HALCON

a product of MVTec

Installation Guide

HALCON 25.11 *Progress*

MVTec Software GmbH

Building Vision for Business

All about installing and licensing HALCON, Version 25.11.0.0

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without prior written permission of the publisher.

Copyright © 2003-2025 by MVTec Software GmbH, Munich, Germany

Protected by the following patents: US 7,751,625, US 7,953,290, US 7,953,291, US 8,260,059, US 8,379,014, US 8,830,229, US 11,328,478. Further patents pending.

AMD and AMD Athlon™ are either trademarks or registered trademarks of Advanced Micro Devices, Inc.
OpenCL™ and the OpenCL logo are trademarks of Apple Inc. used by permission by Khronos.
Arm® is a registered trademark of Arm Limited.
OpenGL® and the oval logo are either trademarks or registered trademarks of Hewlett Packard Enterprise in the United States and/or other countries worldwide.
Intel® , the Intel® logo, OpenVINO™ , the OpenVINO™ logo, and Pentium® are either trademarks or registered trademarks of Intel® Corporation or its subsidiaries.
Linux ® is a registered trademark of Linus Torvalds.
Microsoft, Windows, Microsoft .NET, Visual C++ and Visual Basic are either trademarks or registered trademarks of Microsoft Corporation.
NVIDIA, CUDA, cuBLAS, and cuDNN are either trademarks or registered trademarks of NVIDIA Corporation.
Python® is a registered trademark of PSF.
SUSE® is a registered trademark of Novell, Inc.
CodeMeter® is a registered trademark of WIBU-SYSTEMS AG.

All other nationally and internationally recognized trademarks and tradenames are hereby recognized.

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

About This Manual

The manual provides the necessary information to install HALCON and setup the licensing mechanism successfully. It is divided into the following chapters:

- **Introduction**
  A short overview of the different HALCON versions, available licensing schemes, and the system requirements.
- **Installing HALCON**
  How to install HALCON, either for the first time or in form of an update.
- **Uninstalling HALCON**
  How to uninstall HALCON.
- **Installing Third-Party Components**
  How to install third-party libraries for deep learning on Linux aarch64.
- **All About HALCON Licenses**
  Detailed information about the different types of licenses and how to obtain and install them.
- **Troubleshooting**
  Possible problems and how to solve them.
- **More on the Installation**
  Details like the installed file structure and the relevant environment variables.

Notation

Except for Linux-specific sections, file paths and environment variables are printed in the Windows convention, e.g.,

```
%HALCONEXAMPLES%\extension_package\halconuser
```

to denote the subdirectory `halconuser` containing an example package within the HALCON examples directory referenced by the environment variable `HALCONEXAMPLES` (see section A.4 on page 47 for more information on environment variables). The same expression in Linux convention would look like

```
$HALCONEXAMPLES/extension_package/halconuser
```

Symbols

The following symbols are used within the manual:

This symbol indicates a **tip**.

This symbol indicates an information you should **pay attention** to.

Contents

1 Introduction **7**
- 1.1 HALCON Editions . . . 7
- 1.2 HALCON Configurations . . . 7
- 1.3 Releases and HALCON Versions . . . 8
- 1.4 Supported Platforms and Minimum System Requirements . . . 8
  - 1.4.1 Platform-Specific HALCON Versions . . . 8
  - 1.4.2 Platform-Independent Applications . . . 9
  - 1.4.3 HALCON Variable Inspect (Visual Studio Extension) . . . 9
  - 1.4.4 Requirements for Deep Learning and Deep-Learning-Based Methods . . . 10
  - 1.4.5 Requirements for Scene Engine . . . 11
  - 1.4.6 Requirements for Language Interface Examples . . . 12
- 1.5 Limitations . . . 12
  - 1.5.1 General Limitations . . . 12
  - 1.5.2 Limitations for Dongle-based Licenses . . . 12
  - 1.5.3 Limitations Related to Compute Devices . . . 13
  - 1.5.4 Limitations Related to Image Acquisition . . . 13
  - 1.5.5 Limitations Related to OpenGL . . . 13
  - 1.5.6 Limitations Related to Extension Packages . . . 13
- 1.6 Licensing . . . 14

2 Cybersecurity **15**
- 2.1 Secure Configuration . . . 15
- 2.2 Security Contact . . . 16

3 Installing HALCON **17**
- 3.1 Downloading HALCON . . . 17
- 3.2 Starting the Installation . . . 17
  - 3.2.1 Windows/Linux . . . 17
- 3.3 Switching HALCON Versions . . . 20
- 3.4 Updating HALCON . . . 20
- 3.5 Installing I/O Device and Image Acquisition Interfaces . . . 21
- 3.6 Installing Extension Packages . . . 21
  - 3.6.1 Using an Extension Package Within HDevelop . . . 21
  - 3.6.2 Using an Extension Package in a Stand-Alone Application . . . 21

4 Uninstalling HALCON **23**
- 4.1 Windows . . . 23
- 4.2 Linux . . . 24

5 Installing Third-Party Components **25**
- 5.1 Third-Party Components for Deep Learning on Linux aarch64 . . . 25
  - 5.1.1 Getting the Components . . . 25
  - 5.1.2 Including the Libraries . . . 25
  - 5.1.3 Additional Required General Settings . . . 26

6 All About HALCON Licenses **27**
- 6.1 What is a License? . . . 27
- 6.2 Evaluation Licenses . . . 28

6.3 Development Licenses . . . 29
6.3.1 License Bound to a Network Card . . . 29
6.3.2 License Bound to a Dongle . . . 29
6.3.3 License Bound to a Remote Dongle . . . 30
6.4 Runtime Licenses . . . 30
6.4.1 Getting a Runtime License Restricted to Specific Modules . . . 30
6.4.2 Dynamic Modules for Deep-Learning-Based Applications . . . 31
6.5 How to Upgrade a License . . . 32
6.6 How to Check the Genuineness of HALCON (Windows) . . . 33
6.7 License Watchdog Process For Floating Licenses (Windows) . . . 33

7 HALCON in a Docker Container 35
7.1 Include a HALCON Installation . . . 35
7.1.1 Basic HALCON Packages . . . 35
7.1.2 Required System Packages . . . 35
7.1.3 HALCON Language Interface . . . 36
7.2 Include External Hardware . . . 36
7.2.1 NVIDIA GPUs for Deep Learning . . . 36
7.2.2 NVIDIA GPUs for OpenCL . . . 37
7.3 Licensing HALCON within a Docker Container . . . 37
7.3.1 Remote Dongle . . . 37
7.3.2 Forward USB License Dongle Into Container (Linux Only) . . . 38

8 Troubleshooting 39
8.1 Problems During Installation . . . 39
8.1.1 HALCON Variable Inspect (Visual Studio Extension) . . . 39
8.2 Problems Concerning Licenses . . . 40
8.2.1 Extracting Host IDs . . . 40
8.2.2 HALCON/Python and Floating Licenses under Linux . . . 41
8.3 Troubleshooting in HDevelop or HALCON Applications . . . 41
8.3.1 Troubleshooting with the hinfo utility . . . 41
8.3.2 Startup Errors . . . 41

A More on the Installation 43
A.1 Software Packages . . . 43
A.2 The Installed File Structure . . . 44
A.2.1 Main Directory . . . 45
A.2.2 Machine Configuration Data . . . 46
A.2.3 User Configuration Data . . . 46
A.3 HALCON tools . . . 46
A.4 HALCON's Environment Variables . . . 47
A.4.1 Setting Environment Variables Under Windows . . . 47
A.4.2 Setting Environment Variables Under Linux . . . 47
A.4.3 HALCON-Specific Environment Variables . . . 47
A.4.4 General Environment Variables . . . 48

Index 49

Chapter 1

Introduction

To use HALCON on a computer, you must

1. install HALCON on this computer and
2. obtain a license.

Before looking into the details of these two steps in chapter 3 on page 17 and chapter 6 on page 27, this chapter gives an overview of the different HALCON versions and licensing methods. Finally, it describes the system requirements for running HALCON.

1.1 HALCON Editions

HALCON comes in two editions:

- **HALCON Progress Edition**
  This edition is available by subscription with a six-month release cycle. It does not provide maintenance releases.
- **HALCON Steady Edition**
  This edition is available as a regular purchase with a biennial release cycle. It supports regular maintenance releases and the development version is offered in two variants (see below).

1.2 HALCON Configurations

You can use HALCON in two configurations:

1. **Development version**
   The development version (sometimes also denoted as full version) includes the full spectrum of HALCON, i.e., language interfaces to C, C++, .NET, and Python, interfaces to image acquisition and I/O devices, the Extension Package Interface, which allows you to integrate your own HALCON operators, and, of course, the interactive development environment HDevelop. You need this version whenever you want to develop applications based on HALCON.

   The development version of HALCON Steady is available in two variants: **HALCON Steady** and **HALCON Steady Deep Learning**, which includes the deep learning functionality.
2. **Runtime version**
   If you have finished developing an application based on HALCON, you only need a runtime version of HALCON for each computer where the application is to be run. Since the runtime version is not determined for developing applications it does not include the interactive development environment HDevelop. Furthermore, you can obtain runtime version licenses that include only parts of the functionality (so-called *modules*). For more information, see section 6.4 on page 30 and contact your local sales partner.

1.3 Releases and HALCON Versions

The term *version* has a second meaning: It denotes the HALCON major releases, e.g., HALCON 25.11.0.0 (HALCON Progress), or HALCON 25.05.1.0 (HALCON Steady). For HALCON Steady there might also be so-called *maintenance releases* like HALCON 25.05.2.0 or higher.

The main differences between new *versions* and *maintenance releases* are:

- **Functionality**
  A new HALCON *version* always represents a major step in the functionality. This means that it contains a significant number of new operators, but possibly also new functionality in HDevelop, e.g., new assistants. Furthermore, the functionality of individual operators may be extended or operators have been sped up. Of course, all currently known bugs in the preceding release will have been fixed.

  In contrast, the main intention of a *maintenance release* is to fix all currently known bugs. Nevertheless, such a release typically also brings some speed-ups and minor functional extensions.

- **Compatibility**
  A new HALCON *version* is not downward compatible, with the following implications: First, you must upgrade your HALCON license (see section 6.5 on page 32). Second, if you want to run applications created with an older release under the new version, you must regenerate the applications, as the new HALCON library is not binary compatible to the old one. The term 'applications' includes also image acquisition interfaces and extension packages you created yourself based on an older release. Note that a new version may also be source-code incompatible in some parts, e.g., the signature of an operator or a class method may have been changed. These changes are indicated in the release notes of the HALCON version. In such a case, you must adapt the source code of your application before regenerating it.

  In contrast, a *maintenance release* is in most cases fully downward compatible to its corresponding version. This compatibility includes the license. Please note, however, that some maintenance releases may not be fully binary or source-code compatible because of technical reasons. In such cases, the release notes will contain corresponding warnings and describe how to proceed.

1.4 Supported Platforms and Minimum System Requirements

HALCON runs on Windows and Linux. The minimum system requirements are listed in table 1.1; more details follow below.

1.4.1 Platform-Specific HALCON Versions

For the operating systems listed in table 1.1 on page 9, platform-specific versions of HALCON's executables and libraries are provided. The name of the currently used version is stored in the environment variable `HALCONARCH`. On Windows and Linux, HALCON uses AVX2-optimized code for many operators, when run on a machine that supports AVX2.

Note that HALCON should also run on newer versions of the operating systems than the ones listed; however, we cannot guarantee this.

`HALCONARCH` appears in several directory paths: Executable HALCON programs like `hdevelop`, and DLLs like `halcon.dll` (Windows only), reside in `%HALCONROOT%\bin\%HALCONARCH%`. On Windows systems, this path is therefore automatically included in the environment variable `PATH`;

on a Linux system, you must include it in your login script.

