A system for reliable text recognition in an industrial environment

Evelyne Markey and Ashley Tews

Abstract— **In this paper, a vision system is proposed for high-reliability text label identification on industrial equipment, where complex background, changing conditions and illumination effects pose significant challenges. The system handles images with a low resolution that are taken from a wide field-of-view camera. Due to these challenges, the task of detecting text is divided in two parts. The first part consists of detecting text candidate regions in wide-area images using an edge-based approach. The second part examines these candidate regions in more detail using connected component analysis to identify those containing text. After selecting regions deemed to be highly likely of containing text, the individual characters are recognised using an open source OCR engine. The reliability of the system is improved by adding error correcting codes and by extracting the confidence of recognition of the different characters. Experiments using a code on a curved payload demonstrate the feasibility of using the system for high-reliability label identification.**

I. Introduction

Many industrial and warehousing sites place identification labels and marks on their equipment and products for tracking. Correct identification is important for tracking inventory and to reduce the possibility of delivery of payloads to incorrect locations resulting in loss of productivity.

Common methods of identification include barcodes and RFID tags. Barcodes are used by close proximity, typically hand-held readers. RFID tags offer a more automated solution that requires a tag on each product that can be read by a system onboard a transport vehicle or on site infrastructure. The readers need to be powered and have the tags pass close by in order to read them. Active tags can improve the range and have a reasonable lifespan. However, the data system the readers are connected to only contains information about when the payload passes the location of each reader. Continuous tracking is not possible unless the transport vehicle is carrying a reader and can localise itself, and therefore the payload. Also, to outfit a large industrial area (which could be in the order of kilometers of sheds and transport corridors) with readers and active tags may be infeasible or costly. Finally, the identification marks for some applications need to be human readable from a distance.

Motivated by these constraints, we propose an automated system for reading human-understandable identification marks. Such a system is intended for use in autonomous industrial vehicles, such as a Hot Metal Carrier [1]. In many materials transport-related industries, character-based common labeling methods are already in place on vehicles and mobile infrastructure. Our automated ID recognition system needs the ability to integrate with the existing methods as much as possible to improve the uptake of the technology.

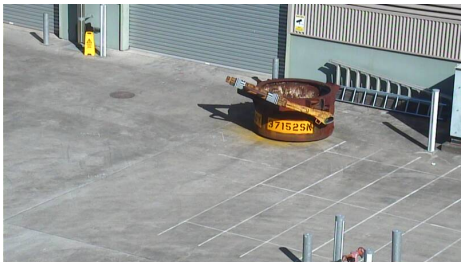


Fig. 1: An example of the spray painted black target text on a rectangular yellow background on the crucible.

Our method involves using a vision-based approach for common text recognition of such identification marks using webcams mounted around the site. These webcams may already be present for surveillance or area monitoring.

In large industrial areas, there may be a substantial distance from a camera's position to the object to be tracked. Consequently, the target text may be difficult to detect without zooming in. Therefore, our system consists of two parts. In the first part, different candidate regions that are likely to contain a textual code are detected. In the next part, the camera zooms in on these regions and increases the confidence if they actually contain the textual ID. Characters are segmented from the background and the result is recognised by an open-source Optical Character Recognition (OCR) engine. This engine is not strong enough to recognise every font (especially broken character fonts often used on containers), thus the engine is trained on a specific font to improve recognition. Confidence values on the recognition of characters are extracted and used to improve the reliability of the system. A check digit system capable of detecting and correcting common errors is also implemented. The system is tested on a crucible container which is used in aluminium smelters for molten metal transport by large forklifts (Figure 1).

The main contribution of our approach is primarily in the methodology and its application to a challenging real-world problem.

The remainder of this paper is organized as follows. In the next section, related work is discussed. The different parts of the system are discussed in more detail in Section III. In Section IV we describe the experiments conducted to evaluate the system and the conclusions are presented in Section V.

II. Related work

There is substantial research into text recognition in offices or outside built environments such as reading street

signs, shop signs and automatic license plate recognition. Generally, the task is approached by a text detection step followed by a text recognition step. Text detection is defined as locating text in an environment with recognition being the actual reading of that text into a chosen (usually human- or machine-readable) format.

Applications similar to ours that deal with recognising container fonts outdoors on infrastructure have to deal with various character sizes, fonts, spaces and alignments, as well as illumination and colour changes and noise due to rust, mud, peeling paint and fading colours. The systems are usually required to be fast and inexpensive as well. To deal with illumination changes and noise, some systems focus on efficiently preprocessing the image. [2] uses a morphological top-hat transformation to preprocess the image. A white top-hat transformation will enhance bright characters on a dark background and a black top-hat transformation will enhance dark characters on a bright background. [3] also uses morphological operations to preprocess the image. The morphological erosion operation eliminates small noises that could be caused by dirty or rusty surfaces. For our application, erosion cannot be used as a preprocessing technique, since the characters in distant images are too small and would be severely deformed or even eliminated.

