“Dynamic Vision App for HALCON”

A few steps from HALCON image processing to the IDS NXT vision app

With the "Dynamic Vision App for HALCON" you can easily execute your image processing tasks developed with HALCON on an IDS NXT camera. Time-consuming vision app development in C++ is no longer necessary for HALCON users! It’s made possible by the "Dynamic Vision App for HALCON". The perfect complement for your image processing development.

With this tech tip you will learn how to create a suitable HALCON library in HDevelop in just a few steps and how to simply upload it to an IDS NXT vegas via the IDS NXT Cockpit and execute it without writing a line of C++ code. The simple image processing example "*example_code2d.hdpl*" searches for Data Matrix codes in the captured image data and marks them in the result images of the vision app. In addition, the code contents are output as text. The IDS NXT Framework, developed by IDS, gives you complete freedom through custom development of vision apps for what you run on vision app-based cameras. You will be supported by the IDS NXT Vision App Development Kit in the development and execution of your individual vision app, which only requires C++ knowledge. However, things are much easier with the newly developed "Dynamic Vision App for HALCON". The perfect complement for your image processing development. In case you only want to execute image processing with the already pre-installed HALCON, this generic app takes over the time-consuming creation of vision apps in C++ completely. It not only executes HALCON image processing written in HDevelop, but also provides the defined input and output parameters via the REST interface. The "Dynamic Vision App for HALCON" also displays these parameters in the IDS NXT Cockpit, allowing you to conveniently configure and check your image processing.