The libraries that you need for linking programs, e.g., `halcon.lib` (Windows) or `libhalcon.so` (Linux) reside in the directory `%HALCONROOT%\lib\%HALCONARCH%`.

Please note that when creating a 64-bit application, both the development computer and the computer on which the application will run must be 64-bit platforms.

To create .NET applications under Linux, you need to install .NET Core or Mono.

	Windows	Linux	
	x64	x64	aarch64
Architecture	64-bit	64-bit	64-bit
Processor	Intel 64 or AMD 64 SSE2 (AVX2 dispatch)	Intel 64 or AMD 64 SSE2 (AVX2 dispatch)	Armv8-A with AArch64 support
Disk Space	4 GB / 1 GB (full installation / Runtime – without deep learning)		
Memory	256 MB		
Display Resolution	1024 × 768	1024 × 768	–
Supported OS Versions	Windows 10 (x64 editions), 11, Windows Server 2016, 2019, 2022, 2025	Linux x86_64	Linux aarch64
Compiler	Visual Studio 2013	gcc 7.5	gcc 7.5
HALCONARCH	`x64-win64`	`x64-linux`	`aarch64-linux`
Specifics			Kernel with hidraw support
Libraries		GLIBC_2.27, GLIBCXX_3.4.24	GLIBC_2.27, GLIBCXX_3.4.24
Application-Specific Requirements		X11R7, freetype 2.4.11, fontconfig 2.10.95, OpenGL 2.0, OpenSSL 3, libdbus-1-3 (HDevelop)	X11R7, freetype 2.4.11, fontconfig 2.10.95, OpenGL 2.0, OpenSSL 3
Compute Device with OpenCL	OpenCL 1.1	OpenCL 1.1	OpenCL 1.1 [1]

Table 1.1: Minimum system requirements. Further requirements occur for the HALCON Variable Inspect and deep learning applications. Please see the corresponding sections below.
[1]: The support of OpenCL depends highly on the platform and the OpenCL driver coming with it, use at your own risk.

1.4.2 Platform-Independent Applications

Even when using a platform-specific version of HALCON, you can still create platform-independent applications, in two ways:

- **With HDevelop**, HALCON's integrated development environment (IDE). HDevelop programs are stored in a platform-independent format, thus, you can run them on any supported platform.
- **With HALCON/.NET**, HALCON's interface to .NET programming languages. Applications written in .NET languages are stored in a platform-independent intermediate language, which is then converted by the so-called common language runtime into platform-specific code.
- **With HALCON/Python**, HALCON's interface to Python script language. Python is an interpreted, high-level and general-purpose programming language that runs your code platform independently on a language interpreter engine.

You can combine both methods by using HDevEngine/.NET to run HDevelop programs from a HALCON/.NET application.

1.4.3 HALCON Variable Inspect (Visual Studio Extension)

HALCON Variable Inspect is shipped in two packages. Which package to use depends on the version of Visual Studio to support. The respective package of HALCON Variable Inspect depends on the following components:

Variable Inspect Extension for Visual Studio 2019 and earlier

- Visual Studio 2017 through Visual Studio 2019
- .NET 4.6
  Older installations of Visual Studio 2013 might be missing the .NET 4.6 framework. If the installation of HALCON Variable Inspect fails with an error, install the .NET 4.6 framework first.

Variable Inspect Extension for Visual Studio 2022 and later

- Visual Studio 2022 and later
- .NET 4.7.2 or higher

1.4.4 Requirements for Deep Learning and Deep-Learning-Based Methods

Disk space requirement for HALCON including deep learning:
Runtime: 3.4 GB (using CUDA 12.8.0).
Full installation: 20 GB.

Applications of deep learning as well as deep-learning-based methods, like Deep OCR, may have further requirements depending on the device (CPU or GPU) on which they are running. These requirements are given below in table 1.3 and table 1.4, respectively.

Whether an implementation is available for a specific device depends on the method and task. An overview for the different deep-learning-based methods is given in table 1.2.

Method (Model *'type'*)	**Training**		**Inference**	
	GPU	CPU	GPU	CPU
3D Gripping Point Detection (*'3d_gripping_point_detection'*)	✓	✓ [1]	✓	✓
Anomaly Detection (*'anomaly_detection'*)	✓	✓ [1]	✓	✓ [1]
Classification (*'classification'*)	✓	✓ [1]	✓	✓
Continual Learning (*'classification'*)	✓	✓ [1]	✓	✓
Deep 3D Matching (*'deep_matching_3d'*)	✓ [3]	×	✓	✓
Deep Counting (*'counting'*)	✓	✓	✓	✓
Deep OCR				
Component: Detection (*'ocr_detection'*)	✓	✓ [1]	✓	✓
Component: Recognition (*'ocr_recognition'*)	✓	×	✓	✓
Global Context Anomaly Detection (*'gc_anomaly_detection'*)	✓	✓ [1,2]	✓	✓
Multi-Label Classification (*'multi_label_classification'*)	✓	✓ [1]	✓	✓
Object Detection and Instance Segmentation (*'detection'*)	✓	✓ [1]	✓	✓
Semantic Segmentation and Edge Extraction (*'segmentation'*)	✓	✓ [1]	✓	✓

Table 1.2: Deep learning: Model dependencies.
[1]: With exception of aarch64.
[2]: Supported but not optimized and as a consequence time-intensive.
[3]: Minimal requirements are GPU with 24 GB RAM..

GPU Applications: When running deep learning as well as deep-learning-based applications on GPU, additional prerequisites apply. Table 1.3 lists additional prerequisites for which HALCON has been tested successfully. In case of multiple entries the ones corresponding have to be used.

HALCON checks which CUDA-version is installed and looks automatically for the corresponding subdirectories with its libraries. Under Windows or Linux x64, these libraries are provided in a separate package (see

		Windows	Linux	
		x64	x64	aarch64
OS Versions		Windows 11	Linux x86_64	Linux aarch64
GPU		NVIDIA GPU with minimal compute capability 5.0		
NVIDIA driver	CUDA	12.8.0	12.8.0	12.6.0 [1]
	min. Version	570.65	570.26	540.4 [1]
Libraries	cuBLAS	12.8.4.1	12.8.4.1	12.6.1.4
	cuDNN	9.10.2	9.10.2	9.3.0

Table 1.3: Requirements for deep-learning-based applications on GPU.
[1]: Install NVIDIA JetPack 6.2.0.

section A.1 on page 43.) Under Linux aarch64 these libraries have to be installed manually as described in chapter 5 on page 25.

CPU Applications: The following table 1.4 lists all platforms that support the execution of deep-learning-based applications.

	Windows	Linux	
Platform	x64 min. SSE4.1	x64 min. SSE4.1	aarch64 [1]

Table 1.4: Requirements for Deep Learning applications on CPU.
[1]: With exception of anomaly detection.

1.4.5 Requirements for Scene Engine

Disk space requirement for the HALCON Scene Engine and included asset library: 35 GB.

The Scene Engine application only runs on GPU. The following table 1.5 lists additional prerequisites for which the Scene Engine has been tested successfully.

	Windows x64	Linux x64
OS Versions	Windows 11	Linux x86_64
GPU	NVIDIA GPU with minimal compute capability 5.0	
NVIDIA driver supporting CUDA	12.8.0	12.8.0
Vulkan	1.2	1.2

Table 1.5: Requirements for Scene Engine on GPU.

1.4.6 Requirements for Language Interface Examples

For HALCON language interface examples additional requirements may apply.

Building HALCON examples with CMake

Minimum requirements depend on the language:

C and C++	Minimum supported CMake version: 3.7.1
.NET	Minimum supported CMake version: 3.8.2

Building Easy Extension Interface for .NET

CMake	Minimum supported CMake version: 3.7.1
C/C++-tooling	For example from Visual Studio
.NET Runtime	At least .NET runtime 6.0. Newer versions may be incompatible.

1.5 Limitations

1.5.1 General Limitations

Below, we list limitations that are relevant for typical application development with HALCON.

- String processing:
  - The HALCON library does not handle multibyte characters internally. File names containing multibyte characters are supported after calling `set_system('filename_encoding','utf8')`.
- Processing of NaNs:
  - Due to performance reasons, the HALCON library does not check for NaNs in input data. Using such input can lead to undefined behavior.
- Maximum image size:
  - HALCON: 32 768×32 768
  - HALCON XL: 1 073 741 824×1 073 741 824 ($2^{30} \times 2^{30}$)
- Maximum number of channels per image:
  - The number of channels per image is limited to 65535.

  Range for coordinates:
  - HALCON: from -32 767 to +32 767
  - HALCON XL: from -1 073 741 823 to 1 073 741 823 (-2^{30} to $2^{30} - 1$)
- Maximum number of windows: 600

1.5.2 Limitations for Dongle-based Licenses

After starting applications based on HALCON 20.05 (or later), the dongle must be removed and inserted again before starting applications based on HALCON 19.11 (or earlier). This problem can be avoided by installing the CodeMeter Runtime from WIBU SYSTEMS AG. The CodeMeter Runtime is not shipped with HALCON and must be downloaded from https://www.wibu.com in the download area.

On Arm platforms (aarch64 architecture) with GLIBC $\geq$ 2.21, it is no longer possible to run both 64-bit and 32-bit HALCON processes in parallel because they might block each other.

1.5.3 Limitations Related to Compute Devices

Operators that are based on texture are limited to the maximum size of the graphics card. Currently this limit is 8192×8192. This is relevant for the following operators:

- projective_trans_image and projective_trans_image_size,
- affine_trans_image and affine_trans_image_size,
- polar_trans_image_inv and polar_trans_image_ext,
- rotate_image,
- mirror_image,
- map_image,
- image_to_world_plane, and
- change_radial_distortion_image.

Further limitations are listed below.

- All other operators are restricted to the maximum allocated block size, e.g., 200 MB on the Tesla C2050.
- edges_sub_pix and lines_gauss are not Open CL accelerated for HALCON XL.
- edges_sub_pix and lines_gauss need a lot of memory on the compute device.

1.5.4 Limitations Related to Image Acquisition

- Maximum number of acquisition interfaces: 128
- Maximum number of open handles per acquisition interface: 256
- Maximum number of interface-specific parameters: 2048

1.5.5 Limitations Related to OpenGL

Operators using OpenGL for visualization (e.g., disp_object_model_3d) require OpenGL 2.1, GLSL 1.2, and the OpenGL extensions "GL_EXT_framebuffer_object" and "GL_EXT_framebuffer_blit". Therefore, those operators cannot be used via Windows Remote Desktop or SSH forwarding. Other operators, like find_shape_model_3d), have an optimized implementation when OpenGL is available. Which operators benefit from OpenGL can be found in the reference documentation of the corresponding operators. To figure out which OpenGL features are available and offered by the driver, HALCON opens a hidden OpenGL window during initialization when the first HALCON operator is called. Thus, if the system has defective OpenGL drivers installed, HALCON may crash during initialization. To work around this issue, HALCON can be told to ignore OpenGL entirely by defining and setting the system environment variable "HALCON_NO_OPENGL" to 1.

1.5.6 Limitations Related to Extension Packages

Maximum number of parameters:

Iconic input parameters:	9
Iconic output parameters:	9
Control input parameters:	20
Control output parameters:	20

1.6 Licensing

To run HALCON on a computer, you need a license. For HALCON Steady, licenses are always issued for a certain HALCON version (i.e., major release, see section 1.3 on page 8). However, a license is not exclusively bound to this version: It is *upward compatible within the version number*, i.e., licenses for major releases are also valid for follow-up maintenance releases. For HALCON Progress, licenses are issued for the subscription period.

