CARMEN® OCR – for ACCR

Automatic Container Code Recognition

(Carmen OCR sdk + ACCR engine)

Reference Manual

For engine ver. 7.2.3 and above

Document version: 30.05.19

Table of Contents

INTRODUCTION

CARMEN® OCR is the common SDK for recognition of all non-license plate related transportation codes.

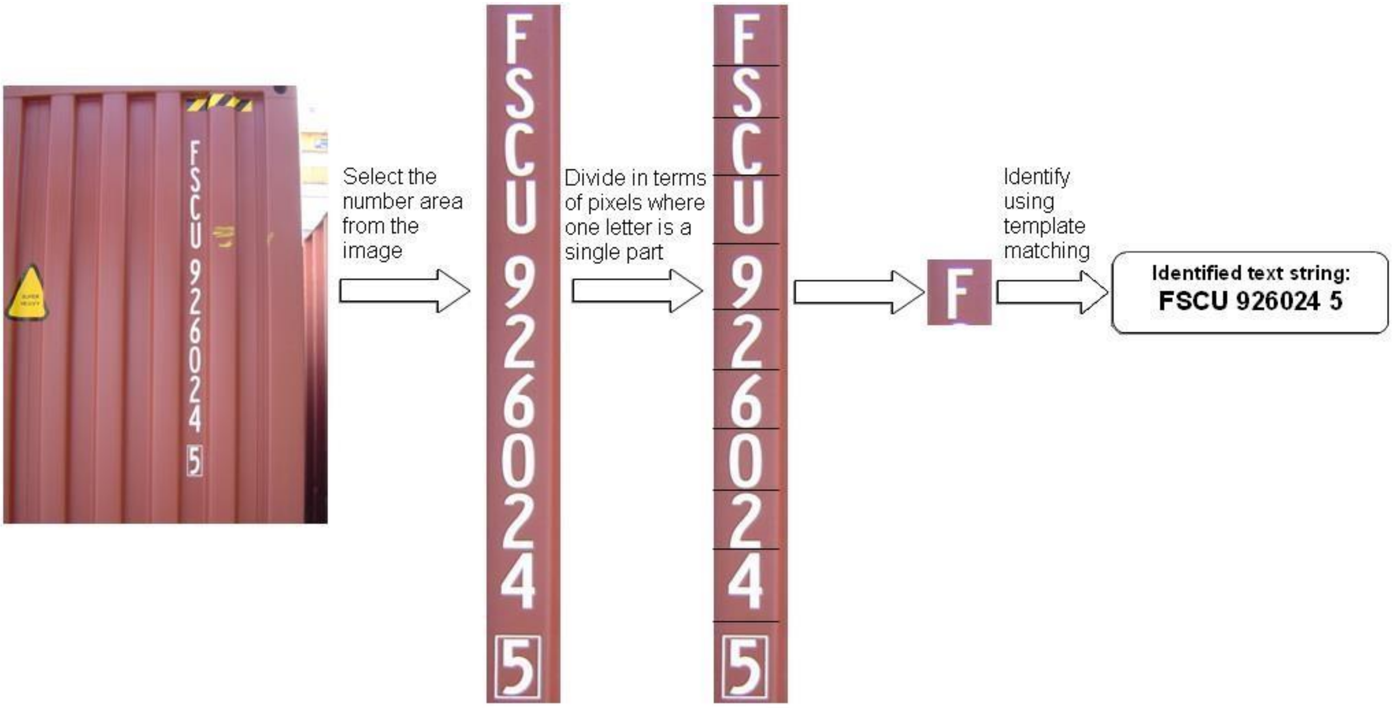
This is possible through using different recognition engines with the sdk, like:

- ACCR - Intermodal shipping container codes, BIC/ISO/ILU,
- UIC - Cargo wagon codes,
- USDOT - US Department of Transportation registration number
- MOCO – Moco container codes (8 digits),
- RUS - Russian and Ukrainian wagon codes (8 digits).
- BRA - Brasil wagon codes (10 or 11 digits)
- CHASSIS - US truck chassis code,
- USA_ACCR - US container codes (6 or 10 digits),

Although the engine properties below are the same for each of the solutions above, this document details the usage and properties of the ACCR (Automatic Container Code Recognition) engine.

