transformation on the image. This transformation is calculated using a reference image of the weighing bridge when no container is present. The position of the camera makes the weighting bridge distorted in the image, but it is known that the weighing bridge is a rectangular with parallel lines. This information is used to calculate the parameters of the affine transformation with the idea that after the transformation the weighing bridge will have parallel sides again. This parameters and the affine transformation are then applied to all the images, canceling part of the distortion. In an ideal situation a truck carrying a container will stop in the same position every time, not more to the left, more to the right, more in the front or more in the back, how it is actually happening in real life. This makes the affine transformation to cancel the whole distortion as presented in Figure 53 case (b), but only a small part of it as in Figure 53 case (a). Stopping the truck in various locations brings also variance in the size of the text, not only different distortions, stopping the truck really far away from the camera resulting in a smaller text. A good example are cases (a) and (c) from Figure 52. In case (a) the truck stops at the gate line and in case (c) much further. Those variations and distortions do not pose a problem for the detection part, but make the recognition of the distorted characters more challenging, when it can be avoided with an easy non-technical solution. The terminal can mark with a line on the ground where the truck driver has to stop, and in this way, all the drivers have to stop in the same spot.

Figure 53: Examples of containers before and after transformation

- The results of this research are not comparable with results of related researches due to the different datasets used. However, the methods used for evaluation give enough insights regarding the performance of the proposed solution.

8 CONCLUSION

In conclusion, and to answer the main research question of this thesis: "To what extent is the automatic identification of the container code possible using machine learning ?", the automatic container code identification is possible, but it faces some challenges. The automatic detection of the container code from container in good shape has a successful rate of 100%, the proposed method being able

to find the code region and deal with all the described challenges and variations. The overall identification of the container code for containers in good shape faces some challenges due to the kNN classifier used, challenges that can be solved increasing the recognition dataset. Regarding the damaged containers, an automatic detection proved to be successful as well, but having some issues with heavily damaged characters at the extremities of a container code. The overall automatic identification for damaged containers is also affected in a negative way by the poor recognition and also by the degree of text damage. Considering that the proposed detection performs as good as a manual detection and the proposed recognition can be easily improved or substituted with another method from the wide available OCR researches, an automatic container code identification system was proven to be successful. To be noted that when the proposed solution outputs a complete container code, that is in 100% of the times the real container code, with 0% false positive rate.

REFERENCES

[1] Jon Almazan, Albert Gordo, Alicia Fornes, and Ernest Valveny. 2014. Word spotting and recognition with embedded attributes. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2014). https://doi.org/10.1109/TPAMI.2014.2339814

[2] Ramya Hebbalaguppe Ehtesham Hassan Lovekesh Vig Ankit Verma, Monika Sharma. 2016. Automatic container code recognition via Spatial Transformer Networks and Connected Component Region Proposals. *Proceedings - 2016 15th IEEE International Conference on Machine Learning and Applications, ICMLA 2016* (2016), 728–733. https://doi.org/10.1109/ICMLA.2016.48

[3] Youngmin Baek, Bado Lee, Dongyoon Han, Sangdoo Yun, and Hwalsuk Lee. 2019. Character Region Awareness for Text Detection. (apr 2019). arXiv:1904.01941 http://arxiv.org/abs/1904.01941

[4] Cong Yao Baoguang Shi, Xiang Bai. 2017. 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* (2017), 1455–1461. https://doi.org/10.1109/TPAMI.2016.2646371

[5] Christian Bartz, Haojin Yang, and Christoph Meinel. 2017. SEE: Towards Semi-Supervised End-to-End Scene Text Recognition. 1 (2017). arXiv:1712.05404 http://arxiv.org/abs/1712.05404

[6] Jason Brownlee. 2018. A Gentle Introduction to the Bootstrap Method. (May 2018). https://machinelearningmastery.com/a-gentle-introduction-to-the-bootstrap-method/

[7] Michal Busta, Lukas Neumann, and Jiri Matas. 2017. Deep TextSpotter: An End-to-End Trainable Scene Text Localization and Recognition Framework. *Proceedings of the IEEE International Conference on Computer Vision* 2017-Octob (2017), 2223–2231. https://doi.org/10.1109/ICCV.2017.242

[8] Michal Buta, Luka Neumann, and Jiri Matas. 2015. FASText: Efficient unconstrained scene text detector. In *Proceedings of the IEEE International Conference on Computer Vision*. https://doi.org/10.1109/ICCV.2015.143

[9] Bernard Gosselin Celine Thillou. 2014. Natural Scene Text Understanding. *Vision Systems: Segmentation and Pattern Recognition* (May 2014), 27. https://doi.org/10.5772/4966

[10] Xilin Chen, Jie Yang, Jing Zhang, and Alex Waibel. 2004. Automatic detection and recognition of signs from natural scenes. *IEEE Transactions on Image Processing* (2004). https://doi.org/10.1109/TIP.2003.819223

[11] Xiangrong Chen and Alan L Yuille. 2004. Detecting and Reading Text in Natural Scenes. 00, C (2004).