Depending on the provided license, a HALCON Steady version can either be used as a Progress or Steady edition.

A license for HALCON Steady cannot be used for HALCON Progress, but starting with HALCON 24.11, a license for HALCON Progress can also be used for the corresponding HALCON Steady version. Note that the corresponding installation is therefore called HALCON Progress-Steady.

The license file is typically called "`license.dat`". Between "license" and ".dat", arbitrary information can be included. Thus, different licenses for different versions (e.g., `license-25.11.dat` for version 25.11) can be stored in the same directory.

The three possible licensing types mainly correspond to the different HALCON versions described in section 1.2 on page 7. Detailed information about HALCON licenses can be found in chapter 6 on page 27.

- **Evaluation license**
  To evaluate the full power of HALCON, you can obtain an evaluation license from your local sales partner free of charge. This type of license is not bound to any computer hardware, i.e., you can use HALCON on any computer you installed it on; however, it is only valid for a limited time, typically for a month. Note that you may not use this license to develop commercial applications.
- **Development license**
  To develop HALCON applications, whether in HDevelop or via a programming language, you need a development license. In contrast to the evaluation license, this license is permanent. Furthermore, this license is bound to a certain hardware component (network card or dongle, see section 6.1 on page 27).

  If you want to use HALCON on multiple computers simultaneously, you need a license for each of them.
- **Runtime license**
  If you finished developing your application based on HALCON and now want to install and run it on a customer's computer, you only need a runtime license. Like development licenses, runtime licenses are permanent and bound to a certain hardware component (network card or dongle).

  As already noted, you can obtain runtime licenses that cover only parts of the functionality (so-called *modules*). Please contact your local sales partner for more information.

Chapter 2

Cybersecurity

At MVTec, we prioritize product quality and customer benefit. Our software is developed by highly skilled engineers at our headquarters in Munich, following industry best practices and stringent guidelines.

We are committed to ensuring that our software:

- Retains recognized industry standards and regulatory requirements.
- Is designed and tested to be free from known vulnerabilities and backdoors.
- Is continuously monitored and updated to address emerging security threats.
- Provides all necessary documentation for full transparency on third-party components and dependencies.

For information about a secure installation of HALCON we refer to the MVTec Software Manager documentation at https://docs.mvtec.com.

2.1 Secure Configuration

The software is available in a development (SDK) and runtime (RTL) version. Only the RTL version is intended to be redistributed to customers. The RTL version is mainly a library which inherently provides functions which are potentially dangerous from a cybersecurity point of view.

In general, all functions which access memory, provide I/O, or file access need to be used with the necessary caution. It is the developer's responsibility to ensure that no functions are used in an unintentionally insecure manner. Only valid files that cannot be manipulated by unauthorized parties or potentially compromised systems should be read. Ensure that all hardware components and shared libraries are obtained from trusted and verified sources. Furthermore, dynamic environment variables may be exploited if not properly managed.

Additionally, be aware that the operator `system_call` can be used to execute system calls and should therefore be handled with caution.

The operator `set_system` can be used to completely disable unwanted functionality using the parameter *'disabled_operators'*. All calls to operators need to be checked by the caller if the parameters are in a secure range.

HALCON provides various ways of communication and it is up to the developer of the final application to choose the security level according to his needs. For example, socket support includes TLS functionality, but this must be actively enabled by the developer.

Telemetry can be enabled in the HDevelop IDE to assist MVTec in developing relevant features. Of course, all transmitted data complies with the General Data Protection Regulation (GDPR) and our data privacy guidelines. In no case any personal data is transmitted to MVTec.

You can find relevant changes in the release notes.

2.2 Security Contact

If you have any security-related questions or concerns, please contact us at security@mvtec.com. Your request will be processed with care and discretion.

For more information and further contact options please refer to the MVTec homepage at https://www.mvtec.com/cybersecurity.

Chapter 3

Installing HALCON

3.1 Downloading HALCON

Note that you first need to register before downloading software.

1. Log in to MVTec's Download Area at https://www.mvtec.com/downloads.
2. Click HALCON DOWNLOADS.
3. Choose PRODUCT VERSION, OPERATING SYSTEM, and, if applicable, the ARCHITECTURE.
4. Download one of the following:
   - "MVTec Software Manager" (SOM)

     A lightweight package manager that downloads only the necessary packages during installation. You can also use the manager to create your own, customized offline installer for the purpose of archiving or unrolling your specific HALCON installation.
   - "Full Version"

     An Offline installer for Windows and Linux includes SOM and all packages required for the full HALCON installation.

3.2 Starting the Installation

3.2.1 Installing HALCON Under Windows/Linux

There are two installation options:

- GUI-based installation, i.e., controlling SOM via your web browser (section 3.2.1.1 on page 18).
- Command line installation, with the possibility for a silent installation (section 3.2.1.2 on page 18).

In both cases, you can run SOM with user permissions ("user mode") or with administrator permissions ("system mode"). Software packages can be installed only for yourself or for all users of the system. In the latter case, SOM will request the credentials for system-wide installation when the corresponding option has been selected.
For more information, see the section "Running SOM" in SOM's internal documentation.

Both installation methods require a valid **login to the download area**. You can only log in from the GUI (start SOM, and click the LOGIN button). The login is remembered across SOM sessions and will be used by the command line installation.

For aarch64-linux only: To use the deep learning functionality on GPUs, two third-party libraries need to be installed manually (see chapter 5 on page 25). For inference on CPUs, the third-party libraries are not required, but it may be necessary to link the correct OpenMP. For more information, see Programmer's Guide, section 2.4 on page 19.

3.2.1.1 GUI-based Installation

To install HALCON using the SOM GUI in the browser, do the following:

1. If you downloaded the "Full Version" or the "Runtime Version", unzip the downloaded archive.
2. Execute `som`. If `som` is not executable under Linux, adapt its file permissions.
   → SOM opens a page in your default web browser.

   If SOM is not yet installed or up-to-date, a welcome dialog offers to install SOM or to update your current installation of SOM. You can skip or postpone this step by closing the dialog.
   To install HALCON, an installation of SOM is not required. Nonetheless, installing SOM is recommended, because it can manage and update your installation(s) and provides easy access to HDevelop and the documentation.
3. Open the page `AVAILABLE`.
   → All available HALCON versions are listed.
4. To start the default installation, click the `INSTALL` button of the HALCON version you wish to install.
   → The "Packages" dialog opens, with all necessary packages already pre-selected.

   Alternatively, click `SELECT PACKAGES` to set up the installation without pre-selections. See section A.1 on page 43 for information about the packages.

   You can install additional packages at a later time by starting SOM again and selecting the packages you need.
5. Click `APPLY`.
   → The installation starts. The installed HALCON version appears under `INSTALLED`.

To complete the HALCON installation, add a license (section 3.2.1.3 on page 19) and, if applicable, configure your installation (section 3.2.1.4 on page 19).

3.2.1.2 Command Line Installation

SOM can also be operated from the command line. This enables script-driven installation and removal of software packages. For more information, see the chapter "Command line usage" of **SOM's internal documentation**. If you want to install HALCON remotely, see also the chapter "Headless operation" .

1. If you downloaded the "Full Version" or the "Runtime Version", unzip the downloaded archive.
2. Run "MVTec Software Manager CLI" if you have installed SOM. Otherwise, open a command prompt, and change to the directory where you saved the `som` executable.
3. List the catalog and available feed URLs:

   ```
   som cat
   ```

4. To install, for example the runtime version of HALCON, type:

   ```
   som -f FEED install rt
   ```

   where `FEED` has to be replaced with the actual feed URL or a unique substring of the feed URL, e.g., `halcon-25.11-progress`.
   → The EULA is shown, followed by a line repeating your last command line with an additional parameter `--accept HASH`.
5. To open the help, enter

   ```
   som help
   ```

6. Copy and paste the line to start the installation, for example:

   ```
   som --accept HASH -f FEED install rt
   ```

To complete the HALCON installation, add a license (section 3.2.1.3 on page 19) and, if applicable, configure your installation (section 3.2.1.4 on page 19).

Silent Installation

You can also use the last command for non-interactive (i.e., silent) installations. To install silently, suppress the command output as usual, i.e., under Windows with `> nul`, and under Linux with `> /dev/null`. For example:

```
som --accept HASH -f halcon-25.11-progress install rt > nul
```

3.2.1.3 License Installation

Installing a License File

You can either place the license file `license.dat` manually into the directory `$HALCONROOT/license`. Alternatively, you can use SOM to install the license file. Note that **SOM overwrites licenses of the same file name**. If you want to avoid this, you have to rename the license file in advance. The correct notation for license files is `license*.dat`.

To install the license file via SOM:

1. Open the page `INSTALLED`.
   → All installed HALCON versions are listed.
2. Open the three-dots pop-up menu of the HALCON version you wish to add the license to.
3. Select `Manage packages`.
4. Click `Browse...` to select the license file.
5. Click `install license file`.
   → HALCON can now be started out of SOM.

Installing a Dongle-bound License

If you are using a CodeMeter dongle and CodeMeter Runtime is running, CodeMeter Runtime needs to be at least version 6.00. Otherwise, the dongle is not recognized correctly, and HALCON throws an error regarding the license file.

Under Linux, you need to allow access to the `hidraw` device for CodeMeter dongles. Therefore, make sure the SOM package containing the hardware support for license dongles is installed.

3.2.1.4 Configuration

Setting Environment Variables

On Windows, the environment variables are set during the installation. On Linux, the following environment variables must be set in order for HALCON to work independent from a SOM session (see section A.4 on page 47 for more information about these and other environment variables):

- `HALCONROOT`: directory you installed HALCON in.
- `HALCONEXAMPLES`: directory the example programs are installed in (`$HALCONROOT/examples`).
- `HALCONIMAGES`: directory the example images are installed in (`$HALCONEXAMPLES/images`).
- `HALCONARCH`: select value corresponding to the used platform (see table 1.1 on page 9).
- `PATH`: this system variable should include `$HALCONROOT/bin/$HALCONARCH`.
- `LD_LIBRARY_PATH`: this system variable should include `$HALCONROOT/lib/$HALCONARCH`.

```
HALCONROOT="/home/foo/MVTec/HALCON-25.11-progress"; export HALCONROOT
HALCONARCH="x64-linux"; export HALCONARCH
PATH="$HALCONROOT/bin/$HALCONARCH:$PATH"; export PATH
LD_LIBRARY_PATH="$HALCONROOT/lib/$HALCONARCH:$LD_LIBRARY_PATH"; export LD_LIBRARY_PATH
```

Figure 3.1: Example for a shell script with environment variables in `sh` syntax, generated when installing HALCON into the directory `/home/foo/MVTec/HALCON-25.11-progress` on a Linux system.

It is recommended that you set the environment variables in a login script or a shell resource script, e.g., `.cshrc` or `.profile`. **The installation script automatically creates an example shell script** `.profile_halcon` in `$HALCONROOT` which contains the necessary settings in `sh` syntax, see figure 3.1 on page 20. The shell script can be included in your login script.

Make sure `LD_LIBRARY_PATH` is set correctly after a reboot, e.g., with:

```
echo $LD_LIBRARY_PATH
```

Some systems disallow setting `LD_LIBRARY_PATH` in `.profile`. If the variable is not set after reboot, you should try to set `LD_LIBRARY_PATH` in other initialization files like `.bashrc`.

Optimizing Parallelization

Optionally, you can optimize HALCON's automatic operator parallelization for your computer as described in the Programmer's Guide, section 2.1.1 on page 15.

