HALCON
a product of MVTec

HALCON for Arm-based Platforms

HALCON 24.05 *Progress*

MVTec Software GmbH

Building Vision for Business

All about using HALCON on Arm®-based platforms, Version 24.05.0.0

More information about HALCON can be found at: http://www.halcon.com

About This Manual

This manual provides supplementary information about using HALCON on Arm®-based platforms.

Chapter 1 briefly describes the specifics for these platforms and the available development methods.

Chapter 2 briefly explains how to install HALCON on the development PC and on the Arm-based platform.

Chapter 3 explains how to develop applications in HDevelop or the programming languages C, C++, C#, Python and HDevEngine.

An up-to-date list of supported platforms can be found on our website http://www.mvtec.com/products/embedded-vision/all-products.

Contents

Chapter 1

Using HALCON on Arm-based Platforms

This chapter provides a brief overview on the specifics of using HALCON on Arm-based platforms. At the end of the chapter you can find a list of other manuals that help you to work with HALCON.

1.1 Requirements

The system requirements can be found in the Installation Guide, section 1.4 on page 8.

For graphical applications, X11 support is required.

The licensing on Arm-based platforms is either based on a product license dongle or a Card IDentification (CID) number of a SD card or other memory cards. In order to access a dongle, the `hidraw` device must exist, i.e., it must be enabled in the kernel configuration. If necessary, a custom kernel needs to be compiled using the kernel option `CONFIG_HIDRAW=y`. Naturally, a USB port is required to connect the dongle. In case of a card identification number, hardware support for the memory device is required. You can query the CID numbers provided by your system by calling the HALCON tool `hhostid -i` from your shell.

1.2 Limitations

The following limitations apply compared to other platforms:

- no XL version
- no MAC-based licensing, no evaluation licenses
- no HDevelop
- no hcomp, thus extension packages have to be cross-compiled on the development PC

An up-to-date list of supported image acquisition interfaces can be found on our website `http://www.mvtec.com/products/interfaces`.

1.3 Differences to Other Platforms

The main difference in developing applications with HALCON for Arm-based platforms compared to other platforms is that you usually do not do this on the Arm-based platform itself, but on a separate development PC, using a cross compiler toolchain to generate the executable. Then you upload the executable to the Arm-based platform and run it there.

HALCON development tools like HDevelop are not available for Arm-based platforms, so you actually need two installations of HALCON:

On the development PC:

- a development version of HALCON

On the Arm-based platform:

- a runtime installation of HALCON

In (the rare) case you have a native toolchain for your Arm-based platform it is of course possible to use it directly with HALCON. But even in this case you need to own a HALCON development license, and the HALCON development tools are not available on your Arm-based platform.

Example programs specific to the Arm-based platform can be found in `examples/arm-linux`. Please note that other example programs can only be compiled if the corresponding makefiles or CMakeLists.txt are adjusted.

In the following we will always assume you are using a cross compiler toolchain as this is the most common way at the time of writing.

1.4 How to Develop HALCON Applications for Arm-based Platforms

There are different ways to develop HALCON applications for Arm-based platforms: The more comfortable way is to develop the machine vision part using HDevelop, HALCON's interactive development environment. Here you can easily find the suitable operators and parameters to solve your vision task. To integrate the machine vision part into your application, you have to either use one of the programming languages or HDevEngine.

Of course, you can also develop the complete application in your favorite programming language.

Unless you use HDevEngine, a cross compiler is needed to create an executable. Finally, you upload it to the Arm-based platform. These development methods are described in more detail in chapter 3 on page 11.

1.5 Where to Find Further Information

This manual concentrates on specific topics for using HALCON on Arm-based platforms. The list below shows you where to find information about other aspects that apply to HALCON in general.

- Installation Guide
  If you have not used HALCON before, this manual shows you how to install HALCON, which is the first step to using HALCON.
- Quick Guide
  This manual is your gate into the world of HALCON. It gives a short introduction to HALCON and presents HALCON example programs from various industries and application areas.
- Solution Guide I
  This manual describes the main machine vision methods and how to use them in HALCON, including many examples.
- Solution Guide II + III
  These manuals consist of multiple Solution Guides, which supply more detailed information about specific machine vision methods.
- HDevelop User's Guide
  This manual is your comprehensive guide to HDevelop, HALCON's integrated development environment.
- Programmer's Guide
  This manual shows you how to use HALCON from a programming language. It contains detailed information about the provided interfaces and their data types, classes, etc., and also explains how to use HDevEngine to directly run HDevelop scripts and procedures from applications written in a programming language.
- HALCON Operator Reference
  The reference manual contains the detailed description of all HALCON operators.

