HALCON provider

MVTec image processing

Version 1.2.0

User’s guide

Jun 28, 2017

[Remark]

[Revision history]

Version	Date	Content
1.0.0.0	2009-06-22	First edition. Supported HALCON 9.0.1.
1.1.0.0	2010-5-11	Supported HALCON 8.0.4 and HALCON 9.0.2. Added CaoTask class implementation.
1.1.0.1	2011-03-11	Added the description of provider registration tool.
1.1.0	2012-07-17	Change the version rule of document.
1.1.0.43	2012-09-28	Supported HALCON 10.0.3 and HALCON 11.0.0.1.
1.1.0.108	2014-08-18	Supported HALCON 11.0.3.
1.1.0.138	2015-06-19	Supported HALCON12, HALCON11.0.4, HALCON10.0.4, and HALCON9.0.4
1.2.0.0	2016-02-25	Supported HALCON12.0.1 and HALCON11.0.5.
1.2.0.7	2016-05-11	Supported HALCON12.0.2.
1.2.0.29	2016-12-06	Supported HALCON13
1.2.0.56	2017-06-28	Supported HALCON13.0.1 and HALCON12.0.3

[Supported models]

Model	Version	Note
HALCON 8	8.0.4	May not work properly if the version is older than 8.0.4 .
HALCON 9	9.0.4	May not work properly if the version is older than 9.0.4.
HALCON 10	10.0.4	May not work properly if the version is older than 10.0.4.
HALCON 11	11.0.5	May not work properly if the version is older than 11.0.5.
HALCON 12	12.0.3	May not work properly if the version is older than 12.0.3.
HALCON 13	13.0.1	May not work properly if the version is older than 13.0.1.

[Attention]

To use this provider, you need to purchase a license separately. Note that this license does not include runtime license of HALCON.

Contents

1. Inroduction

This is a user's guide of HALCON provider that has been developed to use HALCON, which is the image processing library manufactured by MVTec, via ORiN.

Note: To use HALCON provider, you need to install HALCON in your computer. This provider does not support HALCON 64-bit version. You are required to register the provider registry after the installation of HALCON. For about the way of registration, see Table 2-1.

2. Overview of provider

2.1. Overview

HALCON provider is a gateway provider that calls HALCON from ORiN. This provider enables to use HALCON's image processing operators, external procedures, and programs created by HALCON's integrated development environment "HDevelop" on the ORiN CAO engine without any modification. (Figure 2-1).

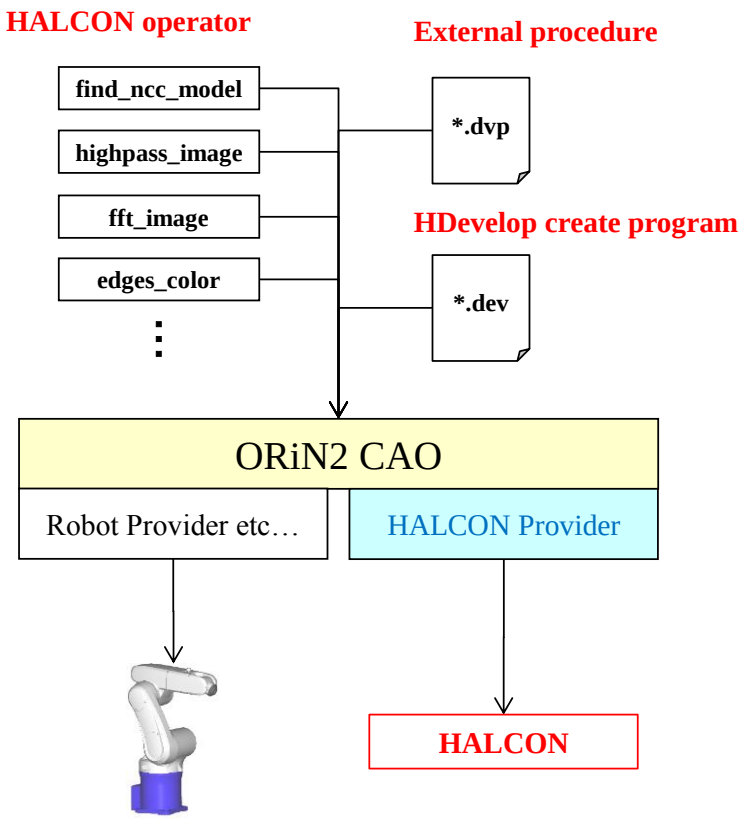

Figure 2-1 Overview of provider

2.2. Installation

The file of provider you use differs depending on the major version of your HALCON. Please register /deregister manually according to your HALCON's version.

