



Tutorial

Halcon-Befehle

Verwendung von Halcon-Befehlen mit dem vicosys

```
Rechenscript (Ruby)
# Run Filter
reval.error.testresults = VC::Command.run(%{
  "HalconScript": {
    "script": {
      "selection": "fixedvalue",
      "fixedvalue": "filter bdev"
    },
    "methods": {
      "Selection": "fixedvalue",
      "fixedvalue": "run_filter"
    },
    "inputs": [
      {
        "identifier": "InputImage",
        "source": "selection",
        "Selection": "himage_from_grey_page",
        "himage_from_grey_page": "-1"
      }
    ],
    "outputs": [
      {
        "identifier": "FilterSize",
        "source": "selection",
        "Selection": "result",
        "result": {
          "first": "11",
          "last": "-1"
        }
      }
    ]
  },
  "puts error"
})
```



Impressum

Herausgeber / Hersteller	Vision & Control GmbH Mittelbergstraße 16 98527 Suhl, Deutschland Telefon: +49 (0) 3681 7974-0 Telefax: +49 (0) 3681 7974-33 www.vision-control.com
Dokumentenname	Halcon-Befehle-de-1.1
Erstausgabedatum	11.10.2021
Änderungsdatum	06.12.2021
Copyright	© Vision & Control 2021

Urheberrecht

Weitergabe sowie Vervielfältigung dieses Dokumentes, Verwertung und Mitteilung seines Inhaltes sind verboten, soweit nicht ausdrücklich gestattet. Zu widerhandlungen verpflichten zu Schadensersatz.

Alle Rechte für den Fall der Patent-, Gebrauchsmuster- oder Geschmacksmustereintragung sowie Nutzungsrechte im Rahmen des Urheberrechts vorbehalten.

vicotar®, vicolux®, pictor®, vicosys® und vcwin® sind eingetragene Warenzeichen der Vision & Control GmbH.

Die Nennung von Produkten und Marken anderer Hersteller oder Anbieter dient ausschließlich zur Information.

Gültigkeit

Das vorliegende Dokument hat Gültigkeit für die folgenden Geräte sowie die abgeleiteten Gerätekonfigurationen.

- Mehrkamerasytem vicosys 5300
- Mehrkamerasytem vicosys 5400
- Mehrkamerasytem vicosys 19001



Inhaltsverzeichnis

1 Vorwort.....	3
2 Bildverarbeitungscode mit HDevelop erstellen.....	4
3 vicosys Halcon Befehlscode erstellen.....	5
4 Befehlscode mittels Rubyscript ausführen.....	6

1 Vorwort

Halcon-Befehle können mittels Rechenscript in die Mehrkamerasysteme vicosys eingebunden werden.

Dieses Dokument erläutert Ihnen den Umgang mit den von Vision & Control zur Verfügung gestellten Tools und gibt ihnen einen kurzen Überblick über die Methoden.

Voraussetzung

Es wird vorausgesetzt, dass ein PC mit einem aktuellen vcwin vorhanden ist und dass Kenntnisse im Umgang mit vcwin und HDevelop bestehen. Sie benötigen folgende Komponenten:

- ein Mehrkamerasystem (vicosys 5x00 oder vicosys 19001 ab Softwareversion 4.16.300 und gültiger Halcon-Lizenz)
- die Bediensoftware vcwin (Version 2.33 oder höher)
- die Programmierumgebung HDevelop

2 Bildverarbeitungscode mit HDevelop erstellen

HINWEIS

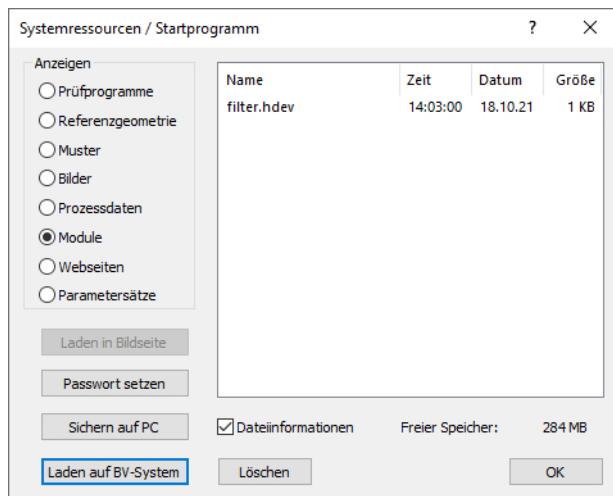
Testcode, wie z.B. das Laden von Testbilder kann in der main bleiben. Dieser wird beim Laden im vicosys nicht ausgeführt. Jedoch verhindern Syntaxfehler im Testcode das Laden des Programms.