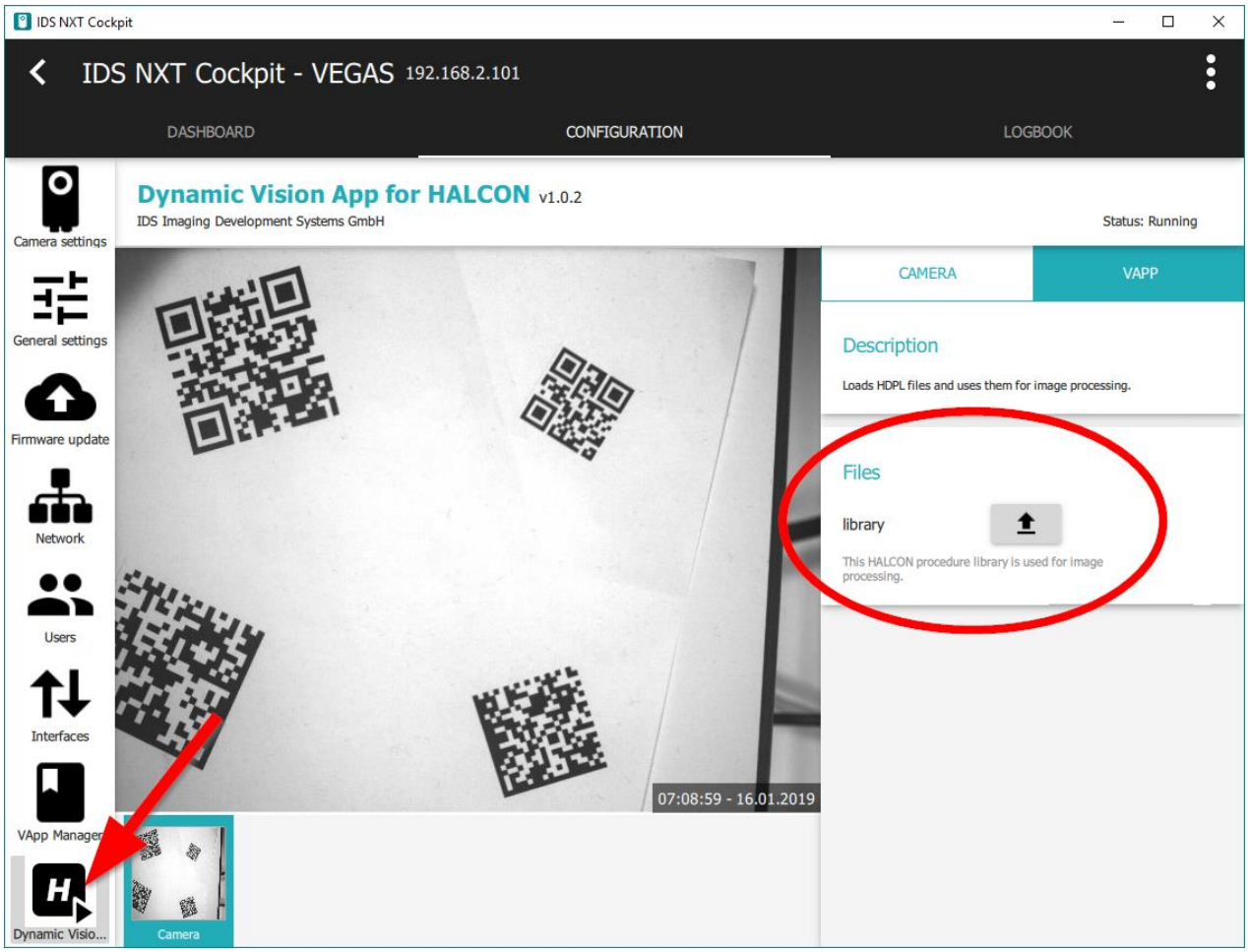


Figure 1 Direct execution of HALCON libraries.

Prerequisites

As preparation for the procedure described here and for a better understanding of the IDS NXT concept, the manual "IDS NXT - Development of vision apps" together with the IDS NXT video tutorials are recommended. To follow the tech tip you need

- IDS NXT vegas (Firmware as of version 1.3)
- Windows PC (with network connection to IDS NXT vegas)
    - Licensed HALCON 13 development environment
    - IDS NXT Cockpit (Firmware as of version 1.2.4)
- "Dynamic Vision App for HALCON"

Install and enable the "Dynamic Vision App for HALCON" on the IDS NXT vegas via the IDS NXT Cockpit. How to install a vision app is described in chapter "Vision App Manager" in the manual "IDS NXT - Operation" .

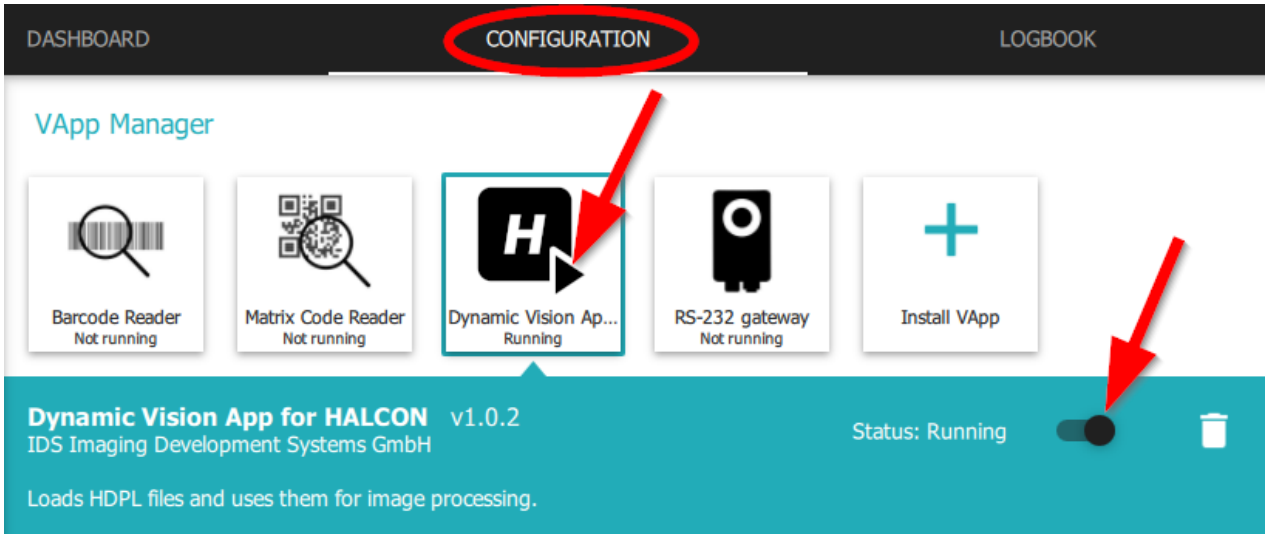


Figure 2 Start the "Dynamic Vision App for HALCON" in the VApp Manager

If the vision app has been started successfully, its configuration dialog appears in the settings area of the IDS NXT Cockpit. Upload compatible HALCON libraries, the development of which is explained in the following.