Container codes are used for identification to keep track of containers in the logistic and inventory management systems. Automatic Container Code Recognition is a scientific know-how, which is able to recognize these unique codes from an image or video signal, and return them in a computer editable format for further processing.

The OCR is an Optical Character Recognition based technology, which means intelligent image processing:


FSCU 926024 5
Select the number area from the image
Divide in terms of pixels where one letter is a single part
Identify using template matching
F
Identified text string:
FSCU 926024 5

CARMEN® OCR is an optical character code recognition software of **ARH Inc**, designed for codes in the cargo and transportation sector. It provides an easy-to-program Application Programming Interface (API) for software developers.

The name of the SDK is **CARMEN® OCR,** which is able to read container codes through the **Carmen ACCR engine**.

The currently supported programming languages and the available interfaces are the following:

- C, C++ (cmaccr.h)
- Java (jcmaccr.jar)
- C# (cmaccrdotnet8.dll) (.NET 4.0 interface)

BASIC FUNCTIONALITY

The functionality of **CARMEN® OCR** is centered upon the ***FindFirstContainerCode*** function. This takes container images and returns the code read, the overall confidence level[1] and separately the read results for each image.

Initialization

On startup, **CARMEN® OCR** loads the ACCR engine and reads its data files from the module folder. The name of the default data file is {engine name}.dat (i.e. ***cmaccr_iso_7.3.2.26.dat*** for the ***cmaccr_iso_7.3.2.26_19Q1*** engine).

Compatibility

In most cases, the **CARMEN® OCR** system retains the backward compatibility with the older versions so, the compiled programs run perfectly well even if the system is upgraded to a newer version.
The evolution of the system adds the necessity of changing the SDK by adding new functions, new parameters to functions, eventually changing the type of the return value or moving the deprecated functions into the compatibility section that may not be installed by default. In this way, users are forced to use the newest version each time they compile their programs with the **CARMEN® OCR** SDK, but there is also the possibility to compile the existing sources using the appropriate version.

What to do if you do not want to update your code?

- The old versions of the interface files are available on the install CD or request them from the Support Team.
- Without updating the ActiveX interfaces (Delphi packages, C++ wrapper class files) the old methods do no work appropriately.
- Without updating the Java (jcmaccr.jar, jcmaccr.dll) and .NET (cmaccrdotnet7.dll) interfaces the old methods will work appropriately.

View the cmaccr.h for full reference and check the Change Log for differences against the previous versions.

For complete understanding of the system internals, consult the GX Reference Manual.

[1] **Confidence level**

The confidence level value represents that to what extent the engine is sure about the result. Dirt and damages on containers or poor image quality increase the risk of mistakes. The greater amount of risk is represented with low confidence level values by the engine.

PROPERTIES OF THE AUTOMATIC CONTAINER CODE READER MODULE

Property list of the CARMEN® OCR engines

PROPERTIES OF THE CARMEN OCR ENGINES	
Properties related to the identification of the current engine	datafile
Properties related to processing time	timeout
	contrast_min
Properties related to the geometry of plates	size
	size_min
	size_max
	slant
	slope
	slope_min
	slope_max
	xtoyres
Properties related to the position of the codes in input images	posfreq
	posfreqhistxs
	posfreqhistys
	posfreqhalflife
	posfreqweight
Properties related to code type	filterlongcodes
	isocode
	checksum

1. DATAFILE – NAME OF DATA FILE OF THE ENGINE

Specifies the name of the module data file located in the module folder. If not set, then the default data file will be used as follows: engine_name.dat (i.e. ***cmaccr_iso_7.3.2.26.dat*** for the ***cmaccr_iso_7.3.2.26_19Q1*** engine). If you have a data file with parameters optimized for some special container code types (for example ***cmaccr_iso_7.3.2.26.dat*** for European ISO container codes) you can use this file as well. Default value: engine_name.dat

Properties related to processing time

2. **TIMEOUT** – RECOGNITION TIME LIMIT

The length of the time interval in milliseconds in which the module tries to find container codes. The interval starts when cmaccr_findfirstcontainercode is called. At the end of this period, the engine tries to finish searching for codes. Zero timeout value means no time limit.

