CONPDS

OCR Engine - Product documentation

iOS 2021.03

INTRODUCTION

The ConPDS OCR engine is not an OCR SDK but a pre-compiled, plug-and-play library, ready-to-use in your software projects. It supports the detection of BIC container codes (ISO 6346), including ISO Size and Type and ILU container codes (EN 13044-1 standard).

HOW IT WORKS

⇒ Capture the image with container code and load it into the OCR engine.

⇒ Preprocessing is making image noise reduction, binarization, image emphasis, and skew correction.

⇒ Detection is our optimized set of algorithms that analyze the image, detect areas with characters in patterns specified in BIC and ILU standards, cleanse the area, and remove any noise in each area.

⇒ The recognition phase identifies and recognizes each character, combines them into a container code, and assigns a confidence factor the result. There can be multiple recognition results and a calculation of a confidence factor for each candidate done.

⇒ Recognition results are returned as JSON response and can be used in further processing or exported. The developer has full control over the OCR results.

METHODS OF PROCESSING

All versions of the ConPDS OCR Engine supports two ways of processing images.

- Image file from storage
- Image binary from memory. It can be single frames from a video stream.

In both cases, it is up to the developer to capture images from a camera (mobile device, stationary camera, etc.) or capture frames from a video stream, e.g., by using RTSP.

CURRENT VERSION

2021.03 – released March 2021

SUPPORTED IMAGE FORMATS

Images for OCR processing should be in one of the following formats:

- JPEG format (https://en.wikipedia.org/wiki/JPEG).
- BMP format (https://en.wikipedia.org/wiki/BMP_file_format)
- PNG format (https://en.wikipedia.org/wiki/Portable_Network_Graphics)

RECOMMENDED IMAGE RESOLUTIONS

For the best and optimal recognition speed, we recommend below image resolutions.

Aspect ratio	Resolutions (HxW or WxH) *)
4:3	1024×768, 1280×960, 1400×1050, 1440×1080
16:10	1280×800, 1440×900, 1680×1050

*) Lower image resolutions are supported but will result in less accuracy in recognitions. Higher image resolutions are also supported, but as the image file size is big (2MB or higher), it will result in slow or even failed recognition.

The average OCR processing time for a JPEG image (1024x768 pixels, 120Kb in file size) is <100ms.

Tip: Invest some hours in optimizing the image capture process so that image resolutions are in recommended resolutions and aim for image file sizes not bigger than 200-250Kb. It will result in the fastest processing time and produce good recognition results.

SUPPORTED CHARACTER RESOLUTIONS

For optimal character recognition, we recommend the following character resolutions.

- 14 pixels – minimum *)
- 20 pixels – acceptable
- 50-100 pixels – recommended
- 200 pixels – maximum *)

*) Character resolution lower or higher will result in poor if any recognition.

SUPPORTED ANGLE RANGE OF IMAGES

The maximum image rotation – X (pitch), Y (yaw), Z (roll) - is ± 30° (recommended ± 15°). An image rotation higher than will result in no recognition.

SYSTEM REQUIREMENTS AND PRE-REQUISITES

- Apple iOS version 9.x or later

PERMISSIONS

- Access to camera or photos library (https://developer.apple.com/documentation/avfoundation/cameras_and_media_capture/requesting_authorization_for_media_capture_on_ios)

PREPARATION FOR USAGE

- Add .framework file to project
- Add .framework file to "Linked frameworks and libraries" to project target

USING THE LIBRARY

To start using the library, add import to top of your header file (.h) or Swift source file (.swift)

Objective-C: import <ConPDS/ConPDS.h>

Swift: import ConPDS

Note: Framework contains both iOS devices (armv7, arm64) and iOS simulator (i386, x86_64) architectures. To deploy an app to a real device or upload the app to AppStore; you need to trim unsupported architectures for the current target. To do this, go in XCode -> Project -> Build Phases add at last Run Script phase.