3.3 Switching HALCON Versions

If multiple versions of HALCON are installed on your system, the active version may be switched using SOM on Windows. To switch the HALCON version:

1. Open SOM.
   → Note the different HALCON versions under `INSTALLED`.
2. Click the radio button in front of `HDevelop` or `HDevelop XL` to register the corresponding HALCON version.
   Please note that you should **use HALCON XL only when you need its features**.

Only one HALCON version can be active at any given time. See also "Handling of HALCON versions" in SOM's internal documentation.

You can also switch the HALCON versions manually:

- On Linux systems, by setting the environment variable `HALCONROOT` accordingly. Note that in order for this method to work, paths based on `HALCONROOT` in other environment variables like `PATH` and `LD_LIBRARY_PATH` must use the variable and not its content. See figure 3.1 on page 20 for an example.
- Under Windows, you must adapt those environment variables that are set during the installation, i.e., `HALCONROOT`, `HALCONARCH`, `PATH`, `HALCONEXAMPLES`, and `HALCONIMAGES`, and those you set yourself (e.g., `HALCONEXTENSIONS`). Please refer to section A.4 on page 47 for more information about setting environment variables.

3.4 Updating HALCON

With HALCON, the term "update" means to install a newer maintenance release over a release based on the same HALCON version. As described in section 1.3 on page 8, you can update HALCON without needing a new license.

To check for available updates, open SOM and watch for the bell icon. This icon signifies available HALCON updates.

To update HALCON:

1. Click on the bell icon or open the pop-up menu and select `Update`.
2. Click `APPLY`.

3.5 Installing I/O Device and Image Acquisition Interfaces

With every HALCON installation, you obtain several already installed I/O device and image acquisition interfaces (see also section A.2 on page 44). Additional interfaces are obtainable from the MVTec/HALCON Download area.

In between HALCON releases, image acquisition interfaces might be updated by MVTec or the manufacturer of an image acquisition device. Such updates are indicated on MVTec's WWW server, to which you can connect by selecting HDevelop's menu entry `Help` ▷ `HALCON News (WWW)` or in the Start Dialog which appears when starting HDevelop. You can then download the interface together with its documentation and HDevelop example programs.

3.6 Installing Extension Packages

The HALCON Extension Package Interface enables you to integrate newly developed image processing algorithms into HALCON in the form of so-called *extension packages*. The same mechanism is used by MVTec to extend the current HALCON release with additional functionality. Which extension packages are currently available can be checked by selecting HDevelop's menu entry `Help` ▷ `HALCON News (WWW)`.

This section describes how to integrate a (downloaded) package named `newextpkg` in order to use it within your HALCON system.

1. Extract the package to a directory of your choice, e.g., `%HALCONROOT%`.
2. Add the *complete* path of the package, e.g., `%HALCONROOT%\packages\newextpkg` to the environment variable `HALCONEXTENSIONS`. Note, that the delimiter between paths in an environment variable is a semicolon on Windows systems and a colon on Linux systems.

Never change the name of an extension package or the corresponding names of the libraries or DLLs contained in it. These names are encoded *within* the libraries/DLLs. If you change the names this information will no longer match. Thus, the loader of the operating system will fail to open the dynamic libraries.

If the package contains images used, e.g., within example programs we recommend including the complete path to the corresponding directory `images` within the package in the environment variable `HALCONIMAGES` (see section A.4 on page 47) to access those images without specifying a complete path.

3.6.1 Using an Extension Package Within HDevelop

To use a new package within HDevelop under Windows, you just need to restart the program. HDevelop automatically integrates all extension packages specified in `HALCONEXTENSIONS`, i.e., the operators contained in a package can be accessed and used like any other HALCON operator.

Under Linux, you must include the package library subdirectory (i.e., `lib/$HALCONARCH`) in the environment variable `LD_LIBRARY_PATH` before starting HDevelop the first time (see table 1.1 on page 9 for the possible values of `HALCONARCH`).

3.6.2 Using an Extension Package in a Stand-Alone Application

If you want to generate a stand-alone application that uses an extension package, you have to link the package libraries (DLLs under Windows, shared libraries under Linux) to the application code, in addition to the HALCON library.

3.6.2.1 Using an Extension Package Under Windows

In order to create new application programs you have to link the corresponding language interface library, e.g., `packagecpp.lib` for a C++ application, to your objects. Furthermore, you will need the HALCON interface library, in the example of a C++ application `halconcpp.lib`, as for any HALCON application.

To be able to link the package DLL to your application program, the *complete* DLL file path of the new package, e.g.,

```
%HALCONROOT%\packages\newextpkg\bin\%HALCONARCH%
```

must be added to the environment variable `PATH` (see table 1.1 on page 9 for the possible values of `HALCONARCH`).

Do not copy a package DLL into the Windows system directories, as it would be loaded twice in this case.

3.6.2.2 Using an Extension Package Under Linux

In order to create new application programs, you must link `libnewextpkg.so` and the corresponding language interface library, e.g., `libnewextpkgcpp.so` for a C++ application, to your objects (besides `libhalcon.so` and the HALCON interface library, in the example of a C++ application `libhalconcpp.so`, as for any HALCON application).

Furthermore, you have to add the path to the package library subdirectory `lib/$HALCONARCH` to the environment variable `LD_LIBRARY_PATH`, otherwise the loader will fail to access the libraries.

Chapter 4

Uninstalling HALCON

4.1 Uninstalling HALCON Under Windows

The preferred method to uninstall HALCON is to use the automatic uninstallation program as described in the following section. If you want to keep track of what is happening to your system, you can follow the instructions given in section 4.1 on page 23.

If you have installed different HALCON versions simultaneously with a compatible HALCON Variable Inspect extension for Visual Studio, uninstalling either of these versions will also uninstall the Variable Inspect extension. As a result, the Variable Inspect extension will be missing in the remaining HALCON version.

Uninstalling Automatically

To uninstall HALCON:

1. Open SOM.
2. Open the pop-up menu of the HALCON version to be uninstalled and select `Uninstall`.

Alternatively, you can remove HALCON via the Control Panel of Windows.

Note that **the uninstallation removes exactly those files that were installed**. This has two implications: If you added files after the installation manually, e.g., new image acquisition interfaces, extension packages, images, or manuals, these files and the corresponding directories will "survive" the uninstallation. On the other hand, if you only modified a file, e.g., an example, without changing its name, the uninstallation will remove it nevertheless. Therefore, you might want to copy such files to another directory before starting the uninstallation.

The uninstallation process will not remove any *user-specific settings*. This means that entries concerning, e.g., the layout of HDevelop or its file history, will be left in the file `%APPDATA%\MVTec\HDevelop.ini`. If you have run the utility hcheck_parallel, AOP information has been stored in `%ProgramData%\MVTec\HALCON 25.11\.aop_info`. You can remove these files manually without risk. Moreover, the uninstaller does not remove the MVTec GigE Vision streaming filter driver if it is installed. This driver needs to be removed separately.

Uninstalling Manually

The commands given in the following description should be entered in a Windows command prompt, which can be obtained by entering `cmd.exe` in the dialog `Start ▷ Run`. You need administrator privileges to perform the uninstallation.

1. Delete the installation directory.
   You can also use Windows Explorer to do this. Note that **the license file and any local additions to this directory will be lost**. A backup of these files is highly recommended.

   ```
   rmdir /S "%HALCONROOT%"
   ```

2. Delete all HALCON registry keys.
   You can also use the Windows registry editor `regedit.exe` to delete the keys. To remove the registry keys for a user installation of HALCON, replace `HKLM` with `HKCU`.

```
reg delete "HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall\MVTec HALCON version"
reg delete HKLM\SOFTWARE\Classes\.hdev
reg delete HKLM\SOFTWARE\Classes\.dev
reg delete HKLM\SOFTWARE\Classes\.hdpv
reg delete HKLM\SOFTWARE\Classes\.dpv
reg delete HKLM\SOFTWARE\Classes\.hdpl
reg delete HKLM\SOFTWARE\Classes\.hobj
reg delete HKLM\SOFTWARE\Classes\HDevelop.Source.File
reg delete HKLM\SOFTWARE\Classes\HDevelop.External.Procedure
reg delete HKLM\SOFTWARE\Classes\HDevelop.Procedure.Library
reg delete HKLM\SOFTWARE\Classes\HALCON.Object
```

   Replace `version` with the version you are uninstalling, e.g., `25.11-Progress`.

3. Delete all environment variables.

```
reg delete "HKLM\SYSTEM\CurrentControlSet\Control\Session Manager\Environment" /V HALCONARCH
reg delete "HKLM\SYSTEM\CurrentControlSet\Control\Session Manager\Environment" /V HALCONEXAMPLES
reg delete "HKLM\SYSTEM\CurrentControlSet\Control\Session Manager\Environment" /V HALCONIMAGES
reg delete "HKLM\SYSTEM\CurrentControlSet\Control\Session Manager\Environment" /V HALCONROOT
```

```
reg delete HKCU\Environment /V HALCONARCH
reg delete HKCU\Environment /V HALCONEXAMPLES
reg delete HKCU\Environment /V HALCONIMAGES
reg delete HKCU\Environment /V HALCONROOT
```

   Please also use the GUI to manually remove the HALCON binary directory from the environment variable `PATH`. See section A.4 on page 47 on how to edit environment variables using the Windows GUI.

4.2 Uninstalling HALCON Under Linux

Uninstalling Automatically

To uninstall HALCON:

1. Open SOM.
2. Open the pop-up menu of the HALCON version to be uninstalled and select `Uninstall`.

Unistalling Manually

Please note: The following procedure will **delete your local additions to the HALCON base directory**.

The actual uninstallation consists of simply removing the content of the HALCON base directory `$HALCONROOT` and all its subdirectories, e.g., by executing

```
rm -rf $HALCONROOT
```

Furthermore, remove the subdirectory `.hdevelop` of the directory referenced by the environment variable `HOME` (see section A.4 on page 47); HDevelop creates this directory to save options, window positions, and the file history.

Finally, delete references to HALCON from the environment variables (see section 3.2.1.4 on page 19).

Chapter 5

Installing Third-Party Components

This chapter describes how to install third-party components that might be required in order to use a specific functionality of HALCON. Currently, this is limited to users of **Deep Learning and deep-learning-based methods** on NVIDIA GPUs.
Please note that for Deep Learning either of the Deep Learning modules Deep Learning Enhanced or Deep Learning Professional must be licensed. This condition does not necessarily apply to deep-learning-based methods as e.g., Deep OCR.

In addition to a valid license, the following third-party components are required for the training of a network or the inference on GPUs:

- NVIDIA GPU and up-to-date graphics driver (see section 1.4 on page 8 for the system requirements),
- the NVIDIA CUDA Basic Linear Algebra Subroutine library from the CUDA Toolkit (cuBLAS),
- the NVIDIA CUDA Deep Neural Network library (cuDNN).

Under **Windows or Linux x64**, these libraries are provided in a separate package via SOM. Therefore, the following sections do not apply for these systems. Under **Linux aarch64** the libraries have to be installed separately, see the section section 5.1 on page 25.

5.1 Third-Party Components for Deep Learning on Linux aarch64

5.1.1 Getting the Components

In order to run Deep Learning, additional requirements apply. Please see section 1.4.4 on page 10.

Running Deep Learning on GPUs on Linux aarch64, the following components must be installed additionally:

- CUDA with the cuBLAS library
- the cuDNN library

The required versions are listed in table 1.2 on page 10.

A convenient way to install them is the NVIDIA "JetPack" installer version 6.2.0, which you can download from `https://developer.nvidia.com/embedded/jetpack`. Make sure to select the CUDA Toolkit and cuDNN Package as to install on the target.
It might still be necessary to install CUDA. In this case, `libcuda.so` is contained in an executable installer. Run `install.sh` on `/home/nvidia/NVIDIA_INSTALLER`.