[12] Zhanzhan Cheng, Fan Bai, Yunlu Xu, Gang Zheng, Shiliang Pu, and Shuigeng Zhou. 2017. Focusing Attention: Towards Accurate Text Recognition in Natural Images. In *Proceedings of the IEEE International Conference on Computer Vision*. https://doi.org/10.1109/ICCV.2017.543

[13] Chucai Yi and YingLi Tian. 2011. Text String Detection From Natural Scenes by Structure-Based Partition and Grouping. *IEEE Transactions on Image Processing* (2011). https://doi.org/10.1109/tip.2011.2126586

[14] Adam Coates, Blake Carpenter, Carl Case, Sanjeev Satheesh, Bipin Suresh, Tao Wang, David J. Wu, and Andrew Y. Ng. 2011. Text detection and character recognition in scene images with unsupervised feature learning. *Proceedings of the International Conference on Document Analysis and Recognition, ICDAR* (2011), 440–445. https://doi.org/10.1109/ICDAR.2011.95

[15] ContainerContainer. [n. d.]. ISO6346 International Shipping Container Standard. ([n. d.]). https://www.containercontainer.com/ISO6346

[16] Yuchen Dai, Zheng Huang, Yuting Gao, Youxuan Xu, Kai Chen, Jie Guo, and Weidong Qiu. 2018. Fused Text Segmentation Networks for Multi-oriented Scene Text Detection. In *Proceedings - International Conference on Pattern Recognition*. https://doi.org/10.1109/ICPR.2018.8546066

[17] Navneet Dalal and Bill Triggs. 2005. Histograms of oriented gradients for human detection. In *Proceedings - 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR 2005*. https://doi.org/10.1109/CVPR.2005.177

[18] Dan Deng, Haifeng Liu, Xuelong Li, and Deng Cai. 2018. PixelLink: Detecting Scene Text via Instance Segmentation. (2018). arXiv:1801.01315 http://arxiv.org/abs/1801.01315

[19] Bureau International des Containers et du Transport Intermodal. 2019. Marking of containers. (2019). https://www.bic-code.org/marking-of-containers/

[20] DongQuin Zhang and Shih-Fu Chang. 2003. A Bayesian framework for fusing multiple word knowledge models in videotext recognition. https://doi.org/10.1109/cvpr.2003.1211512

[21] Boris Epshtein, Eyal Ofek, and Yonatan Wexler. 2010. Detecting text in natural scenes with stroke width transform. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. https://doi.org/10.1109/CVPR.2010.5540041

[22] Pedro F. Felzenszwalb and Daniel P. Huttenlocher. 2005. Pictorial structures for object recognition. *International Journal of Computer Vision* (2005). https://doi.org/10.1023/B:VISI.0000042934.15159.49

[23] Rohith Gandhi. 2018. R-CNN, Fast R-CNN, Faster R-CNN, YOLO — Object Detection Algorithms. (2018). https://towardsdatascience.com/r-cnn-fast-r-cnn-faster-r-cnn-yolo-object-detection-algorithms-36d53571365e

[24] Rohith Gandhi. 2018. Support Vector Machine — Introduction to Machine Learning Algorithms. (2018). https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47

[25] Yunze Gao, Yingying Chen, Jinqiao Wang, Ming Tang, and Hanqing Lu. 2019. Reading scene text with fully convolutional sequence modeling. *Neurocomputing* (2019). https://doi.org/10.1016/j.neucom.2019.01.094

[26] Michael Garbade. 2018. Understanding K-means Clustering in Machine Learning. (2018). https://towardsdatascience.com/understanding-k-means-clustering-in-machine-learning-6a6e67336aa1

[27] Ross Girshick. 2015. Fast R-CNN. In *Proceedings of the IEEE International Conference on Computer Vision*. https://doi.org/10.1109/ICCV.2015.169

[28] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. 2014. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. https://doi.org/10.1109/CVPR.2014.81

[29] Albert Gordo. 2015. Supervised mid-level features for word image representation. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. https://doi.org/10.1109/CVPR.2015.7298914

[30] Ankush Gupta, Andrea Vedaldi, and Andrew Zisserman. 2016. Synthetic Data for Text Localisation in Natural Images. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. https://doi.org/10.1109/CVPR.2016.254

[31] Onel Harrison. 2018. Machine Learning Basics with the K-Nearest Neighbors Algorithm. (2018). https://towardsdatascience.com/machine-learning-basics-with-the-k-nearest-neighbors-algorithm-6a6e71d01761