[4] discusses a commercially employed system for vehicle and container number recognition (VECON) that uses histogram equalisation for normalising the image intensity and motion effect removal to preprocess the images. To detect container codes in an image, many of these systems perform Connected Component Analysis (CCA) and send features of character-like blobs in the image to a neural network ([4],[2]) or use them to train a classifier [5]. A large database of different images has to be available for training. CCA works well to detect text characters in images at a close distance. At a large distance though, the characters are very small and their connected components are hard to segment and therefore this technique is not applicable to these images.

Assuming that the target text is made up of short horizontal and vertical edges, an edge detection method is used in this paper. Apart from being able to be detected at a distance, edges also have the advantage of being quite robust under illumination changes since text often presents sharp contrast against its background. Edge detection is a well known technique in the field of text detection ([6],[7]) and applications in automatic license plate recognition ([8], [9])

Based on the above discussion, there has been substantial research on detecting and reading textual identification marks. However, these systems do not present a robust and effective method for detecting identification marks at a distance which is a requirement for our system.

III. System Description

In this section, the different components of the system are described. Figure 2 shows a flowchart of the main steps of the text recognition system: candidate selection on a distant image, text detection on a close-up image, text recognition and text evaluation, where it is decided if the text recognition results are valid.

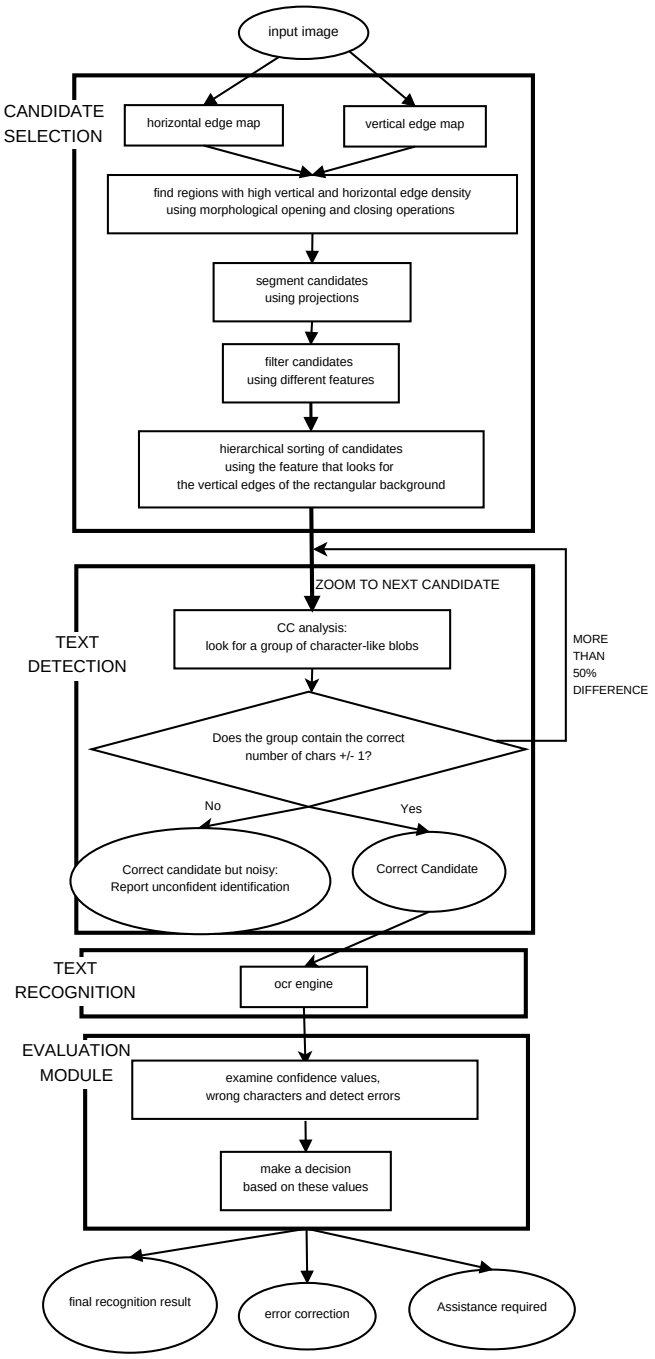


Fig. 2: Flowchart of the system.

In the candidate selection stage, regions that *may* contain target text are identified in the image. These regions are investigated in more detail in the second stage through zoomed-in images, using CCA.

If the candidate region is verified to contain the textual identification mark, the text is localised and the different characters are segmented from the background and subsequently binarised. The resultant binary image contains only textual information which is a prerequisite requirement for

recognition by an OCR engine, which occurs in the third stage. The OCR engine that is being used in this paper is the open-source software Tesseract [10]. Since the application requires high reliability, any errors that can occur in the result should be eliminated. Therefore extra check characters are added to the characters of the textual identification mark. In the final stage, confidence values relating to the recognition score of individual characters are determined and an evaluation of the check characters is performed. If any mistakes in the recognition are identified in this stage, the system attempts to correct the error or notifies the user of the type of error.