⚠ The installation and deinstallation of IDS NXT Vision Apps can only be done by the administrator user!

HALCON Development

The "Dynamic Vision App for HALCON" establishes the connection between a "HDevelop procedure library" (HDPL) and the IDS NXT framework in a Vision app-based camera of the IDS NXT family. Control parameters and image processing results are thus made available to the camera REST interface and the IDS NXT Cockpit. In this way, image processing can be operated directly like a complete vision app.

To ensure the compatibility of the HALCON library with the dynamic vision app, HALCON procedures with special names must exist, whose input and output parameters (procedure interface) are exactly defined (semantic types). Only then the "Dynamic Vision App for HALCON" can automatically select the appropriate configuration or result variables that will be provided via the REST interface.

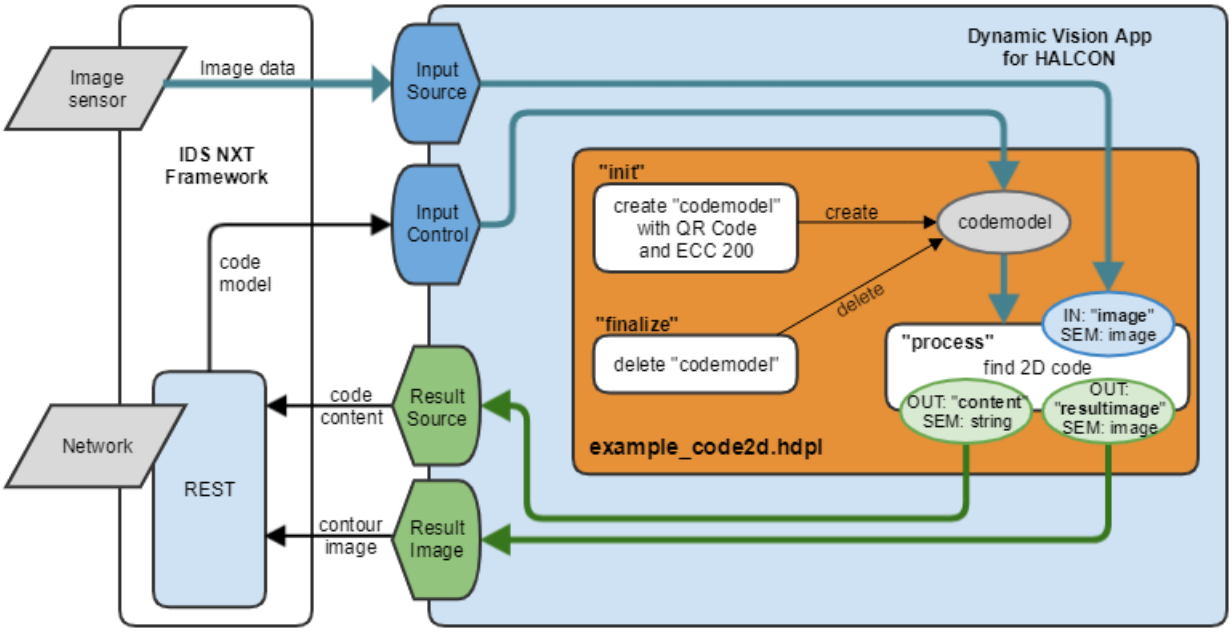


Figure 3 The "Dynamic Vision App for HALCON" establishes the connection between HALCON and the IDS NXT Framework.

HALCON library structure

For the communication with the dynamic vision app the HALCON image processing requires the following HDPL procedures as entry points:

- "**init**" - is executed when the library is loaded. InputCtrlParams (variables, control parameters) and OutputCtrlParams (displays, results) are registered here.
- "**finalize**" - is called before the library is unloaded and cleans up all data structures created by init.
- "**train**" - allows a training of the HALCON image processing. This procedure is optional. If available, the corresponding training is performed.
- "**process**" - is the processing function of the actual image processing.

