

"A container code verification apparatus"

The present invention relates to a container code verification apparatus and more specifically to an apparatus for verifying an identification code on a cargo container against a target code.

Cargo containers each have a unique identification code (ID code) which is painted or applied by other means onto the sides of the container surface. The ID code must be read and verified against a computer record whenever a container is towed in or out of a container yard in a port. This ensures that the correct truck is towing the correct container, or containers, through a gate of a cargo terminal. Currently, the code is read and verified manually as the truck passes through the gate. The truck must be stopped at the gate whilst each ID code on each container being towed by the truck is checked. Human inspection of the ID code is slow and prone to oversights and errors. Each gate must be manned by at least one operator thus necessitating significant man power.

One object of the present invention is to provide a container number recognition apparatus which automates the present verification process.

Another object of the present invention is to provide a container number recognition apparatus which improves the efficiency of the verification process and to require the presence of only one gate operator to man multiple gates at the port.

Accordingly one aspect of the present invention provides a container code verification apparatus comprising: image capture means (1, 2, 3) to capture an image of a container (8) surface carrying a displayed code (16); data processing means (5) to form a pixel array from the captured image; scanning means to scan the pixel array; detection means to detect potential characters; selection and grouping means to select and group together potential characters of the displayed codes; and a neural network to determine from the pixel values of the potential characters a set of recognised characters constituting a recognised code, the pixel array being divided into a plurality of windows, the neural network comprising a plurality of input nodes defining an input layer and at least one output node defining an output layer, one or more intermediate layers being disposed between the input layer and the output layer, in which neural network the input layer is divided into a plurality of discrete areas each corresponding to a respective window; the pixel value of each pixel in a window represents the input for a corresponding input node in an area of the input layer corresponding to the respective window; each node within the network computes an output in accordance with a predetermined function; the outputs of the nodes in each area of the input layer are connected to specified nodes in a first intermediate layer which are not connected to the outputs of the nodes in another area of the input layer; the outputs of the nodes in the first and subsequent intermediate layers being connected to the inputs of the nodes in the immediately following layer; and the output nodes of the last intermediate layer are connected to the inputs of the output node or nodes of the output layer, the container code verification apparatus further comprising: comparison means to compare the target code with the recognised code; and verification means to verify or reject the recognised

code in dependence upon the results of the comparison between the target code and the recognised code.

In order that the present invention may be more readily understood, an embodiment thereof is now described, by way of example, with reference to the accompanying drawings in which :

Figures 1a and 1b are examples of container identification codes which have been highlighted, the ID code in Figure 1a in a light colour on a dark background and the ID code of Figure 1b in a dark colour on a light background;

Figure 2 is a diagrammatic representation of one embodiment of the present invention;