[32] Dafang He, Xiao Yang, Wenyi Huang, Zihan Zhou, Daniel Kifer, and C. Lee Giles. 2017. Aggregating local context for accurate scene text detection. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. https://doi.org/10.1007/978-3-319-54193-8_18

[33] Dafang He, Xiao Yang, Chen Liang, Zihan Zhou, Alex G. Ororbia, Daniel Kifer, and C. Lee Giles. 2017. Multi-scale FCN with cascaded instance aware segmentation for arbitrary oriented word spotting in the wild. In *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*. https://doi.org/10.1109/CVPR.2017.58

[34] Kaiming He, Georgia Gkioxari, Piotr Dollar, and Ross Girshick. 2017. Mask R-CNN. In *Proceedings of the IEEE International Conference on Computer Vision*. https://doi.org/10.1109/ICCV.2017.322

[35] Pan He, Weilin Huang, Tong He, Qile Zhu, Yu Qiao, and Xiaolin Li. 2017. Single Shot Text Detector with Regional Attention. In *Proceedings of the IEEE International Conference on Computer Vision*. https://doi.org/10.1109/ICCV.2017.331

[36] Tong He, Zhi Tian, Weilin Huang, Chunhua Shen, Yu Qiao, and Changming Sun. 2018. An End-to-End TextSpotter with Explicit Alignment and Attention. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. https://doi.org/10.1109/CVPR.2018.00527

[37] Wenhao He, Xu Yao Zhang, Fei Yin, and Cheng Lin Liu. 2017. Deep Direct Regression for Multi-oriented Scene Text Detection. In *Proceedings of the IEEE International Conference on Computer Vision*. https://doi.org/10.1109/ICCV.2017.87

[38] Fu-Yu Hsu Yu-San Lin Yu-Te Wu Yung-Nien Sun Hsin-Chen Chen, Chih-Kai Chen. 2011. A computer vision system for automated container code recognition. *Proceedings of the Internatioanal MultiConference of Engineers and Computer Scientists 2011* (2011), 470–474. https://www.scopus.com/inward/record.uri?eid=