Creating Procedures

Create the required procedures of type "Library procedure (.hdpl)" in HDevelop and save them in the same library. For each **procedure**, specify the required **input** or **output parameters** (iconic and controlling) and their parameter documentation (**semantics**).

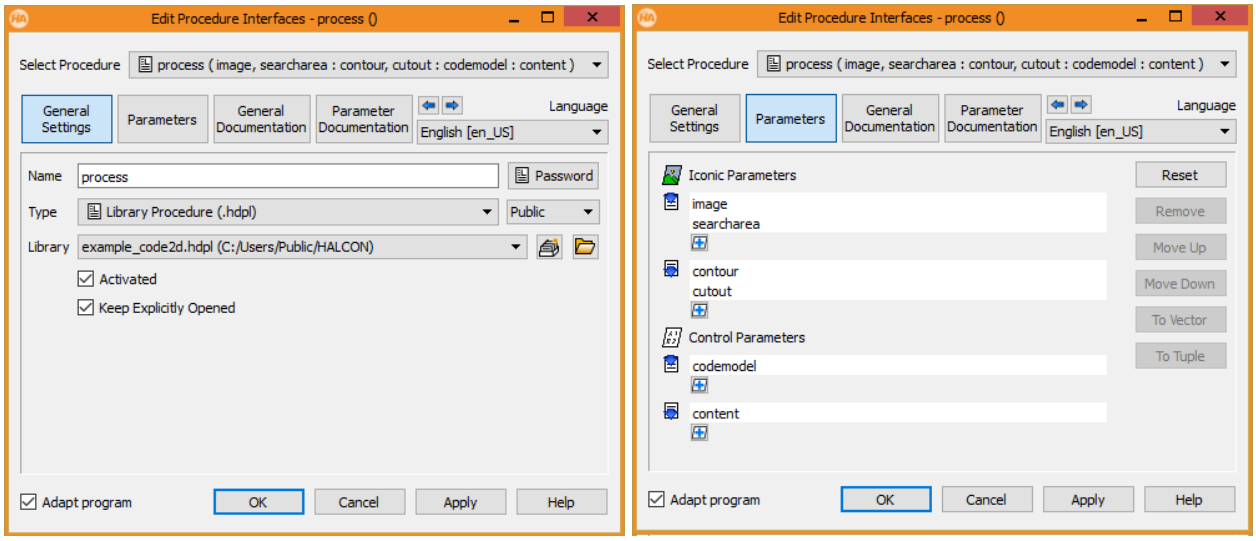


Figure 4 Fully defined procedure interfaces with parameter documentation (semantics) are crucial.

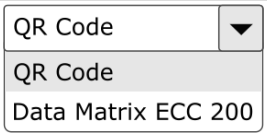
Important! The correct "semantic types" must be assigned to the input and output parameters of the HALCON procedures in the tab "Parameter Documentation"!

Create the following procedure parameters for the image processing example of this tech tip (*example_code2d.hdpl*):

Procedure	Parameter	Type	Semantic
init	codetype	Input: Control parameter	string (Value: QR Code (Default), Data Matrix ECC 200
	codemodel	Output: Control parameter	datacode_2d
process	image	Input: Iconic parameter	image
	searcharea	Input: Iconic parameter	region
	codemodel	Input: Control parameter	datacode_2d
	contour	Output: Iconic parameter	xld
	cutout	Output: Iconic parameter	image
	content	Output: Control parameter	string
finalize	codemodel	Input: Control parameter	datacode_2d
train	image	Input: Iconic parameter	image
	firstarea	Input: Iconic parameter	image
	codemodel	Input: Control parameter	datacode_2d
	content	Output: Control parameter	string
	sizemin	Output: Control parameter	integer
	sizemax	Output: Control parameter	integer

Parameter semantics

HALCON parameter semantics is particularly important. This defines which parameter types are transferred or queried by the dynamic vision app to the individual HALCON procedures. According to the semantics of a HALCON parameter, the vision app dynamically generates a suitable interaction parameter for the REST interface. The following interaction parameters are generated in the example:

HALCON procedure parameters	IDS NXT Interaction parameters	
codetype		Combo Box Ctrl
image		image
contour		Image with marker
content	**Code content:** IDS-Techtip	label
searcharea		Adjustable region (image area)