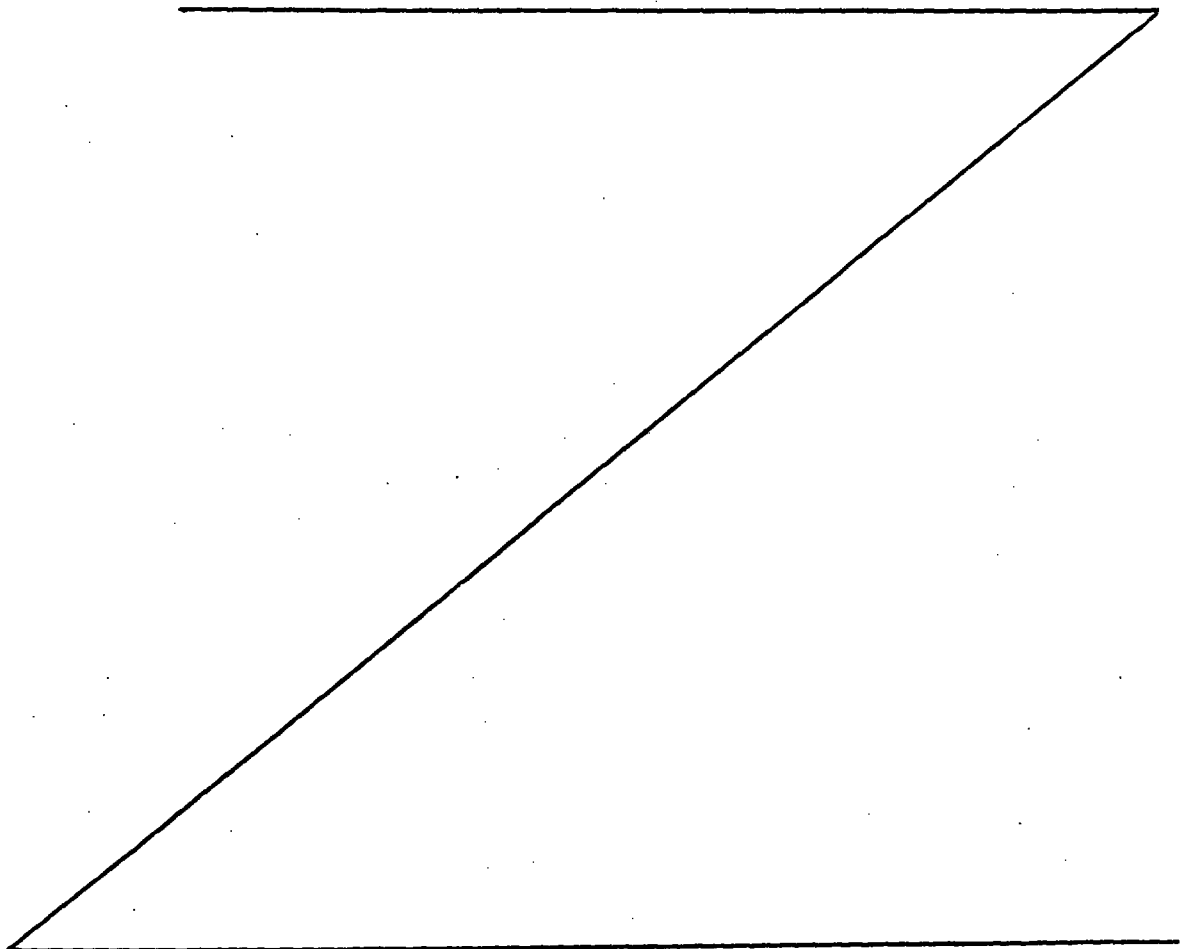


Figure 3 illustrates a truck in position at a gate installed with an embodiment of the present invention;

Figure 4 illustrates vertical and horizontal segments in a pixel array in accordance with one embodiment of the present invention;

Figure 5 illustrates a right boundary vertical segment in accordance with one embodiment of the present invention;

Figure 6 is a schematic representation of a multi-layer neural network structure as incorporated in one embodiment of the present invention;

Figure 7(i) illustrates a window-based neural network architecture for use with one embodiment of the present invention;

Figure 7(ii) illustrates a conventional neural network architecture;

Figure 8 illustrates an example of a matching problem;

Figure 9 illustrates a matching path found for the example of Figure 8; and

Figure 10 illustrates the local continuity restraints applicable to an embodiment of the present invention.

A cargo container, such as a standard ISO container, comprises a rigid metal box container. Each container is identified by a unique ID code comprising a string of alphanumeric characters which is usually painted onto at least one surface of the container. Other information is also applied to the container surface, such as gross weight, net weight, country of origin and the like. Thus, the ID code is located amongst other characters containing the other information. The containers themselves, due to the environment in which they are used, often become marked, dirty or dented. The

presence of corrugations, structural bars, smears and other noise may distort the characters. Thus, the variation in character and background intensity and the inherent lack of adequate contrast pose problems for a reliable method and apparatus for ID code character extraction, recognition and verification. The intensity and contrast of the characters and the background varies with respect to the illumination of the container surface in different conditions, such as during daylight, night time and cloud cover. Also, the characters of the ID code may be presented to the recognition apparatus at an angle, thus resulting in characters which are skewed as shown in Figures 1a and 1b which also illustrate other characters in addition to the ID code which are present on each container.

Referring to Figure 2, the apparatus embodying the invention comprises a plurality of cameras 1, 2 and 3, three in this case, a multiplexer 4 which transmits the signals from the cameras 1, 2 and 3 to a transputer network 5 via a BNC connection 6. The flow of data from the transputer network 5 is connected to and controlled by a host computer 7 via an AT bus 8. The host computer 7 may be, for example a PC-AT 386 microcomputer which is in turn connected by an RS-232 serial link 9 to a gate computer 10 which stores container information obtained from a mainframe computer (not shown).