Note: In case of multiple images load with the same container code, timeout applies for recognition of the full image series. If timeout runs out before all images processed, only the codes for the number of images fit into the timeout frame will be resulted.
Default value: 1000

3. **CONTRAST_MIN** – MINIMAL CONTRAST

The supposed minimal difference between the grayscale value of the characters and the background.

Properties related to the geometry of the container codes

By setting the below properties of the engine appropriately, the recognition time of the engine can be decreased. Properties effecting recognition time significantly are **size_min**, **size_max** and **contrast_min**. These are set appropriately if the intervals defined by them are as narrow as possible (suited to camera environment).

4. **SIZE** – AVERAGE HEIGHT OF THE CHARACTERS

The average height of the container code characters in the image in pixels. No default value.

NOTE: size_min ≤ size ≤ size_max. Therefore, the size value has to be equal to or greater than size_min and equal to or less than size_max otherwise the engine may return no data.

5. **SIZE_MIN** – MINIMUM HEIGHT OF THE CHARACTERS

The minimum height of the container code characters in the image in pixels. No default value.

6. **SIZE_MAX** – MAXIMUM HEIGHT OF CHARACTERS

The maximum height of the container code characters in the image in pixels. No default value.

7. **SLOPE** – AVERAGE SLOPE OF THE CONTAINER CODE CHARACTER

The average slope of the container code character in the image. This value is represented in percent (%) and it is positive if the horizontal axis of the character slopes downwards viewing from left to right.

NOTE: slope_min ≤ slope ≤ slope_max
So, the slope value has to be equal to or greater than slope_min and equal to or less than slope_max otherwise the engine may return no data.

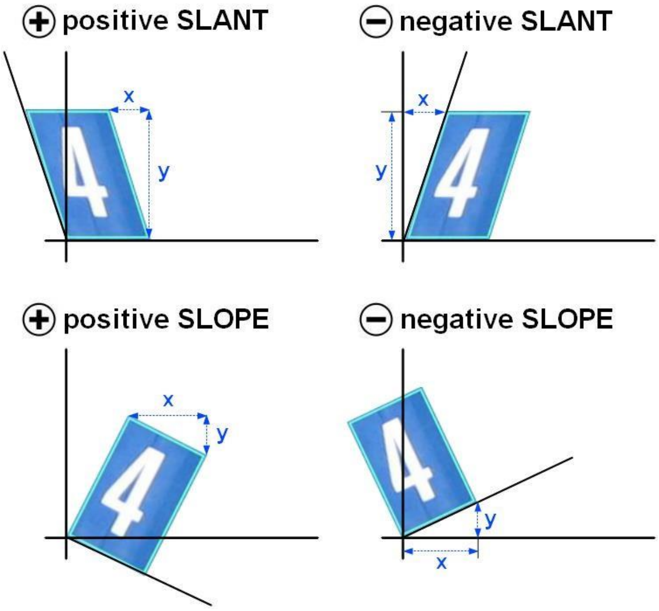
8. **SLOPE_MIN** – MINIMUM SLOPE OF THE CONTAINER CODE CHARACTER

The minimum slope of the container code character in the image. This value is represented in percent (%) and it is positive if the horizontal axis of the character slopes downwards viewing from left to right.

9. **SLOPE_MAX** – MAXIMUM SLOPE OF THE CONTAINER CODE CHARACTER

The maximum slope of the container code character in the image. This value is represented in percent (%) and it is positive if the horizontal axis of the character slopes downwards viewing from left to right.

10. **SLANT** – AVERAGE SLANT OF THE CONTAINER CODE CHARACTER



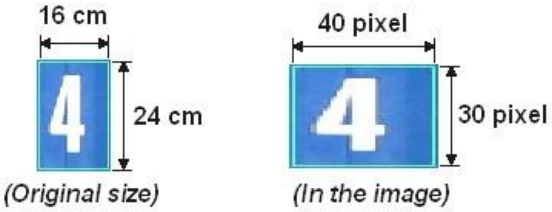
11. **XTOYRES** – THE RATIO OF HORIZONTAL AND VERTICAL RESOLUTIONS

The average slant of the container code character in the image. This value is represented in percent (%) and it is positive if the vertical axis of the characters slants to the left viewing from bottom to top.

The ratio of the horizontal and vertical resolutions of the frame of the character area. This value is represented in percent (%). The horizontal resolution is the ratio of the width of the character frame in the image and the real width of the character. The vertical resolution is the ratio of the height of the character frame in the image and the real height of the character.