Table 2-1 HALCON provider

File name	CaoProvHALCON.dll
ProgID	CaoProv.HALCON
Registration[1]	regsvr32 CaoProvHALCON.dll
Deregistration	regsvr32 /u CaoProvHALCON.dll

[1] This is not registered/deregistered automatically even if it is installed/uninstalled by ORiN SDK. Please register/deregister manually.

2.3. Adding a license

To activate this provider, install HALCON and ORiN2 SDK, and then enter HALCON Provider's license. If you do not have a legitimate HALCON provider license, use the following trial license.

HLGY-YZZ2-QL95-MDLG (valid for three months)

[How to add a license]

1. Start CaoConfig (Figure 2-2) and then select [Cao Provider] tab.
2. From the provider list, select [HALCON CAO Provider].
3. On the License pane, click […] button.
4. On the ORiN2 License Manager, click [Add] button.
5. Enter a license key and then click [OK] button.
6. Click [Close] button to close CaoConfig.

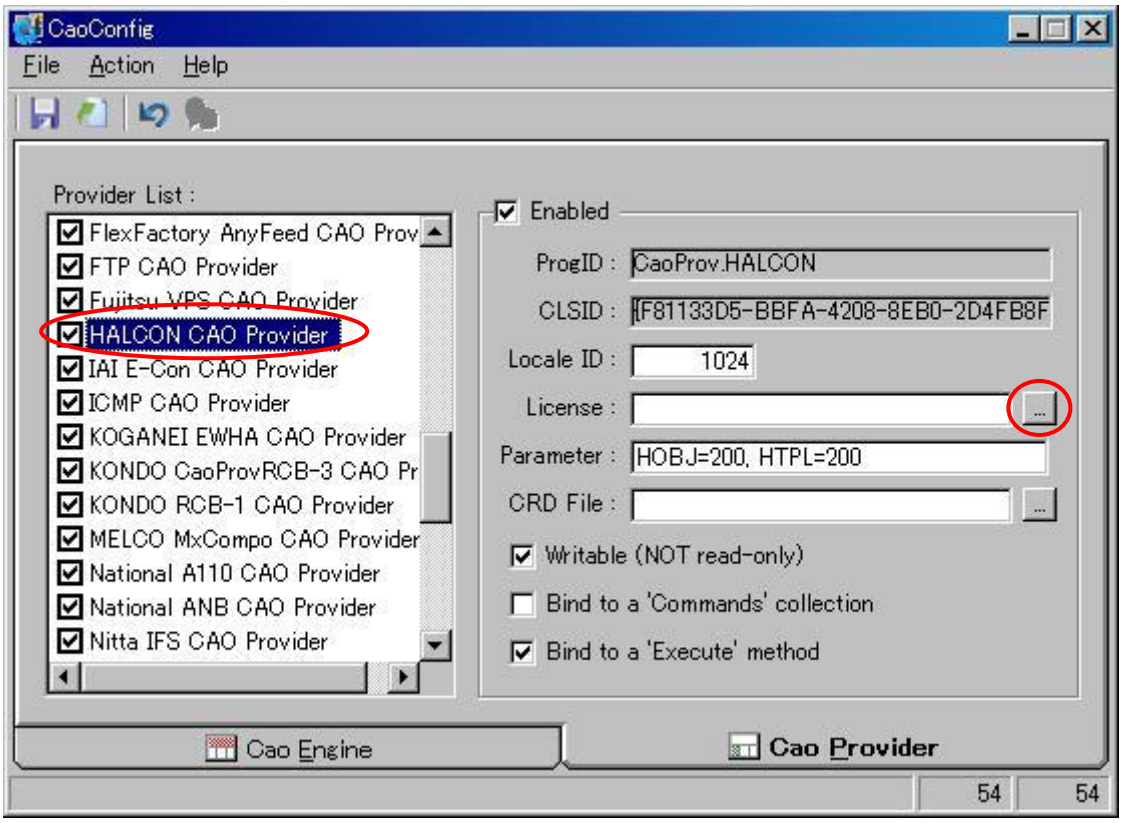

Figure 2-2 Adding a license on CaoConfig window

2.4. Registration

Register the internal memory settings in registry. To change the settings, use CaoConfig (Figure 2-2). The following table shows how to register the parameters.