When a truck loaded with a container approaches a gate and stops at the gate (as shown in Figure 3), the gate operator initiates the verification process on the host PC 7. Three closed circuit TV cameras 1, 2 and 3 are used to capture images of the container 8. Each image comprises an array of 720 x 512 pixels, each pixel being capable of 256 grey levels. The images are sent via the multiplexer 4 to the transputer network 5. The transputer

network 5 comprises a monochrome frame grabber 11 (Figure 20) which acts as a master transputer and controls the flow of execution commands to a number of worker transputers 12,13 AND 14 (in this case three), and a root transputer 15 which is connected to the host PC 7 and allows communication between the host PC 7 and the other transputers. The host PC 7 is responsible for file saving, input and output via a keyboard and VDU and for communication with the gate mainframe computer which stores a record of container information obtained from a mainframe computer which can be interrogated by the host PC 7. The host PC 7 also displays the results of the verification to the operator. When a truck with a container 8, or containers, arrives at the gate, the gate mainframe computer sends information about the container 8 expected at the gate, including which container ID code 16 is expected, to the gate PC 10.

The appropriate camera 1,2 and 3 is chosen to focus on the rear of the container 8 and the frame grabber contrast setting is adjusted to obtain a good level of contrast between the characters on the container 8 and the background. A segmentation process then locates and extracts a bounding box for each ID character of the ID code 16 from the container image. The extracted bounding box is normalised to a standard size and a character pixel map of this information is passed to a network character recogniser which calculates the alphanumeric character most likely to be represented by the character pixel map. A proposed ID character code comprising the recognised characters derived from the image information extracted from the container 8 is matched with the expected ID code provided by the gate computer 10 and the two codes are compared. The comparison of the two ID codes produces a confidence measure which indicates the degree to which the two ID codes match. The operator can then decide to hold

back or allow the truck to pass through depending on the confidence measure determined by the matching comparison between the two ID codes. If the confidence measure is over a pre-set threshold then the truck may continue, if not, then the truck is held at the gate.

The container number recognition system can handle the entire range of container sizes. This includes containers of length 20 ft, 40 ft and 45 ft (6.1m, 12.2m and 13.7m respectively) as well as heights varying from 8 ft to 9.5 ft (2.4m and 2.9m respectively). Also, the length of the chassis on which the container, or containers, are placed can be 20 ft, 40 ft or 45 ft (6.1m, 12.2m and 13.7m respectively). Therefore, the number and location of the cameras is such that the container images captured are of adequate quality to enable the ID code characters to be resolved.

The operation of the apparatus is now discussed in more detail:

During daytime operation the natural level of illumination of containers 8 at the gate is normally sufficient and as such no external lighting is necessary. However, in operation during the evening and the night, fluorescent lights 17, 18 are used to illuminate the containers 8. As shown in Figure 3, a rear light 17 is located behind the container 8 facing the gate house 19 to improve the contrast of the image of the container 8, against the background. A frontal illumination 18 source is located facing the container 8 to improve the illumination of features on the surface of the container such as the ID code characters 16. If the characters which are to be extracted from the surface of the container are to be resolvable from the background of the container

surface then it has been found that there must be an intensity contrast between the background and the subject of at least 15 - 20% of the overall image intensity. The closed circuit TV cameras 1,2 and 3 have an auto-iris capability to compensate for small variations in illumination.

Upon receipt of a target ID code from the gate computer 10 the appropriate camera 1,2 or 3 is chosen to focus on the rear of the container 8. The selected camera 1,2 or 3 triggers the frame grabber 11 which applies an algorithm to adjust the contrast setting of the frame grabber 11 so that the mean grey level for a predetermined image window 20 defined by the camera's field of view is at an optimal value. The algorithm uses Newton's method to compute the new value of the contrast setting according to equation (i):

$$i) \text{ new contrast} = \text{old contrast} + \text{slope} (\text{mean} - \text{optimum mean})$$

$$\text{slope} = \Delta \text{ contrast} / \Delta \text{ mean at last iteration}$$

The delta operator (Δ) indicates the change in contrast or mean grey level respectively between the last iteration and the current one. The optimal mean value and the slope have been experimentally determined.