If the original height of a character is (Ys) 24 cm and the width is (Xs) 16 cm but in the image the height of the character is (Ysi) 30 pixels and the width is (Xsi) 40 pixels, than

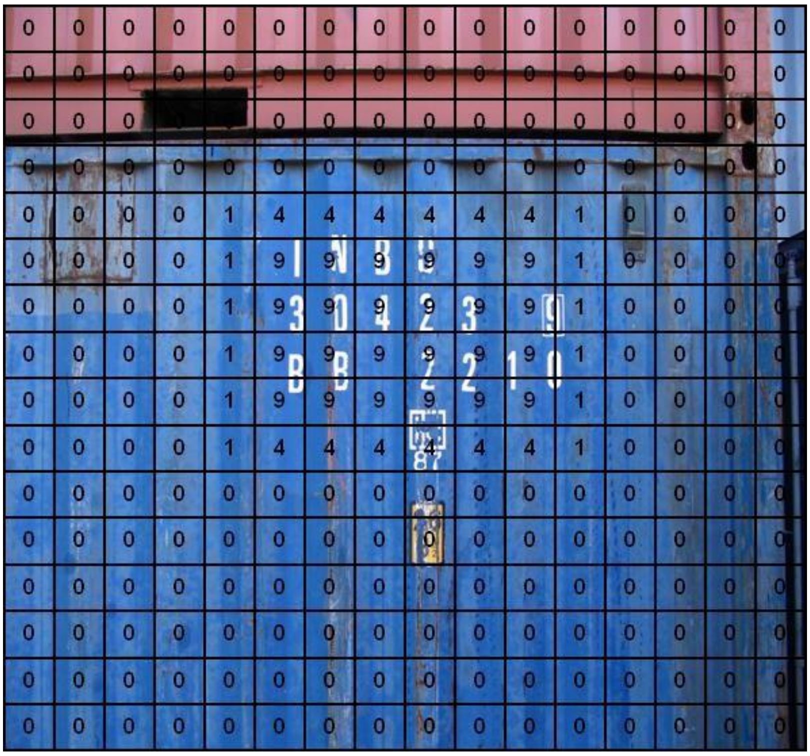
$$xtoyres = 100 \cdot \frac{\frac{Xsi}{Ysi}}{\frac{Xs}{Ys}} = 100 \cdot \frac{\frac{40}{30}}{\frac{16}{24}} = 200$$



Default value: 100

Properties related to the position of the container codes

An area can be specified for the ACCR engine, which is more superior to the others.
Please check the following example, which indicates superior areas in the center of the image:



0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	4	4	4	4	4	4	1	0	0	0	0
0	0	0	0	1	9	9	9	9	9	9	1	0	0	0	0
0	0	0	0	1	9	9	9	9	9	9	1	0	0	0	0
0	0	0	0	1	9	9	9	9	9	9	1	0	0	0	0
0	0	0	0	1	9	9	9	9	9	9	1	0	0	0	0
0	0	0	0	1	4	4	4	4	4	4	1	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

With the following parameters, the algorithm can be set to search for container codes on specific parts of the image. Moreover, some parts can be differentiated according to probability of the occurrence of the container code. The essence of the method is that the image is divided into zones with identical areas and each zone has a weight.

The correct value assignment of the weight increases the effectiveness of the searching process. With the growth of the weight of the appointed zone, the probability of finding the code increases and the time required for reading the code decreases.

The weights can be initialized by the equal spread function, by the assigning of the weight values of zones and finally by defining a polygon that should contain all the codes.
There is a possibility for adoption of the weights as well. In this case, each found code increases the weight of that zone which contains the code.

12. **POSFREQ -** CONTAINER CODE POSITION FREQUENCY

String – which can be an empty string – may contain numbers divided by ',' and ';' within a row.
If the string is empty, it initializes the grid with uniform distribution (the weight of each zone will be the same positive number).

Otherwise, ',' separates the numbers while the ';' separates the line wrap.
If there are two columns and at least three rows, the string defines a polygon. Otherwise, it defines zones, where the given numbers are the starting weights of the zones.
The data is invalid if the rows are not the same length.