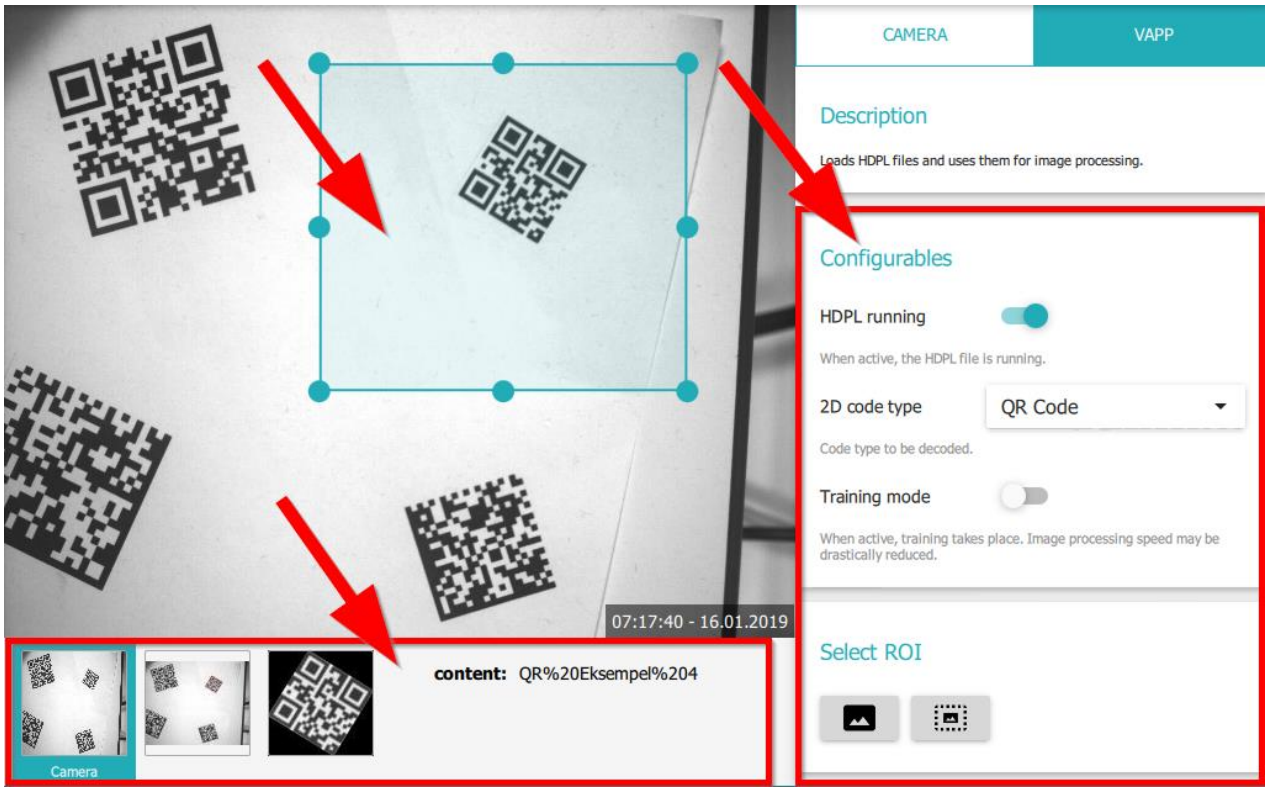


Figure 5 Interaction parameters are displayed as graphical elements of the vision app.

Image processing

> This tech tip only describes the exchange between the "Dynamic Vision App for HALCON" and the HALCON library using the procedure interfaces.