Having determined the contrast setting, the camera 1,2 or 3 captures an image which is sent via the multiplexer 4 to the frame grabber 11 where the image is processed further. The image consists of an array of 720 x 512 pixels, each pixel being capable of a quantisation level of between 0 and 255.

To determine the location of all potential characters within the image and to establish a surrounding

rectangle or bounding box around each potential character, the captured image is scanned by the frame grabber one column at a time. The grey level of each pixel in each column is quantised and horizontal and vertical segments are created based on the quantised value. A horizontal segment (see Figure 4) is defined as a part of a row of the image in a column for which the adjacent pixels (in the horizontal direction) have the same quantised value. Similarly, a vertical segment (see Figure 5) is a part of a column for which adjacent pixels (in the vertical direction) have the same quantised value. Therefore, the image has now been divided into vertical and horizontal segments consisting of a number of pixels which are associated by their respective quantised values and the image can now be scanned segment by segment thus saving both time and memory space.

The vertical segments are scanned and all segments which are on the right hand boundary of a character are detected. These are defined as right boundary vertical segments (see Figure 4) and each of these segments must satisfy two conditions : there must a corresponding vertical segment (V2) to the left of the vertical segment (V1) of which at least a part must be adjacent to the first segment (V1) and the quantised value of the corresponding vertical segment (V2) must be different from that of the right boundary vertical segment (V1).

The threshold or extent of each potential character is based on the pixel grey level at the boundary defined by the boundary segments. If the background colour of the image is white, then the threshold is selected as the lowest grey level. If the background colour of the image is black, then the threshold is selected as the highest grey level. The threshold is used to determine if the

potential character is connected to any other components which may belong to the same character. An example of connected horizontal segments is shown in Figure 4 which shows three connected horizontal components h1 to h3. By limiting the character size to a predetermined level, the number of connected components will be reduced. Each character is surrounded by a bounding box which defines the spatial extent of a character.

This process creates a character pixel map comprising horizontal and vertical segments. Features based on the aspect ratio and the histogram of the character pixel map are computed. If the value of the feature defined by the character pixel map is not in an acceptable range which has been defined by heuristic rules, then the feature is eliminated as noise.

The resultant features or characters which are in the acceptable range are processed further to define specific groupings of characters. This process determines the location of all potential characters in the image and finds the surrounding rectangle (bounding box) for each potential character. The characters are thus grouped together by proximity and height based on their relative spatial coordinates in order to retain only those characters which should belong to the ID code.

Figures 1a and 1b show the ID code characters in one group which has been highlighted. The other characters are also grouped together, but not highlighted.

Thus, having established the location of all the potential characters and surrounded each character with a boundary box, all the boundary boxes are sorted into horizontal (x) and vertical (y) directions. This allows the characters to be further sorted into groups of

horizontal rows. The ID code usually occurs in one, two or three horizontal rows. Any single isolated characters are discarded at this stage.

5 The polarity of the characters in each group is determined, i.e. whether the characters are white on black or black on white. This information is determined based on the uniformity of the height and width of the characters in the group assuming that there are more uniform characters of one polarity than of another. This information is determined by using a belief or uniformity measure as disclosed in the journal of Mathematical Analysis and Applications 65:531-542, Nguyen HT (1978), "On random sets and belief functions".

15 The grouped characters are selected starting from the top-most row of the image. Those groups which have a belief uniformity measure above a predetermined level are chosen. The number of character groups selected is guided by the total number of characters present in the target ID code provided by the mainframe computer as well as the uniformity measure.

20 At this stage, groups of potential characters have been identified from the image. However, these extracted characters may be skewed, of different sizes and on different backgrounds. Therefore, a normalisation step is carried out which firstly converts the pixel maps of each character to a white character on a black background and secondly standardises the size of each character pixel map using scale factors computed in both x and y directions. Usually, this results in a linear compression since the size of the extracted characters is larger than the standard size.

The normalised grey-scale character pixel map is presented as the input to a neural network. The quantised grey level of each pixel is normalised to a value between 0 and 1. The grey level value is not binarised to either 0 or 1 as the choice of the binarisation threshold would influence the shape of the character defined by the pixel map. This may result in portions of a character being artificially connected or broken.