For example:

Defining a zone:

<posfreq value=“1,1,1;4,9,4;1,1,1”/>

1	1	1
4	9	4
1	1	1

The image is divided into 3×3 zone with the given starting weights.

Defining a polygon:

<posfreq value="100,100;500,100;500,500;100,500"/>

(The order of the coordinates has to be set clockwise)

In this case, a square is designated in the center of the image, on the score of which the algorithm produces the zones (posfreqhistxs*posfreqhistys is the number of zones, 16*16 by default) in such a way that:

- the starting weight of the zones – contained by the polygon – will be maximal,
- the weight of the zones intersected by the polygon will be lower in proportion to the intersection.
- finally, the weight of the outer (untouched by the polygon) zones will be 0.

It does not return any character from the zones with 0 weight.

13. POSFREQHISTXS,POSFREQHISTYS - [2..64],[2..64]

The number of the rows and columns.
Both are 16 by default.

14. POSFREQHALFLIFE - [0..1048576]

If its value is 0, the weights will not be adopted (it does not learn from the previous cases). It will use the original settings all the time.
Otherwise, after 'posfreqhalflife' number of evaluations, the starting information will be half lapsed and the new information will be half freshened.
Half-life: after evaluation of many images, the total weight of the histogram will be twice as much.

15. POSFREQWEIGHT - [0..100]

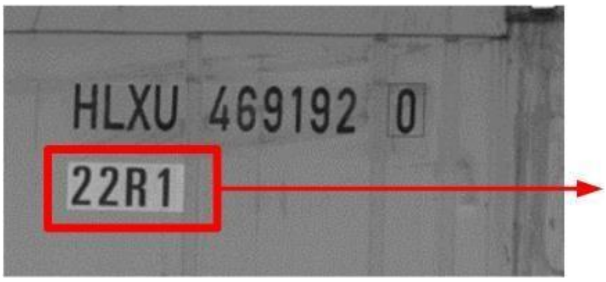
This parameter defines how the system has to take into account the position of the container code.
If this parameter is 0, the system doesn’t distinguish between the non-0 weight zones. In this case, the searching does not exploit the distribution of the position of container code.
If this parameter is 100, the system tries to exploit maximally the distribution of the position of container code.

Properties related to code type

16. **FILTERLONGCODES** – LONG CODE FILTERING

If there is no code on a container but there is other kind of text on it then the engine may return with false (container) codes. The FilterLongCodes option helps to reduce the occurrence of these false codes, increases runtime length (approx 5%), and worsens the recognition of container codes on pictures having a container code (approx 0.5%). If this option is set to 1 then filtering is enabled. (If set to 0, filtering is disabled) Default value: 1.

17. **ISOCODE** – ISO CODE RECOGNITION



1 – the engine searches for the last four digits of the entire ISO code as well.

Note: only works with ACCR - ISO engine.

Possible values: 0 – the engine does not search for the last four digits.

Default value is 1.

18. **CHECKSUM** – FLAG TO MARK THE PRESENCE OF CHECKSUM

Earlier, this parameter used to determine the recognised codes. Recently, this is fixed for each recognition engine **– do not change this property**.

1 - USA ACCR: 6 number code,
2 - MOCO: 8 character MOCO codes,
4 - USA ACCR: 10 character codes,
5 - USA ACCR: for 6 and 10 character codes,
8 - ISO / ILU: 11 character codes – ISO 6346 (+ optional 4 ISO code character)
16 - RUS: (8 characters, numeric - Russian and Ukrainian wagon codes).
32 - BRA container
64 - USDOT
256 - UIC
512 - Chassis – US truck chassis code