2-s2.0-79960601061&partnerID=40&md5=52ece29270f6ac841fc122067d3de6b0
[39] Weilin Huang, Zhe Lin, Jianchao Yang, and Jue Wang. 2013. Text localization in natural images using stroke feature transform and text covariance descriptors. In *Proceedings of the IEEE International Conference on Computer Vision*. https://doi.org/10.1109/ICCV.2013.157
[40] Weilin Huang, Yu Qiao, and Xiaoou Tang. 2014. Robust scene text detection with convolution neural network induced MSER trees. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. https://doi.org/10.1007/978-3-319-10593-2_33
[41] Zhida Huang, Zhuoyao Zhong, Lei Sun, and Qiang Huo. 2018. Mask R-CNN with Pyramid Attention Network for Scene Text Detection. (nov 2018). arXiv:1811.09058 http://arxiv.org/abs/1811.09058
[42] Adaptive Recognition Hungary. [n. d.]. General description of ACCR. ([n. d.]). http://www.ocrtech.com/container_code_description.html
[43] ICDAR. [n. d.]. Robust Reading Competition. ([n. d.]). https://rrc.cvc.uab.es/
[44] Max Jaderberg, Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. 2014. Deep Structured Output Learning for Unconstrained Text Recognition. (2014), 1–10. arXiv:1412.5903 http://arxiv.org/abs/1412.5903
[45] Max Jaderberg, Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. 2016. Reading Text in the Wild with Convolutional Neural Networks. *International Journal of Computer Vision* (2016). https://doi.org/10.1007/s11263-015-0823-z
[46] Anil K. Jain and Bin Yu. 1998. Automatic text location in images and video frames. *Pattern Recognition* (1998). https://doi.org/10.1016/S0031-3203(98)00067-3
[47] Seong-Whan Lee Alan Yuille Christof Koch Jung-Jin Lee, Pyoung-Hean Lee. 2011. AdaBoost for text detection in natural scene. *Proceedings of the International Conference on Document Analysis and Recognition, ICDAR* (2011), 429–434. https://doi.org/10.1109/ICDAR.2011.93
[48] D. Karatzas and A. Antonacopoulos. 2004. Text extraction from web images based on a split-and-merge segmentation method using colour perception. In *Proceedings - International Conference on Pattern Recognition*. https://doi.org/10.1109/ICPR.2004.1334328
[49] Anil K. Jain Keechul Jung, Kwang In Kim. 2003. Text information extraction in images and video : a survey Pattern Recognition. (2003), 977–997.
[50] Kwang In Kim, Keechul Jung, and Jin Hyung Kim. 2003. Texture-Based Approach for Text Detection in Images Using Support Vector Machines and Continuously Adaptive Mean Shift Algorithm. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2003). https://doi.org/10.1109/TPAMI.2003.1251157
[51] Eui-Young Cha Kyung-Mo Koo. 2012. A novel container ISO-code recognition method using texture clustering with a spatial structure window. *International Journal of Advanced Science and Technology* (2012), 83–92.
[52] Qing Liu Pingping Lu Langqi Mei, Jianming Guo. 2017. A Novel Framework for Container Code-Character Recognition Based on Deep Learning and Template Matching. *Proceedings - 2016 International Conference on Industrial Informatics - Computing Technology, Intelligent Technology, Industrial Information Integration, ICIICII 2016* (2017), 78–82. https://doi.org/10.1109/ICIICII.2016.0030
[53] Chen Yu Lee and Simon Osindero. 2016. Recursive Recurrent Nets with Attention Modeling for OCR in the Wild. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. https://doi.org/10.1109/CVPR.2016.245
[54] Seonghun Lee and Jin Hyung Kim. 2013. Integrating multiple character proposals for robust scene text extraction. *Image and Vision Computing* (2013). https://doi.org/10.1016/j.imavis.2013.08.007
[55] Huiping Li, David Doermann, and Omid Kia. 2000. Automatic text detection and tracking in digital video. *IEEE Transactions on Image Processing* (2000). https://doi.org/10.1109/83.817607
[56] Hui Li, Peng Wang, and Chunhua Shen. 2017. Towards End-to-End Text Spotting with Convolutional Recurrent Neural Networks. *Proceedings of the IEEE International Conference on Computer Vision* 2017-Octob (2017), 5248–5256. https://doi.org/10.1109/ICCV.2017.560 arXiv:1707.03985
[57] Minghui Liao, Baoguang Shi, Xiang Bai, Xinggang Wang, and Wenyu Liu. 2016. TextBoxes: A Fast Text Detector with a Single Deep Neural Network. (2016). https://doi.org/10.1162/jocn arXiv:1611.06779
[58] Minghui Liao, Zhen Zhu, Baoguang Shi, Gui Song Xia, and Xiang Bai. 2018. Rotation-Sensitive Regression for Oriented Scene Text Detection. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. https://doi.org/10.1109/CVPR.2018.00619
[59] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng Yang Fu, and Alexander C. Berg. 2016. SSD: Single shot multibox detector. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. https://doi.org/10.1007/978-3-319-46448-0_2
[60] Wei Liu, Chaofeng Chen, and Kwan-Yee K Wong. 2018. Char-Net: A Character-Aware Neural Network for Distorted Scene Text Recognition. In *Aaai*.
[61] Xuebo Liu, Ding Liang, Shi Yan, Dagui Chen, Yu Qiao, and Junjie Yan. 2018. FOTS: Fast Oriented Text Spotting with a Unified Network. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. https://doi.org/10.1109/CVPR.2018.00595