The neural network is used to recognise the patterns defined by the standard size characters. This method is utilised because it offers good tolerance to noise and deformations in the input characters. Specifically a multi-layer feed-forward window-based neural network model is used. The general architecture of the neural network is shown in Figure 6.

The neural network consists of an input layer, one or more hidden intermediate layers and an output layer. Defined areas or windows of the normalised grey level character pixel map are used as the input for the neural network. In one embodiment, shown in Figure 7, two windows are used which are defined by the upper left and lower right co-ordinates of two rectangles of equivalent size. Together, the two rectangles bound the entire character pixel map. The windows are fed to the input layer of the neural network and each pixel within the boundaries of each window constitutes an input node of the input layer.

Each node of the network computes the following function:

$$ii) \quad y_i = f \left(\sum_{j=1}^N w_{i,j} x_j - \theta_i \right)$$

where y_i = output activation value of node i

x_j = j^{th} signal input to node i

$w_{i,j}$ = connection weight from node j to node i

θ_i = bias

In the above equation, $f(x)$ is a monotonic increasing, differentiable function, the output of which is bounded between 0 and 1.

The input nodes of the input layer are connected to specified ones of the first hidden intermediate nodes of the first hidden intermediate layer. Thus, input nodes of the first window are connected to a first set of specified intermediate nodes and input nodes of the second window are connected to a second set of specified nodes. In this manner the input nodes associated with one window are fully connected to specified hidden intermediate nodes in the first layer but not to hidden intermediate nodes in the first layer associated with other windows. The two sets of this network architecture is shown in Figure 7(i) together with a conventional network as shown in Figure 7(ii). For subsequent layers after the input layer and the first intermediate layer, the nodes of consecutive layers are fully connected.

In a particular embodiment of the invention only the following characters need to be recognized: A to Z and 0 to 9, i.e. a total of 36 characters, 26 letters and 10 numerals. Therefore, the output layer would consist of 36 nodes each representing a particular character.

Using a window-based neural network of this type allows windows to be strategically located in the character pixel map to discriminate between confusing characters and improve the accuracy with which the characters are classified. Also, compared to the conventional fully connected network shown in Figure 7(ii), less synaptic connections are required between the input and first intermediate layers thus reducing processing time and

memory requirements. Experimental results have shown that the time required to train such a system can be substantially reduced yet the recognition performance is improved slightly.

The network is trained using the popular sigmoidal function:

$$\text{iii)} \quad f(x) = \frac{1}{1 + e^{-x/T}}$$

and the network is operated using an approximation to the training function:

$$\text{iv)} \quad f(x) = \frac{T + x + |x|}{2(T + |x|)}$$

In both cases, T is a parameter used to control the non-linearity. Both functions behave similarly in that they are both bounded between 0 and 1, are monotonically increasing and differentiable. Moreover, when T approaches 0, both become step functions. When T approaches ∞ , both approximate to a horizontal line passing through $f(x) = 0.5$. Because the second function requires more iterations to attain a similar level of recognition performance compared to the first function, the first function is used for training purposes. However, after training, the second function replaces the first function without any degradation of recognition performance.

In a preferred embodiment, the neural network does not attempt to distinguish between "O" and "0" or "I" and "1" because some container companies use identical character fonts for both characters. However, the matching procedure which is to be discussed below can resolve any ambiguity that arises due to this confusion.

As previously discussed, the system is intended to verify the ID code of a particular container and this is finally achieved by a last processing step which compares the character string output from the neural network with the target ID code character string sent from the mainframe computer. However, because other processing steps are involved which may introduce errors, the two character strings may not be simply compared. For example, in the segmentation step, ID code characters may be discarded as noise resulting in the omission of a character from the recognised character string or, noise such as smears or marks on the container surface may be treated as ID code characters resulting in characters being inserted into the recognised character string. The segmentation step may determine a character boundary incorrectly and thereby split a single character into two. Also, the neural network may recognise a character incorrectly thus substituting a particular character with an erroneous character. It can clearly be seen that the recognised character string may be potentially erroneous and so a matching step is included to establish the best match between the recognised character string and the target ID code string.