Insert following script:

```
APP_PATH="${TARGET_BUILD_DIR}/${WRAPPER_NAME}"
find "$APP_PATH" -name '*.framework' -type d | while read -r FRAMEWORK
do
   FRAMEWORK_EXECUTABLE_NAME=$(defaults read "$FRAMEWORK/Info.plist" CFBundleExecutable)
   FRAMEWORK_EXECUTABLE_PATH="$FRAMEWORK/$FRAMEWORK_EXECUTABLE_NAME"
   echo "Executable is $FRAMEWORK_EXECUTABLE_PATH"

   EXTRACTED_ARCHS=()
   for ARCH in $ARCHS
   do
   echo "Extracting $ARCH from $FRAMEWORK_EXECUTABLE_NAME"
   lipo -extract "$ARCH" "$FRAMEWORK_EXECUTABLE_PATH" -o "$FRAMEWORK_EXECUTABLE_PATH-
   $ARCH"
   EXTRACTED_ARCHS+=("$FRAMEWORK_EXECUTABLE_PATH-$ARCH")
   done

   echo "Merging extracted architectures: ${ARCHS}"
   lipo -o "$FRAMEWORK_EXECUTABLE_PATH-merged" -create "${EXTRACTED_ARCHS[@]}"
   rm "${EXTRACTED_ARCHS[@]}"

   echo "Replacing original executable with thinned version"
   rm "$FRAMEWORK_EXECUTABLE_PATH"
   mv "$FRAMEWORK_EXECUTABLE_PATH-merged" "$FRAMEWORK_EXECUTABLE_PATH"
done
```

RECOGNIZE TO JSON

- Create an instance of class **RecognitionWrapper**
- Call method **'recognizeToJSON' or 'recognizeToJSONFromPictureBuffer'**
- The method returns pointer to instance of **NSString.** It is **'nil'** if recognizing failed. If recognition is successful, it contains **JSON** with the objects documented in the section "OCR ENGINE OUTPUT."

LICENSING

- Use static instance of class License
- Use method '**getPublicKeyFromServer:completion**' to get server public key. Completion contains **public key** (optional) **and error** (optional)
- Use method '**sendLicenseRequest:serverPublicKey:apiKey:licenseApiKey:completion**' to get information about license. Completion contains instance of class **LicenseInfo**(optional) and **error** (optional)
- Or simply use method 'getLicenseInfo(apiKey:licenseApiKey:completion' to get information about license. Completion contains instance of class LicenseInfo(optional) and error (optional)

License checking errors:

- **invalidServerPublicKey**: Invalid server public key
- **invalidJSONResponse**: Invalid JSON response
- **invalidResponse**: Invalid response from server
- **invalidURL**: Invalid server URL
- **invalidDeviceId**: Error while getting device ID
- **timeOut**: Request timeout.
- **invalidAPIKey**: Invalid API key

Example of usage

```
let recognizer = RecognitionWrapper()

// Recognize to JSON from Image
let json = try? recognizer.recognize(toJSON: image)

//Recognize from image buffer to JSON
if let imageData = UIImagePNGRepresentation(image) {
let json = try? recognizer.recognizeToJSON(fromPictureBuffer: imageData)
}
```

OCR ENGINE OUTPUT

Below is a description of JSON response from the OCR engine

Field	Type and description
license_api_state	*Integer. Contains code of error*
license_api_message	*String. Contains message of error*
license	*String. Missing*
result	*String. Missing*
engine_version	*String. Contains version number of OCR Engine*
processing_time_ms	*integer.* Total time in milliseconds OCR engine used on recognition.
recognition_type	*String.* BIC of ILU. Detected container code standard. BIC (ISO 6346) or ILU (EN 13044-1)
result	*String.* The recognized container code ([A-Z]{3}\[ABDEJKZU]) with the highest confidence factor.
checksum	*String.* "1" = The checksum calculation of recognized characters is correct. "0" = The checksum calculation of recognized characters is incorrect. "-1" = Cannot do the checksum calculation due to missing characters.
iso_size_type	*String.* If recognition_type is BIC, then the value will be the recognized ISO Size Type (4 alphanumerics) https://en.wikipedia.org/wiki/ISO_6346#Size_and_Type_Codes
confidence	*String.* This value indicates the total confidence factor of the recognition. A value of 84 and above is the same as 95% accuracy of recognition. Values below 84 indicate lower accuracy than 95%.
confidence_characters	This section gives information about the recognized character and the confidence factor of each.
character[1-15]	*String.* Reference to the identified character.
confidence	*String.* Confidence factor of the recognized character.
recognition_area	This section gives information about the area of detection.
ul_x	*Integer.* Upper left X coordinate
ul_y	*Integer.* Upper left Y coordinate