[62] Yuliang Liu and Lianwen Jin. 2017. Deep matching prior network: Toward tighter multi-oriented text detection. In *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*. https://doi.org/10.1109/CVPR.2017.368
[63] Zichuan Liu, Yixing Li, Fengbo Ren, Hao Yu, and Wangling Goh. 2018. Squeezed-Text: A Real-time Scene Text Recognition by Binary Convolutional Encoder-decoder Network. *Proceedings of the Association for the Advancement of Artificial Intelligence, AAAI* (2018).
[64] Canjie Luo, Lianwen Jin, and Zenghui Sun. 2019. MORAN: A Multi-Object Rectified Attention Network for scene text recognition. *Pattern Recognition* 90 (2019), 109–118. https://doi.org/10.1016/j.patcog.2019.01.020 arXiv:arXiv:1901.03003v1
[65] Michael R. Lyu, Jiqiang Song, and Min Cai. 2005. A comprehensive method for multilingual video text detection, localization, and extraction. *IEEE Transactions on Circuits and Systems for Video Technology* (2005). https://doi.org/10.1109/TCSVT.2004.841653
[66] Pengyuan Lyu, Minghui Liao, Cong Yao, Wenhao Wu, and Xiang Bai. 2018. Mask textspotter: An end-to-end trainable neural network for spotting text with arbitrary shapes. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 11218 LNCS (2018), 71–88. https://doi.org/10.1007/978-3-030-01264-9_5 arXiv:1807.02242
[67] Pengyuan Lyu, Cong Yao, Wenhao Wu, Shuicheng Yan, and Xiang Bai. 2018. Multi-Oriented Scene Text Detection via Corner Localization and Region Segmentation. (2018), 7553–7563. https://doi.org/10.1109/CVPR.2018.00788 arXiv:1802.08948
[68] Jianqi Ma, Weiyuan Shao, Hao Ye, Li Wang, Hong Wang, Yingbin Zheng, and Xiangyang Xue. 2018. Arbitrary-oriented scene text detection via rotation proposals. *IEEE Transactions on Multimedia* (2018). https://doi.org/10.1109/TMM.2018.2818020
[69] J Matas, O Chum, M Urban, and T Pajdla. 2001. Distinguished Regions for Wide-baseline Stereo. *Research Reports of CMP, Czech Technical University in Prague* 33 (2001).
[70] MathWorks. [n. d.]. Getting Started with R-CNN, Fast R-CNN, and Faster R-CNN. ([n. d.]). https://nl.mathworks.com/help/vision/ug/getting-started-with-r-cnn-fast-r-cnn-and-faster-r-cnn.html
[71] Max Jaderberg. 2003. Synthetic Data and Artificial Neural Networks for Natural Scene Text Recognition. *Diabetes Research and Clinical Practice* 60, 3 (2003), 183–189. https://doi.org/10.1016/S0168-8227(03)00035-4 arXiv:1406.2227v4
[72] Andrew Zisserman Koray Kavukcuoglu Max Jaderberg, Karen Simonyan. 2015. Spatial Transformer Networks. (2015), 1–15. https://doi.org/10.1038/nbt.3343
[73] Anand Mishra, Karteek Alahari, and Cv Jawahar. 2012. Scene Text Recognition using Higher Order Language Priors. In *Procedings of the British Machine Vision Conference 2012*. https://doi.org/10.5244/C.26.127
[74] Anand Mishra, Karteek Alahari, and C. V. Jawahar. 2011. An MRF model for binarization of natural scene text. In *Proceedings of the International Conference on Document Analysis and Recognition, ICDAR*. https://doi.org/10.1109/ICDAR.2011.12
[75] Anand Mishra, Karteek Alahari, and C. V. Jawahar. 2012. Top-down and bottom-up cues for scene text recognition. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. https://doi.org/10.1109/CVPR.2012.6247990
[76] Lukas Neumann and Jiri Matas. 2011. A method for text localization and recognition in real-world images. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. https://doi.org/10.1007/978-3-642-19318-7_60
[77] Lukas Neumann and Jiri Matas. 2012. Real-time scene text localization and recognition. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. https://doi.org/10.1109/CVPR.2012.6248097
[78] Luka Neumann and Jiri Matas. 2013. On combining multiple segmentations in scene text recognition. In *Proceedings of the International Conference on Document Analysis and Recognition, ICDAR*. https://doi.org/10.1109/ICDAR.2013.110
[79] Shigueo Nomura, Keiji Yamanaka, Osamu Katai, Hiroshi Kawakami, and Takayuki Shiose. 2005. A novel adaptive morphological approach for degraded character image segmentation. *Pattern Recognition* (2005). https://doi.org/10.1016/j.patcog.2005.01.026
[80] Tatiana Novikova, Olga Barinova, Pushmeet Kohli, and Victor Lempitsky. 2012. Large-lexicon attribute-consistent text recognition in natural images. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. https://doi.org/10.1007/978-3-642-33783-3_54
[81] Christopher Olah. 2015. Understanding LSTM Networks. (2015). https://colah.github.io/posts/2015-08-Understanding-LSTMs/
[82] World Customs Organization. 2017. *Custom Convention on Containers 1972*. Manning Publications, 109–110.
[83] Savan Patel. 2017. Chapter 2 : SVM (Support Vector Machine) — Theory. (May 2017). https://medium.com/machine-learning-101/chapter-2-svm-support-vector-machine-theory-f0812effc72
[84] Trung Quy Phan, Palaiahnakote Shivakumara, Shangxuan Tian, and Chew Lim Tan. 2013. Recognizing text with perspective distortion in natural scenes. In