Referring to Figure 8, the top row of characters represents the target ID string and the second row represents the character string achieved after the segmentation step. The neural network outputs a range of scores, each of which represents the degree of similarity between the input pixel map and the 36 character output classes of the neural network (i.e. the degree of recognition for each character class). Rows 3 and 4 of Figure 8 show the best and second best scores respectively. For each entry in these rows, the character denotes the recognized code while the number in parenthesis indicates

the corresponding score. As shown in the example, the neural network commits two substitution errors since the best score for the letter "U" in the target string is given as "O" (0.7) and the target character of "2" is given by the neural network as "7"(0.6). In this example, the segmentation step has also introduced two errors, the addition of the character "#" in response, presumably, to some noise and has omitted the character "0" from the target ID code character string. The procedure described below has been designed to resolve these ambiguities in both the recognition and segmentation steps by converting the problem of verifying the target ID code character string into a two dimensional path searching problem, as shown in Figure 9 which indicates the matching path found for the example of Figure 8, the goal of which is to find the optimal matching path.

In the preferred embodiment, the optimal path is found by a technique known as Dynamic Time Warping as discussed in Itakura F. (1975) "Minimum prediction residual principle applied to speech recognition", IEEE Transactions on Acoustics, Speech and Signal Processing 23:67-72 and in Satoe H, Chiba S (1978) "Dynamic programming algorithm optimisation for spoken word recognition", IEEE Transactions on Acoustics Speech and Signal Processing 26:43-49. In this embodiment local continuity constraints as shown in Figure 10 are defined to restrict the search area. The constraints specify that to reach point (i, j) , five local paths are possible (paths 1-5). Under normal circumstances i.e. no insertion or deletion of characters at this point, path 3 should be taken. If path 2 is chosen, the algorithm suggests that the $(i - 1)^{th}$ character in the recognition string is an insertion error. Likewise, if path 4 is selected, it indicates that there is an omission between the $(i - 1)^{th}$ and i^{th} position. Path 1 and

path 5 are included to take care of boundary conditions. Path 1 caters for extraneous characters positioned before or after the target ID string whereas path 5 is needed for cases where the first or last few ID characters are missing. Thus paths 1 and 5 are invoked only near the beginning or the end of the search area.

Once the local continuity constraints are specified, the global search area is determined. To find the best path, a cumulative matching score $D_{i,j}$ is maximised. This score can be recursively computed as follows:

$$v) \quad D_{i,j} = \max \begin{cases} D_{i-1,j} & ; \text{path 1} \\ D_{i-2,j-1} + S - p & ; \text{path-2} \\ D_{i-1,j-1} + S & ; \text{path-3} \\ D_{i-1,j-2} + S - q & ; \text{path-4} \\ D_{i,j-1} & ; \text{path-5} \end{cases}$$

where S is the neural network response for the target character j , p is the penalty for an insertion error and q for an omission error. The penalties are set to 0.2 in this embodiment.

The basic algorithm to find the optimal matching path is as follows:

Let I be the number of characters in the recognized string and J the number of characters in the target ID string

Step 1: for $j = 1$ to J

 initialize $D_{0,j}$ to a negative large number

$D_{0,0} = 0.0$

Step 2: for $i = 1$ to I
 for $j = 1$ to $J + 1$
 compute $D_{i,j}$ according to the equation (v)
 register the local path taken to an array $path_{i,j}$

Step 3: Starting from $j = J + 1$
 while ($j \geq 0$)
 trace back the $path_{i,j}$ array and register the
 matching character in the recognition string.

At the end of Step 3, the optimal matching pairs between the target and the recognition string are determined. The result could be directly reported to the operator. However, to make the system easier for the operator to use, it is better to aggregate all the types of errors into a single performance index. This is achieved by defining a confidence measure as discussed below.

The confidence measure must be able to reflect the actual accuracy of the entire system without penalising for insignificant mistakes. For example, insertion errors are not serious mistakes though they should be avoided as far as possible. As for character recognition, although the highest score produced by the recognizer may not correspond to the correct character, characters having the second or third highest scores may correspond. Further, the difference between this score and the highest score may be small. To take care of such situations, a penalty is introduced which depends on the extent to which the character has been mis-recognized.