Chapter 2

Getting Started

This chapter guides you through your first steps with HALCON for Arm-based platforms, mentioning the particularities concerning the steps

- installing the necessary components (section 2.1),
- licensing (section 2.2)

Please note that throughout this manual we assume that you already know how to access the Arm-based platform from your development PC (see the corresponding documentation from the platform's vendor for details).

2.1 Installing the Necessary Components

On your development PC, you must install the following components:

- A development version of HALCON including a license.
- The HALCON libraries compiled for the target architecture (aarch64-linux or armv7a-linux).
- Compiler, cross compiler for the target architecture for aarch64-linux and armv7a-linux: gcc 5.5.

2.1.1 Installing HALCON on the Development PC Running Under Linux

On the development PC, a HALCON installation (with component selection "Full" or "Custom") is required. The installation procedure is described in the Installation Guide, section 2.2.1 on page 15. The installation script supports the installation of the required files for cross-compilation.

To test whether the installation and licensing was successful, we recommend to start HDevelop. The menu item `Help` ▷ `About` shows which HALCON version is installed, thus, you can check whether you successfully installed HALCON 24.05.0.0.

The following directories on the development PC will contain content specific for HALCON on Arm-based platforms:

- `doc/pdf/manuals`: this manual
- `lib/aarch64-linux`: the HALCON libraries for platforms based on AArch64
- `lib/armv7a-linux`: the HALCON libraries for platforms based on Armv7-A
- `examples/arm-linux`: the examples for HALCON for Arm-based platforms

2.1.2 Installing HALCON on the Arm-based Platform

The runtime installer for Arm-based platforms is available as part of the full installation archive (in the subdirectory `misc/linux`) or as a separate download. The installation process is the same as with the full version: Extract the archive on the Arm-based platform and run the installation script `install-linux.sh`. Again, see the Installation Guide, section 2.2.1.2 on page 16 for more information on the installation procedure.

In order to use HALCON on the Arm-based platform, the following environment variables must be set in order for HALCON to work:

- `HALCONROOT=`<directory you installed HALCON in, e.g., /opt/halcon>
- `HALCONARCH=aarch64-linux` or `HALCONARCH=armv7a-linux`
- `LD_LIBRARY_PATH=$LD_LIBRARY_PATH:${HALCONROOT}/lib/${HALCONARCH}`
- `PATH`: this system variable should include `$HALCONROOT/bin/$HALCONARCH`.

2.2 Licensing

To use HALCON for Arm-based platforms you need two licenses:

- For the development version on the development PC you need a HALCON development license (see Installation Guide, section 5.3 on page 27).
- For HALCON on the Arm-based platform, you need a HALCON run-time license (see Installation Guide, section 5.4 on page 28). If you choose to use a CID bound license, you can query the available CID id nodes by calling `hhostid -i`.

2.2.1 Installing the License on the Arm-based Platform

If you installed HALCON for Arm-based platforms yourself and obtained the license file separately from your distributor, you must rename it to `license.dat` (or similar, see the Installation Guide, section 1.6 on page 14 for the naming conventions) and place it in the subdirectory `license` of the directory where you installed HALCON on the Arm-based platform (`$HALCONROOT`).

2.3 Uninstalling HALCON

HALCON provides no uninstallation script for Linux systems, therefore you must perform the uninstallation manually. Please see the Installation Guide, section 3.2 on page 22 for the specific steps.

Chapter 3

Creating Applications

This chapter explains how to create applications for Arm-based platforms using HALCON.

There are the following ways of developing:

- **developing in HDevelop**, exporting the script or procedures to C, C++ or C#, and including the code in your application (see section 3.2 on page 14)
- **using HDevEngine** (see section 3.7 on page 15)
- **using HALCON/.NET** (see section 3.5 on page 15)
- **programming in Python** (see section 3.6 on page 15)
- **programming in C or C++** (see section 3.4 on page 14, section 3.3 on page 14)

Note that you develop the application on the development PC, not on the Arm-based platform, using the gcc compiler to create the executable.

3.1 Configuration Files and Directory Structure

The directory `$HALCONROOT/examples/arm-linux` on your development PC contains sample code to build a simple HALCON application using C and C++. Also, sample code for a program that can execute HDevelop scripts using HDevEngine is included.

In order to build the samples, two options are at your disposal.

3.1.1 Cross-Compile

In this section we describe the steps necessary to compile an example on a development PC in order to run on Arm-based platforms ('cross-compilation'). We will do this with the aid of the datacode example, for which also the exported C and C++ code is included in the example directory.

We assume HALCON is installed on your development PC (and the aarch64-linux or armv7a-linux libraries have been installed during the installation).

First the steps needed on your development PC. They are done the easiest under a Debian or Ubuntu distribution and in the following we will assume you are using such a system.