Proceedings of the IEEE International Conference on Computer Vision. https://doi.org/10.1109/ICCV.2013.76

[85] Andrei Polzounov, Artsiom Ablavatski, Sergio Escalera, Shijian Lu, and Jianfei Cai. 2018. Wordfence: Text detection in natural images with border awareness. In *Proceedings - International Conference on Image Processing, ICIP*. https://doi.org/10.1109/ICIP.2017.8296476

[86] David Doermann Qixiang Ye. 2015. Text Detection and Recognition in Imagery: A Survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2015), 1480–1500.

[87] Abilash R. 2018. Applying Random Forest (Classification) -Machine Learning Algorithm from scratch with real datasets. (2018). https://medium.com/@ar.ingenious/applying-random-forest-classification-machine-learning-algorithm-from/-scratch-with-real-24ff198a1c57

[88] Sunil Ray. 2017. Understanding Support Vector Machine algorithm from examples. (2017). https://www.analyticsvidhya.com/blog/2017/09/understaing-support-vector-machine-example-code/

[89] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. [n. d.]. You Only Look Once: Unified, Real-Time Object Detection. ([n. d.]). arXiv:arXiv:1506.02640v5

[90] Joseph Redmon and Ali Farhadi. 2017. YOLO9000: Better, faster, stronger. *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017* 2017-Janua (2017), 6517–6525. https://doi.org/10.1109/CVPR.2017.690 arXiv:arXiv:1612.08242v1

[91] Muhammad Rizwan. 2018. LeNet-5 – A Classic CNN Architecture. (2018). https://engmrk.com/lenet-5-a-classic-cnn-architecture/

[92] Jose Rodriguez and Florent Perronnin. 2014. Label embedding for text recognition. https://doi.org/10.5244/c.27.5

[93] Jose A. Rodriguez-Serrano, Albert Gordo, and Florent Perronnin. 2015. Label Embedding: A Frugal Baseline for Text Recognition. *International Journal of Computer Vision* (2015). https://doi.org/10.1007/s11263-014-0793-6

[94] Xuejian Rong, Chucai Yi, and Yingli Tian. 2017. Unambiguous text localization and retrieval for cluttered scenes. In *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*. https://doi.org/10.1109/CVPR.2017.349

[95] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. 2015. U-net: Convolutional networks for biomedical image segmentation. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 9351 (2015), 234–241. https://doi.org/10.1007/978-3-319-24574-4_28 arXiv:1505.04597

[96] Adrian Rosebrock. 2017. Intersection over Union (IoU) for object detection. (2017). https://www.pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection/

[97] Partha Pratim Roy, Umapada Pal, Josep Lladós, and Mathieu Delalandre. 2009. Multi-oriented and multi-sized touching character segmentation using dynamic programming. In *Proceedings of the International Conference on Document Analysis and Recognition, ICDAR*. https://doi.org/10.1109/ICDAR.2009.124

[98] Sumit Saha. 2018. A Comprehensive Guide to Convolutional Neural Networks. (2018). https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

[99] Cong Yao Shangbang Long, Xin He. 2018. Scene Text Detection and Recognition: The Deep Learning Era. (2018), 1–20. http://arxiv.org/abs/1811.04256

[100] Ross Girshick Shaoqing Ren, Kaiming He and Jian Sun. 2017. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (May 2017), 1137–1149. https://doi.org/10.1109/TPAMI.2016.2577031

[101] Karthik Sheshadri and Santosh Divvala. 2012. Exemplar Driven Character Recognition in the Wild. https://doi.org/10.5244/c.26.13

[102] Baoguang Shi, Xiang Bai, and Serge Belongie. 2017. Detecting oriented text in natural images by linking segments. In *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*. https://doi.org/10.1109/CVPR.2017.371

[103] Cunzhao Shi, Chunheng Wang, Baihua Xiao, Yang Zhang, Song Gao, and Zhong Zhang. 2013. Scene text recognition using part-based tree-structured character detection. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. https://doi.org/10.1109/CVPR.2013.381

[104] Mitsuaki Tamagawa Hiroaki Ikeda Koji Kan Shintaro Kumano, Kazumasa Miyamoto. 2004. Development of a container identification mark recognition system. *Electronics and Communications in Japan, Part II: Electronics (English translation of Denshi Tsushin Gakkai Ronbunshi)* (2004), 38–50.

[105] Palaiahnakote Shivakumara, Souvik Bhowmick, Bolan Su, Chew Lim Tan, and Umapada Pal. 2011. A new gradient based character segmentation method for video text recognition. In *Proceedings of the International Conference on Document Analysis and Recognition, ICDAR*. https://doi.org/10.1109/ICDAR.2011.34

[106] Palaiahnakote Shivakumara, Trung Quy Phan, and Chew Lim Tan. 2011. A Laplacian approach to multi-oriented text detection in video. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2011). https://doi.org/10.1109/TPAMI.2010.166

[107] Gidi Shperber. 2018. A gentle introduction to OCR. (2018). https://towardsdatascience.com/a-gentle-introduction-to-ocr-ee1469a201aa

[108] Karen Simonyan and Andrew Zisserman. 2015. Very deep convolutional networks for large-scale image recognition. *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings* (2015), 1–14. arXiv:arXiv:1409.1556v6

[109] Manik Varma Teofilo E. de Campos, Bodla Rakesh Babu. 2016. Character recognition in natural scene images. *2015 International Conference on Information and Communication Technologies, ICICT 2015* (2016). https://doi.org/10.1109/ICICT.2015.7469575

[110] Zhi Tian, Weilin Huang, Tong He, Pan He, and Yu Qiao. 2016. Detecting text in natural image with connectionist text proposal network. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. https://doi.org/10.1007/978-3-319-46484-8_4

[111] GVCT-Grand View Container Trading. 2011. How is the check digit of a container calculated? (2011). http://www.gvct.co.uk/2011/09/how-is-the-check-digit-of-a-container-calculated/