A. Candidate Selection

In the candidate selection stage, the input image is evaluated for regions that may contain text. By not constraining the locations of the object-of-interest that contains the text being sought, a more general approach is allowed. The candidate selection stage has to deal with the following requirements:

- the camera being used to detect the code has a low resolution (in the order of 800 $\times$ 450 pixels) with the system expected to undertake text recognition at moderate distances (up to 30 m).
- the method has to be robust to various natural and artificial lighting conditions and in environments where the text may be covered in dust.
- the method has to be able to recognise skewed characters on a round surface.
- the method has to be reliable as a false positive can be costly for the target application.

Regarding the first requirement, there is a trade-off between text size, camera resolution and range. If the characters have a height of 180 mm, this would result in a height of approximately 2 pixels at a distance of 30 meters.

This is a very small size to be able to do reliable detection. Therefore, in this stage, only candidate regions that *appear* text-like are detected.

In the case of textual identification marks on industrial objects, we have found robust results by spraying the characters in black on a yellow rectangular background as illustrated in Figure 1. This is similar to the colours of warning signs used on roads. The choice of font (FF Container) is also commonly found in industrial environments.

The selected colours ensure a better contrast for edge detection and the rectangle makes horizontal/vertical edge detection easier. It also has the effect of allowing greater distances for detection. Next, the candidate selection steps shown in Figure 2 are described in more detail and the system is illustrated using the example image in Figure 3.

Step 1: Edge detection and morphological operations
The target font's characters are made up of short horizontal and vertical edges that are connected to each other. By finding regions in the image with a high density of vertical and horizontal edges, plausible candidate text regions are found. To this end, vertical and horizontal edge maps are first computed using the Sobel vertical and horizontal masks [11] on the Canny edge version of the image. Long lines (these are lines that are longer than the width of the identification code) are deleted from these maps, so as to rule out long edges from objects such as buildings or gates.

To extract those regions in the maps that possess a higher edge density, a morphological closing operation is used. Morphological operations are used in image processing to understand the structure and shape of areas in an image, for example through contours, skeletons and convex hulls. These operations are executed by probing the image with an element with a pre-defined shape - a structural element. A closing operation will fill up small holes and cracks in foreground objects (i.e. smaller than the structural element) and slightly increase their contour. Vertical edges are closed in horizontal direction and horizontal edges are closed in vertical direction using a rectangular structural element. Regions that are covered by both the vertical and horizontal closing are thus considered to be regions with a high density of short horizontal and vertical edges [6]. As a final operation in this step, a morphological opening operation is performed on the image in order to isolate rectangular regions. Opening smooths out foreground objects and removes small areas from the object (i.e. areas smaller than the structural element) (Figure 3b).

Step 2: Segmenting through projection
At this stage, regions with background clutter could be considered as text regions. Since it is assumed that the region with the textual identification mark forms a reasonably continuous rectangular-shape region, this region can be isolated by segmenting at a location where a sudden discontinuity occurs. The derivatives of the horizontal and vertical projections of the different regions are computed. A change in this feature points to a sudden discontinuity in the region, thus the region is segmented at this point in the horizontal or vertical direction (Figure 3c). This is undertaken in a series of vertical and horizontal projections at different resolutions to remove noise, and to segment blobs into uniform rectangular regions.

Step 3: Filtering and hierarchical structuring using morphology and geometric features
In this step, the remaining candidate regions from the previous steps are organised in order of the most likely to contain the target text, determined by evaluating the features in each.

First the remaining regions are filtered on size, aspect ratio and solidity, and candidates that are too close to the borders are filtered out.

Next, five features are calculated with thresholds between 0 and 1 to evaluate and rank the regions in order of their likelihood of containing target text.

- **Feature 1 - region solidity.** The feature evaluates whether candidate regions contain a rectangular background similar to the identification code. To achieve this, it uses the modified tophat operation, as discussed in [12]. This operation filters out small details like the text on the background, but retains the rectangular background. For the feature, the edges of the modified tophat image were detected. Next the bounding box

(a) (b) (c)

Fig. 3: Illustration of the system. (a) original image with the top 5 resulting text candidates displayed. Note the location of the target text on the payload is skewed and in shadow. (b) overlapping areas with high vertical and horizontal edge density, (c) after segmenting the possible text regions from background clutter.

of the candidate region is shrunk and expanded. The area outside the shrunken and inside the expanded bounding box should contain the edge of the rectangular background. If the number of pixels in this area is similar to the contour length of the candidate region, this feature has a value close to 1. In practice, this feature was not overly useful, and was given a low threshold of acceptance (greater than 0.1).