5.1.2 Including the Libraries

HALCON needs to find the libraries. For this, firstly you need to locate the shared objects.

- The cuBLAS libraries:

- libcublas.so
- libcublasLt.so

They can be found where the CUDA Toolkit has been installed.

- The cuDNN libraries:
  - libcudnn.so
  - libcudnn_adv_train.so
  - libcudnn_ops_infer.so
  - libcudnn_cnn_infer.so
  - libcudnn_ops_train.so
  - libcudnn_adv_infer.so
  - libcudnn_cnn_train.so

  They are generally contained in the lib64 subdirectory.

Secondly, you need to make them detectable for HALCON. To do so, you have the following options, whereof we recommend the first one:

a Extend the environment variable LD_LIBRARY_PATH to include the respective libraries.
This may be done e.g., with the following commands:

```
LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/lib/aarch64-linux-gnu
LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/local/cuda/targets/aarch64-linux/lib
```

b Copy the required libraries libcuda.so and the mentioned cuBLAS and cuDNN library to the third party directory:
$HALCONROOT/lib/$HALCONARCH/thirdparty/cuda12_6.
All libraries needed for CUDA 12 have to be in the subdirectory cuda12_6. If the subdirectory thirdparty/cuda12_6 does not exist, create it first.

Additionally extend the environment variable LD_LIBRARY_PATH to include the respective libraries.
This may be done e.g., with a command like the following:

```
LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$HALCONROOT/lib/$HALCONARCH/thirdparty/cuda12_6
```

5.1.3 Additional Required General Settings

The HALCON process must be granted read-write access on:
/dev/nvhost-ctrl
This can be achieved e.g., via membership in the "video" group.

Chapter 6

All About HALCON Licenses

Section 1.6 on page 14 already contained an overview of the possible licensing schemes. In this chapter, you will find detailed information about how to obtain and install

- evaluation licenses (section 6.2 on page 28),
- development licenses (section 6.3 on page 29), and
- runtime licenses (section 6.4 on page 30),

Finally, section 6.5 on page 32 shows how to upgrade a license.

6.1 What is a License?

HALCON licenses are stored in so-called *license files*. The content of these files specifies

- what is licensed (e.g., development version, runtime version, etc.)
- which modules are licensed (e.g., Calibration, Matching, etc.)
- whether the license is temporary (e.g., evaluation license) or permanent
- the hardware to which the license is bound (see below)

License files are named `license.dat` (or `license-25.11.dat` or similar, see section 1.6 on page 14) and reside in the subdirectory `license` of the directory where you installed HALCON. **Note that HALCON will not run if you modify the license keys within the license file manually!**

A single license allows you to run a specific version of HALCON by one user on one system.

The system is identified by a unique host ID. Depending on the type of host ID, you may install HALCON on any number of computers, but in any case you must use it only on the one computer to which the host ID is attached. Further information about licenses can be found in the EULA.

The license is not bound to a specific user. Consequently, different users may use the licensed HALCON version, but not simultaneously. To have multiple users run HALCON at the same time, the corresponding number of licenses are needed. Note that each system service or daemon which executes HALCON is also counted as user. Thus, if only one license is available and a system service executes HALCON, the regular user is not able to run another instance of HALCON in parallel.

Network Card Versus Dongle Binding

As noted in section 1.6 on page 14, development and runtime licenses are bound to a certain hardware component. This is either the *network card* (see section 6.3.1 on page 29) or a *dongle* (see section 6.3.2 on page 29).

Dongle-bound licenses allow using HALCON on different computers by moving the dongle. Of course, network cards can also be switched between computers, but in practice they can be regarded as fixed. Thus, if you want to develop applications with HALCON on more than one stand-alone computer at different times, the easiest solution is to obtain a dongle-bound license.

Identifying the Hardware

The license manager software identifies a network card or a HALCON dongle by a so-called *host ID*. A valid host ID is the unique, immutable, machine-readable identification of an actual piece of hardware as devised by the hardware vendor.

HALCON includes a tool to query the host IDs from the command line. Open a Command Prompt window or a Linux shell, and then enter the following command:

```
hhostid
```

This command produces a list of host IDs, which are available for licensing. For example:

```
0d00cafebabe 3-3123456
```

Use `hhostid -i` to get additional information, such as the type of a certain host ID:

```
hhostid -i
#0: 0d00cafebabe [MAC] <Local Area Connection>
#1: 3-3123456 [CM] <CM Container>
```

Each output line lists the following properties: First, the running number followed by the actual host ID is displayed. In the case of network adapters, the *permanent* MAC address is displayed. The host ID type is displayed in square brackets (`MAC` for network adapters, `CM` for HALCON dongles). The value in angle brackets shows the interface name as reported by the operating system. If the currently assigned MAC address of a network adapter differs from its permanent address, it is reported here as well. HALCON only supports MAC-based licenses bound to universally administered, permanent MAC addresses. Locally administered MAC addresses are rejected.

See section 8.2.1 on page 40 for a detailed description on solving problems extracting the host ID.

HDevelop automatically checks whether any network cards or HALCON dongles are present and displays their host IDs in the menu item:

```
Help ▷ About.
```

For example, the following information will be displayed on a computer equipped with a network card and a dongle:

```
HALCON 25.11 Progress

HDevelop version: 25.11.0.0 (03.11.2025)
HALCON version: 25.11.0.0 (03.11.2025)

Platform version: x64-win64 (avx2)

(c) 1996-2025
MVTec Software GmbH, Munich, Germany

The host IDs of this computer are:
0d00cafebabe                        (network card ID)
3-3123456                           (dongle ID)
```

The first line shows the HALCON edition (Progress, Steady, or Steady Deep Learning, see section 1.1 on page 7).

6.2 Evaluation Licenses

As already noted in section 1.6 on page 14, with an evaluation license you can evaluate the full functionality of HALCON free of charge on any computer. The only restrictions are that evaluation licenses are valid only for a limited time (typically a month), and no commercial applications may be developed.

Evaluation licenses differ according to the platform on which they unlock HALCON. This description refers to Intel-based platforms. For Arm-based platforms, please refer to HALCON for Arm-based Platforms, section 2.2 on page 10.

Step 1: Obtain the license

You can obtain an evaluation license from your local sales partner.

Step 2: Install the license

"Installing" the license simply means placing the license file into the subdirectory `license` of the directory where you installed HALCON. If necessary, rename the file to `license.dat` (or `license-25.11.dat` or similar, see section 1.6 on page 14).

Note that you can evaluate HALCON on any computer where you installed HALCON just by copying the evaluation license file into the corresponding subdirectory `license`. You can also evaluate HALCON under different operating systems.

6.3 Development Licenses

A development license allows you to use the full functionality of your HALCON edition (see section 1.2 on page 7) including the development tools like HDevelop (see also section 1.6 on page 14). It must be bound to a certain hardware component (see also section 6.1 on page 27). The following sections describe how to proceed to obtain and install a

- license bound to a network card (section 6.3.1)
- license bound to a dongle (section 6.3.2 on page 29)

6.3.1 License Bound to a Network Card

Step 1: Extract the host ID

The easiest way to extract the host ID is to execute the following command from a Windows command prompt or a Linux shell.

```
hhostid -i
```

All host IDs of type MAC can be used for licensing. See section 6.1 on page 28 for more information.

Step 2: Obtain the license

Send the host ID of the network card to your local sales partner. The sales partner then sends you a *license file*.

Step 3: Install the license

Place the license file into the subdirectory `license` of the directory where you installed HALCON. If necessary, rename the file to `license.dat` (or `license-25.11.dat` or similar, see section 1.6 on page 14).

6.3.2 License Bound to a Dongle

Step 1: Obtain dongle and license

Please note that **you cannot use any dongle but only those supplied by MVTec via your local sales partner**. Currently, HALCON supports USB dongles.

The sales partner will send you the dongle together with a corresponding *license file*. The dongle ID is printed on the dongle.

Step 2: Install the license

Place the license file into the subdirectory `license` of the directory where you installed HALCON. If necessary, rename the file to `license.dat` (or `license-25.11.dat` or similar, see section 1.6 on page 14).

If you want to use HALCON on more than one computer by switching the dongle between them, repeat this step for every computer.

6.3.3 License Bound to a Remote Dongle

Starting with HALCON 21.05, you can also use a dongle connected to a remote computer over the network.

The remote dongle must be plugged into a computer with a running CodeMeter Runtime, version 6.50 or later, that is configured to allow remote clients to acquire licenses. For security reasons, MVTec recommends using version 7.10 or later. Please see the CodeMeter Runtime documentation on how to configure the CodeMeter Runtime correctly.

Make sure the communication between client and server is possible, i.e., not blocked by a firewall. By default, TCP port 22350 is used.

You must also tell HALCON how to contact the dongle. This is done by adding the `server=<hostnames>` tag to the end of the license string for the dongle in the license file. `<hostnames>` is a comma-separated list of hostnames to contact. These can be either fully qualified domain names, IP addresses, or the special keyword `_local`. `_local` instructs HALCON to use a dongle connected to the local computer.

The following example license file shows a HALCON runtime license for the dongle ID 3-3668597. HALCON first looks for the dongle locally, then attempts to contact the CodeMeter Runtime at the IPv4 address 10.0.0.5 and, if this fails, contacts the CodeMeter Runtime at license.example.com:

```
LICENSE MVTec_HALCON 25.11 MODULES=fc123bdomit ID=CM:3-3668597 \
        FLAGS=PROGRESS_OK SESSIONS=1 SIGNATURE="F155 C6B7 5E24 5FAF 224D B779 \
        93D7 65B1 1F0C CCD9 CE94 03D0 0D79 066D C66F F869 A92D EC52 B7A2 04A7 \
        874C B503 BE19 B150 C9C1 592F 41EB C808 8479 380C CE95 1408" \
        server=_local,10.0.0.5,license.example.com
```

Note that the `hhostid` tool and the `get_system`('hostids') HALCON operator call can only find dongles connected directly to the computer on which HALCON is running and will not find remote dongles.

6.4 Runtime Licenses

In contrast to a development license, a runtime license only allows you to run HALCON applications. Furthermore, a runtime license can be restricted to a set of selected modules if your application does not require the full functionality of HALCON.

6.4.1 Getting a Runtime License Restricted to Specific Modules

In the following we list the steps to determine the used modules, obtain an appropriate runtime license and install it.

Step 1: Extract the required modules

To extract the modules that are used by an application proceed as follows:

1. If the application is running in HDevelop, select the menu item `File` ▷ `Properties`, which will open a dialog. In its tab `Used Modules` the used modules are listed (see the HDevelopUser's Guide, section 6.14 on page 111, for more information). Figure 6.1 shows the result for an OCR application.

   If you click `Copy to Clipboard`, the required modules are saved in the clipboard, from where you can insert them in other applications.

   Please note that this method determines the list of used modules by inspecting *all* operators of the current program, no matter if they can be reached or not. If the program contains operator calls that are never executed, it is recommended to deactivate the corresponding program lines using F4 before opening this dialog to get a correct list of used modules.

2. If the application is written in a programming language (C, C++, C#, VB.NET, etc.), insert the operator `get_modules` (see the corresponding entry in the HALCON Reference Manuals for more information) at the end of the program. Note that **the operator `get_modules` will only return the correct modules if all HALCON operators used in the application are executed at least once**.