[112] Sik-Ho Tsang. 2018. SSD — Single Shot Detector (Object Detection). (2018). https://towardsdatascience.com/review-ssd-single-shot-detector-object-detection-851a94607d11

[113] Seiichi Uchida. 2014. Text Localization and Recognition in Images and Video. *Handbook of Document Image Processing and Recognition* (2014), 43. https://doi.org/10.1007/978-0-85729-859-1

[114] Rein van den Boomgaard. 2017. Histogram of Oriented Gradients. (2017). https://staff.fnwi.uva.nl/r.vandenboomgaard/IPCV20172018/20172018/LabExercises/HOG.html

[115] Steffen Wachenfeld, Hans Ulrich Klein, and Xiaoyi Jiang. 2006. Recognition of screen-rendered text. In *Proceedings - International Conference on Pattern Recognition*. https://doi.org/10.1109/ICPR.2006.974

[116] Toru Wakahara and Kohei Kita. 2011. Binarization of color character strings in scene images using K-means clustering and support vector machines. In *Proceedings of the International Conference on Document Analysis and Recognition, ICDAR*. https://doi.org/10.1109/ICDAR.2011.63

[117] Cong Wang, Fei Yin, and Cheng Lin Liu. 2018. Scene Text Detection with Novel Superpixel Based Character Candidate Extraction. In *Proceedings of the International Conference on Document Analysis and Recognition, ICDAR*. https://doi.org/10.1109/ICDAR.2017.156

[118] Fangfang Wang, Liming Zhao, Xi Li, Xinchao Wang, and Dacheng Tao. 2018. Geometry-Aware Scene Text Detection with Instance Transformation Network. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. https://doi.org/10.1109/CVPR.2018.00150

[119] Kai Wang, Boris Babenko, and Serge Belongie. 2011. End-to-end scene text recognition. In *Proceedings of the IEEE International Conference on Computer Vision*. 1457–1464. https://doi.org/10.1109/ICCV.2011.6126402

[120] Kai Wang and Serge Belongie. 2010. Word spotting in the wild. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. https://doi.org/10.1007/978-3-642-15549-9_43

[121] Tao Wang, D. Wu, A. Coates, and A. Ng. 2012. End-to-end text recognition with convolutional neural networks. *Pattern Recognition (ICPR), 2012 21st International Conference on* (2012), 3304–3308. http://www-cs.stanford.edu/people/ang/papers/ICPR12-TextRecognitionConvNeuralNets.pdf

[122] Jerod J. Weinman, Erik Learned-Miller, and Allen Hanson. 2007. Fast lexicon-based scene text recognition with sparse belief propagation. In *Proceedings of the International Conference on Document Analysis and Recognition, ICDAR*. https://doi.org/10.1109/ICDAR.2007.4377061

[123] Wikibooks. 2013. Delaunay triangulation. (2013). https://en.wikibooks.org/wiki/Trigonometry/For_Enthusiasts/Delaunay_triangulation

[124] Wikipedia. 2017. Connectionist temporal classification. (2017). https://en.wikipedia.org/wiki/Connectionist_temporal_classification

[125] Wikipedia. 2019. Adaptive histogram equalization. (2019). https://en.wikipedia.org/wiki/Adaptive_histogram_equalization

[126] Wikipedia. 2019. Container terminal. (2019). https://nl.wikipedia.org/wiki/Containerterminal

[127] Wikipedia. 2019. Histogram of oriented gradients. (2019). https://en.wikipedia.org/wiki/Histogram_of_oriented_gradients

[128] Wikipedia. 2019. ISO 6346. (2019). https://en.wikipedia.org/wiki/ISO_6346

[129] Wikipedia. 2019. Recurrent neural network. (2019). https://en.wikipedia.org/wiki/Recurrent_neural_network

[130] Wikipedia. 2019. Shipping container. (2019). https://en.wikipedia.org/wiki/Shipping_container#Intermodal_freight_containers

[131] Dao Wu, Rui Wang, Pengwen Dai, Yueying Zhang, and Xiaochun Cao. 2018. Deep Strip-Based Network with Cascade Learning for Scene Text Localization. In *Proceedings of the International Conference on Document Analysis and Recognition, ICDAR*. https://doi.org/10.1109/ICDAR.2017.140

[132] Weijia Wu, Jici Xing, and Hong Zhou. 2019. TextCohesion: Detecting Text for Arbitrary Shapes. (apr 2019). arXiv:1904.12640 http://arxiv.org/abs/1904.12640

[133] Yue Wu and Prem Natarajan. 2017. Self-Organized Text Detection with Minimal Post-processing via Border Learning. In *Proceedings of the IEEE International*

Conference on Computer Vision. https://doi.org/10.1109/ICCV.2017.535

[134] Cong Yao, Xiang Bai, Wenyu Liu, Yi Ma, and Zhuowen Tu. 2012. Detecting texts of arbitrary orientations in natural images. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. https://doi.org/10.1109/CVPR.2012.6247787