- **Feature 2 - region rectangularity.** This next feature evaluates how rectangular the background of the region is. Vertical and horizontal lines are analysed in the Canny edge image. Lines are found by applying a Sobel operator over the region. Next, lines are filtered to obtain strong, strictly vertical lines with a looser requirement of straightness for the horizontal lines considering they occur on a curved surface of the target object. If the distance between likewise oriented pairs of these lines is similar to the width/height of the candidate region, this features receive a value close to 1. A threshold of 0.7 is used on this feature.
- **Feature 3 - Black Top Hat feature.** This feature uses the black tophat morphological operation. The black tophat image is the difference between the closed image and the original image. Since the closing operation fills up small holes and cracks in foreground objects (i.e. objects lighter than the background), this operation will enhance small dark elements on a lighter background. The edge map of the black tophat image is calculated, and the edges are closed. Then the candidate regions are compared to this image. If the pixels of the closed image fill up a certain candidate, then this candidate gets a value close to 1. A threshold of 0.45 is set to filter out candidates that do not meet these requirements.
- **Feature 4 - Regions in shadow.** If a shadow falls over the target area, Otsu binarisation [13] of a modified tophat image will not display the the rectangular background, only part of it. By applying an adaptive binarisation of the modified tophat image, the shadows are effectively removed, at the cost of introducing noise. The target area will appear white in the adaptive image and the candidate regions are compared with this image. A threshold of 0.6 is used on this feature.
- **Feature 5 - Vertical projections.** The final feature counts the number of separate strictly vertical blobs that are present in the candidate area. It does this for the normal edge image and for the edge image of the black tophat image. Low thresholds were set in order not to filter out candidates that are in shadow or very far off candidates where it is harder to locate these vertical edges.

To compensate for potentially large variations in character height in the images and also for the large variation in lighting conditions (edges in shadow regions are harder to detect) the method is executed six times with varying parameters for the Canny operation and the rectangular structural elements. The resulting accumulated non-overlapping regions are ranked according to how closely they satisfy the feature evaluations above. In the example image (Figure 3a), the top five candidates are shown. In the next stage the camera zooms in on each of those regions in order of their ranking.

B. Text Detection

The previous Candidate Selection stage provides an estimate of possible locations of the target text. In the Text Detection stage, a more accurate location is found and the characters are segmented from the background.

1) Text Localisation: From the ranked candidate regions, each region is zoomed in according to its size, for further evaluation (e.g. Figure 4). A black tophat morphological operation is applied to enhance the characters even if a shadow pattern is distorting them. The black tophat image is binarised using Otsu binarisation and a Connected Component Analysis (CCA) is then performed. The connected components are filtered on ratio and size.

Since the target font consists of broken characters, the characters will be represented by separate blobs. In the next step, blobs that might belong to the same character are grouped together.

Several features are used to group them:

Fig. 4: Example of a zoomed in region for the Text Localisation and identification stages.

- *stroke width*: The stroke width of every blob in the image is calculated and blobs from the same character are assumed to have a similar stroke width [14]. The stroke width of a blob is calculated by first calculating the Sobel gradients in order to determine the gradient of every edge point of the blob. The next encountered pixel in the direction of the gradient that has a gradient roughly opposite is a good candidate to measure the stroke width. The stroke width is the length of the line between the two pixels. Every pixel on the line is given this value. In the next step, every pixel of a line is given the average value of all the pixel values along the line. The average of all the pixel values from the blob is the final stroke width. A counter keeps track of how many good candidates measuring the stroke width in the blob were found. A low count means the shape of the blob is irregular and the value of the stroke width is unreliable.
- *distance:* Neighbouring blobs from the same character will be closer to each other than to blobs of neighbouring characters. For every pair of blobs, the distance between the two closest points is calculated.
  For the target font, this distance has to be smaller than 0.7 times the biggest stroke width of the characters. The number of neighbouring pixels that meet these distance requirements are counted. This amount has to be smaller than 0.6 times the largest stroke width.

Once the different characters are formed, they are grouped to form the identification code. Several features are used to group characters together:

- *distance:* The distance between two neighbouring characters in a group has to be smaller than half the smallest character height.
- *similar height:* Neighbouring characters from the same group must have a relatively similar height.
- *ratio:* Characters have a certain height/width ratio.
- *similar baseline:* The baseline of a character is defined as the lowest horizontal line of their bounding box. Neighbouring characters of the same group have to have baselines at a similar height.

After the groups are formed, the number of characters in each is analysed. If the number is +-1 of the required number of characters, it is accepted as a correct candidate. The characters in the group are binarised (Figure 5) and the image is sent to the OCR-engine for text recognition.

Fig. 5: Result after zooming in, localising and segmenting.

Fig. 6: 'B' could easily be mistaken for 'I3', 'N' could be mistaken for 'IV' and 'W' could be mistaken for 'VV'.

If a group has more than half, or less than 1.5 times the required amount of characters, it is considered a correct, but noisy candidate.