lr_x	*Integer.* Lower right X coordinate
lr_y	*Integer.* Lower right Y coordinate
recognition_area_characters	This section has a subsection for each recognized character.
character[1-15]	There will be one section for each recognized character. Each section gives information about the area of recognition.
ul_x	*Integer.* Upper left X coordinate
ul_y	*Integer.* Upper left Y coordinate
lr_x	*Integer.* Lower right X coordinate
lr_y	*Integer.* Lower right Y coordinate
candidates	A set of subsections that holds information about additional candidates ordered by confidence factor: each section has the following fields.
recognition_type	*String.* BIC of ILU.
Result	*String.* [A-Z]{3}\[ABDEJKZU]
checksum	*String.* "1" = The checksum calculation of recognized characters is correct. "0" = The checksum calculation of recognized characters is incorrect. "-1" = Cannot do the checksum calculation due to missing characters.
iso_size_type	*String.*
confidence	*Integer.* Confidence factor of recognition. As this is a candidate, this will always be lower than the main result.
image_metadata	**This section will output some image metadata.**
image_resolution	It contains information about the resolution of the processed image.
width	*Integer.* The width of the image
height	*Integer.* The height of the image.

LICENSE ACTIVATION

ConPDS OCR engine supports license activation for both Internet (online) and non-Internet (offline) connected devices.

Online activation (Internet-connected activation)

For online activation, there should be access to the ConPDS licensing server (https://licensing.conpds.com) via the Internet (TCP port 443). Once a license is issued and activated, there is no longer a requirement for connection to the Internet unless a new API or License key has to be applied or a Trial License has expired and needs reactivation.

Offline activation (Non-Internet connected activation)

Send an email to support@conpds.com and request login to the ConPDS licensing server. Once granted access, you should log in to https://licensing.conpds.com and download "Server.key" which is found in the drop-down in the upper right corner. Put the "Server.key" file inside the same folder as your executable file (*.exe)

Run recognition. If no internet connection, the library will create "manualRequest" file inside the same folder as your executable file. Log in to http://licensing.conpds.com with username/password select "License request" in the left side panel. Click the "Manual request" link, upload the license request file and click "Create" and then an offline "License" file is generated. Put this "License" file into the same folder as your executable file (*.exe). The device is licensed and can start recognition.

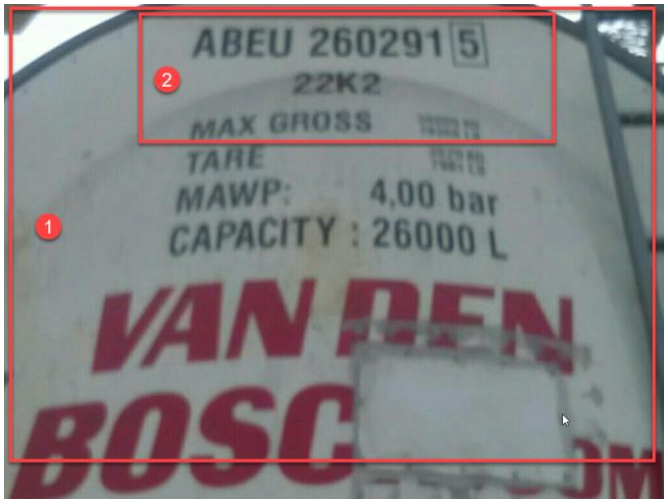
In case of any questions to the above-described procedure, always feel free to contact us. We are here to assist you promptly.

UNDERSTAND OCR PROCESSING AND HOW TO IMPROVE RESULTS

Example 1: Bad or no recognition

Reason:
(1) A poor image capture that can't be fixed by image preprocessing.
(2) The detection phase will try to identify patterns in an incorrect area and will result in failed recognition.