[135] Cong Yao, Xiang Bai, Nong Sang, Xinyu Zhou, Shuchang Zhou, and Zhimin Cao. 2016. Scene Text Detection via Holistic, Multi-Channel Prediction. (2016), 1–10. arXiv:1606.09002 http://arxiv.org/abs/1606.09002

[136] Cong Yao, Xiang Bai, Baoguang Shi, and Wenyu Liu. 2014. Strokelets: A learned multi-scale representation for scene text recognition. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. https://doi.org/10.1109/CVPR.2014.515

[137] Qixiang Ye, Wen Gao, Weiqiang Wang, and Wei Zeng. 2003. A robust text detection algorithm in images and video frames. In *ICICS-PCM 2003 - Proceedings of the 2003 Joint Conference of the 4th International Conference on Information, Communications and Signal Processing and 4th Pacific-Rim Conference on Multimedia*. https://doi.org/10.1109/ICICS.2003.1292567

[138] Xu Cheng Yin, Wei Yi Pei, Jun Zhang, and Hong Wei Hao. 2015. Multi-Orientation Scene Text Detection with Adaptive Clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2015). https://doi.org/10.1109/TPAMI.2014.2388210

[139] Xu-Cheng Yin, Xuwang Yin, Kaizhu Huang, and Hong-Wei Hao. 2014. Robust Text Detection in Natural Scene Images. *{IEEE} Trans. Pattern Anal. Mach. Intell.* 36, 5 (2014), 970–983. https://doi.org/10.1109/TPAMI.2013.182 arXiv:1301.2628

[140] Xiaobing Wang Shuli Yang Wei Li Hua Wang Pei Fu Yingying Jiang, Xiangyu Zhu and Zhenbo Luo. 2017. R2CNN: Rotational Region CNN for Orientation Robust Scene Text Detection. (2017), 8.

[141] Xiang BAI Yingying ZHU, Cong YAO. 2016. Scene text detection and recognition: recent advances and future trends. *Frontiers of Computer Science* (2016), 19–36. https://doi.org/10.1007/s11704-015-4488-0

[142] Hosub Yoon Jaehong Kim Youngwoo Yoon, Kyu-Dae Ban. 2016. Automatic container code recognition from multiple views. *ETRI Journal* (2016), 767–775. https://doi.org/10.4218/etrij.16.0014.0069

[143] Liu Yuliang, Jin Lianwen, Zhang Shuaitao, and Zhang Sheng. 2017. Detecting Curve Text in the Wild: New Dataset and New Solution. (2017). arXiv:1712.02170 http://arxiv.org/abs/1712.02170

[144] Honggang Zhang, Kaili Zhao, Yi Zhe Song, and Jun Guo. 2013. Text extraction from natural scene image: A survey. *Neurocomputing* (2013). https://doi.org/10.1016/j.neucom.2013.05.037

[145] Sheng Zhang, Yuliang Liu, Lianwen Jin, and Canjie Luo. 2017. Feature Enhancement Network: A Refined Scene Text Detector. (2017), 2612–2619. arXiv:1711.04249 http://arxiv.org/abs/1711.04249

[146] Zheng Zhang, Chengquan Zhang, Wei Shen, Cong Yao, Wenyu Liu, and Xiang Bai. 2016. Multi-oriented Text Detection with Fully Convolutional Networks. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. https://doi.org/10.1109/CVPR.2016.451

[147] Hongqing Ma Zhiwei He, Jilin Liu and Peihong Li. 2005. A New Automatic Extraction Method of Contaienr Identity Codes. *IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS* (2005), 72–78.

[148] Yu Zhong, Kalle Karu, and Anil K. Jain. 1995. Locating text in complex color images. *Proceedings of the International Conference on Document Analysis and Recognition, ICDAR* 1 (1995), 146–149. https://doi.org/10.1109/ICDAR.1995.598963

[149] Xinyu Zhou, Cong Yao, He Wen, Yuzhi Wang, Shuchang Zhou, Weiran He, and Jiajun Liang. 2017. EAST: An efficient and accurate scene text detector. (2017), 5551–5560. arXiv:arXiv:1704.03155v2

[150] Zhiwei Zhou, Linlin Li, and Chew Lim Tan. 2010. Edge based binarization for video text images. In *Proceedings - International Conference on Pattern Recognition*. https://doi.org/10.1109/ICPR.2010.41

[151] Xiangyu Zhu, Yingying Jiang, Shuli Yang, Xiaobing Wang, Wei Li, Pei Fu, Hua Wang, and Zhenbo Luo. 2018. Deep Residual Text Detection Network for Scene Text. In *Proceedings of the International Conference on Document Analysis and Recognition, ICDAR*. https://doi.org/10.1109/ICDAR.2017.137