APPENDICES

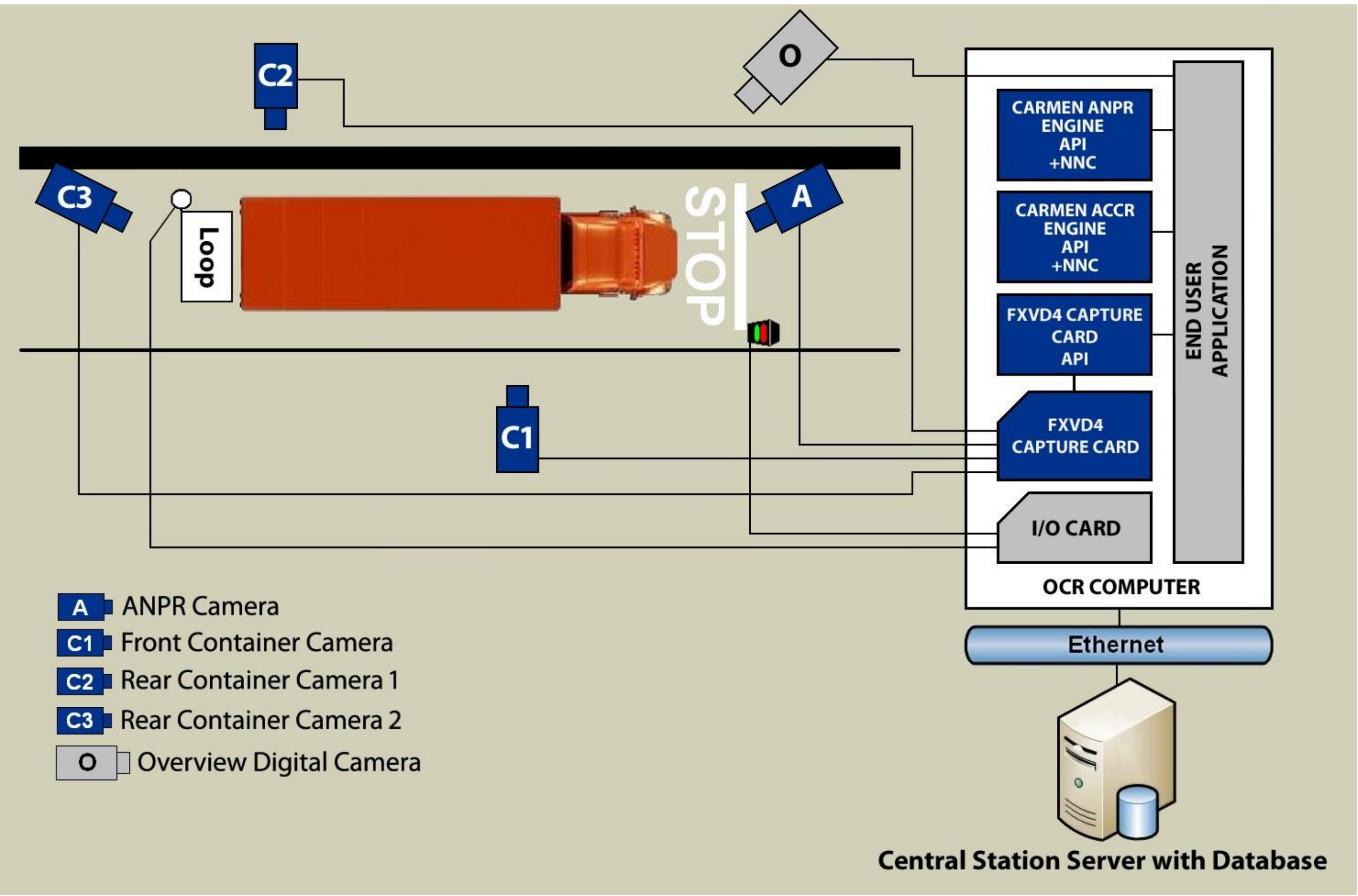
Confidence Level Calculation

$$\frac{k}{n+1} *$$ ***the highest image confidence***

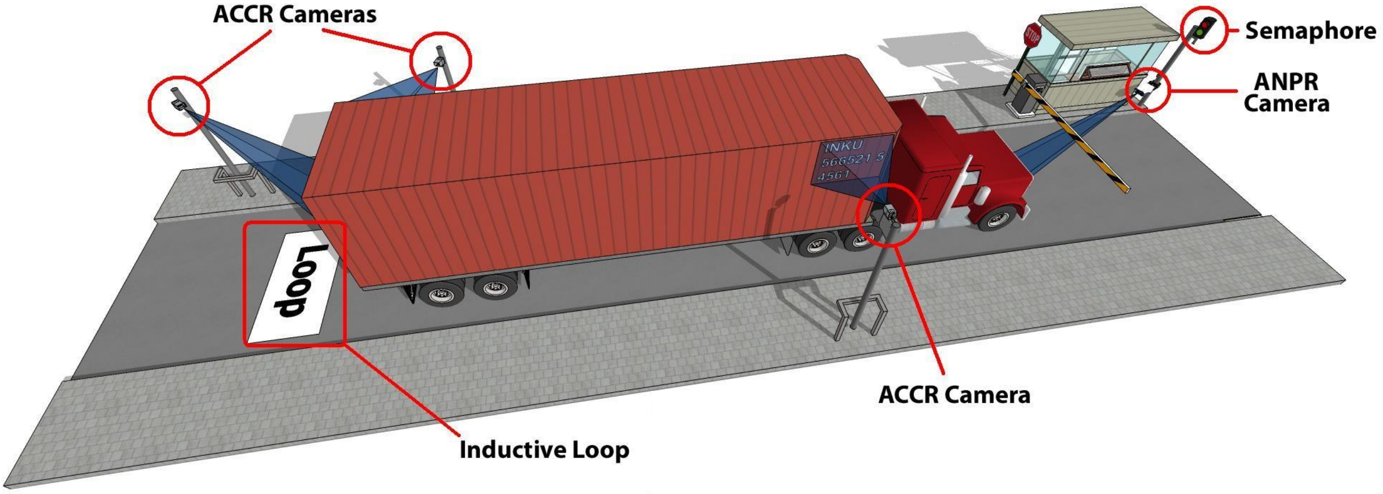
Where:

- **k** stands for the number of images, where the SAME result was read
- **n** stands for the number of all images of the container
- **"the highest image confidence" =** highest confidence of the image series where the confidence level of one image equals the minimum confidence of the characters on the actual image

Sample Installation - Truck Lane


C2
O
C3
Loop
STOP
A
C1
CARMEN ANPR ENGINE API +NNC
CARMEN ACCR ENGINE API +NNC
FXVD4 CAPTURE CARD API
FXVD4 CAPTURE CARD
I/O CARD
END USER APPLICATION
OCR COMPUTER
Ethernet
Central Station Server with Database
A ANPR Camera
C1 Front Container Camera
C2 Rear Container Camera 1
C3 Rear Container Camera 2
O Overview Digital Camera

NOTE: The FXVD4 capture card and API is only needed in case of analog CCTV cameras.



In the above image, a sample application can be seen for ACCR purposes. In many cases the container codes are damaged, therefore we suggest to capture multiple images from different sides of the container and handle them together as an image sequence. Three container cameras are recommended to reach the most precise reading results. To provide optimal conditions for taking an image, we suggest using two external white LED illuminator devices. With the help of an additional ANPR camera, the truck, carrying the container, can be identified.

To reduce consumption of the computer resources we suggest using external triggering to capture images only if the container code and the license plate are in a good position.