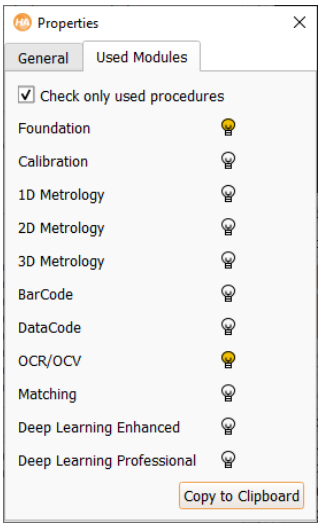

Figure 6.1: Used modules for an OCR application.

Note that there is a special case regarding the modules for deep-learning-based methods, where the required license does *not* depend on the used operator, but on the used model type (dynamic module). See section 6.4.2 on page 31 for further details.

Step 2: Extract the host ID

Please refer to section 6.3.1 on page 29 (network card) for information about how to extract the host ID. If you choose a dongle-bound license, no further action is required as you get the dongle together with the license (see section 6.3.2 on page 29).

Step 3: Obtain the license

Send the determined module names and – except in case of a dongle-bound license – the extracted host ID to your local sales partner.

The sales partner then sends you a license file. If you requested a dongle-bound license, you will also receive the dongle.

Step 4: Install the license

Place the license file into the subdirectory `license` of the directory where you installed HALCON. If necessary, rename the file to `license.dat` (or `license-25.11.dat` or similar, see section 1.6 on page 14).

6.4.2 Dynamic Modules for Deep-Learning-Based Applications

For some HALCON deep learning operators, the required licensed module is dynamic. This is the case if the respective operator can be used for methods of different modules. Thus, the license needed does not depend on the operator, but on the used model type of the used deep learning network. What kind of licenses may be required by an operator can be queried using `get_operator_info`.

See table 6.1 on page 32 for an overview which license modules enable inference and/or training for different types of deep learning models.

'type'	DLE [1]	DLP [2]	OCR/OCV [3]	3D [4]	Matching [5]
'3d_gripping_point_detection'					
Training				✓	
Inference				✓	
'anomaly_detection'					
Training		✓			
Inference		✓			
'classification'					
Training		✓			
Inference	✓[6]	✓			
'counting'					
Training					✓
Inference					✓
'deep_matching_3d '					
Training				✓	
Inference				✓	
'detection'					
Training		✓			
Inference	✓	✓			
'gc_anomaly_detection'					
Training		✓			
Inference		✓			
'generic'					
Training		✓			
Inference	✓	✓			
'multi_label_classification'					
Training		✓			
Inference		✓			
'ocr_detection'					
Training			✓		
Inference			✓		
'ocr_recognition'					
Training			✓		
Inference			✓		
'segmentation'					
Training		✓			
Inference	✓	✓			

Table 6.1: Deep learning model type-specific dynamic license modules.
[1]: Module 'Deep Learning Enhanced'
[2]: Module 'Deep Learning Professional'
[3]: Module 'OCR/OCV'
[4]: Module '3D Metrology'
[5]: Module 'Matching'
[6]: Exception: Not included if model is extended with Out-of-Distribution Detection.

6.5 How to Upgrade a License

If you upgrade a HALCON license to a newer version, your local sales partner provides you with a new license file which contains new license keys. This new license file should replace the old one in `%HALCONROOT%\license\license.dat`.

6.6 How to Check the Genuineness of HALCON (Windows)

To verify that your HALCON installation is genuine, you can check the digital signature of the file `halcon.dll`. Make sure that your Windows installation is fully updated. Otherwise, the signature verification might not work as expected.

- Right-click the file `%HALCONROOT%\bin\%HALCONARCH%/halcon.dll`.
- Select “Properties”.
- Select the tab “Digital Signature”.
- Select the signature “MVTec Software GmbH” and click “Details”.

If your version of HALCON is genuine, the dialog reads “This digital signature is OK”. Otherwise, your version of HALCON has been tampered with. This might lead to unexpected behavior of HALCON.

6.7 License Watchdog Process For Floating Licenses (Windows)

On Windows, socket communication is not possible during process termination. As a result, a process using HALCON on Windows cannot return a floating license automatically when it is terminated.

To avoid the need for HALCON applications to manually return floating licenses on Windows, HALCON provides the license watchdog `hlwd`. The watchdog is started as a separate process whenever HALCON acquires a floating license. If the HALCON process is terminated before it could return its license, the watchdog returns the license instead and then terminates itself.

Chapter 7

HALCON in a Docker Container

When developing or deploying your application using Docker containers on Linux, you need to include the HALCON software in your container image. This chapter describes how to modify your Dockerfile to install and run HALCON in a Docker container. To do so, you need to place the following files next to your Dockerfile:

- HALCON offline installer for Linux (see the "MVTec Software Manager" (SOM) documentation for details)
- Valid license file (see section 7.3 for the licensing options available)

It may also be helpful to look at the Docker example under `examples/docker/halcon-application` of your HALCON installation.

7.1 Include a HALCON Installation

7.1.1 Basic HALCON Packages

HALCON can be easily installed into a Docker image by adding a command execution line of the offline MVTec Software Installer (SOM) into the Dockerfile. To do this, you need to place the installer zip file (e.g., halcon-25.11.0.0-x64-linux.zip) next to your Dockerfile. You can then use the following line to install the HALCON Runtime package along with the Deep Learning package and the udev rules files, for example:

```
RUN unzip halcon-25.11.0.0-x64-linux.zip -d /tmp
RUN cd /tmp/halcon-25.11..0.0-x64-linux && \
    ./som install -f halcon-25.11.0 runtime --accept <hash value> && \
    ./som install -f halcon-25.11.0 deep-learning-thirdparty --accept <hash value> && \
    ./som install -f halcon-25.11.0 udev-rules --accept <hash value>
```

To complete the installation you will need to accept the EULA by adding the specific EULA hash <hash value> to the installation call. See the SOM documentation for details on how to obtain the hash value and the different installation and package options.

SOM will locate the HALCON installation in the directory `/opt/MVTec/Halcon-25.11-Progress`. In case your application does not use the rpath linking option to find the HALCON libraries you might have to add the library path to make sure that the HALCON language interface library is found (see also the corresponding description for your HALCON language interface in the Programmer's Guide)

```
ENV LD_LIBRARY_PATH=/opt/MVTec/Halcon-25.11-Progress/lib/$HALCONARCH
```

7.1.2 Required System Packages

To make sure that all system requirements listed in table 1.1 on page 9 needed by HALCON are installed you should also install the according system packages. Use the package manager of the Linux distribution your Docker image is based on. E.g., for Ubuntu 22.04, using the apt installation manager, this call could be:

```
RUN apt-get update && apt-get install -y --no-install-recommends \
    libx11-6 \
    libxext6 \
    && rm -rf /var/lib/apt/lists/*
```

For other versions of Ubuntu or other Linux distributions, the names of the packages may be different. Consult the packages website of your distribution at hand for the correct names.

7.1.3 HALCON Language Interface

HALCON/C and HALCON/C++ language interfaces already come with the runtime package of a basic HALCON installation and no additional work will be necessary. However, HALCON/Python and HALCON/.NET interfaces have to be installed with the according package manager of the specific language framework. E.g., you install HALCON/Python by using pip

```
RUN apt-get update && apt-get install -y --no-install-recommends \
    python3-pip \
    && rm -rf /var/lib/apt/lists/*

RUN pip install mvtec-halcon==25110 pillow numpy
```

Use the system package manager to install a package for `dotnet-sdk` if supported (like `apt` does for Ubuntu since 22.04). In case support is missing, you can download it from Microsoft. Afterwards, you can install HALCON/.NET using the NuGet package manager for .NET.

```
RUN apt-get update && apt-get install -y --no-install-recommends \
    wget && \
    wget https://packages.microsoft.com/config/ubuntu/22.04/packages-microsoft-prod.deb \
      -O packages-microsoft-prod.deb && \
    dpkg -i packages-microsoft-prod.deb && \
    rm packages-microsoft-prod.deb && \
    apt-get update && apt-get install -y --no-install-recommends \
    dotnet-sdk-7.0 \
    && apt remove -y wget \
    && rm -rf /var/lib/apt/lists/*

RUN dotnet add package MVTec.HalconDotNet -v 25110
```

See the Programmer's Guide for further language-specific requirements that might need to be set in the Docker image.

7.2 Include External Hardware

7.2.1 NVIDIA GPUs for Deep Learning

Using NVIDIA GPUs requires some additional setup on the host system: The latest graphics driver for the GPU at hand needs to be installed. NVIDIA Container Toolkit needs to be installed in order to allow Docker access to the GPUs. See the official NVIDIA documentation on installing the NVIDIA Container Toolkit for details.

On an Ubuntu system, the necessary steps are:

- Make the package manager aware of the external repository and install the NVIDIA Container Toolkit package
- Configure the Container Toolkit using the `nvidia-ctk` command
- Restart the Docker daemon

You can then verify that the setup worked by running `nvidia-smi` within the container:

```
sudo docker run --rm --runtime=nvidia --gpus all ubuntu nvidia-smi
```

The flags `-runtime=nvidia -gpus=all` are required in your Docker run command line for all containers using NVIDIA GPUs.

7.2.2 NVIDIA GPUs for OpenCL

The host system setup described in the previous subsection is also required here. In addition you need to install `clinfo` and verify that your GPU is reported correctly. Within the image, you need to install some packages for OpenCL and perform setup to make OpenCL aware of the NVIDIA GPUs:

```
RUN apt-get update && apt-get install -y --no-install-recommends \
    ocl-icd-libopencl1 \
    opencl-headers \
    clinfo \
    && rm -rf /var/lib/apt/lists/*

RUN mkdir -p /etc/OpenCL/vendors && \
    echo "libnvidia-opencl.so.1" > /etc/OpenCL/vendors/nvidia.icd
```

7.3 Licensing HALCON within a Docker Container

This section shows the different licensing options that are available for running HALCON in a Docker container. They describe licensing HALCON by

- using a remote dongle (section 7.3.1 on page 37) and
- forwarding a USB dongle into the container (section 7.3.2 on page 38).
- using a MVTec License Server for floating or cloud licenses (see documentation of the MVTec License Server product)

You can also use an evaluation license. See section 6.2 for further details.

In any case, you will have to copy the license file to the license directory of your HALCON installation in your Docker image. Therefore, add the following line to your Dockerfile:

```
COPY <license file> /opt/MVTec/Halcon-25.11/license/license.dat
```

If you intend to reuse the image on different machines, you may want to manage the license separately from the Docker image. In this case, you can optionally bind-mount the license directory as a volume.

The following subsections list additional installation steps for specific license types.

7.3.1 Remote Dongle

The use of the remote dongle is explained in section 6.3.3. Please read this chapter first.

As described, you have to modify the license file to add the "Remote Dongle" functionality. Add the additional line: `"server=host.docker.internal"`. The DNS name resolves to your host system IP address.

```
LICENSE MVTec_HALCON 25.11 MODULES=fc123bdomit VALID=2024-11-30 \
        ID=CM:3-3666322 FLAGS=PROGRESS_OK SESSIONS=1 \
        SIGNATURE="519B 0815 E95D C60C 954E \
        06FB 06FB 06FB 06FB 06FB 06FB 06FB DE1D E41F 2037 7637 F86E F55F 8C45 \
        BAC1 5FD4 8408 1CEA 06FB 34E2 36FC 06FB 9DE6 1B01 01BD 7444 DE09" \
        server=host.docker.internal
```

The remote dongle feature also requires the configuration of the CodeMeter runtime by enabling its "Network Server" feature. Then, grant client access to the IP range used by Docker's network. The IP range of Docker's default bridge network can be obtained using

```
docker network inspect bridge | grep -i subnet
```

When using a Linux host machine you may need to add `--add-host=host.docker.internal:host-gateway` to your Docker run command to allow Docker to communicate with the host's network via `host.docker.internal`.