C. Text Recognition: The OCR Engine Tesseract

Apart from being capable of delivering high accuracy for certain text recognition tasks, Tesseract has the following advantages:

- It is possible to train Tesseract for different languages, fonts or even for a selected set of characters. This is particularly useful for the present application, since a limited alphabet in a specific font is used. Training Tesseract for this particular *language* can significantly improve the results.
- Because Tesseract is open source, it is possible to extract extra information out of the engine such as confidence values for individual characters. These are used in the Text Evaluation stage (III-D).

Although Tesseract can recognise broken characters, there can be substantial confusion in the recognition, since some characters could be easily interpreted as others (see the example in Figure 6). To reduce these types of errors, Tesseract can be trained to recognise only certain characters in a particular font. The following section discusses how this is undertaken.

1) Training Tesseract: For the current application, Tesseract is trained to recognise characters from a specific broken font (FF Container) and a specific character set. Only digits and capital letters are used. The letters 'O' and 'I' are omitted to reduce ambiguity and an extra letter 'F' is omitted because only 23 letters are required (Section III-D.2). The choice of font and character set is termed the *language*.

The training dataset consists of words that are a random combination of 1 to 6 letters and digits. The words are randomly placed in 10 image files and range from a height of 60 pixels to a height of 110 pixels. In the dataset, every letter and every digit appears at least 20 times.

A big challenge in the system is the recognition of skewed and rotated characters. The code is sprayed on the curved

surface of the crucible (Figure 1) and can be viewed at various angles. Therefore, the amount of skew variation that Tesseract can manage was tested by training on orthogonally viewed dataset without rotated or skewed characters. The trained system was tested on skewed images and showed that Tesseract can recognise skewed and rotated characters up to 15 degrees. Supplementary training on skewed and rotated datasets improves this performance to 18 degrees.

D. Text Evaluation

The last stage of the system is Text Evaluation, as illustrated in Figure 2. In this stage, the confidence of the recognition is evaluated through confidence values and check digits. If it finds a high possibility of an error, the system will attempt to correct it or annotate what the error is.

1) Confidence of the recognition: Tesseract returns a recognition result that contains characters which only have a resemblance with the training data. For a reliable system, the characters whose recognition does not return a high confidence value either need to be removed or corrected depending on whether they are noise or text.

2) Check digits: An error detection and correction system is implemented to further improve the reliability. The system has to provide reliable detection and correction without adding too many additional symbols to the target code. A Hamming check character system [15] was chosen which adds characters to the code. These characters are included in the language Tesseract has been trained on.

Error Detection For error detection, an additional character b is added to the five digit code. It is obtained by solving:

$$(b + w_1 d_1 + w_2 d_2 + ... + w_m d_m) \bmod n = 0 \qquad (1)$$

where d_i is the identification code and w_i are weights. n has to be a prime number and the weights cannot be factors of n. This ensures that all single-digit errors (i.e. one digit in the code is changed) are detected. An error is undetected when the result is a multiple of n. Since n is a prime number and the weights are not a factor of n, this can never be the case for a single-digit error. Under these circumstances, all single-digit errors are thus recognized.

The check character is chosen to be a letter instead of a digit to improve the error detection[1] and to distinguish the check character from the code. This means that for our application n = 23 (the largest prime number smaller or equal to 26) and m = 5 since the code consists of 5 digits.

The letters are mapped to the calculated check digit using a simple mapping where A=1, B=2, C=3,... The weights are chosen to be: $(w_{1..5}) = (18..22)$.

Apart from detecting single-digit errors, exhaustive testing shows the approach also detects 95.7% of double, triple, quadruple and quintuple errors.

Error Correction By adding a second check character a, detected errors may be corrected. Since we have established w_i using Eq 1, the resulting two check characters a and b, can be obtained by solving:

$$(a + b + d_1 + d_2 + d_3 + d_4 + d_5) \bmod 23 = 0 \qquad (2)$$

$$(b + 18d_1 + 19d_2 + 20d_3 + 21d_4 + 22d_5) \bmod 23 = 0 \qquad (3)$$

The first check digit a gives the magnitude of the error and the second digit b returns the location of the error. An error means that digit d_i changes to the erroneous digit b_i. If this is substituted in (2):

$$\begin{aligned}(a + b + d_1 + ... + d_i + ... + d_5) \bmod 23 \\ = ((a + b + d_1 + ... + d_i + ... + d_5) + \\ (b_i - d_i)) \bmod 23 \\ = 0 + (b_i - d_i) \bmod 23 \\ = b_i - d_i \\ = \Delta \end{aligned} \qquad (4)$$

This gives the magnitude Δ of the error. If the digit change is substituted in (3)

$$\begin{aligned}(b + 18d_1 + ... + (i + 17)d_i + ... + 22d_5) \bmod 23 \\ = (b + 18d_1 + ... + (i + 17)d_i + ... + 22d_5 \\ +(i + 17)(b_i - d_i)) \bmod 23 \\ = 0 + (i + 17)\Delta \bmod 23 \end{aligned} \qquad (5)$$

This gives the location i of the error.