Thus, having considered all these factors, the performance index or confidence measure is computed as follows:

$$CM = \frac{1}{J} \sum_{j=1}^N \frac{\text{score}(i)}{\text{best score}(i)}$$

where N is the number of matching character pairs found by the dynamic time warping procedure, score(i) is the similarity score of the ith character. If there is any omission error, then N will be less than J. If character i is correctly recognised, then score(i) is the best score and equals best score and the contribution of that character to the sum is 1. It can be seen that if all the characters are segmented and recognized correctly, then the sum of the N elements is J and hence the confidence measure, $CM = J/J = 1$. For insertion errors which are detected by the dynamic time warping process, there will be no character in the ID string associated with them. Hence these extra characters will not enter into the computation.

Thus, the final result of the entire process is the CM score. Its value lies between 0 and 1. The operator can therefore set a specific confidence measure threshold so that the container ID code verification is deemed successful if the CM score is higher than the threshold. Otherwise, the operator is notified and can investigate the matter.

In the preferred embodiment, the processes described above have been implemented on a five transputer network which is hosted on a PC-AT microcomputer. The code was written in Parallel C Ver. 2.1.1. using the 3L Compiler. The character extraction and recognition

processes were parallelised while the normalization and matching procedures were handled by the master transputer. In order to implement the character segmentation process in parallel the entire image is divided into five equal vertical strips. Each processor is allocated one image strip. Four of the vertical strips overlap in the x direction by an amount equal to the maximum allowable width of a character minus 1. This is to ensure that every character is extracted in full by a transputer. The master transputer distributes the image strips to the root and the three workers. When all the processors have finished and sent back the potential bounding boxes the master proceeds with the grouping phase of the character extraction. The recognition process takes place in parallel by distributing the character maps one by one to the worker processors. The master transputer controls and coordinates the sending of pixel maps and receiving of recognition results.

While determining the performance of the system, the character extraction and recognition processes were separately evaluated in addition to testing the accuracy of the overall system. This was done in order to test the accuracy of the individual processes, since the errors produced during character extraction are propagated to the recognition phase. The segmentation and character extraction process was tested on 191 character images under varying illumination conditions. The results of computer extraction i.e. the bounding boxes of characters, were compared to the coordinates of characters extracted manually. Table 1 shows that the number of characters correctly segmented out comprises 91.365% of the total characters present in the images.

A larger database of 441 character images was used to evaluate the recognition process. The statistical

distribution of characters was found to be uneven. For example, characters 'Q' and 'V' do not occur at all while 'U' appears very frequently. On the other hand, since the sample size is sufficiently large, such statistics reflect the true probability of occurrence of each individual character during actual operation. In the recognition experiments, the database is partitioned into two sets: DS1 for training and DS2 for testing. DS1 consists of 221 images and 2231 alphanumeric characters, while DS2 has 2167 characters in the remaining 220 images. There are roughly twice as many numeral characters as alphabetic characters in both sets.

The estimated verification time for each container image is approximately 13.5 seconds. This includes the time elapsed from when the container information is received from the Gate PC till the recognition results and confidence measure are sent back to the Gate PC. Figures 1 (a) and (b) show examples of container codes extracted by the above described method. The extracted characters and their circumscribing rectangles are shown after the thresholding process.

Modifications are envisaged to improve the robustness of the character extraction process and increase its accuracy when the container image is noisy. Various approaches are being considered to recover characters with poor contrast, merged characters and partially obliterated characters.

5

CLAIMS:

1. A container code verification apparatus comprising: image capture means (1, 2, 3) to capture an image of a container (8) surface carrying a displayed code (16); data
10 processing means (5) to form a pixel array from the captured image; scanning means to scan the pixel array; detection means to detect potential characters; selection and grouping means to select and group together potential characters of the displayed codes; and a neural network to
15 determine from the pixel values of the potential characters a set of recognised characters constituting a recognised code, the pixel array being divided into a plurality of windows, the neural network comprising a plurality of input nodes defining an input layer and at least one output node
20 defining an output layer, one or more intermediate layers being disposed between the input layer and the output layer, in which neural network the input layer is divided into a plurality of discrete areas each corresponding to a respective window; the pixel value of each pixel in a
25 window represents the input for a corresponding input node in an area of the input layer corresponding to the respective window; each node within the network computes an output in accordance with a predetermined function; the outputs of the nodes in each area of the input layer are
30 connected to specified nodes in a first intermediate layer which are not connected to the outputs of the nodes in another area of the input layer; the outputs of the nodes in the first and subsequent intermediate layers being connected to the inputs of the nodes in the immediately
35 following layer; and the output nodes of the last intermediate layer are connected to the inputs of the output node or nodes of the output layer, the container code verification apparatus further comprising: comparison means to compare the target code with the recognised code;