The " init " procedure control parameter "codetype" is created to select the code type and a 2D code model "codemodel" is created, which is also used as input parameter in other procedures. Furthermore, a file with training data is loaded or created in order to generate a more detailed code model.

```
if (FileExistsTraining)
    read_data_code_2d_model(TrainingFilePath, codemodel)

[...]
else
 create_data_code_2d_model(codetype, [], [], codemodel)
```

In the "process" procedure, the image data are first reduced with help of the "searcharea" set by the user. Then HALCON searches for 2D codes in this region using the (trained) "code model" and transfers the results in the "content" string and their image position as XLD in the "contour" image. The "cutout" image shows the cut out code.

```
find_data_code_2d (image, contour, codemodel, [], [], ResultHandles, content)
[...]
reduce_domain(image, Region, cutout)
```

Within the "train" procedure, the code model is further improved and the training data is saved in a file.

```
find_data_code_2d(ImageReduced, contour, codemodel, ['train'], ['all'],
ResultHandles, content)
[...]
write_data_code_2d_model(codemodel, TrainingFilePath)
```

The "finalize" procedure cleans up the memory by deleting the "codemodel".

```
clear_data_code_2d_model(codemodel)
```

Upload HALCON Library

The created HALCON library can now be used. Use the "Upload" button ( ) of the dynamic vision app to upload the previously created HALCON library. A message informs you when the library has been loaded successfully. In this case, the dynamic vision app displays the input and output parameters of the HALCON library in the IDS NXT Cockpit. The image processing can be used immediately.

The HALCON library remains permanently in memory even after the next restart, until it is removed from the system via the "Delete" button ( ) or overwritten by the upload of another library. Only one library can be loaded and active at a time. However, you can change the image processing as often as you like.

Hints

Performance

The combination of the generic vision app and a HALCON library is not speed optimized and can therefore not be compared to the programming of a specially implemented C++ vision app. The tech tip *example_code2d.hdpl* reaches about 70% of the speed of the specialized matrix code reader vision app. The "Dynamic Vision App for HALCON" offers the advantage that HALCON image processing can be tested very quickly and easily on a vision app-based camera without having to write one line of C++ source code and cross compile it for the camera platform.

Exception Handling

If an unhandled exception occurs during the execution of a HALCON procedure, the vision app restarts and unloads the HALCON library. To avoid a crash, make sure that every occurring error is handled within the HALCON procedures (exception handling with try...catch).

Tips

Avoid restarting

If the vision app crashes during the development of the HALCON image processing library, it would make sense to avoid automatically restarting it again and again. You can prevent this with a REST command. Then you decide manually when the app should start.

```
curl -u <user credentials> -X PUT
/vapps/activated/dynamic_vision_app_for_halcon?temporary=true <device URL>
```

> This REST command is only valid once for the current execution of the vision app!

Show log

To show a detailed log of the "Dynamic Vision App for HALCON", e.g. to track down a malfunction of the vision app or the image processing, call the following URL in the browser:

```
http://<Geräte IP>/vapps/dynamic_vision_app_for_halcon/log
```

Figure 6 This log shows an error-free process when loading a HALCON library.

The "Dynamic Vision App for HALCON" analyzes the procedure parameters defined in the library and creates the required vision app interaction elements (see yellow markers), such as configurables, inputs and results. Execution errors in HALCON image processing can be identified and corrected more easily with this application log.

By using the standard output file, HALCON messages can also be integrated into this log.

```
open_file('standard', 'output', stdout)
result := 'NIO'
fwrite_string (stdout, result + '\n')
```

Summary

The "Dynamic Vision App for HALCON" provides a fast and easy way to run HALCON-based image processing on a Vision App-based camera such as the IDS NXT vegas. This saves you the time-consuming C++ programming of your own vision app, allowing you to concentrate on creating the HALCON routines. The simple upload mechanism enables you to change the purpose of the IDS NXT Vision app-based camera as often as you like during runtime. You do not need to be familiar with C++ or vision app programming. Embedded image processing "it's so easy".

If you want to learn more about the IDS NXT vision app concept and how to use it, read the IDS NXT vision app development guide and watch the IDS NXT video tutorials. An introduction to the simple and intuitive programming of your own HALCON machine vision application can be found on the HALCON product pages of MVTec.

Further tech tips and application reports can be found on the IDS website. There you can also find more information about the new vision app-based industrial cameras of the IDS NXT family.

Author

Dipl.-Ing. (FH) Heiko Seitz
Technical Writer

IDS Imaging Development Systems GmbH
Dimbacher Str. 6-8
74182 Obersulm
Germany

T: +49 7134 96196-0
E: h.seitz@ids-imaging.de
W: www.ids-imaging.com