In the system, a and b are calculated using equation (2) and (3) and added to the 5-digit code. When the code is recognised erroneously, a and b can be used to calculate the magnitude of the error Δ and $(i+17)\Delta$ mod 23. Beforehand, a table was generated that contains the value of $(i + 1)$ for every value of Δ and $(i + 17)\Delta$ mod 23. This table is used to find the position i of the error after calculating Δ and $(i + 17)\Delta$ mod 23. When the position and the magnitude of the error are known, the code is corrected by changing b_i to $(b_i - \Delta)$ mod 23.

This is illustrated with an example. The code 47365 receives the characters P (=16) and E (=5) as check characters. Equation 3 is executed with $d_{1..m} = 47365$ to receive the check character *P* and Equation 2 is executed with $d_{1..m} = 47365$ and $b = 16$ to receive the check character *E*. Next an error is introduced in the code, for example *4* is recognised as *9*. In Equation 2, the result of the modulus is not longer 0 but 5. This is the magnitude Δ of the error. Equation 3 returns 21. This has to be equal to $(i + 17)\Delta$ mod 23 as seen in Equation 5. Using the look-up table, the result for the position i is 1 which is where the error occurred. If two digits are misrecognised, for example $47365PE$ changes to $97385PE$, the error is detected (Δ = 7), but the position will not be determined due to the look-up table only having entries for values of the position from 0 to 6.

3) Evaluating the result: The most obvious errors are handled first such as a wrong number of recognised char-

[1]There are 26 letters in the English alphabet and only 10 digits, which improves the detection of double-digit errors by 5.7%. This value is obtained by a comparison of the detection performance of all possible double-digit errors.

acters in the result. Usually this indicates that noise was recognised as a character. Noise-induced characters typically have low confidence values (i.e. negative). Any character with a confidence below a threshold (-6), is considered as noise and deleted. If on the other hand there are too many characters and none have a confidence less than -4.5, the system flags an unconfident result and notifies the user because it is unclear where the error is located.

A second obvious error is if there are letters detected instead of numbers or numbers at the position of the check letters. Since the check letters are crucial for the reliability of the system, any possible error with them is flagged as an unsuccessful identification. This is also true when the check characters have a low confidence (less than -4.5). While it is possible to correct a letter in a number position, for now an error flag is set and the position of the error is noted.

In a subsequent step, error detection is performed with its results, combined with the confidence values and the error flags, used to decide whether the recognition result should be accepted, corrected or defined as unconfident. If unconfident, an error report is returned.

IV. Experiments

A. Experimental Setup

A series of preliminary evaluations of system performance were conducted using the black text code on a yellow background on the target object (e.g. Figure 1). Initially, the effect of distance and skew were evaluated with a low resolution camera (768 $\times$ 576 pixels) over a range of distances and horizontal viewing angles. Subsequent testing was conducted on different codes, in different environment conditions and with distortions to the code to determine its ability to manage imperfections.

The prototype software was implemented in Python and run on a standard PC connected to an internet webcam located outside a building at our industrial site. The time for candidate detection depends on the number of candidate regions and scene complexity, however, for most of our testing, it was typically 25 seconds. The time for text identification depends on the rank of the target text region after the candidate selection phase although if it was the first ranked, it would take approximately 20 seconds for identification.

B. Distance and Skew Testing

The system was tested over different angles and distances. Confidence values are extracted for every digit and the angle and distance at which Tesseract starts returning incorrect recognition results is represented. The experiments were conducted on just one set of numbers and thus do not represent confidence values or recognition results for specific digits. That would require conducting experiments with every possible digit at every one of the 5 locations at least once and which would require the crucible to be painted again for every set of numbers. This would be infeasible. Figures 7 and 8 represent the confidence values of the numeral code *47365*. For Figure 7, images were taken at distances varying from 1 to 10 metres, with intervals of 1 metre. For Figure 8, images were taken at 3 metres distance, at rotations varying from 0 to 50 degrees, with intervals of about 3.5 degrees and with 0 degrees defined as an image taken with the camera perpendicular in front of the center of the numeral code. Values close to zero are the most confident. The first appearance of an incorrect recognition result as returned by Tesseract is represented by a vertical black line.

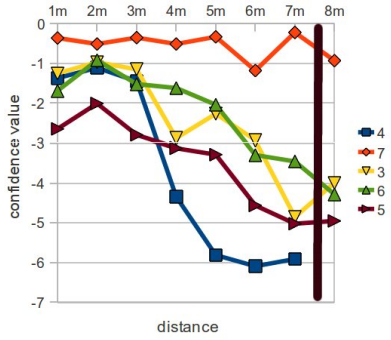


Fig. 7: Confidence values for '47365' at varying distance.

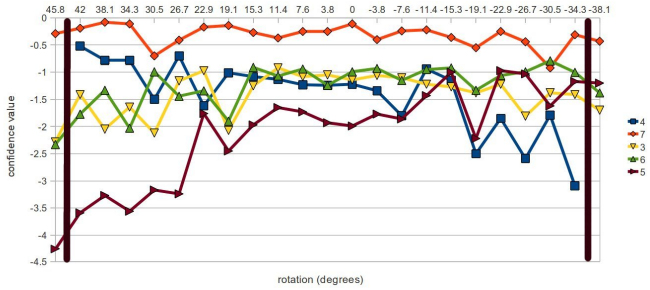


Fig. 8: Confidence values for '47365', at varying rotation.