1. Make sure you have the cross-compilation toolchain installed. If this is not the case, you can install it via the package streams:

   For aarch64-linux:

   ```
   sudo apt-get install gcc-aarch64-linux-gnu
   sudo apt-get install g++-aarch64-linux-gnu
   ```

For armv7a-linux:

```
sudo apt-get install gcc-arm-linux-gnueabihf
sudo apt-get install g++-arm-linux-gnueabihf
```

2. Change to your HALCON installation directory.
3. Set the HALCON environment variables, e.g., by typing

```
source .profile_halcon
```

4. Specify the HALCON target architecture (your Arm-based platform). For this, you have two options:
   (a) Re-set HALCONARCH to the one of your Arm-based platform by typing

```
export HALCONARCH=aarch64-linux
```

   or

```
export HALCONARCH=armv7a-linux
```

   (b) Provide the HALCONARCH an argument `HALCON_ARCHITECTURE` to CMake.
5. Change to the application directory, in our case `$HALCONROOT/examples/arm-linux`.
6. Create a new build directory for your compiled example and move into it.

```
mkdir build
cd build
```

7. Specify a toolchain to CMake. For the HALCON architectures armv7a-linux and aarch64-linux, you can find example toolchain files in `$HALCONEXAMPLES/cmake/toolchains/`
8. Compile the application by typing:
   For aarch64-linux:

```
cmake .. \
  -DHALCON_ARCHITECTURE=aarch64-linux \
  -DCMAKE_TOOLCHAIN_FILE=../../cmake/toolchains/cross-aarch64-linux-gnu.cmake
cmake --build .
```

   For armv7a-linux:

```
cmake .. \
  -DHALCON_ARCHITECTURE=armv7a-linux \
  -DCMAKE_TOOLCHAIN_FILE=../../examples/cmake/toolchains/cross-arm-linux-gnueabihf.cmake
cmake --build .
```

9. Copy the compiled files onto your Arm-based platform (the files are in `./bin/aarch64-linux` or `./bin/armv7a-linux/`).
10. The datacode example uses images. Therefore, the image sequence file `$HALCONROOT/examples/images/datacode/ecc200/ecc200_cpu.seq` and all images referenced by it must be in a location readable on the Arm-based platform. The easiest way is to copy the directory `$HALCONROOT/examples/images/datacode/ecc200` to your Arm-based platform.

Now, on your Arm-based platform:

1. Change to your runtime installation directory.
2. Set the necessary HALCON environment variables `HALCONROOT`, `LD_LIBRARY_PATH` (e.g., by `source .profile_halcon`), and if images are used, `HALCONIMAGES`.
3. Execute the program which you have copied from the development PC, e.g., `datacode_cpp`.

3.1.2 Compile and Run on a Native Machine

In this subsection we describe how you can compile the exported C and C++ code using the compiler of your Arm-based platform. We will do this with the aid of the datacode example.

For this, you will need to install HALCON differently on your Arm-based platform:

1. Install HALCON using the MVTec Software Manager on your Arm-based platform. Select at least the components Runtime files and Runtime files (general).

   Be aware, such an installation does not contain header files, CMake scripts, and example source code. They have to be copied from a full HALCON installation.

2. Install further required components. On your Linux PC, create an archive of the required directories:

   ```
   cd $HALCONROOT
   tar -cf dev-files.tar include/examples/cmake examples/arm-linux
   ```

   Copy this archive to your Arm-based platform and extract the archive in your HALCON installation:

   ```
   cd $HALCONROOT
   tar -xf dev-files.tar
   ```

Now the Arm-based platform is set up and we can compile the code. Therefore, execute the following steps on your Arm-based platform:

1. Change to the directory `$HALCONROOT/examples/arm-linux`.
2. Set a required environment variable:

   ```
   export HALCONEXAMPLES=$HALCONROOT/examples
   ```

3. Create a new build directory for your compiled example and move into it.

   ```
   mkdir build
   cd build
   ```

4. Compile the application by typing:

   ```
   cmake ..
   cmake --build .
   ```

   Note that for native compilation it is not necessary to provide CMake arguments for the HALCON architecture and the toolchain.

   To ensure that `make` does not attempt to run HDevelop, the date of the C file `datacode.c` and the C++ file `datacode_cpp.cpp` must be newer than the corresponding HDevelop file `datacode.hdev`.

In order to execute the program, carry out the following steps:

1. Set the environment variable `HALCONIMAGES` appropriately.
2. Execute the program, e.g., `datacode_cpp`.
3. You can also execute the example with `hrun`.