Table 2-2 Parameter character string of CaoConfig

Parameter	Description
HOBJ=<Array size>	Specify the size of Hobject array. Specify 1 or larger value. (Default:200)
HTPL=<Array size>	Specify the size of HTuple array. Specify 1 or larger value. (Default:200)

2.5. Internal memory

HALCON provider has two fixed-length Data type[2] memory areas; Hobject and HTuple. Inside of this provider, HALCON's image processing operators are executed based on the Hobject and HTuple data. These data are stored in the memory addresses (indices) specified by the caller as input values. Execution results will be stored in the memory addresses (indices) that the caller of provider specifies as output destination.

2.6. Calling HDevelop external procedures

Programs created by HALCON's integrated development environment "HDevelop" can be called by CaoController::Execute method. To call such programs, convert them to external procedure files (*.dvp). For how to create HALCON's external procedure file (*.dvp), please refer to HDevelop Users Guide of HALCON.

To call an external procedure file, you are required to register the folder path of the file. To add a new path, use CaoController::Execute("AddProcedurePath", <bstrPath:BSTR >).

2.7. Calling HDevelop program

To call an HDevelop program file (*.dev), which is created by HALCON's integrated development environment HDevelop, generate a CaoTask instance of the program, and then call the instance with CaoTask::Start method.

2.8. Message transfer function

With a message transfer function of Cao engine, you can store any image data transferred by other provider into the internal memory. Image data of transferred message must be VARIANT type data (VT_UI1 | VT_ARRAY) of DIB (Device Independent Bitmap) format. At the timing of message reception, OnMessage event of CaoController class is issued to inform the reception state. To specify the storage address of received data, use CaoVariable::put_ID()property of a system variable "@MSG".

[2] Hobject and HTuple are data types defined in HALCON.

2.9. Method and Properties

2.9.1. CaoWorkspace::AddController method

This method allocates the internal memory areas of Hobject and HTuple in the inside of provider and initializes HALCON. The internal memory allocation is done only the first time of AddController execution. Once this method has been called, this internal memory areas will be used from the next time no matter what application calls the internal memory areas.

Syntax AddController(<bstrCtrlName:BSTR>, <bstrProvName:BSTR>,
<bstrPcName:BSTR > [, <bstrOption:BSTR>])

bstrCtrlName : [in] Controller name
bstrProvName : [in] Provider name. fixed to ”CaoProv.HALCON”
bstrPcName : [in] Option character string ＝”<option 1>, <option 2>, …”
bstrOption : [in] Option character string ＝”<option 1>, <option 2>, …”

Table 2-3 Option character strings of CaoWorkspace::AddController

Option	Description
HOBJ=<Array size>	Specify the size of Hobject array. Specify 1 or larger value. If this option is not specified, the size is determined by the same name’s parameter that has been registered in the registry. (Default:200)
HTPL=<Array size>	Specify the size of HTuple array. Specify 1 or larger value. If this option is not specified, the size is determined by the same name’s parameter that has been registered in the registry. (Default:200)

2.9.2. CaoController::AddVariable method

System variable described in Table 2-4 is implemented.

Syntax AddVariabler(<bstrVarName:BSTR> [, <bstrOption:BSTR>])

bstrVarName : [in] Variable name
bstrOption : [in] Option character string ＝”<option 1>, <option 2>, …”

Table 2-4 Option character strings of CaoController::AddVariable

Option	Description
ID=<Internal memory address >	For the system variable “@MSG.” only. Specify the internal memory address of Hobject to be stored. If this option is not specified, the end of data array will be specified. (default: 200)

2.9.3. CaoController::AddTask method

Generate an execution task of a program that is created by HALCON's integrated development environment "HDevelop".