Fig. 9: Image at 8 metres distance at which the recognition fails.

The confidence values clearly deteriorate with growing distance and growing angle. The recognition starts failing at 8 metres (Figure 9) and at an angle of about 40 degrees (Figures 10 and 11). Also recognition of characters at the edges are the least confident (at further distance or larger

Fig. 10: Image at -38 degrees at which the recognition fail.

Fig. 11: Image at 42 degrees at which the recognition fails.

rotation) and if recognition fails, these are usually the first characters that are misrecognised. This is expected since the characters are sprayed on a curved surface. The result indicates that it in order to obtain a more confident recognition the check digits could be placed in the middle of the code instead of at the end.

Since recognition of the textual identification mark starts failing at 8 metres, it is clear that a method that zooms in on the text area was necessary, or a higher resolution camera required.

C. Reliability Testing

A limited set of experiments were conducted to determine the system's performance over a variety of codes, environment conditions, object locations and lighting conditions (Figure 12). The goal was to determine the reliability of the output of the system and how well it manages variations in operating conditions. Since the target application is code detection on the side of a curved payload, it is not feasible to test a large number of different codes. Three codes (85016DY, 37152SN, 69421LQ) were trialed with plans for further testing. The accumulated results of testing are presented in Table I according to the conditions. Note that at the time of testing and the location of the target object, full sunlight was never on the text and its background region.

The columns in Table I are the number of trials (Trials), the number of trials that underwent some form of error correction (Corrections), and the number of trials where the system flagged it was not a reliable output (Flagged err). For the *in-shadow* trials, the flagged error was due to the sun's reflection on the camera dome as shown in Figure 13. For the *cross shadow* trials, it was due to noise introduced by the strong shadows that fell across the text. For the *night* tests, the zoomed in image was very grainy since the camera was in greyscale night mode which made the image noisy. In all tests, the text was reliably detected by the system. The rank order of appearance of the text region in the output of the Candidate Selection stage was typically: in the top 3 for sunny and rainy conditions; top 5 for overcast; and top 7 for cross shadow conditions.

	Trials	Corrections	Flagged err.
in-shadow	16	5	5
cross shadow	14	1	5
evening	6	0	2
night	2	0	2
rain	2	0	1

TABLE I: Results of the text detection stage.

Fig. 13: Example image where the system had difficulty in correctly identifying the text due to the sun's reflection on the camera dome.

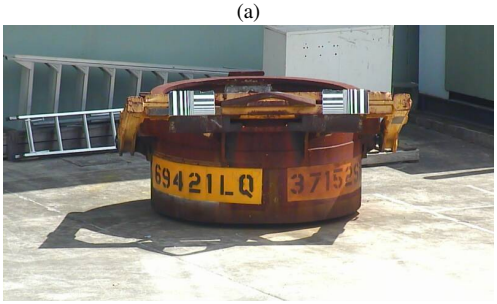
The next set of tests were conducted to determine how the system handled degradation of the text in terms of incomplete characters and a rusting or dusty surface. This is representative of the potential problems to be encountered in the target application. For the character degradation tests, parts of various characters were removed, such as one of the arms of the 'Y'. In these cases, the error detection and correction mechanism should either repair the text or annotate the error to the end user. Tests were conducted with the codes in sunny, shadowy, overcast and rainy conditions. A total of 10 tests were conducted with only one successful correction on a damaged '8' and the rest reported unreliable results due to incorrect characters being identified. There were no false positives which is a fundamental requirement of the system.

The rusting surface tests consisted of spraying rust-coloured paint over the text region. The colour matched the rusty colour of the crucible and was applied in light, medium

(a) (b) (c) (d)

Fig. 12: Examples of images of the various testing conditions. From left to right - in-shadow, cross shadow, night and rainy.

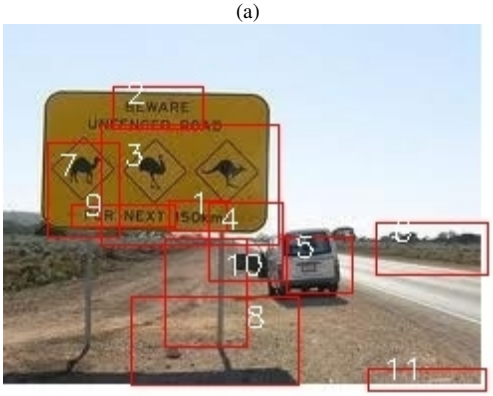
(a)



(b)

Fig. 14: Moderate (a) and heavy (b) simulated rust on the crucible

(a)



(b)

Fig. 15: Example results from applying the text detection stage to random static images. The red boxes show the candidate regions with their ranking annotated in white. Although non-text regions are candidates, these would be discarded in downstream processing if the full system was applied to them.

and heavy intensity (examples shown in Figure 14).