- Erstellen Sie den Bildverarbeitungscode mit HDevelop.
- Lagern Sie Ihre Bildverarbeitungscode in eine Prozedur aus.



```
Program Window - run_filter()
run_filter( InputImage : OutputImage : FilterSize : )
gauss_filter( InputImage, OutputImage, FilterSize )
return ()
```

- Übertragen Sie das gespeicherte *.hdev Programm in den Ordner **Module** auf dem vicosys. (Menü **Kommunikation > Systemressourcen / Startprogramm > Module**)



3 vicosys Halcon Befehlscode erstellen

Den JSON-Befehlscode für den vicosys Befehl können Sie mit dem Hilfsprogramm DemoGUI erstellen oder selber schreiben. Das Hilfsprogramm DemoGui befindet sich auf Ihrem vicosys.

Öffnen Sie im Browser den Pfad: [http://\[IP-vicosys\]/jsongui/index.html](http://[IP-vicosys]/jsongui/index.html)

Commands

Halcon Script ausführen 1

Parameters

Script Datei 2

fester Text
Text: filter.hdev

Methode
fester Text
Text: run_filter

Inputs 3

Input Element
Bezeichner: InputImage
Quelle: HImage aus Graubild
Bildseite: -1

Input Element
Bezeichner: FilterSize
Quelle: Zahl oder Vector aus Ergebnis(en)
Ergebnis Variablen
Liste
erstes Ergebnis: 5
letztes Ergebnis:

Ergebnisse speichern 4

Ergebnis
in Bildseite
Bezeichner: OutputImage
Bildseite: 2
Kamera: 0

Generate JSON 5

1. Wählen Sie im Drop-Down Menü "Halcon Script ausführen"
2. Tragen Sie unter **Script Datei** und **Methode** die betreffende *.hdev Datei und die Methode ein.
3. Stellen Sie unter **Inputs** alle Eingabeparameter der Methode ein. Die Reihenfolge der Parameter ist frei wählbar. Die Zuordnung erfolgt über die Bezeichner.
4. Tragen Sie unter **Ergebnisse speichern** gewünschte Ausgabeparameter der Methode ein.
5. Wenn alle Parameter eingestellt sind, erstellen Sie mit der Schaltfläche [Generate JSON] den JSON-Befehlscode. Markieren Sie den Inhalt des Fensters und kopieren Sie ihn in die Zwischenablage.

4 Befehlscode mittels Rubyscript ausführen

1. Fügen Sie in Ihrem vcwin Testprogramm an der gewünschten Stelle einen Rechenscriptbefehl ein.
2. Verwenden Sie den `VC::Command.run` Befehl um den JSON-Befehlscode über Rubyscript auszuführen.

```
# run halcon filter
retval,error,testresults = VC::Command.run( %{