and verification means to verify or reject the recognised code in dependence upon the results of the comparison between the target code and the recognised code.

2. A container code verification apparatus according to Claim 1, wherein each node in the neural network computes the function:

$$y_i = f \left(\sum_{j=1}^N w_{i,j} x_j - \theta_i \right)$$

where y_i = output activation value of node i
 x_j = j^{th} signal input to node i
 $w_{i,j}$ = connection weight from node j to node i
 θ_i = bias

3. A container code verification apparatus according to Claim 1 or 2, wherein the neural network is operated using the function:

$$f(x) = \frac{T + x + |x|}{2(T + |x|)}$$

4. A container code verification apparatus according to any one of Claims 1 to 3, wherein the neural network is trained using the function:

$$f(x) = \frac{1}{1 + e^{-x/T}}$$

5. A container code verification apparatus according to any preceding claim wherein the neural network is included in a transputer network (5).

6. A container code verification apparatus according to any preceding claim wherein the output layer of the neural network provides a set of scores indicating the degree of recognition between the character defined by the



pixel array and the classes of character recognisable by the neural network.

5 7. A container code verification apparatus according to any preceding claim wherein means are provided to surround the detected potential character with a bounding box.

10 8. A container code verification apparatus according to Claim 7, wherein each bounding box is divided into a plurality of windows and the quantisation value of the pixels contained in each window comprises the input values for a corresponding discrete area defined in the input layer of the neural network.

15 9. A container code verification apparatus according to any preceding claim wherein means are provided to detect the potential characters by identifying horizontal segments (h1...n) comprising horizontally adjacent pixels with substantially equivalent quantisation levels and vertical segments (v1...n) comprising vertically adjacent pixels of substantially equivalent quantisation levels and horizontal and vertical segments (h1...n, v1...n) which are connected together defining a potential character.

20 10. A container code verification apparatus according to any preceding claim wherein the container surface carries potential characters which are spatially located horizontally adjacent one another and the grouping means group such horizontally adjacent characters together.

25 11. A container code verification apparatus according to Claim 10, wherein the selection means select only the horizontally grouped potential characters.



12. A container code verification apparatus according to any preceding claim wherein the selection means does not select any single isolated potential characters.

5 13. A container code verification apparatus according to any preceding claim wherein means are provided to determine the polarity of the foreground and background of each potential character and to convert the polarity to a standard polarity.

14. A container code verification apparatus according to any preceding claim wherein means are provided to scale the potential characters to a standard size.

15 15. A container code verification apparatus according to any preceding claim wherein means are provided to normalise the quantisation level of each pixel to a value between 0 and 1.

20 16. A container code verification apparatus according to any preceding claim wherein means are provided to compare the recognised code and the target code by obtaining a confidence measure representing the degree of match between the two codes and comparing the confidence
25 measure with a predetermined threshold.

17. A container code verification apparatus according to any preceding claim further comprising means to position the container (8) carrying the displayed code (16) adjacent
30 an image capturing means (1, 2, 3) to capture the image of the container surface.

18. A container code verification apparatus according to any preceding claim wherein artificial lighting is



provided to illuminate the container to increase the image contrast.

5 19. A container code verification apparatus according to any preceding claim wherein the image is captured by at least one closed circuit TV camera (1, 2, 3) connected to a frame grabber (11).

10 20. A container code verification apparatus according to Claim 19, wherein the frame grabber has a contrast parameter which is determined by Newton's iterative method.

15 21. A container code verification apparatus according to any preceding claim wherein a remote mainframe computer provides the target code.

20 22. A container code verification apparatus substantially as hereinbefore described with reference to, and as shown in, the accompanying drawings.