Seven tests were undertaken in a variety of environment conditions for light and medium rust in overcast and sunny conditions. The text was detected on each occassion with three trials flagging unreliable results. These were all in partly shadowy conditions. For the heavy rust simulation, the text region could not be detected under any lighting conditions which demonstrates an expected limit of the system.

We have also applied the text detection stage to pictures taken from the internet to determine how the system performs on other industrial text. The other parts of the system cannot be tested as it requires a camera to zoom in on candidate regions for further analysis. This highlights the dependance between system components to produce a reliable system since the candidate regions may not contain target text and will be discarded in the text detection stage. Two examples are shown in Figure 15.

V. Conclusions

We have developed a preliminary image-based system for identifying textual labels on the curved surface of infrastructure. This application is typical in an industrial environment for identifying and tracking product locations. A standard container font was used and a two part strategy developed for identifying text. The first part involves detecting the text in wide-angle images where text regions are typically small. Candidate regions are then zoomed in to refine the detection and undertake text recognition. Recognition is undertaken with a well-used OCR program and we have modified the open-source code to extract confidence values relating to the identification 'score' of text characters. The system was demonstrated in a typical real-world example on a large molten metal transport container which has curved surfaces. The results indicate the range and angular limits of the approach, but demonstrate highly reliable performance in text recognition and knowing when the results are unreliable.

Experiments conducted with a small range of identification codes, under varying lighting conditions, and with deformations to the text region have shown the system to have robust text detection and show a high degree of reporting only true positives. This is paramount for applications where a wrongful identification can lead to incorrect payload tracking and delivery.

There are many avenues for future work including improving the confidence calculation of the candidate regions by extracting better text features, and finding a more robust method of selecting the thresholds. The system will also be exercised over more variations in codes, environments and environment conditions.

VI. ACKNOWLEDGMENTS

This project was funded by CSIRO's Light Metals Flagship.

REFERENCES

[1] A. Tews, C. Pradalier, and J. Roberts. Autonomous hot metal carrier. In *Robotics and automation, IEEE International Conference on*, pages 1176–1182, Roma, Italy, April 2007.

[2] I.S. Igual, G.A. Garcia, and A.P. Jimenez. Preprocessing and recognition of characters in container codes. In *16th International Conference on Pattern Recognition, ICPR*, volume 3, Quebec City, QC, Canada, August 2002.

[3] Z. He, J. Liu, H. Ma, and P. Li. A new localization method for container auto-recognition system. In *Neural Networks and Signal Processing. Proceedings of the 2003 International Conference on*, volume 2, pages 1170–1172, December 2003.

[4] J.C.M. Lee. Automatic character recognition for moving and stationary vehicles and containers in real-life images. In *Neural Networks, IJCNN. International Joint Conference on*, volume 4, pages 2824–2828, July 1999.

[5] L. Zini, A. Destrero, and F. Odone. A classification architecture based on connected components for text detection in unconstrained environments. In *Proceedings of the 2009 Sixth IEEE International Conference on Advanced Video and Signal Based Surveillance, AVSS*, Washington, DC, USA, 2009.

[6] D. Chen, J-M. Odobez, and H. Bourlard. Text detection and recognition in images and video frames. *Pattern Recognition*, 37:595–608, 2004.

[7] C. Liu, C. Wang, and R. Dai. Text detection in images based on unsupervised classification of edge-based features. In *Proceedings of the Eight International Conference on Document Analysis and Recognition*, 2005.

[8] S. Saha, S. Basu, M. Nasipuri, and D. K. Basu. License plate localization from vehicle images: An edge based multi-stage approach. *International Journal of Recent Trends in Engineering (IJRTE)*, 1:284–288, 2009.

[9] B. Hongliang and L. Changping. A hybrid license plate extraction method based on edge statistics and morphology. In *Proceedings of the Pattern Recognition, 17th International Conference on (ICPR)*, volume 2, pages 831–834, Washington, DC, USA, 2004.

[10] R. Smith. An overview of the tesseract ocr engine. In *Proceedings of the Ninth International Conference on Document Analysis and Recognition, ICDAR*, volume 2, pages 629–633, Washington, DC, USA, 2007. IEEE Computer Society.

[11] R. Gonzalez and R. Woods. *Digital Image Processing*. Prentice Hall, 2nd edition, 2002.

[12] N. Ezaki, M. Bulacu, and L. Schomaker. Text detection from natural scene images: Towards a system for visually impaired persons. In *Proceedings of 17th International Conference on Pattern Recognition, ICPR*, volume 2, pages 683–686, Cambridge, UK, August 2004. IEEE Computer Society.

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

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

[15] S. Barnett. *Some Modern Applications of Mathematics*. Ellis Horwood, Hemel Hempstead, 1995.