7.3.2 Forward USB License Dongle Into Container (Linux Only)

This option requires a Linux host system as it is not possible to forward USB devices from Windows hosts into Linux containers.

Prior to interacting with Docker, you need to ensure that `hhostid` on the host system reports the dongle correctly. For help with this see section 8.2.1 on page 40.

You need the Linux kernel to assign a hidraw device to the dongle in order for this method to work. To make this happen, you need to ensure that the codemeter service is not running. To deactivate it use:

```
sudo systemctl disable --now codemeter.service
```

Note that for a non-root user in the container a Linux udev rule needs to be set up in the image to enable the access to this hostid.

```
RUN mkdir -p /etc/udev/rules.d/
RUN cp /opt/MVTec/Halcon-25.11/misc/linux/udev/rules.d/59-halcon-codemeter.rules \
    /etc/udev/rules.d/59-halcon-codemeter.rules
```

To grant the container access to the dongle when running the container, you need to forward the corresponding hidraw device. This can be obtained for instance by running the following after connecting the dongle to the host system:

```
grep HID_NAME /sys/class/hidraw/*/device/uevent | grep WIBU
```

To start the container (/dev/hidraw4 is an example here) you have to pass the device, e.g.:

```
docker run --rm -it --runtime=nvidia --gpus=all -v $PWD/src:/home/src \
    --device /dev/hidraw4 <docker_image>
```

Chapter 8

Troubleshooting

This chapter offers help for problems with the licensing mechanism (section 8.2 on page 40), when starting HDevelop or your own HALCON applications (section 8.3 on page 41), and other miscellaneous problems.

Note that throughout the chapter the environment variable `HALCONARCH` is referenced. See table 1.1 on page 9 for the possible values of this variable.

8.1 Problems During Installation

8.1.1 HALCON Variable Inspect (Visual Studio Extension)

Problem

HALCON Variable Inspect (VSIX) cannot be installed.

Note that you can use HALCON Variable Inspect 25.11 with previously installed versions of HALCON as well. To be able to use the new version with a previous version of HALCON Steady, rename the license file of HALCON 25.11 to, e.g., `license-25.11.dat`, and then copy it to the `%HALCONROOT%\license` directory of the older HALCON Steady installation.
See the Programmer's Guide, section 3.4 on page 24 for more information about HALCON Variable Inspect.

Cause

The problem is caused if users with different permissions are involved. If an older version of HALCON Variable Inspect has been installed as administrator, it cannot be upgraded by a general user.

Solution

The old HALCON Variable Inspect has to be uninstalled before the SOM package can be installed. To uninstall HALCON Variable Inspect from your Visual Studio:

1. Open Visual Studio.
2. Open the menu `Extensions` ▷ `Manage Extensions`.
3. Change to `Installed`.
4. Select `HALCON version Variable Inspect` and click `Uninstall`.
5. Try installing HALCON Variable Inspect again. If the installation still fails, continue with the next step.
6. Open a command prompt as administrator.
7. Execute the following command to uninstall all instances of HALCON Variable Inspect. Note that this will remove all instances of HALCON Variable Inspect in all installed versions of Visual Studio:

```
VSIXBootstrapper.exe /admin /uninstall:4159242D-A658-4E33-BD2F-7F4ED4CFC420
```

8. Try installing HALCON Variable Inspect again. If the installation still fails, continue with the next step.
9. Open a command prompt as user.
10. Execute the following command to uninstall all instances of HALCON Variable Inspect. Note that this will remove all instances of HALCON Variable Inspect in all installed versions of Visual Studio:

```
VSIXBootstrapper.exe /uninstall:4159242D-A658-4E33-BD2F-7F4ED4CFC420
```

8.2 Problems Concerning Licenses

If you encounter problems with your HALCON license even though your license file exists and is located in the correct directory, a first step is always to check if the host ID identifying your network card or dongle matches the entry in the license file (see the corresponding sections in chapter 6 on page 27). If the two do not match, please send the new identifying information to your local sales partner. See section 8.2.1 if you encounter problems with extracting the identifying information.

8.2.1 Extracting Host IDs

- `hhostid` **does not return a dongle ID**

  Please note that only HALCON dongles (orange) are reported by `hhostid`. Other dongles (e.g., blue MERLIC dongles) are not supported.

- `hhostid` **does not return a MAC address**

  This might happen if you call `hhostid` from a virtualized environment using locally administered MAC addresses. HALCON only supports MAC based licenses bound to universally administered, permanent MAC addresses. Locally administered MAC addresses are rejected.

- `hhostid` **shows the dongle ID on Linux only when run as root**

  If HALCON is run without root permissions, HALCON needs read/write permissions on the `hidraw` device created for the dongle. The following udev rule is required to achieve this:

```
# Udev rules to allow access to hidraw device for CodeMeter dongles

ACTION=="remove", GOTO="halcon-cm_end"

SUBSYSTEM=="hidraw", ATTRS{idVendor}=="064f", ATTRS{idProduct}=="2af9", MODE="666"

LABEL="halcon-cm_end"
```

  You can add this rule to `/etc/udev/rules.d` yourself, or copy the file `$HALCONROOT/misc/linux/udev/rules.d/59-halcon-codemeter.rules` to that directory. Note that the UDEV rule must be activated after installation which can be achieved by a reboot, for example.

- `hhostid` **shows no dongle ID on Linux even though a HALCON dongle is plugged in**

  In order to be able to use the dongle on Linux, you must be running kernel version 2.6.39 or later with the following configuration options enabled:

  - `CONFIG_HID`
  - `CONFIG_HIDRAW`
  - `CONFIG_TMPFS`

  These options are usually enabled by most Linux distributions. A common exception are embedded systems, for which one or more of these options may be disabled. In this case, `hhostid` is not able to access the dongle and a custom kernel including the required components must be built.
  To check if the running kernel has been built with `CONFIG_HIDRAW` enabled, run the following command:

```
gunzip -c /proc/config.gz | grep CONFIG_HIDRAW
```

  The other configuration options can be checked analogously.

8.2.2 HALCON/Python and Floating Licenses under Linux

If you are using HALCON/Python and floating licenses under Linux and encounter internal licensing errors (`HALCON error #2082: Internal licensing error: Could not initialize HTTP session for URL`), this might indicate that there is a problem loading the `libcurl.so` library shipped with HALCON. This library uses the OpenSSL libraries, which are usually provided by the system. Since some systems do not ship the OpenSSL 3 libraries, HALCON provides them as an extra SOM package. In case you encounter the licensing error above, ensure that this extra SOM package is not installed, by verifying that the `$HALCONROOT/lib/x64-linux/thirdparty` folder does **not** contain the `libssl.so.3` and `libcrypto.so.3` files. The problem arises because Python has already loaded the system `libcrypto.so.3` library, which can be incompatible with the version shipped with HALCON. In general, we always recommend using the system OpenSSL libraries if possible.

8.3 Troubleshooting in HDevelop or HALCON Applications

8.3.1 Troubleshooting with the hinfo utility

If you have trouble starting HDevelop or a HALCON Application, you can run the hinfo utility. This program will display various information about HALCON, including the values of environment variables relevant to HALCON, the version of HALCON, the license being used, etc. If no valid HALCON license can be found, hinfo will display the license and the error message explaining why the license cannot be used for any license it can find.

8.3.2 Startup Errors

This section explains miscellaneous error messages when starting HDevelop or your own HALCON applications and their reasons.

- **Error using license file**

  This error message might have several reasons:

  - The file `%HALCONROOT%\license\license.dat` is missing and/or not readable.
  - Your license is not valid on this machine.
  - If there is more than one user trying to use HALCON via remote access, the second user gets an error message.

- **HALCON stops recognizing the dongle**

  The HALCON dongle will lock itself if it is repeatedly unplugged and plugged back in within a short time (more than 50 times within 20 minutes). If this happens, the dongle will not respond for five minutes. Any attempt to access the dongle during this time will restart the five minute timeout.

  On Windows, if Wibu CodeMeter Runtime is not installed, starting and stopping HALCON is detected as being plugged in and out by the dongle. This means that if HALCON processes are rapidly started and stopped, the dongle will lock itself. Installing Wibu CodeMeter Runtime will avoid this problem as HALCON will then communicate with the dongle via the CodeMeter Runtime instead of accessing the dongle directly.

- **No license for this operator**

  The operator which you try to execute belongs to a HALCON module that is not licensed (see section 6.4 on page 30. Obtain a new license including this module.

- **hdevelop: Command not found** (Linux)

  Check your system environment variable `PATH`. It must include the path `$HALCONROOT/bin/$HALCONARCH`.

- **lib* : can't open file** (Linux)

  Check the system variable `LD_LIBRARY_PATH` (see section A.4 on page 47).

- **No help files for package <package-name> in directory <directory>**

  Possible reasons for this error message are:

- No files %HALCONROOT%\help* (if the package name is "system") or no help files in one of the user packages.
- If the package name is "system": Wrong HALCONROOT.
- Check the file permissions. Probably HDevelop cannot access important files.

- **Help file for package <package-name> is corrupt**
  Possible reasons for this error message are:
  - If the package name is "system": Inconsistent version of %HALCONROOT%\help* or wrong HALCONROOT.
  - If the package name is that of a user package: Inconsistent version of the help files of this package.

- **Can't open display** (Linux)
  If you see an error message like this you may have a wrong system variable DISPLAY and/or your program is not allowed to open a window by the specified X-server.

- **No refresh of window content on a Linux system**
  On some Linux systems the default behavior regarding occluded windows may be set in an inconvenient way for HALCON. The result is that if a window is temporarily occluded by another window, its content is not saved and restored anymore, i.e., windows remain "black" after uncovering. An example for this are all SuSE Linux distributions $\geq$ 7.0. The corresponding property is called "backing-store"; you can check the current setting of this property by typing (the following example corresponds to a SuSE 8.2 Linux system):

  ```
  xdpyinfo | grep backing-store
  ```

  which should result in the output like

  ```
  options:    backing-store YES, save-unders YES
  ```

  if the window content is saved and restored. You can change this behavior by changing the configuration file of your X server. It usually resides in /etc/X11/xorg.conf if you are using Xorg, or in /etc/X11/XF86Config if you are using XFree86. Consult your system's documentation if in doubt.

  You will probably need to become root to modify this file. Open the file in a text editor, find the section named "Device", and add the following option to this section:

  ```
  Section "Device"
    ...
    Option        "BackingStore" "True"
  EndSection
  ```

  Alternatively, you can modify the file Xservers residing in the directory /usr/lib/X11/xdm (or /opt/kde3/share/config/kdm in case of newer Linux versions), see your system's documentation. Note, that you probably need root privileges to modify this file. Append the option +bs (i.e., "plus backing-store") to the line that starts the local X server:

  ```
  :0 local /usr/X11R6/bin/X :0 vt07 +bs
  ```

  Now, stop and start the X server again (by using the appropriate commands or by rebooting your computer); the command xdpyinfo now should yield the output shown above.

Appendix A

More on the Installation

A.1 Software Packages

Depending on your operating system, the following meta packages are available via SOM:

Development
: The development version of HALCON, including interfaces to supported programming languages, the full documentation, and all example programs, including all necessary images and 3D models.

Runtime
: The runtime version of HALCON, i.e., only the set of libraries that are necessary to run a HALCON application. Naturally, this type of installation encompasses neither documentation, examples nor images. However, it contains HDevEngine and the provided external procedures.

AI Accelerator Interfaces
: Plugins for optimizing the performance of deep learning, like the “TensorRT Inference PlugIn”.