The inductive loop (other triggering possibilities: laser sensor, microwave sensor, magnetic sensor or infrared barrier) serves as a sensor, which indicates the arrival of the vehicle.

As the vehicle stops at the “STOP” sign, all cameras identify the container code and the license plate of the truck. For more information, feel free to contact our Support Team.

CONTACT INFORMATION

HEADQUARTERS:
ARH Inc.
Alkotás utca 41
HU-1123 Budapest
Hungary
Phone: +36 1 201 9650
Fax: +36 1 201 9651
Web: www.arh.hu

ARH AMERICA:
ARA Corp.
28059 US Highway 19 North Suite 203
Clearwater, FL 33761
Phone: (727) 724-4219
Fax: (727) 724-4290
Web: www.adaptiverecognition.com

SERVICE ADDRESS:
ARH Inc.
Ipari Park HRSZ 1113/1
HU-2074 Perbál
Hungary
Phone: +36 1 2019650

ARH Technical Support System (ATSS) is designed to provide you the fastest and most proficient assistance, so you can quickly get back to business.

Information regarding hardware, software, manuals and FAQ are easily accessible for customers who previously registered to enter the dedicated ATSS site. Besides offering assistance, the site is also designed to provide maximum protection while managing your business information and technical solutions utilized.

NEW USER

If this is your first online support request, please create an account by clicking on this link.

RETURNING USER

All registered ATSS customers receive a personal access link via e-mail. If you previously received a confirmation message from ATSS, it contains the embedded link that allows you to securely enter the support site.

If you need assistance with login or registration, please contact atsshelp@arh.hu for help.