Syntax AddTask(<bstrTaskName:BSTR> [, <bstrOption:BSTR>])

bstrTaskName : [in] Absolute path of a program file (*.dev) created by HDevelop.

bstrOption : [in] Option character string (not used)

2.9.4. CaoController::Execute method

This method executes a HALCON's image processing operator or an external procedure specified by the first argument of this syntax. Parameter array consists of four elements and each of them corresponds to the HALCON operator's four arguments (input object, output object, input control, and output control) from the top. For each element of parameter array, specify a memory address where the parameter is stored. To specify a memory address, use VT_I4 (| VT_ARRAY). To specify a null character, use VT_BSTR. For information about parameter specification of HALCON's image processing operator, please refer to the HDevelop Operator Reference Manual provided with HALCON.

Syntax Execute(<bstrCmd:BSTR > [, <vntParam:VARIANT>])

bstrCmd : [in] HALCON operator name

vntParam : [in] Parameter array

vntParam = Array(< vntP1:VARIANT >, < vntP2:VARIANT >,
< vntP3:VARIANT >, < vntP4:VARIANT >)

vntP1 : [in] Input Hobject. To specify two or more objects, use an array. If this entry is not necessary, enter a null character.

vntP2 : [out] Output Hobject. To specify two or more objects, use an array. If this entry is not necessary, enter a null character.

vntP3 : [in] Input HTuple. To specify two or more objects, use an array. If this entry is not necessary, enter a null character.

vntP4 : [out] Output HTuple. To specify two or more objects, use an array. If this entry is not necessary, enter a null character.

Example :

```
Execute( "reduce_domain", Array( Array( 1, 2 ), 10, "", "" ) )
Execute( "set_ncc_model_origin", Array( "", "", Array( 11, 12, 13 ), "" ) )
Execute( "get_image_pointer3", Array( 1, "", "", Array( "", "", "", 5, 6, 7 ) )
```

2.9.5. CaoController::get_VariableNames property

Return the list of variable names supported by CaoController object.

2.9.6. CaoController::OnMessage event

The following table shows OnMessage events implemented in CaoController class.

Table 2-5 Implementation status of OnMessage events

No.	Data type	Content	Description
1	VT_BSTR	Error message	Return the execution status of HDevelop program task. End normally: NULL, End abnormally: Error message
2	VT_I4	Message reception state	Return the reception status of In-process message. Received successfully: 0, Received invalid data[3]: 1, Failed to receive: other than 0

2.9.7. CaoVariable::get_ID property

When this property is used for the system variables @HOBJ, @HTPL, and @MSG, this property obtains the memory address allocated to the current variable object. Data type is VT_I4. This property cannot be used if it is created on a CaoTask object.

2.9.8. CaoVariable::put_ID property

When this property is used for the system variables @HOBJ, @HTPL, and @MSG, this property directly changes the memory address allocated to the current variable object to the specified value. If this property is created on a CaoTask object, the content of the current variable object is copied and pasted to the specified memory address.

2.9.9. CaoVariable::get_Value property

Obtain a value stored in the memory that is allocated to the current variable object. When the data type is Hobject, if the stored data is HALCON's image type, it will be converted to VARIANT type data (VT_UI1 | VT_ARRAY) of DIB (Device Independent Bitmap) format, and then output. If the stored data is other HALCON data type, an error will occur. When the data type is HTuple, it will be converted to VARIANT type data and then output.

2.9.10. CaoVariable::put_Value property

Specify a value to the memory allocated to the current variable object. When the data type is Hobject, data to be specified must be VARIANT type data (VT_UI1 | VT_ARRAY) of DIB format. If other data type is used,

[3] Any data other than VARIANT type data (VT_UI1 | VT_ARRAY) that expresses DIB (Device Independent Bitmap) .

an error will occur. When the data type is HTuple, VARIANT type data (integer, point, character string) can be set as-is. This property cannot be used if it is created on a CaoTask object.

2.9.11. CaoTask::AddVariable method

This method implements variables that have the same names as the ones used in HDevelop program which is created as a task. If an option character string is not specified, the specified variable name will be recognized as a local variable's name. For operation about object, please refer to CaoVariable object.

Syntax AddVariable(<bstrVarName:BSTR> [, <bstrOption:BSTR>])

bstrVarName : [in] Variable name used in the inside of HDevelop program.

bstrOption : [in] Option character string

Table 2-6 Option character strings of CaoTask::AddVariable

Option	Description
GLOBAL	Specified variable name is recognized as a name of global variable.