```

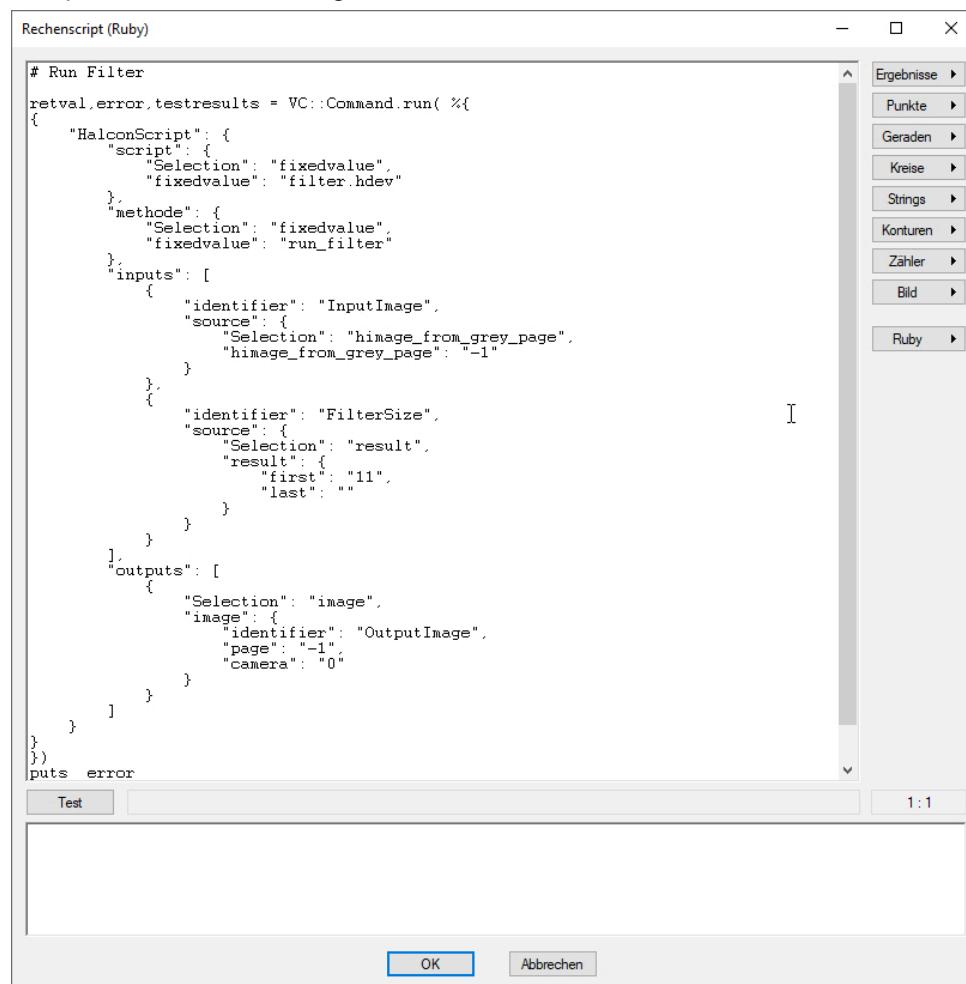
3. Fügen Sie im Scriptfenster des Dialogs Rechenscript den kopierten JSON-Befehlscode ein.
4. Schließen Sie das Script mit dem Code für die Fehlerbehandlung ab.

```
    }
    puts error
    return (retval==0)
}
```

`retval` ist der ErrorCode den VCWin Befehle zurückgeben. Der Ruby-Script-Befehl schlägt fehl, wenn das HalconScript fehlschlägt.

In der Ausgabe von `puts error` werden die Fehler dargestellt, z.B. die Exception Message falls Halcon eine Exception geworfen hat.

5. Mit der Schaltfläche [Test] können Sie den aktuellen Code auf seine syntaktische Richtigkeit überprüfen und sich das Ergebnis des Befehls rechts neben der Schaltfläche anzeigen lassen.



```
Rechenscript (Ruby)

# Run Filter
retval,error,testresults = VC::Command.run( %{
{
  "HalconScript": {
    "script": {
      "Selection": "fixedvalue",
      "fixedvalue": "filter.hdev"
    },
    "method": {
      "Selection": "fixedvalue",
      "fixedvalue": "run_filter"
    }
  },
  "inputs": [
    {
      "identifier": "InputImage",
      "source": {
        "Selection": "himage_from_grey_page",
        "himage_from_grey_page": "-1"
      }
    },
    {
      "identifier": "FilterSize",
      "source": {
        "Selection": "result",
        "result": {
          "first": "11",
          "last": ""
        }
      }
    }
  ],
  "outputs": [
    {
      "Selection": "image",
      "image": {
        "identifier": "OutputImage",
        "page": "-1",
        "camera": "0"
      }
    }
  ]
}
})
puts error
```