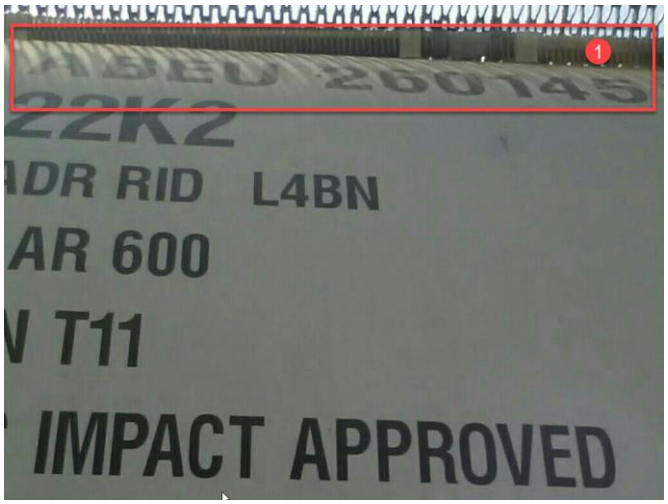
Tip: Better image capture.



Example 2: Much text and details

What happens:
(1) Image capture contains a big area of text, but preprocessing and detection of well-known structure and pattern of container code, recognition will detect the correct part of the image …
(2) … and return the correct recognition result.

Tip: In this example, recognition is successful, but to avoid incorrect recognition, improve image capture by focussing on the area with the container code and avoid the area with useless content.



Example 3: Much test and image angle out of range

Reason:

(1) Image capture contains much text and does not include a complete and visible container code due to an unsupported angle. The preprocessing of the image will not be able to improve to include full container code, and the detection phase will not find a well-known structure and pattern of container code. The result is incorrect or no recognition result.

Tip: Change the angle of the camera, and make sure a complete container code is present.

Example 4: Container code incomplete

Situation:

(1) Characters of the container code are scratched. Image preprocessing will improve the image, and in this example, recognition will return the correct container code as characteristics of the scratched character can be recognized.

Tip: If characters are more scratched or even completely missing, then take an image of one of the other container codes located on the unit.

Example 5: Container code is changed

Situation:
(1) The container code is changed, and the previous container code is still visible. Image preprocessing will enhance image and recognition will return the correct container code.

Tip: Same as in example 4.

Example 6: Incomplete image capture

Situation:
(1) Bad image capture as a complete container code is not present. Image preprocessing can improve image quality, but detection will fail as well known structure of container code is not present.

Tip: Improve image capture. A successful recognition requires an image with a complete container code.

Example 7: Incomplete image capture

Situation:

(1) Image capture does not contain a complete container code. The detection phase will find the well-known structure of the container code. In this example, the OCR engine will return the correct result as the characteristics of the incomplete check digit character is most likely the character "7".

Tip: Improve image capture, so images always contain complete container code. In current example recognition was successful, but could have been incorrect if check digit character had the same characteristics as the characters "0" or "6".

.

Example 8: Container code characters are scratched or corroded

Situation:

(1) Quality of characters are bad, but image preprocessing will improve image quality, and recognition will return the correct result.

Tip: Improve image capture by taking an image of one of the other container codes located on the unit.

Example 9: Image out of focus

Situation:

(1) Image capture is out of focus. Image preprocessing will improve the quality, and the detection part will find a well-structured container code. Recognition will return the correct result.

Tip: Better image capture so that initial image quality is better.

SUPPORT SERVICES

Support

If you are experiencing difficulties with our products, you can contact ConPDS Support directly via support@conpds.com.

Maintenance

Current maintenance unlocks access to Technical Support, Product Updates, and Upgrades. Maintenance is extended automatically at the end of each 12 months, unless you, as a customer specifies otherwise.

Product Upgrades and Service Releases

Product Upgrades and Service Releases – To upgrade your version of the ConPDS OCR engine to the latest available version, you can download from the download link provided when purchasing the product. If you no longer have the link, please contact ConPDS Support.

CONTACT

Address

Vestervang 21
8700 Horsens
Denmark

Telephone

+45 6040 7000

Email

info@conpds.com

VAT no.

DK35828192