2.9.12. CaoTask::Start method

Execute an HDevelop program that is created as a task. Specifying the option character string will determine whether the task executes synchronously or asynchronously. If asynchronous execution is selected, OnMessage event of CaoController class is issued when the task is completed.

Syntax Start(<lMode:I4> [, <bstrOption:BSTR>])

lMode : [in] Start mode (1 : 1 cycle, 2 : Repeat, 3 : Step, 4 : Step back)

bstrOption : [in] Option character string ＝"<option 1>, <option 2>, …"

Table 2-7 Option character strings of CaoTask::Start

Option	Description
ASync	Execute tasks asynchronously. (default setting)
Sync	Execute tasks synchronously.
<Timeout period>	Control is returned to the caller once this timeout period [ms] has passed.

2.9.13. CaoTask::Stop method

Stop an HDevelop program that is created as a task.

Syntax Stop(<lMode:I4> [, <bstrOption:BSTR>])

lMode : [in] Stop mode(0 : default, 1 : Instantaneous, 2 : Step, 3 : Cycle, 4 : Initialization)

bstrOption : [in] Option character string (not used)

2.9.14. CaoTask::Delete method

Delete a task and unload HDevelop program being loaded.

2.9.15. CaoTask::get_FileName property

Return the absolute path of the currently loaded HDevelop program file.

2.9.16. CaoTask::get_VariableNames property

Obtain the variable name list used in an HDevelopment program that is created as a task. If option character strings is not specified, all local variable names used in the inside of HDevelop program will be returned.

Syntax get_VariableNames([<bstrOption:BSTR>])

bstrOption : [in] Option character string ＝”<option 1>, <option 2>, …”

Table 2-8 Option character strings of CaoTask::get_VariableNames

Option	Description
HOBJ	Return an Hobject local variable name used in the inside of program only.
HTPL	Return an HTuple local variable name used in the inside of program only.
GLOBAL_HOBJ	Return an Hobject global variable name used in the inside of program only.
GLOBAL_HTPL	Return an HTuple global variable name used in the inside of program only.
GLOBAL	Return all global variable names used in the inside of program.

2.10. Variable list

2.10.1. Controller class

Table 2-9 Controller class system variable list

Variable name	Data type	Description	Attribute	
			get	put
@HOBJ	VT_UI1 | VT_ARRAY	System variable to operate Hobject array directly.	✓[4]	✓[5]
@HTPL	VT_VARIANT	System variable to operate HTuple array directly.	✓	✓
@HOBJMAX	VT_I4	A variable to get the maximum size of Hobject array.	✓	
@HTPLMAX	VT_I4	A variable to get the maximum size of HTuple array.	✓	
@MSG	VT_UI1 | VT_ARRAY	System variable that stores a received message.	✓	✓

[4] Available only when Hobject type data expresses image data.
[5] Available only when VARIANT type data is "VT_UI1 | VT_ARRAY" data that expresses DIB (Device Independent Bitmap).

Appendix A. Error code list

Error codes of provider are HRESULT type. For details about HRESULT type, seethe following URL.

<http://msdn2.microsoft.com/en-us/library/bb401631.aspx>

Error codes used in this provider is classified into two types; “Standard error” which is defined in Microsoft Windows as a standard and “Custom error” which is defined exclusive to the provider. Standard errors are common error codes that are defined in Winerror.h. Custom errors are local error codes that are defined in the provider. The following table shows the list of error codes.

Table A-1 HALCON provider error code

Standard error (quoted)		
Number	Macro name	Description
0x00000000	S_OK	No error occurred
0x80004001	E_NOTIMPL	Not implemented
0x80070057	E_INVALIDARG	One or more arguments are invalid
0x80004005	E_FAIL	Unspecified error
Custom error		
Number	Category	Description
0x**80014**201	Provider license error	Provider license is invalid. Please purchase a license.
0x**80014**001 or larger	Provider internal exception error.	This error occurs when process ends abnormally in the inside of provider.
0x**8017**0000 or larger	HALCON operator error	If an error occurs at the HALCON operator execution, the sum of an error code returned by HALCON operator and 0x**8017**0000 will be returned. For information about error code of HALCON operator, refer to HALCON’s manual.