Image Acquisition Interfaces
: The most widely used interfaces. Additional interfaces can be downloaded under https://www.mvtec.com/products/interfaces.

Digital I/O Interfaces
: Interfaces for several I/O devices. Additional interfaces can be downloaded under https://www.mvtec.com/products/interfaces.

Variable Inspect Extension for Visual Studio
: Extension that simplifies the debugging of applications in Visual Studio.

The packages above are based on the following packages:

EULA
: The End User License Agreement.

Installation maintenance
: Support files for installation and uninstallation.

Example programs
: Example programs for all supported programming languages.

Example data
: 3D-Models and images.

Runtime files
: All files needed for runtime installation, and the HALCON tools “hhostid” and “hcheck_parallel”.

Runtime files (general)
: Support files needed for runtime installation (Linux).

Development files
: Include files for development, the entire product documentation, and the HALCON tools “hbench”, “hinfo”, and “hcomp”.

HDevelop
: HALCON’s interactive development environment (HDevelop) and a command line tool to run HDevelop scripts (hrun).

HDevelop XL
: The large-image version of HDevelop and a command line tool to run HDevelop scripts (hrun).

Deep learning core
: Pretrained classifiers for deep learning applications.

Deep learning data
: Images and pretrained classifiers for deep learning example programs.

Deep learning third party library
: CUDA libraries.

Microsoft Visual C++ Redistributable
: Files for the development with C++.

A.2 The Installed File Structure

The default installation targets depend on the operating system. For Windows and Linux users they also depend on the installation mode (“User mode” or “System mode”).

Windows

User mode

Install target (programs)	`%LOCALAPPDATA%\Programs\MVTec`
Install target (data)	`%APPDATA%\MVTec`

System mode

Install target (programs)	`%PROGRAMFILES%\MVTec`
Install target (data)	`%PUBLIC%\Documents\MVTec`

Linux

User mode
Install target (programs) `$HOME/MVTec`
Install target (data) `$HOME/MVTec`

System mode
Install target (programs) `opt/MVTec`
Install target (data) `opt/MVTec`

A.2.1 Main Directory

In the following, the most important directories and files are described briefly. Note that, depending on your installation, not all directories may be present.

`bin`

Contains HALCON programs and HALCON tools, for example HDevelop, in subdirectories corresponding to the different platforms. The subdirectories `dotnet20` and `dotnet35` contain the HALCON/.NET assemblies based on .NET Framework 2.0, and .NET Framework 3.5, respectively.

Windows only: This directory also contains the DLLs of the HALCON libraries, the DLLs for the supported image acquisition interfaces, and I/O device interfaces.

`calib`

Contains description files for the calibration plates, which you can use to calibrate your camera.

`doc`

Contains the whole documentation in English, e.g., the User's Manuals and the Operator Reference.

`doc_de_DE`

Contains the German version of the Operator Reference.

`doc_ja_JP`

Contains the Japanese version of the Operator Reference.

`examples`

Contains example programs, images, and models. **To experiment with the examples without modifying the distributed versions, you can create a private copy in your own working directory.**

`filter`

Contains predefined filter masks.

`genicam`

Contains the underlying GenAPI runtime software for the GigE Vision, GenICamTL, and USB3Vision image acquisition interfaces.

`help`

The files in this directory act as the HALCON database, i.e., they provide information about HALCON to HDevelop and to all HALCON applications. In particular, they contain the operator database. The XML files starting with `index_manuals` contain the index data of the manuals. The XML files starting with `index_examples` contain the data for the `Browse Examples` dialog in HDevelop and those starting with `tip_of_the_day` contain the information for the Tip of the Day that appears in the HDevelop Start Dialog.

`include`

Contains the header files that are necessary to use HALCON within the programming languages C or C++.

`lib`

Contains the HALCON libraries.

`license`

The license file must be placed here (see section 6.1 on page 27.)

`lut`

Contains predefined look-up tables.

`misc`

Contains miscellaneous files for the installer, e.g., the GigE Vision filter driver.

Windows only: In addition, the installer of the HALCON `Variable Inspect` can be found here. Please refer to the Programmer's Guide, section 3.4 on page 24, for more information about this extension.

`ocr`

Contains pretrained fonts.

`procedures`

Contains external procedures for HDevelop and HDevEngine.

`som.d`

Database for installed SOM packages, i.e., all installed packages are registered in this subdirectory.

A.2.2 Machine Configuration Data

The following directory contains machine-specific configuration data, e.g., results of the utility hcheck_parallel or the operator `optimize_aop`.

Windows	`C:\ProgramData\MVTec\HALCON-25.11` (`%ProgramData%...`)
Linux	`/opt/halcon` (`$HALCONROOT`)

A.2.3 User Configuration Data

The following directory contains user-specific configuration data, e.g., HDevelop preferences (`HDevelop.ini`).

Windows	`%APPDATA%/MVTec`
Linux	`$HOME/.hdevelop/MVTec`

A.3 HALCON tools

The following HALCON tools are delivered:

`hbench`	A benchmark tool for the HALCON machine vision library to compare the capability of various machines. E.g., memory transfer can be measured with the `hbench` option (`-memory`).
`hcheck_parallel`	A tool checking a multiprocessor computer about its potential for speeding up HALCON operators by parallel processing, see Programmer's Guide, section 2.1.1 on page 15.
`hcomp`	The HALCON compiler. See Extension Package Programmer's Manual, section 7.1 on page 97
`hhostid`	A tool to find available host IDs for licensing of HALCON, see section 6.1 on page 28.
`hrun`	A command line utility to run HDevelop scripts, see HDevelopUser's Guide, appendix C.2 on page 326.

These tools are in the directory `bin` or a subdirectories corresponding to the platforms of your HALCON installation, see section A.2 on page 44.

The following command displays possible options and a synopsis for every tool:

```
toolname --help
```

A.4 HALCON's Environment Variables

Most of the configuration necessary to work with HALCON amounts to setting environment variables, e.g., to tell HALCON the directories where to find images or extension packages etc. These environment variables are described below, after some information regarding the different platforms.

A.4.1 Setting Environment Variables Under Windows

SOM automatically sets the necessary environment variables, e.g., `HALCONROOT`, `HALCONEXAMPLES`, `HALCONIMAGES`, and `PATH` (see below). To take a look at these settings, search for "Environment" using the system search, and select the search result "Edit the system environment variables" or "Edit environment variables for you account". You can add or modify a variable by entering the name of a variable and the desired value. If a value consists of multiple items, e.g., the variable `PATH`, which may contain multiple directories, those items must be separated by *semicolons*.

A.4.2 Setting Environment Variables Under Linux

As described in section 3.2.1.4 on page 19, you must set the necessary environment variables in a login script or a shell resource script.

A.4.3 HALCON-Specific Environment Variables

- `HALCONROOT`
  This is the most important environment variable. It designates the directory where HALCON is installed.

  If this variable is unset at the time HDevelop is run, or when the HALCON library is loaded, its value will be inferred from the path the executable or the library resides in, respectively. From this path the trailing part `bin\%HALCONARCH%` or `lib\%HALCONARCH%` will be removed. The variable `HALCONROOT` will then be set to the resulting path temporarily.

  Based on this variable, the system switches to subdirectories, which are important for running HALCON. Some of them are listed below; the HALCON file structure is described in section A.2 on page 44.

  - `%HALCONROOT%\help`
    The files in this directory act as the HALCON information database (see section A.2 on page 44 for more information).
  - `%HALCONROOT%\doc\html\reference\operators`
    HDevelop expects the HTML files of the operator reference in this directory.
  - `%HALCONROOT%\license`
    This directory contains the *license file* necessary for using HALCON (see chapter 6 on page 27).
  - `%HALCONROOT%\examples`
    If the variable `HALCONEXAMPLES` (see below) is not set, the system looks for example programs in this directory.
  - `%HALCONEXAMPLES%\images`
    If the variable `HALCONIMAGES` (see below) is not set, the system looks for image files in this directory.
- `HALCONEXAMPLES`
  This environment variable designates the directory where HALCON example programs are installed.
- `HALCONIMAGES`
  The system uses this environment variable to search for image files specified by a relative path. As a rule it contains several directory names, separated by semicolons (Windows) or colons (Linux).
- `HALCONARCH`
  This variable designates the used platform. More details can be found in section 1.4 on page 8.

- HALCONEXTENSIONS
  This is a list of directories in which user-defined extension operators (so-called *extension packages*) are kept. Each package consists of a number of operators linked into a shared library, plus the additional operator documentation in help files and HTML files. See section 3.6 on page 21 for information on how to install an extension package, and the Extension Package Programmer's Manual for details on creating your own extension packages.

- HALCONSPY
  If this environment variable is defined (regardless of the value) *before you start* a HALCON program, the HALCON debugging tool *HALCON Spy* is activated. This corresponds to call the HALCON operator `set_spy` with the parameters `"mode","on"` *within* a HALCON program. The main difference between the two modes for activating HALCON Spy is that by defining HALCONSPY it is possible to monitor an already linked HALCON program during runtime without modifications. For further information on how to use HALCON Spy and how to parameterize it via this environment variable please refer to the Programmer's Guide, section 3.1 on page 23.

- HALCON_LICENSE_FILE
  This environment variable can be set to the full path of a specific license file. In this case, the given license file is used and no further search is performed. Usually, the license file is searched in the current working directory and the installed `license` directory.

A.4.4 General Environment Variables

- PATH
  Windows: During the installation, the directory %HALCONROOT%\bin\%HALCONARCH% is automatically added to the system variable PATH.

  Linux: If you want to start HDevelop from an arbitrary directory, you must include the HALCON program path $HALCONROOT/bin/$HALCONARCH in the system variable PATH.

- LD_LIBRARY_PATH (Linux only)
  Please include the HALCON library path $HALCONROOT/lib/$HALCONARCH in the system variable LD_LIBRARY_PATH. This is necessary both for running HDevelop and for creating stand-alone applications.

- DISPLAY (Linux only)
  The system uses this environment variable to open windows. It is used in the same way as for other X applications.

- HOME (Linux only)
  This system variable points to your home directory.

Index

compatibility (HALCON), 8
cuBLAS library, 25
cuDNN library, 25
cybersecurity, 15

Deep learning third-party components, 25
development license, 29
development version, 7
DISPLAY (environment variable), 48
dynamic modules in HALCON, 31

environment variables (general), 48
environment variables (HALCON), 47
evaluation license, 28

get host ID manually, 28
get host ID of network card, 29
get required HALCON modules, 30

HALCON limitations, 12
HALCON_LICENSE_FILE (environment variable), 48
HALCONARCH (environment variable), 47
HALCONEXAMPLES (environment variable), 47
HALCONEXTENSIONS (environment variable), 48
HALCONIMAGES (environment variable), 47
HALCONROOT (environment variable), 47
HALCONSPY (environment variable), 48
hhostid, 28, 40
HOME (environment variable), 48

install extension package, 21
install HALCON, 17
install image acquisition interface, 21
installed file structure, 44

LD_LIBRARY_PATH (environment variable), 48
license, 27
    installation, 19
    overview, 14
    troubleshooting, 40, 41

maintenance release, 8

node-locked development license bound to dongle, 29
node-locked development license bound to network card, 29

PATH (environment variable), 48

runtime license, 30

runtime version, 7

set environment variables
    Linux, 47
    Windows, 47
Software Manager (SOM), 17
software packages, 43
switch between HALCON versions, 20
system mode, 17
system requirements, 8

troubleshooting (miscellaneous), 41
troubleshooting for get host ID, 40

uninstall HALCON, 23
update HALCON, 20
upgrade license, 32
user mode, 17

Variable Inspect Extension for Visual Studio
    troubleshooting, 39
version (HALCON), 8

WOW64, 8