3.2 Developing in HDevelop

If you use HDevelop to develop your application, you have to use HDevelop on your development PC. For your final application, you then integrate your HDevelop script into a C, C++, .NET Core, or Python application.

For this, there are two alternatives:

- You can **export** the complete script or individual procedures to C, C++, or C# and integrate the relevant parts of the exported code. To export an HDevelop script or procedure, open the menu item `File` ▷ `Export` and then select the C, C++, or C# language in the combo box. Detailed information can be found in the HDevelop User's Guide. Note that exporting a library project or entire HDevelop program to the Python programming language is currently not supported.
- You can call the complete script or individual procedures from a C++, C#, or Python application using HDevEngine (see section 3.7, Programmer's Guide, section 23.2.1 on page 155, or Programmer's Guide, section 24.1.1 on page 179, respectively).

Both methods have their advantages and disadvantages. Using HDevEngine is more comfortable, especially if you develop the complete image processing part in HDevelop and keep the interface to the rest of the application as small as possible. As a second advantage, you can modify the HDevelop script without needing to re-compile and link the application (if you don't change the interface). However, the application may run slightly slower because the HDevelop script is interpreted by HDevEngine.

For a quick start with HDevelop, please refer to the Quick Guide. For detailed information, see the HDevelop User's Guide.

3.3 Creating C++ Applications

Detailed information about HALCON's C++ interface can be found in the Programmer's Guide. To summarize the necessary extensions to the normal application development on your Arm-based platform, you

- include the main header file in your application and add the namespace:

```
#include "HalconCpp.h"
using namespace HalconCpp;
```

- compile the application with the include paths

```
-I$(HALCONROOT)/include -I$(HALCONROOT)/include/halconcpp
```

- and link the HALCON libraries for the aarch64-linux or armv7a-linux architecture

```
-L$(HALCONROOT)/lib/$(HALCONARCH) -lhalconcpp -lhalcon
```

3.4 Creating C Applications

Detailed information about HALCON's C interface can be found in the Programmer's Guide. To summarize the necessary extensions to the normal application development on your Arm-based platform, you

- include the main header file in your application:

```
#include "HalconC.h"
```

- compile the application with the include paths

```
-I$(HALCONROOT)/include
```

- and link the HALCON libraries for the aarch64-linux or armv7a-linux architecture

```
-L$(HALCONROOT)/lib/$(HALCONARCH) -lhalconc -lhalcon
```

3.5 Creating C# Applications

Detailed information about HALCON/.NET interface can be found in the Programmer's Guide. Furthermore, make sure that you have installed the .NET Core SDK for your system architecture. To summarize the necessary extensions to the normal application development on your Arm-based platform, you

- add the namespace:

```
using HalconDotNet;
```

- and add the `MVTec.HalconDotNet` NuGet package to your application:

```
dotnet add package MVTec.HalconDotNet -v 24050
```

How to create a simple HALCON application with .NET Core is demonstrated in Programmer's Guide, section 9.1 on page 62.

3.6 Creating Python Applications

Detailed information about HALCON's Python interface can be found in the Programmer's Guide. To try out the Python interface, make sure that the CPython implementation of the Python programming language is installed on the system. To summarize the necessary extensions to the normal application development on your Arm-based platform, you

- add HALCON/Python to your project:

```
pip install mvtec-halcon==24050
```

- import the HALCON/Python module:

```
import halcon as ha
```

How to create a simple HALCON application with HALCON/Python is demonstrated in Programmer's Guide, chapter 14 on page 95.

3.7 Using HDevEngine

With HDevEngine, you can load and run complete HDevelop scripts or individual procedures from your application.

Detailed information about HDevEngine can be found in the Programmer's Guide.

HDevEngine with C++

To summarize the compilation and linking process for C++, you

- include the main header file for HALCON and HDevEngine in your application and add the namespace:

```
#include "HalconCpp.h"
#include "HDevEngineCpp.h"

using namespace HalconCpp;
using namespace HDevEngineCpp;
```

- compile the application with the include paths

```
-I$(HALCONROOT)/include -I$(HALCONROOT)/include/halconcpp \
  -I$(HALCONROOT)/include/hdevengine
```

- and link the HALCON libraries and the HDevEngine library for the aarch64-linux or armv7a-linux architecture

```
-L$(HALCONROOT)/lib/$(HALCONARCH) -lhdevenginecpp -lhalconcpp -lhalcon
```

HDevEngine with Python

Using HDevEngine with Python on Arm-based platforms does not differ from other platforms. To summarize the process for Python, you

- ensure that CPython is available on the device
- add HALCON/Python to your project:

```
pip install mvtec-halcon==24050
```

- import the HALCON/Python module:

```
import halcon as ha
```

For more information, see the Programmer's Guide, chapter 24 on page 179.